Simultaneous learning : New Textbook

Math & Python

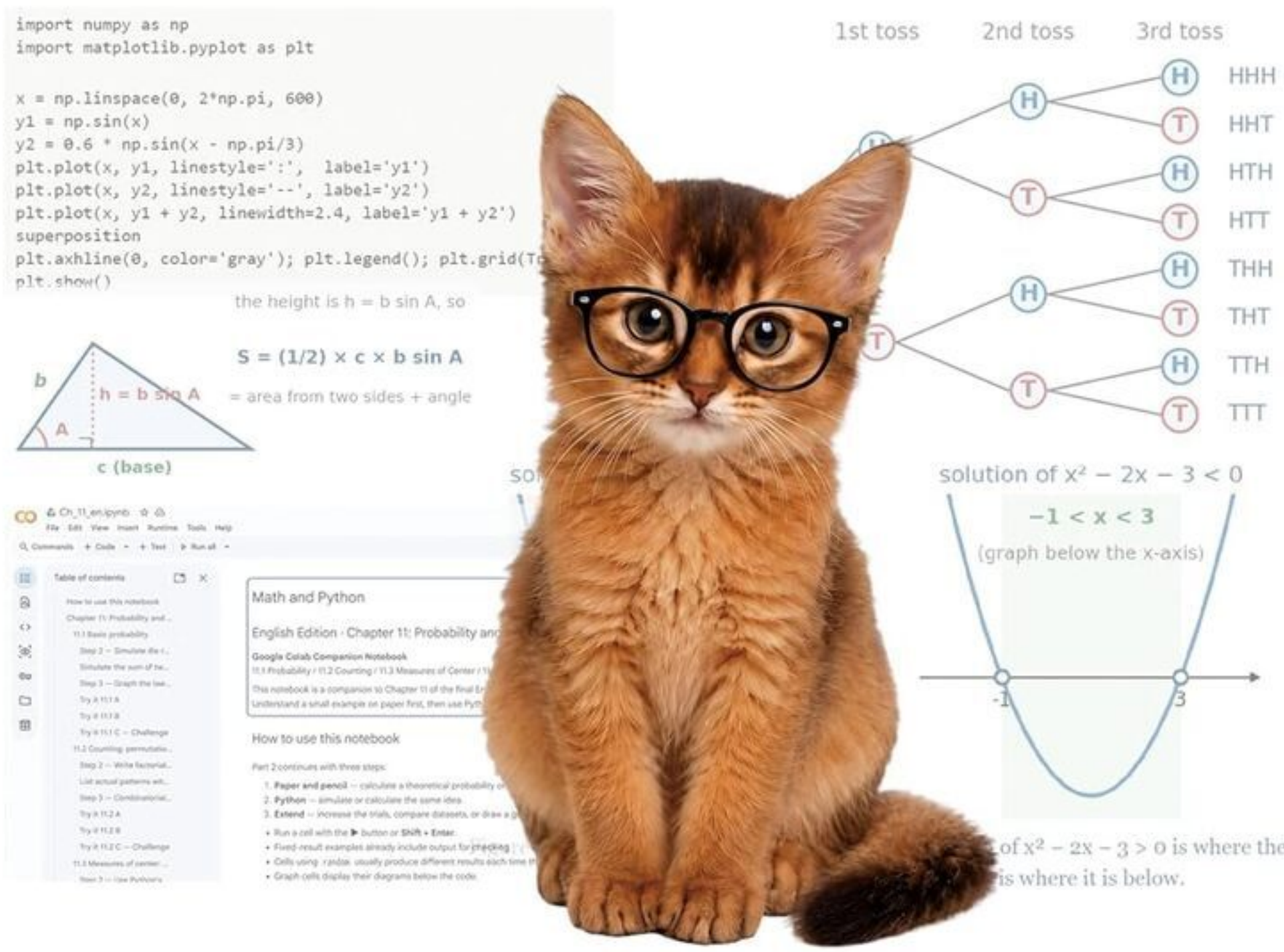
Level 2. "Deepening your understanding of mathematics with Python"

From trigonometric functions to complex numbers and higher-order equations ....

Level 2

"Curiosity about mathematics,

the foundation of everything in the AI era."



Two free downloadable files : Hands-On Math with Python

* Efficient learning with Colab note files linked to this book.

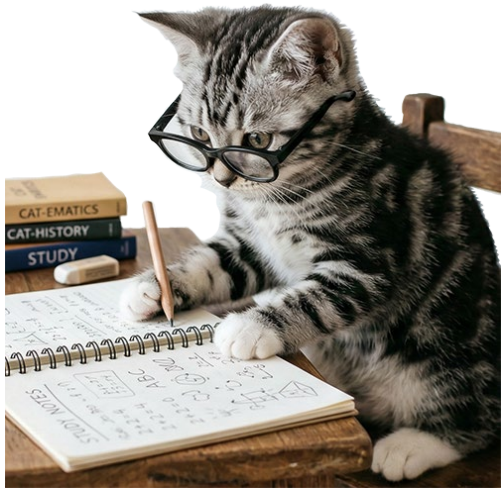
* Beginner-friendly Python code explained step-by-step.

FilesStart Instantly in Your Browser—No Setup!

CHIC INSTITUTE

Hideaki Miyoshi / Geraldine Angaga Lickas

Simultaneous learning of Math and Python



Google Colaboratory, Google Colab, and the Google logo are trademarks or registered trademarks of Google LLC.
Python is a trademark or registered trademark of the Python Software Foundation.

Preface

Today, we live in an era where AI (Artificial Intelligence) is part of our everyday lives. Instead of just watching this big change, we need the power to ride this wave ourselves. To do that, we must understand AI not as "magic," but as a "system."

This book was written to help you learn two powerful basic skills at the same time: "Mathematics," and "Python" (programming) to make it work in the real world. By combining the logical thinking of math with the creative power of programming, abstract formulas turn into real tools you can actually "touch and move."

I want young people who will create the future to read this book. But I also strongly recommend it to adults who gave up on math in the past, or who never had the chance to learn. It is never too late to start learning.

I hope this book becomes a bridge that gives you back the "joy of understanding" and the "power to create." And I truly wish that readers of this book will step forward to seek the truth of the world and build a new future.

Hideaki Miyoshi / Geraldine Angaga Lickas

How to Use This Book — Paper, Pen, and Python

The way to learn with this book is very simple. You will always move between these three things.

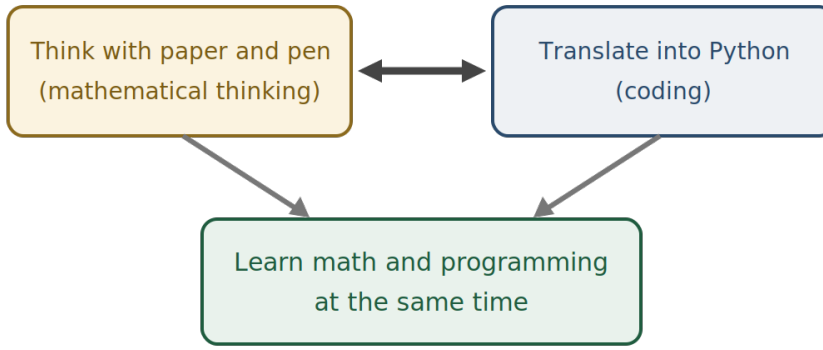


Figure 1.1 The learning flow of this book

① 🖍️ **Think with paper and pen.** First, use your own hands. Write equations, draw pictures, and ask yourself, "What does this mean?" This is **mathematical thinking**.

② 💻 **Translate it into Python.** Next, "translate" your ideas into Python code. The computer can calculate and draw graphs in a moment. This is **coding**.

③ ⭐ **Learn both at the same time.** The equation on paper and the result on the screen match — "They are exactly the same!" When you repeat this experience, you will learn math and programming **at the same time, and deeply**.

Do you remember the three steps from the level 1 version: "🖍️ Paper → 🔗 Connect → 💻 Python"? The level 2 version adds time to think about "Why?" and a bridge to your future. So now there are four steps.

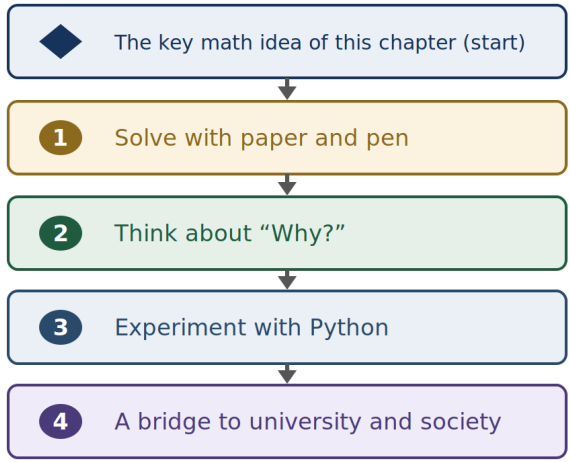


Figure 1.2 The four-step learning

At the start of each chapter, you first catch “the key math idea of this chapter” (the starting point). Then you learn through the four steps below. You do not have to do all of them every time. At first, just moving between 🖍️ and 💻 is enough.

🎯 **The key math idea of this chapter (start)** Catch “what you will learn, and why” in a few words.

Step 1 🖍️ **Solve with paper and pen** Calculate with your own hands and change the form of equations. When you move your hands, the ideas become part of you.

Step 2 🔍 **Think about “Why?”** Think about what a formula means and why a definition is made that way. This is the most interesting part of high school math.

Step 3 💻 **Experiment with Python** Make a guess, then check it with code. “I tried it, and it really happened!” — this makes your understanding strong.

Step 4 🚀 **A bridge to next level** Take a quick look at how what you learn will be useful in the future.

Different skill levels are OK

Maybe you are good at programming, or maybe it has been a long time. This book explains everything with both pictures and code. If the code feels difficult, just follow the pictures and the “paper and pen” explanations first. If you are worried about math, run the Python code and check the results with your own eyes — it will become much easier to understand. Start from the entrance that fits you.

Before You Start — Let’s Use Google Colab

In this textbook, we will learn Python with a tool called “Google Colab.”

Google Colab is a useful free service. Anyone can use Python with just a browser (Chrome, Safari, Edge, etc.) connected to the internet. You do not need to install anything on your computer.

<**Notice**> Colab notebook files linked to the book can be downloaded from the following address:

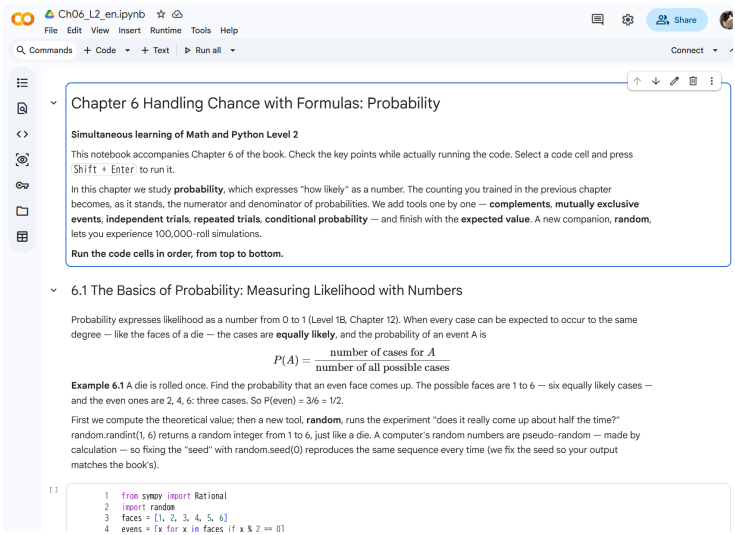
https://chic.institute/file-downloads/

Please use this by selecting "Upload Notebook" from the Colab file menu.

■ Two types of Colab notebook files (download)

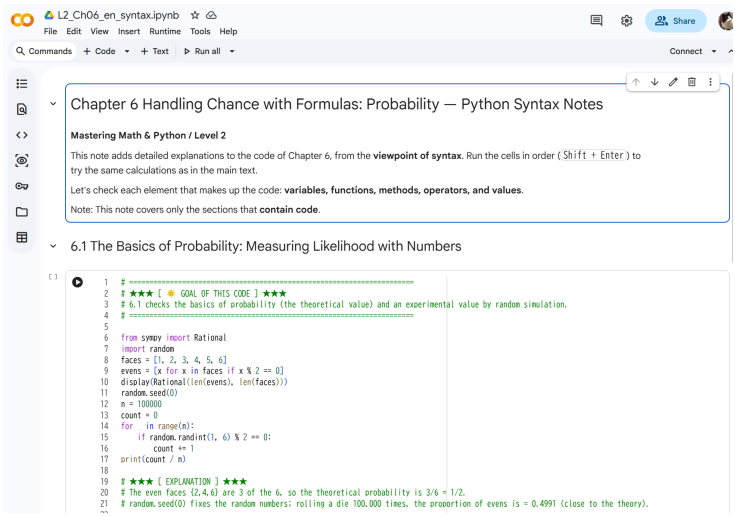
- **Book-linked Colab notebook**

  Learn efficiently while running the code



- **Python Syntax Notes**

  Explains the Python code syntax at each step



Nice points of Google Colab

- No need to install Python on your computer
- You can use it on a smartphone or tablet
- Your programs are saved to Google Drive automatically
- You can start right away with a Google account

■ Step 1: Open Colab

First, get a Google account ready. A school account or your own Gmail account is fine.

Next, go to this address in your browser.

```
https://colab.research.google.com/
```

If it asks you to log in to your Google account, please log in.

■ Step 2: Write and Run Code

A notebook has input boxes called “cells.” You write your Python code in these cells.

Running the code is very easy.

1. Type your code in a cell
2. Click the ▷ button on the left of the cell (or press Shift + Enter)
3. The result appears under the cell



▲ *Figure 0.1 The Google Colab screen and the names of its parts*

To add a new cell, click the "+ Code" button at the top of the screen. Add more and more cells and try your code.

■ Useful Shortcuts to Remember

Keys	What it does
`Shift + Enter`	Run this cell and move to the next cell
`Ctrl + Enter`	Run only this cell
`Ctrl + S`	Save the notebook (usually automatic)
`Ctrl + M, B`	Add a new cell below

> **For Mac users**
>
> On a Mac, use the Command (⌘) key instead of the Ctrl key.
>
> • Shift + Enter is the same
>
> • Ctrl + S becomes ⌘ + S
>
> • Ctrl + Enter becomes ⌘ + Enter

■ Let's Try It Now!

When you are ready, type this code in the first cell and run it with the ▷ button.

```
print("Start learning Python!")
```

If you see “Start learning Python!” under the cell, you are ready. Now let’s move on to 1.1!

■ Official Links

If you want to know more, check the official Google pages, too. They show the latest information and useful tips.

Page	URL
Colab (login)	https://colab.research.google.com/
Colab official site	https://colab.google/
Google for Developers	https://developers.google.com/colab

> **Bookmark It First**
>
> You will open Colab many times while you study.
>
> If you add it to your browser’s bookmarks (favorites), you can open it quickly next time.

Now, let’s start our journey into the world of math and Python!

Contents

How to Use This Book — Paper, Pen, and Python 3
Before You Start — Let's Use Google Colab 5
Chapter 1 Gateway to Level 2 Math: Organizing Numbers and Expressions 15
1.1 The World of Real Numbers: Sorting Our Numbers 15
1.2 Computing with Radical Expressions 19
1.3 Absolute Value: Distance on the Number Line 22
1.4 Expanding Expressions 24
1.5 Factoring: Walking Expansion Backwards 27
1.6 Linear Inequalities: Solving Order Relations 28
Chapter 2 Thinking with Logic: Sets and Propositions 38
2.1 The Basics of Sets: Turning Collections into Mathematics 38
2.2 Operations on Sets: Intersection, Union, and Complement 41
2.3 Propositions and Conditions: Statements with a Definite Truth Value 44
2.4 Converse, Inverse, and Contrapositive: Turning a Proposition Around 46
2.5 Necessary and Sufficient Conditions 49
2.6 The Idea of Proof by Contradiction 51
Chapter 3 Building a Solid Foundation for Functions: Quadratic Functions 60
3.1 The Idea of a Function: Naming the Rule of Correspondence 61
3.2 The Graph of a Quadratic Function: Mastering the Parabola 62
3.3 Completing the Square: Making the Vertex Appear 65
3.4 Maximum and Minimum: The Highest and Lowest Points 67
3.5 The Link to Quadratic Equations: Where the Graph Crosses the x-axis 68
3.6 Quadratic Inequalities: Solving with Graphs 71

Chapter 4 Measuring Figures with Numbers: Trigonometric Ratios..... 81
4.1 What Trigonometric Ratios Are: sin, cos, and tan 81
4.2 Relations Among the Trigonometric Ratios: Know One, Know Them All 85
4.3 Trigonometric Ratios of Obtuse Angles: Beyond the 90° Wall 86
4.4 The Law of Sines: A Bridge Between Angles and Sides (Part 1) 88
4.5 The Law of Cosines: A Bridge Between Angles and Sides (Part 2) . 90
4.6 The Area of a Triangle and Measuring Figures 92
Chapter 5 Making Counting Systematic: The Number of Cases 101
5.1 The Principle of Counting: No Omissions, No Double Counting... 102
5.2 The Addition Rule: Split into Cases, Then Add.....103
5.3 The Multiplication Rule: Split into Stages, Then Multiply 105
5.4 Permutations: Choose and Arrange..... 107
5.5 Combinations: Just Choosing — Order Does Not Matter 109
5.6 Counting with Repetition..... 111
Chapter 6 Handling Chance with Formulas: Probability 120
6.1 The Basics of Probability: Measuring Likelihood with Numbers ...121
6.2 Complements and Mutually Exclusive Events: Probability by Subtracting and Adding..... 123
6.3 Independent Trials: Probabilities Joined by Multiplication..... 126
6.4 Repeated Trials: Trying the Same Thing Again and Again..... 127
6.5 Conditional Probability: Information Changes Probability..... 129
6.6 Expected Value: How Much on Average?.....131
Chapter 7 Deepening the Properties of Shapes: Plane and Solid Geometry 140
7.1 Properties of Triangles: Special Lines and Special Points.....141
7.2 Properties of Circles: Inscribed Angles and the Power of a Point . 144
7.3 Constructions: Seen as "Sets of Equidistant Points"..... 146
7.4 Ceva's Theorem and Menelaus's Theorem 148
7.5 Basics of Solid Geometry: Polyhedra and Measurement 151

7.6 Viewing Geometry Logically: A Theorem and Its Converse..... 154
Chapter 8 Extending Expressions: Complex Numbers and Higher-Degree Equations 164
8.1 Computing with Polynomials: Division and the Remainder Theorem 165
8.2 Algebraic Fractions: Fractions in the World of Expressions 167
8.3 The Binomial Theorem: Finding Expansion Coefficients at Once 169
8.4 Complex Numbers: Extending the World of Numbers..... 170
8.5 Quadratic Equations Revisited: The Discriminant and the Relation between Roots and Coefficients..... 173
8.6 Higher-Degree Equations: Lowering the Degree with the Factor Theorem 176
Chapter 9 Handling Shapes with Coordinates: Geometry and Equations 185
9.1 Points and Distances: Basics of the Coordinate Plane 186
9.2 Equations of Lines..... 188
9.3 Two Lines: Parallel, Perpendicular, and Distance 190
9.4 Equations of Circles 192
9.5 Loci: Collections of Points Satisfying a Condition 194
9.6 Regions: The World Described by Inequalities..... 196
Chapter 10 Reading Periodic Change: Trigonometric Functions..... 206
10.1 General Angles and Radian Measure: Widening the World of Angles 207
10.2 Trigonometric Functions: Ratios Set Free on the Unit Circle..... 209
10.3 Graphs and Periods of Trigonometric Functions 211
10.4 The Addition Formulas: Adding Angles 214
10.5 Double-Angle, Half-Angle, and Combining Waves 215
10.6 Trigonometric Equations and Inequalities: Solving on the Unit Circle..... 218
Chapter 11 Handling Rapid Change: Exponential and Logarithmic

Functions 227
11.1 Extending Exponents: Zeroth, Negative, and Fractional Powers 228
11.2 Exponential Functions: Graphs of Explosive Change 230
11.3 Logarithms: Exponents Read Backwards 232
11.4 Laws of Logarithms: Turning Products into Sums 233
11.5 Logarithmic Functions and Common Logarithms: Naming the Digits 235
11.6 Exponential and Logarithmic Equations: Match the Bases 237
Appendix 1 A Summary of Python Basics 247
A1.1 Running Python 247
A1.2 Arithmetic in Python 248
A1.3 Variables 249
A1.4 Conditionals 250
A1.5 Repeating with Loops 251
A1.6 Working with Collections of Data 251
A1.7 Functions 252
A1.8 Drawing Graphs 253
A1.9 Creating Classes 255
Appendix 2 Essential LaTeX (MathJax) Notation 258
A2.1 Math Mode Basics 258
A2.2 Superscripts and Subscripts: Exponents and Indices 259
A2.3 Fractions and Roots 260
A2.4 Greek Letters and Symbols 261
A2.5 Function Names: sin, cos, log, and Friends 261
A2.6 Brackets, Spacing, and Text 262
A2.7 Notation You Will Meet Later: Sums, Integrals, Limits 263

Chapter 1
Gateway to Level 2 Math: Organizing Numbers and Expressions

Real numbers

Rational and irrational numbers

Expressions with radicals

Absolute value

Expanding expressions

Factoring

Linear inequalities

Chapter 1 Gateway to Level 2 Math: Organizing Numbers and Expressions

What You Will Learn

In Levels 1A and 1B, we toured the whole world of middle school mathematics: positive and negative numbers, algebraic expressions, equations, square roots, quadratic equations, functions, geometry, and statistics. Level 2, which starts here, is the gateway to high school mathematics. Before climbing this new mountain, this chapter reorganizes the "numbers" and "expressions" you have learned so far, viewed from one step higher up.

Reorganizing is not mere review. For example, the integers, fractions, decimals, and square roots you studied separately can all be seen as residents of one big world called the real numbers. Expansion and factoring can be understood together, as operations that run in opposite directions. Just as a map makes travel easier, holding a complete map of numbers and expressions will make your Level 2 journey much smoother.

The way we work stays the same as before. First understand each calculation with pen and paper, then run the same calculation in Python to check it. SymPy, the symbolic math library that served us so well in Level 1B, plays a starring role in this chapter too.

By the end of this chapter, you will be able to:

- explain the relationship between real, rational, and irrational numbers with a diagram
- simplify and compute with radical expressions such as $\sqrt{18}$
- explain absolute value on the number line and compute with it
- expand expressions using the multiplication formulas
- factor expressions using common factors and the formulas
- solve linear inequalities and show the solutions on a number line
- check expansion, factoring, and inequalities with SymPy's expand, factor, and solve

1.1 The World of Real Numbers: Sorting Our Numbers

Let us begin by organizing all the numbers we have met so far. The first numbers we ever encountered were 1, 2, 3, ... for counting things. These are called the **natural numbers**. Then 0 and the negative numbers joined in, expanding our world to the **integers**: ..., −2, −1, 0, 1, 2, ... (Level 1A, Chapter 10).

Integers alone cannot express "splitting 2 cakes among 3 people." That is where the fraction $\frac{2}{3}$ comes in. A number that can be written in the form $\frac{a}{b}$, using an integer a and a nonzero integer b, is called a **rational number**. The integer 5 is rational too, since it can be written as $\frac{5}{1}$. So is the decimal 0.75, since it equals $\frac{3}{4}$.

> **Term** **Rational number** A number that can be written as a fraction $\frac{a}{b}$ with an integer a and a nonzero integer b. The word comes from "ratio": a rational number is a ratio of integers.

However, as we learned in Level 1B, Chapter 2, there are numbers in the world that cannot be written as fractions. For example, $\sqrt{2}$ = 1.41421356... cannot be expressed exactly by any fraction. Such numbers are called **irrational numbers**. $\sqrt{3}$ and the circle constant π = 3.14159265... are also irrational.

> **Term** **Irrational number** A number that cannot be written as a fraction. As a decimal, it goes on forever without repeating. Examples: $\sqrt{2}$, $\sqrt{3}$, π.

Putting all the rational and irrational numbers together gives the world of the **real numbers**. The real numbers correspond to all the points on the number line: a world of numbers packed together with no gaps at all.

> **Term** **Real number** Any rational or irrational number. The real numbers correspond one-to-one with the points on the number line. Level 2 mathematics takes place mainly in this world.

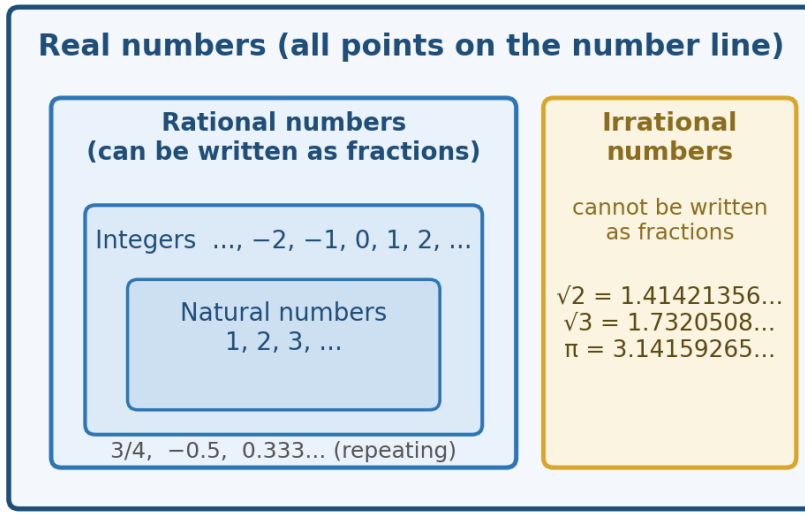


Figure 1.1 Classifying the real numbers. The rationals contain the integers, and the integers contain the natural numbers.

Figure 1.1 is the complete map of our number world. The natural numbers sit inside the integers, the integers inside the rationals, and the rationals together with the irrationals make up the reals: a nested structure, like matryoshka dolls.

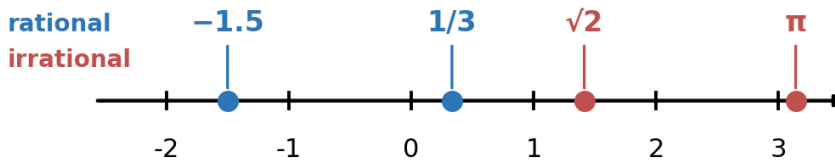


Figure 1.2 Various numbers on the number line. Rational and irrational numbers alike have exact positions on the line.

Telling Them Apart by Their Decimals: Terminating and Repeating

You can tell rational and irrational numbers apart by what they look like as base-10 decimals — the everyday decimal notation we normally use. A rational number, written as a decimal, is always one of the following two kinds.

- **Terminating decimal**: a decimal that ends somewhere, like $\frac{7}{8} = 0.875$

- **Repeating decimal**: a decimal in which the same block of digits repeats forever, like $\frac{5}{12} = 0.41666\ldots$

An irrational number, by contrast, becomes a decimal that neither ends nor repeats. Remember it this way: "ends or repeats → rational; never ends and never repeats → irrational."

Example 1.1

Classify the following six numbers as natural numbers, integers, rational numbers, or irrational numbers (some belong to more than one group).

$$5, \quad -3, \quad 0.75, \quad \frac{1}{3}, \quad \sqrt{2}, \quad \pi$$

With pen and paper Place each number on the map in Figure 1.1. 5 is a natural number (and also an integer and a rational). -3 is an integer (and a rational). 0.75 is rational because it can be written as $\frac{3}{4}$. $\frac{1}{3}$ is a fraction itself, so it is rational (as a decimal it repeats: 0.333…). $\sqrt{2}$ and π cannot be written as fractions, so they are irrational.

Check it with Python With SymPy, fractions and radicals stay exact.

```
from sympy import sqrt, Rational  # exact values
print(Rational(3, 4))    # 0.75 as a fraction
print(sqrt(2))           # square root of 2, exact form
print(sqrt(2).evalf())   # as a decimal (approximation)
```

```
Output ▸
3/4
sqrt(2)
1.41421356237310
```

Rational(3, 4) keeps the fraction $\frac{3}{4}$ exactly as it is. Likewise, SymPy does not rush sqrt(2) into a decimal; it keeps the exact form sqrt(2). Calling evalf() shows 1.41421356…, but that is only an approximation — the true $\sqrt{2}$ goes on forever. Being able to compute with exact forms is the big difference between SymPy and a calculator.

Example 1.2

Write the fractions $\frac{7}{8}$ and $\frac{5}{12}$ as decimals, and say whether each is terminating or repeating.

With pen and paper To convert a fraction to a decimal, divide the

numerator by the denominator. Long division gives 7 ÷ 8 = 0.875, which terminates. But 5 ÷ 12 = 0.41666..., with the 6 repeating forever — a repeating decimal.

In fact, you can predict the outcome from the denominator. If the prime factors of the reduced denominator are only 2s and 5s, the decimal terminates; if any other prime factor (3, 7, ...) appears, it repeats. Here 8 = 2×2×2, so it terminates; 12 = 2×2×3 contains a 3, so it repeats (we studied prime factorization in Level 1A, Chapter 11).

Check it with Python This time let us also try fractions, a module from Python's standard library.

```
from fractions import Fraction  # standard module
print(Fraction(7, 8))    # the fraction 7/8
print(7 / 8)             # divide to get a decimal
print(5 / 12)            # and this one...
```

Output ▸

```
7/8
0.875
0.4166666666666667
```

7 / 8 ended neatly at 0.875. The true value of 5 / 12 is 0.41666... forever, but a computer can hold only finitely many digits, so the last digit is rounded up to 7 in the display. When you read a decimal result, keep in mind that what you see may be an approximation.

1.2 Computing with Radical Expressions

Recap (Level 1B, Chapter 2) For a positive number a, the positive number whose square is a is written $\sqrt{a}$ (root a) and called the positive square root of a. For example, $(\sqrt{2})^2 = 2$. The symbol √ is called the radical sign.

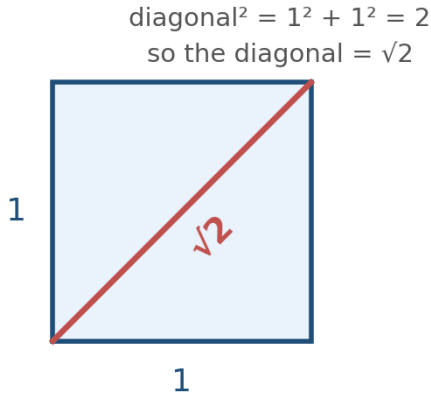


Figure 1.3 The diagonal of a unit square has length $\sqrt{2}$. Irrational numbers live inside everyday shapes.

As Figure 1.3 shows, $\sqrt{2}$ is a length you can draw right in front of you: the diagonal of a square with side 1. Irrational numbers are not exotic at all — they are all around us. In this section, let us become fluent at computing with radical expressions.

Rules for Radicals

For a > 0 and b > 0, the following rules hold.

$$\sqrt{a} \times \sqrt{b} = \sqrt{ab} \qquad \frac{\sqrt{a}}{\sqrt{b}} = \sqrt{\frac{a}{b}}$$

In other words, multiplication and division can be combined under one radical sign. Reading this rule backwards lets you make the number under the radical smaller: if a square factor (4, 9, 16, 25, ...) is hiding inside, you can pull it out.

$$\sqrt{a^2 b} = a\sqrt{b} \quad (a > 0, b > 0)$$

Caution: addition under a radical cannot be split apart For example, $\sqrt{9 + 16} = \sqrt{25} = 5$, but $\sqrt{9} + \sqrt{16} = 3 + 4 = 7$. Therefore $\sqrt{a + b} = \sqrt{a} + \sqrt{b}$ does not hold in general. Only multiplication and division may be combined under one radical sign.

Example 1.3

Write $\sqrt{18}$ in the form $a\sqrt{b}$.

With pen and paper Factor 18 and look for square factors: $18 = 9 \times 2 = 3^2 \times 2$. Since 9 is 3^2, it can come out of the radical.

$$\sqrt{18} = \sqrt{9 \times 2} = \sqrt{9} \times \sqrt{2} = 3\sqrt{2}$$

Check it with Python SymPy's sqrt performs this simplification automatically.

```
from sympy import sqrt
print(sqrt(18))   # simplified automatically
```

```
Output ▸
3*sqrt(2)
```

The output 3*sqrt(2) means $3\sqrt{2}$. It matches our hand calculation.

Adding and Subtracting Radical Expressions

Radicals with the same number inside can be combined, just like like terms in algebra (Level 1A, Chapter 12). $2\sqrt{3} + 3\sqrt{3} = 5\sqrt{3}$ works exactly like 2x + 3x = 5x. When the numbers inside differ, first rewrite each radical in the form $a\sqrt{b}$ — the same radical may then appear.

Example 1.4

Compute $\sqrt{12} + \sqrt{27}$.

With pen and paper As they stand, the radicals differ, so we cannot add them. Rewrite each in the form $a\sqrt{b}$.

$$\sqrt{12} = \sqrt{4 \times 3} = 2\sqrt{3} \qquad \sqrt{27} = \sqrt{9 \times 3} = 3\sqrt{3}$$

The same $\sqrt{3}$ has appeared, so we can combine them.

$$2\sqrt{3} + 3\sqrt{3} = 5\sqrt{3}$$

Check it with Python

```
from sympy import sqrt
print(sqrt(12) + sqrt(27))   # combined automatically
```

```
Output ▸
5*sqrt(3)
```

SymPy simplifies $\sqrt{12}$ and $\sqrt{27}$ separately, then combines the matching radicals — all in one step.

Rationalizing the Denominator

When a fraction has a radical in its denominator, you can remove it by multiplying both the numerator and the denominator by that same radical. This is called **rationalizing the denominator**. With an integer denominator, the size of the number is easier to estimate and further calculation becomes easier.

Example 1.5

Rationalize the denominator of $\frac{6}{\sqrt{3}}$.

With pen and paper Multiply the numerator and denominator by $\sqrt{3}$.

$$\frac{6}{\sqrt{3}} = \frac{6 \times \sqrt{3}}{\sqrt{3} \times \sqrt{3}} = \frac{6\sqrt{3}}{3} = 2\sqrt{3}$$

The key point is that $\sqrt{3} \times \sqrt{3}$ in the denominator becomes 3. Reducing the fraction gives $2\sqrt{3}$.

Check it with Python SymPy rationalizes automatically the moment you enter the division.

```
from sympy import sqrt, radsimp
print(6 / sqrt(3))        # rationalized automatically
print(radsimp(2 / (sqrt(5) - sqrt(3))))  # use radsimp
```

Output ▸

```
2*sqrt(3)
sqrt(3) + sqrt(5)
```

The second line is a slightly advanced example. Even when the denominator is a difference like $\sqrt{5} - \sqrt{3}$, the function radsimp (radical simplify) can rationalize it. The trick: multiply numerator and denominator by $\sqrt{5} + \sqrt{3}$, and the denominator becomes $(\sqrt{5} - \sqrt{3})(\sqrt{5} + \sqrt{3}) = 5 - 3 = 2$ — the radicals vanish. This "sum times difference" pattern is one of the multiplication formulas we study in Section 1.4.

1.3 Absolute Value: Distance on the Number Line

Recap (Level 1A, Chapter 10) The distance between a number and 0 on the number line is called the absolute value of that number, written with the symbol $|\ |$. For example, $|-3| = 3$ and $|3| = 3$.

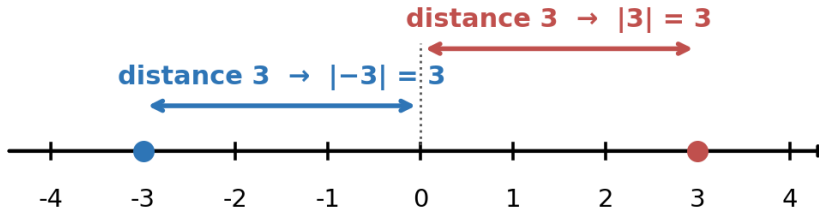


Figure 1.4 Absolute value is "distance from 0." Both −3 and 3 are at distance 3 from 0.

As Figure 1.4 shows, −3 and 3 both sit at distance 3 from 0. Distance ignores direction and is never negative, so an absolute value is never negative either. In Level 2 we will use absolute values inside algebraic expressions, so let us restate the definition as a formula.

> **Term** **Absolute value** The distance between a number and 0 on the number line, written $|a|$. The inequality $|a| \geq 0$ always holds.

The absolute value of a number a can be written in two cases, according to the sign of a.

- If $a \geq 0$, then $|a| = a$ (e.g. $|5| = 5$, $|0| = 0$)
- If $a < 0$, then $|a| = -a$ (e.g. $|-5| = -(-5) = 5$)

Watch the −a in the second case. Since a itself is negative, attaching a minus sign makes −a positive. "To remove absolute value bars, split into cases by the sign of what is inside" — this technique will return again and again in Level 2 equations and functions.

Example 1.6

Find the following values.

(1) $|-7|$ (2) $|0.5|$ (3) $|3-8|$

With pen and paper (1) The distance between −7 and 0 is 7. (2) 0.5 is positive, so it stays 0.5. (3) Compute the inside first: $3 - 8 = -5$, and $|-5| = 5$. The golden rule: evaluate what is inside the bars first, then take the absolute value.

Check it with Python Python has a built-in function abs (absolute value).

```
print(abs(-7))     # absolute value of -7
print(abs(0.5))    # positive numbers stay the same
print(abs(3 - 8))  # the inside 3-8 is computed first
```

Output ▸

```
7
0.5
5
```

abs evaluates the expression in the parentheses first, then takes the absolute value — the same order as our "inside first" rule by hand.

Absolute value as distance The absolute value of a difference, |a − b|, gives the distance between a and b on the number line. For example, |3 − 8| = 5 says that 3 and 8 are 5 apart. This point of view will be useful again in functions and data analysis from Level 2B onward.

1.4 Expanding Expressions

Recap (Level 1A, Chapter 16) Rewriting a product of parentheses as a sum without parentheses (a polynomial) is called expanding. The foundation of expansion is the distributive law $a(b + c) = ab + ac$.

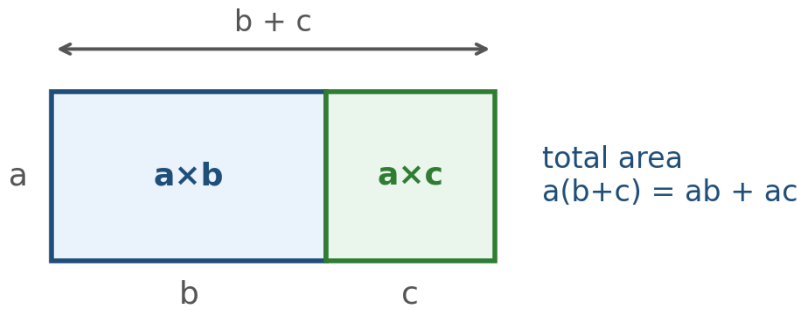


Figure 1.5 Area model of the distributive law: a rectangle of width b+c and height a, split into two parts.

As Figure 1.5 shows, the distributive law can be made visible with rectangle areas. A rectangle of width b + c and height a has area a(b + c). Splitting it into left and right rectangles gives ab + ac. These are two ways of writing the same area, so they must be equal.

A product of two parentheses, $(a + b)(c + d)$, can also be expanded by using the distributive law twice. First treat (a + b) as a single block M, so that M(c + d) = Mc + Md. Then restore M and distribute once more.

$$(a + b)(c + d) = ac + ad + bc + bd$$

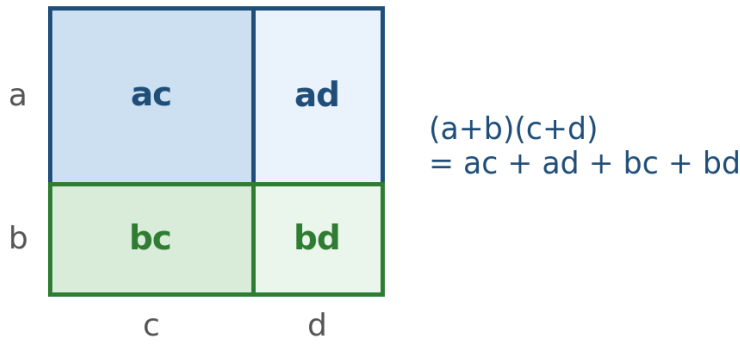


Figure 1.6 Area model of (a+b)(c+d): one large rectangle divided into four rooms.

Look at Figure 1.6. The large rectangle, with height a + b and width c + d, divides into four rooms: ac, ad, bc, bd. The trick is to multiply every pair without missing any: firsts, outers, inners, lasts.

Multiplication Formulas: Frequent Patterns Become Formulas

Four expansion patterns appear so often that memorizing them as **multiplication formulas** makes your calculations faster and more reliable.

Table 1.1 Multiplication formulas

Formula	Name
$(x + a)(x + b) = x^2 + (a + b)x + ab$	sum and product
$(a + b)^2 = a^2 + 2ab + b^2$	square of a sum
$(a - b)^2 = a^2 - 2ab + b^2$	square of a difference
$(a + b)(a - b) = a^2 - b^2$	sum times difference

Every one of these formulas can be derived from the four-room area diagram in Figure 1.6. Rather than rote memorization, understand them as "results of the distributive law with names attached" — then you can rebuild any of them on the spot if you forget.

Example 1.7

Expand $(x + 3)(x - 5)$.

With pen and paper Use the sum-and-product formula with a = 3 and b = −5. The sum is 3 + (−5) = −2 and the product is 3 × (−5) = −15.

$$(x + 3)(x - 5) = x^2 + (3 - 5)x + 3 \times (-5) = x^2 - 2x - 15$$

When negative numbers are involved, watch the signs of the sum and the product (the sign rules are just as in Level 1A, Chapter 10).

Check it with Python Use SymPy's expand. To use x as a mathematical letter, first declare it with symbols.

```
from sympy import symbols, expand
x = symbols('x')      # declare x as a math symbol
print(expand((x + 3) * (x - 5)))   # expand the product
```

Output ▸

```
x**2 - 2*x - 15
```

Recall that x**2 in the output means x^2 (** is exponentiation). It matches our hand calculation.

Example 1.8

Use the multiplication formulas to expand the following.

(1) $(2x + 3)^2$ (2) $(x + 4)(x - 4)$

With pen and paper (1) Use the square-of-a-sum formula with a = 2x and b = 3.

$$(2x + 3)^2 = (2x)^2 + 2 \times 2x \times 3 + 3^2 = 4x^2 + 12x + 9$$

Note that $(2x)^2 = 4x^2$: the whole of 2x is squared, so the coefficient 2 gets squared too. (2) is exactly the sum-times-difference formula.

$$(x + 4)(x - 4) = x^2 - 4^2 = x^2 - 16$$

Check it with Python

```
from sympy import symbols, expand
x = symbols('x')
print(expand((2*x + 3)**2))      # square of a sum
print(expand((x + 4) * (x - 4))) # sum times difference
```

```
Output ▸
4*x**2 + 12*x + 9
x**2 - 16
```

Both match the hand calculations. Note that 2x must be written 2*x in code. Mathematics lets you omit the multiplication sign, but Python always requires the *.

1.5 Factoring: Walking Expansion Backwards

Now we go the other way. Rewriting a polynomial as a product of expressions is called **factoring**, and each expression in the product is called a **factor**.

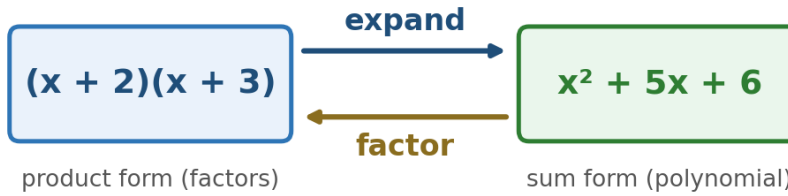


Figure 1.7 Expansion and factoring are inverse operations: rightward is expansion, leftward is factoring.

> **Term** **Factoring** Rewriting a polynomial as a product of factors (expressions); the inverse operation of expansion. Example: $x^2 + 5x + 6 = (x + 2)(x + 3)$

As Figure 1.7 shows, expansion and factoring are one-way streets running in opposite directions. Expansion is mechanical — just follow the distributive law — but factoring asks you to deduce "what product was this?", so it takes a little more thought. The deduction goes in two steps.

- **Step 1**: look for a common factor and pull it outside the parentheses
- **Step 2**: then read the multiplication formulas (Table 1.1) from right to left and look for a matching pattern

Example 1.9

Factor the following expressions.

(1) $6x^2 + 9x$ (2) $x^2 + 7x + 12$

With pen and paper (1) This is Step 1. Since $6x^2$ = 3x × 2x and 9x = 3x × 3, the common factor 3x is hiding in both terms. Pulling it out,

$$6x^2 + 9x = 3x(2x + 3)$$

(2) There is no common factor, so we move to Step 2. Why do we look for two numbers whose sum is 7 and whose product is 12? Expanding $(x + m)(x + n)$ gives $x^2 + (m + n)x + mn$. So to factor $x^2 + 7x + 12$, we need two numbers with m + n = 7 and mn = 12. Searching, we find 3 and 4 (3 + 4 = 7, 3 × 4 = 12). Therefore,

$$x^2 + 7x + 12 = (x + 3)(x + 4)$$

You can always check your factorization by expanding it again: expand (x + 3)(x + 4), and if you get back the original expression, your answer is correct.

Check it with Python Use SymPy's factor.

```
from sympy import symbols, factor
x = symbols('x')
print(factor(6*x**2 + 9*x))      # common factor
print(factor(x**2 + 7*x + 12))   # factoring by formula
```

Output ▸

```
3*x*(2*x + 3)
(x + 3)*(x + 4)
```

expand and factor are inverse functions: applying expand to the result of factor returns the original. Whenever you are unsure about a hand calculation, they make a dependable checker.

1.6 Linear Inequalities: Solving Order Relations

Last comes inequalities. In Level 1A, Chapter 13, we solved equations — statements joined by the equals sign =. Here we solve statements joined by inequality signs: **inequalities**.

Term **Inequality** A statement that compares numbers or expressions using an inequality sign (<, >, ≤, ≥). The range of values of the letter that makes the statement true is called the solution of the inequality.

Let us confirm the four inequality signs. a < b means "a is less than b"; a > b means "a is greater than b"; a ≤ b means "a is less than or equal to b"; and a ≥ b means "a is greater than or equal to b".

Solving an inequality works almost exactly like solving an equation. In place of the properties of equality, we use the following properties of inequalities so that only x remains on the left-hand side.

Table 1.2 Properties of inequalities

Property	Direction of the sign
Adding or subtracting the same number on both sides	unchanged
Multiplying or dividing both sides by the same positive number	unchanged
Multiplying or dividing both sides by the same negative number...	**reverses!**

Multiplying by a negative number reverses the sign $2 < 3$ is a true inequality. Multiply both sides by −1 and you get −2 and −3 — but on the number line, −2 is to the right (larger). So −2 > −3. Multiplying (or dividing) by a negative number flips the order — this is the one and only difference from equations, and the most important.

Example 1.10

Solve the following linear inequalities and show the solutions on a number line.

(1) $3x - 5 < 7$ (2) $-2x + 3 \leq 9$

With pen and paper (1) Proceed just as with an equation. Move −5 to the right side: 3x < 12. Divide both sides by the positive number 3: x < 4. The inequality sign stays as it is.

$$3x - 5 < 7 \quad \rightarrow \quad 3x < 12 \quad \rightarrow \quad x < 4$$

(2) Move the 3: −2x ≤ 6. Now divide both sides by −2. Dividing by a negative number reverses the inequality sign.

$$-2x \leq 6 \quad \rightarrow \quad x \geq -3$$

The solution can be shown as a range on the number line. When drawing it, an endpoint that is not included is marked with an open circle ○, and an endpoint that is included with a filled circle ●.

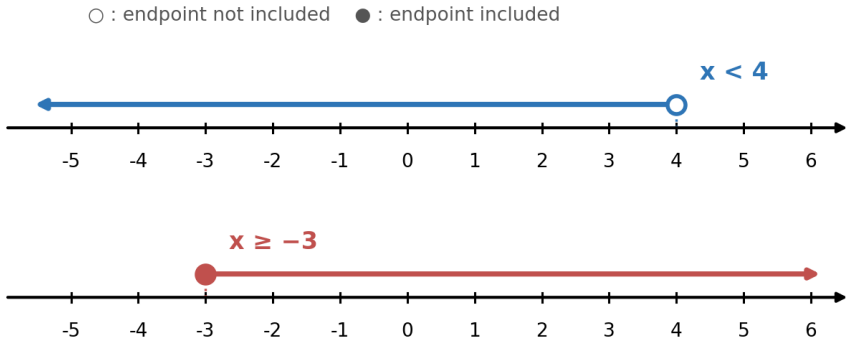


Figure 1.8 Showing solutions of inequalities on a number line: ○ for an excluded endpoint, ● for an included one.

As Figure 1.8 shows, $x < 4$ does not include 4, so it gets an open circle ○, while $x \geq -3$ includes -3, so it gets a filled circle ●. "The solution is not a single number but a range" — that is what makes inequalities special.

Check it with Python solve handles not only equations but inequalities too.

```
from sympy import symbols, solve
x = symbols('x')
print(solve(3*x - 5 < 7, x))   # inequalities work too
# <= means less than or equal to
print(solve(-2*x + 3 <= 9, x))
```

Output ▸

```
(-oo < x) & (x < 4)
(-3 <= x) & (x < oo)
```

In the output, oo is infinity (∞) and & means "and". The first line says "x is greater than $-\infty$ and less than 4", in other words $x < 4$. The second says "x is greater than or equal to -3", that is, $x \geq -3$. Note that in code, ≤ is written <= and ≥ is written >=.

Column The Secret of Hippasus: Discovering the Irrational

Around the fifth century BC in ancient Greece, there was a school of mathematicians known as the Pythagoreans. Their creed was "all is number," and they believed every quantity could be expressed as a ratio of

integers — that is, as a fraction, a rational number. But one of their members, Hippasus, discovered that the diagonal of a unit square, $\sqrt{2}$, cannot be expressed by any fraction at all. The discovery overturned the school's creed from its foundations, and legend has it that Hippasus was drowned at sea for revealing the secret to outsiders. True or not, the story shows what an earth-shaking event the discovery of irrational numbers was. The simple line sqrt(2) is connected to 2,500 years of mathematical history.

Chapter Summary

- Numbers expand from the natural numbers to the integers to the rationals; the rationals and irrationals together form the real numbers, which correspond to all points on the number line.
- A rational number can be written as a fraction (its decimal terminates or repeats). An irrational number cannot (its decimal never ends and never repeats).
- $\sqrt{a^2 b} = a\sqrt{b}$ simplifies what is under a radical. Radicals with the same number inside combine like like terms. A radical in a denominator can be removed by rationalizing.
- The absolute value |a| is the distance from 0 on the number line. When a < 0, remove the bars with |a| = −a, splitting into cases.
- All expansion rests on the distributive law. The most frequent patterns are the multiplication formulas (Table 1.1) — learn to use them fluently.
- Factoring is expansion in reverse: first pull out common factors, then read the formulas backwards. Every answer can be checked by expanding.
- Inequalities are solved by the same steps as equations, except that multiplying or dividing by a negative number reverses the inequality sign. Solutions are ranges on the number line.

Table 1.3 Python tools used in this chapter

Python tool	What it does
`Rational(3, 4) / Fraction(7, 8)`	exact fractions
`sqrt(2)`	radicals, kept in exact form
`.evalf()`	show an exact value as a decimal approximation
`radsimp(expr)`	rationalize a denominator
`abs(x)`	absolute value
`expand(expr)`	expand an expression
`factor(expr)`	factor an expression
`solve(inequality, x)`	solve an inequality

Practice Problems

Solve each problem with pen and paper first, then write Python code in an empty Colab cell to check your answer. When using SymPy, remember to start with from sympy import … .

Problem 1

Classify the following numbers as rational or irrational.

$$-6, \quad \frac{2}{7}, \quad \sqrt{5}, \quad 0.272727\ldots(repeating), \quad \pi$$

Problem 2

Write $\frac{3}{8}$ and $\frac{5}{6}$ as decimals, and say whether each is terminating or repeating.

Problem 3

Write $\sqrt{50}$ in the form $a\sqrt{b}$.

Problem 4

Compute $\sqrt{8} + \sqrt{32} - \sqrt{18}$.

Problem 5

Rationalize the denominator of $\frac{10}{\sqrt{5}}$. Also compute $(\sqrt{7} - \sqrt{2})(\sqrt{7} + \sqrt{2})$.

Problem 6

Find the values of $|-9|$, $|2 - 8|$, and $|\sqrt{2} - 2|$ (recall that $\sqrt{2} = 1.41\ldots$).

Problem 7

Expand $(x + 6)(x - 2)$ and $(3x - 1)^2$.

Problem 8

Expand $(2a + 5b)(2a - 5b)$.

Problem 9

Factor the following expressions.

(1) $4x^2 - 12x$ (2) $x^2 - x - 20$ (3) $x^2 - 16$

Problem 10

Solve the following linear inequalities.

(1) $5x + 2 > 17$ (2) $4 - 3x \geq -5$

Answers and Explanations

Answer 1

Rational: -6, $\frac{2}{7}$, 0.272727... (a repeating decimal can always be written as a fraction, here $\frac{27}{99} = \frac{3}{11}$). Irrational: $\sqrt{5}$, π.

```
from sympy import sqrt, Rational
print(Rational(2, 7))    # stays a fraction = rational
print(sqrt(5).evalf())   # never repeats = irrational
```

Output ▸

```
2/7
2.23606797749979
```

The decimal expansion of sqrt(5) goes on forever with no repeating block.

Answer 2

$\frac{3}{8} = 0.375$, terminating. $\frac{5}{6} = 0.8333\ldots$, repeating. The denominator $8 = 2\times2\times2$ has only the prime factor 2, so the decimal terminates; the denominator $6 = 2\times3$ contains a 3, so it repeats.

```
from fractions import Fraction
print(Fraction(3, 8))
print(3 / 8)
print(5 / 6)
```

```
Output ▸
3/8
0.375
0.8333333333333334
```

The final 4 in the display of 5 / 6 comes from rounding at the limit of the computer's digits (in truth the 3s continue forever).

Answer 3

Since $50 = 25 \times 2 = 5^2 \times 2$, we get $\sqrt{50} = 5\sqrt{2}$.

```
from sympy import sqrt
print(sqrt(50))
```

```
Output ▸
5*sqrt(2)
```

Answer 4

$\sqrt{8} = 2\sqrt{2}$, $\sqrt{32} = 4\sqrt{2}$, and $\sqrt{18} = 3\sqrt{2}$, so $2\sqrt{2} + 4\sqrt{2} - 3\sqrt{2} = 3\sqrt{2}$.

```
from sympy import sqrt
print(sqrt(8) + sqrt(32) - sqrt(18))
```

```
Output ▸
3*sqrt(2)
```

Once everything is in the form $a\sqrt{2}$, the terms combine like like terms.

Answer 5

Multiply numerator and denominator by $\sqrt{5}$: $\frac{10}{\sqrt{5}} = \frac{10\sqrt{5}}{5} = 2\sqrt{5}$. The second part is a sum times a difference: $(\sqrt{7} - \sqrt{2})(\sqrt{7} + \sqrt{2}) = 7 - 2 = 5$.

```
from sympy import sqrt, expand
print(10 / sqrt(5))
print(expand((sqrt(7) - sqrt(2)) * (sqrt(7) + sqrt(2))))
```

```
Output ▸
2*sqrt(5)
5
```

The sum-times-difference formula works for radical expressions too — watching the radicals vanish so cleanly is one of the small pleasures of algebra.

Answer 6

$|-9| = 9$. $|2-8| = |-6| = 6$. Since $\sqrt{2} = 1.41\ldots$ is less than 2, $\sqrt{2} - 2$ is negative, so $|\sqrt{2} - 2| = -(\sqrt{2} - 2) = 2 - \sqrt{2}$.

```
from sympy import sqrt, Abs
print(abs(-9))
print(abs(2 - 8))
print(Abs(sqrt(2) - 2))   # exact form: use SymPy's Abs
```

Output ▸

```
9
6
2 - sqrt(2)
```

SymPy's Abs recognizes that the inside is negative and rewrites it as −(inside) — the same case analysis we did by hand.

Answer 7

By the sum-and-product formula, $(x + 6)(x - 2) = x^2 + 4x - 12$. By the square of a difference, $(3x - 1)^2 = 9x^2 - 6x + 1$.

```
from sympy import symbols, expand
x = symbols('x')
print(expand((x + 6) * (x - 2)))
print(expand((3*x - 1)**2))
```

Output ▸

```
x**2 + 4*x - 12
9*x**2 - 6*x + 1
```

Do not forget that $(3x)^2 = 9x^2$: the coefficient gets squared too.

Answer 8

Sum times difference: $(2a + 5b)(2a - 5b) = (2a)^2 - (5b)^2 = 4a^2 - 25b^2$.

```
from sympy import symbols, expand
a, b = symbols('a b')   # declare two symbols at once
print(expand((2*a + 5*b) * (2*a - 5*b)))
```

Output ▸

```
4*a**2 - 25*b**2
```

Answer 9

(1) Pull out the common factor 4x: $4x^2 - 12x = 4x(x - 3)$. (2) Two numbers with sum −1 and product −20 are 4 and −5, so $x^2 - x - 20 = (x + 4)(x - 5)$. (3) Read sum-times-difference backwards: $x^2 - 16 = (x + 4)(x - 4)$.

```
from sympy import symbols, factor
x = symbols('x')
print(factor(4*x**2 - 12*x))
print(factor(x**2 - x - 20))
print(factor(x**2 - 16))
```

Output ▸

```
4*x*(x - 3)
(x - 5)*(x + 4)
(x - 4)*(x + 4)
```

For (2), SymPy writes the factors in the order (x - 5)*(x + 4), but multiplication can be done in any order, so this is the same answer as (x + 4)(x − 5).

Answer 10

(1) Move the 2: 5x > 15. Divide both sides by 5: $x > 3$. (2) Move the 4: −3x ≥ −9. Dividing both sides by −3 reverses the sign: $x \leq 3$.

```
from sympy import symbols, solve
x = symbols('x')
print(solve(5*x + 2 > 17, x))
print(solve(4 - 3*x >= -5, x))
```

Output ▸

```
(3 < x) & (x < oo)
(-oo < x) & (x <= 3)
```

The first line reads "3 < x", that is, x > 3; the second reads "x ≤ 3". SymPy's answer confirms that the inequality sign reversed in (2).

Chapter 2
Thinking with Logic: Sets and Propositions

Sets

Propositions

Conditions

Converse, inverse, and contrapositive

Necessary and sufficient conditions

The idea of proof by contradiction

Chapter 2 Thinking with Logic: Sets and Propositions

What You Will Learn

In Chapter 1 we organized the world of numbers and expressions. In this chapter we study two tools that support the very way mathematics thinks: **sets** and **propositions**. A set handles "collections of things," and a proposition handles "statements that are definitely true or false," both without any ambiguity.

At first glance this chapter has little calculation in it, but the words you learn here — "implies," "converse," "contrapositive," "necessary and sufficient" — form the foundation of proofs and explanations in every later chapter of Level 2. And here is the happy part: Python comes with a set type for sets and a bool type for truth values, built right in. The sets and logic of mathematics can be written in Python almost exactly as they are.

The way we work stays the same: first understand the ideas with pen and paper, then run the same ideas in Python to check them.

By the end of this chapter, you will be able to:

- explain sets, elements, and subsets with symbols and diagrams
- find the intersection $A \cap B$, the union $A \cup B$, and the complement, using Venn diagrams and Python sets
- decide whether a proposition is true or false, and find the truth set of a condition
- form the converse, inverse, and contrapositive of a proposition, and explain why the contrapositive always matches the original
- decide necessary and sufficient conditions using the inclusion of truth sets
- explain the flow of proof by contradiction and follow the proof that $\sqrt{2}$ is irrational
- check sets and logic with Python's set, in, all, and more

2.1 The Basics of Sets: Turning Collections into Mathematics

A collection of things in which it is completely clear what belongs and what does not — like "the divisors of 6" or "the even numbers up to 10" — is called a **set**. Each individual thing in a set is called an **element** of the set.

> **Term** **Set and element** A collection of things with a clearly decided range is called a set. Each thing in a set is an element of that set. A collection like "the big numbers," where membership depends on who you ask, is not a set.

A set is written by listing its elements inside curly braces { }. For example, the set A of divisors of 6 is written

$$A = \{1, 2, 3, 6\}$$

Membership is written with the symbol ∈. Since 2 is an element of A, we write $2 \in A$; since 5 is not, we write $5 \notin A$ (a slash through a symbol means its negation).

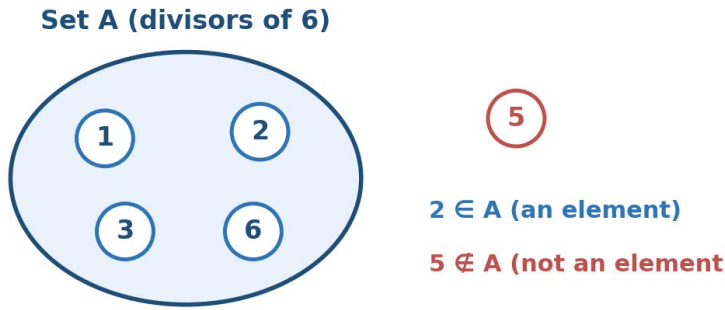


Figure 2.1 A set and its elements. The numbers inside A are its elements: $2 \in A$, $5 \notin A$.

As Figure 2.1 shows, a set becomes easy to picture as "elements inside a bag." This kind of picture is called a Venn diagram, and it will appear again and again as the star of this chapter.

Example 2.1

For the set of divisors of 6, A = {1, 2, 3, 6}, confirm that $2 \in A$ and $5 \notin A$ are correct, and state the number of elements of A.

With pen and paper The divisors of 6 are the natural numbers that divide 6 exactly. Since $6 \div 1$, $6 \div 2$, $6 \div 3$, and $6 \div 6$ all divide exactly, the divisors are 1, 2, 3, 6 — four of them (we studied divisors in Level 1A, Chapter 11). 2 is in this list, so it is an element; 5 is not, so it is not an

element.

Check it with Python Python has a **set type** for sets. Just as in mathematics, you build one by enclosing the elements in curly braces { }. Membership is tested with the in operator, and the answer comes back as True or False. The function len counts the elements.

```
A = {1, 2, 3, 6}        # the set of divisors of 6
print(2 in A)            # is 2 an element of A?
print(5 in A)            # is 5 an element of A?
print(len(A))            # number of elements
```

Output ▸

```
True
False
4
```

The mathematical statement 2 ∈ A became, almost symbol for symbol, the Python expression 2 in A. True and False are the bool type (truth values) from the Python Basics part of Level 1B. In this chapter, True and False take center stage.

Subsets: A Set Inside a Set

When every element of a set A is also in a set B, we say A is a **subset** of B and write A ⊂ B. For example, with A = {1, 2} and B = {1, 2, 3, 6}, both elements 1 and 2 of A are in B, so A ⊂ B.

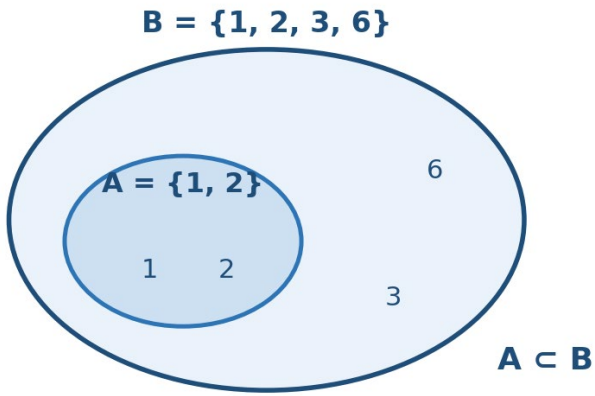


Figure 2.2 A subset. When every element of A is in B, we write A ⊂ B.

Term Subset When every element of set A is an element of set B, A is called a subset of B, written $A \subset B$. The set with no elements at all is called the empty set, written $\emptyset$. The empty set is considered a subset of every set.

Example 2.2

For $A = \{1, 2\}$ and $B = \{1, 2, 3, 6\}$, decide whether $A \subset B$, and whether $B \subset A$.

With pen and paper The elements 1 and 2 of A are both in B, so $A \subset B$ is correct. On the other hand, the elements 3 and 6 of B are not in A, so $B \subset A$ is not correct.

Check it with Python For Python sets, the subset test is written with the same <= used for comparing numbers: "A <= B" means "A is a subset of B."

```
A = {1, 2}
B = {1, 2, 3, 6}
print(A <= B)            # is A a subset of B?
print(B <= A)            # is B a subset of A?
```

Output ▸

```
True
False
```

The <= that compares the sizes of numbers is reused for sets to test containment. The image "smaller set $\subset$ bigger set" lines up nicely with the symbol — a clever reuse of notation.

2.2 Operations on Sets: Intersection, Union, and Complement

Just as numbers have addition and multiplication, sets have "operations" too. Let us look at the standard ways of making a new set from two given sets.

Term Intersection and union The set of all elements that belong to both of two sets A and B is called their intersection, written $A \cap B$. The set of all elements that belong to at least one of them is called their union, written $A \cup B$.

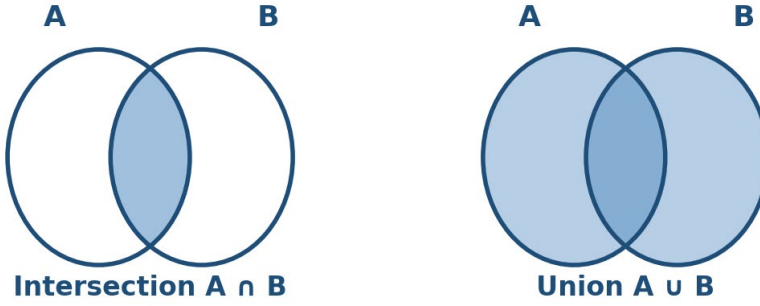


Figure 2.3 Intersection and union. ∩ is "the part in both"; ∪ is "everything combined."

The Venn diagrams in Figure 2.3 make it clear at a glance. ∩ (cap) is the overlapping part of the two circles, and ∪ (cup) is the two circles combined. Remember it as "∩ means both, ∪ means at least one." The union ∪ contains everything that is in A only, in B only, or in both.

Example 2.3

For the set of divisors of 12, A = {1, 2, 3, 4, 6, 12}, and the set of divisors of 18, B = {1, 2, 3, 6, 9, 18}, find A ∩ B and A ∪ B.

With pen and paper For the intersection, pick out the numbers that are in both sets: 1, 2, 3, and 6 appear in both, so A ∩ B = {1, 2, 3, 6}. For the union, gather every element of both (counting repeats only once): A ∪ B = {1, 2, 3, 4, 6, 9, 12, 18}.

By the way, A ∩ B = {1, 2, 3, 6} is exactly the set of common divisors of 12 and 18. The greatest common divisor, 6, appears as the largest number in it — a connection back to Level 1A, Chapter 11.

Check it with Python With Python sets, the intersection is computed with & and the union with |. Because the display order of a set is not fixed, we pass the result to sorted, which returns a list in increasing order (that is why the output appears in square brackets [ ]).

```
A = {1, 2, 3, 4, 6, 12}    # divisors of 12
B = {1, 2, 3, 6, 9, 18}     # divisors of 18
print(sorted(A & B))        # intersection, sorted
print(sorted(A | B))        # union
```

Output ▸

```
[1, 2, 3, 6]
[1, 2, 3, 4, 6, 9, 12, 18]
```

The mathematical ∩ corresponds to & and ∪ to |. The results match what we picked out by hand.

The Complement: The Set of "Everything Else"

The set of everything under consideration is called the **universal set**, written U. Within U, the set of all elements that are not in A is called the **complement** of A, written $\bar{A}$ (an overline is drawn above the A).

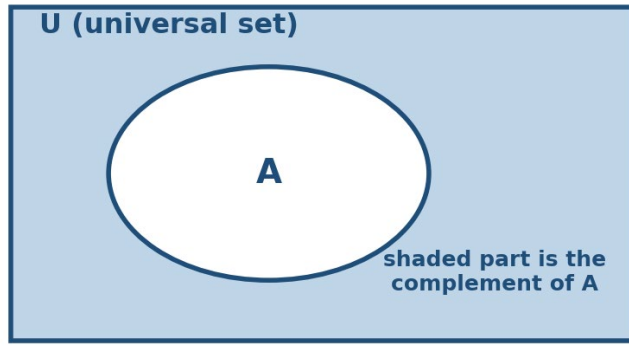


Figure 2.4 The complement. Inside the universal set U, the part outside A is the complement of A.

> **Term** **Universal set and complement** The set of everything under consideration is the universal set U. The set of elements of U that are not in A is the complement of A, written $\bar{A}$.

Example 2.4

Let the universal set be U = {1, 2, ..., 10} (the natural numbers from 1 to 10), and let A be the set of even numbers up to 10. Find the complement $\bar{A}$.

With pen and paper A = {2, 4, 6, 8, 10}. Removing the elements of A from U leaves only the odd numbers: $\bar{A}$ = {1, 3, 5, 7, 9}. "Not even" is "odd" — the complement corresponds to the negation of a condition.

Check it with Python The complement is found by set subtraction, U - A. The expression set(range(1, 11)) builds the set of integers from 1 to 10 (range(1, 11) means from 1 up to, but not including, 11).

```
U = set(range(1, 11))       # universal set 1 to 10
A = {2, 4, 6, 8, 10}        # even numbers up to 10
print(sorted(U - A))        # complement of A
```

Output ▸
```
[1, 3, 5, 7, 9]
```

The meaning of the complement — "what remains when A is taken away from the whole" — becomes, quite literally, a subtraction. Table 2.1 summarizes the correspondence between the symbols and Python so far.

Table 2.1 Set symbols and their Python counterparts

Mathematical symbol	In Python
$x \in A$	`x in A`
$A \subset B$	`A <= B`
$A \cap B$	`A & B`
$A \cup B$	`A | B`
complement $\bar{A}$	`U - A`
empty set $\emptyset$	`set()`

2.3 Propositions and Conditions: Statements with a Definite Truth Value

From here on, we make "statements" themselves an object of mathematics. A statement or formula that is definitely either true or false — one or the other, with no ambiguity — is called a **proposition**.

> **Term Proposition** A statement or formula that is definitely true (correct) or false (incorrect). True and false together are called its truth value. "5 > 3" is a true proposition and "10 < 2" is a false one. "Math is fun" depends on the person, so it is not a proposition.

The key test for a proposition is: **would everyone who judges it reach the same answer?** "Mt. Fuji is tall" depends on the person, so it is not a proposition. "Mt. Fuji is taller than 3000 m" is definitely true, so it is one.

Example 2.5

Decide whether each statement is true or false.

(1) 5 is greater than 3 (2) 10 is less than 2 (3) 2 + 3 equals 5

With pen and paper (1) is correct, so it is true. (2) is incorrect, so it is false. (3) 2 + 3 = 5, so it is true.

Check it with Python Recap (Python Basics part of Level 1B): the comparison operators >, <, and == return their result as the bool type, True or False. In other words, a Python comparison expression is itself a proposition.

```
print(5 > 3)          # "5 is greater than 3"
print(10 < 2)         # "10 is less than 2"
print(2 + 3 == 5)     # "2 + 3 equals 5"
```

Output ▸

```
True
False
True
```

True for true, False for false. Mathematical truth values and Python's bool type correspond perfectly — even the names match. Remember that testing equality uses == rather than = (a single = is assignment to a variable).

Conditions and Truth Sets

A statement like "x > 6," which contains a letter x and becomes true or false once a value of x is chosen, is called a **condition** on x. The condition x > 6 is true for x = 8 and false for x = 2. Its truth value depends on x — that is what distinguishes a condition from a proposition.

Term **Condition and truth set** A statement or formula containing a letter, whose truth is decided once the letter's value is chosen, is called a condition. The set of all values that make a condition true is called its truth set.

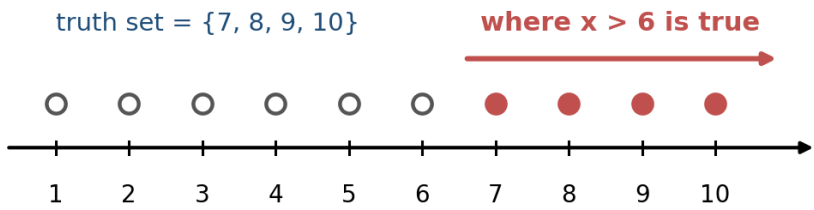


Figure 2.5 The truth set of the condition x > 6. Among 1 to 10, the condition is true for {7, 8, 9, 10}.

As Figure 2.5 shows, a condition can be seen on the number line as "the region where it is true." Conditions and sets are connected through truth sets — and this connection will do real work in the logic of the coming

sections.

Example 2.6

Let x be a natural number up to 10. Find the truth set of the condition "x > 6" and of the condition "x is even."

With pen and paper Test the numbers from 1 to 10 one by one. x > 6 is true for 7, 8, 9, 10. "x is even" is true for 2, 4, 6, 8, 10.

Check it with Python "The set of values satisfying a condition" can be built in one line with a **set comprehension**: writing {x for x in range(1, 11) if condition} collects exactly the x in the range for which the condition is True.

```
# truth set of the condition x > 6
S = {x for x in range(1, 11) if x > 6}
print(sorted(S))
# truth set of the condition "x is even"
T = {x for x in range(1, 11) if x % 2 == 0}
print(sorted(T))
```

Output ▸

```
[7, 8, 9, 10]
[2, 4, 6, 8, 10]
```

x % 2 == 0 means "the remainder when x is divided by 2 is 0," that is, "x is even" (% is the remainder operator). A set comprehension is the "condition → truth set" conversion, written directly as code.

2.4 Converse, Inverse, and Contrapositive: Turning a Proposition Around

A proposition of the form "if p then q," joining two conditions p and q, is written $p \Rightarrow q$. For example, "if $x = 3$ then $x^2 = 9$" has this form. We call p the hypothesis and q the conclusion.

From one proposition $p \Rightarrow q$, three related propositions can be made by swapping the hypothesis and conclusion, or by negating them (negation is written with the symbol ¬).

Table 2.2 Converse, inverse, and contrapositive

Name	Form
original proposition	$p \Rightarrow q$
converse (swap hypothesis and conclusion)	$q \Rightarrow p$
inverse (negate both)	$\neg p \Rightarrow \neg q$
contrapositive (swap and negate both)	$\neg q \Rightarrow \neg p$

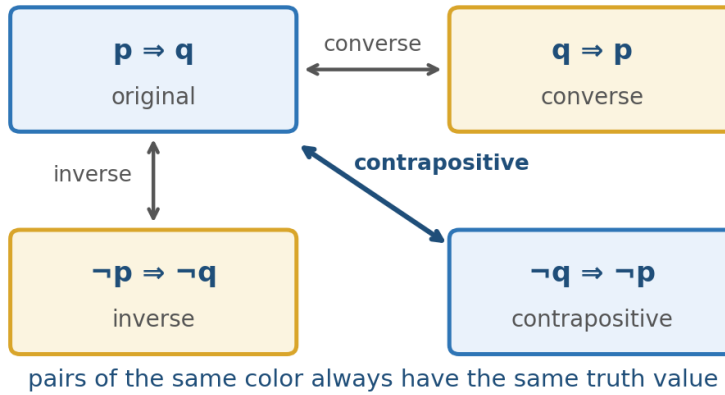


Figure 2.6 How the four propositions relate. Pairs of the same color (original and contrapositive; converse and inverse) always have the same truth value.

Figure 2.6 shows how the four propositions relate. One property here is essential: **the original proposition and its contrapositive always have the same truth value** (likewise, the converse and the inverse match each other). On the other hand, even when the original is true, the converse need not be.

Why does the contrapositive always match? Truth sets explain it. Saying p ⇒ q is true means that the truth set P of p fits entirely inside the truth set Q of q ($P \subset Q$). But then every element outside Q is also outside P — and that is exactly ¬q ⇒ ¬p, the contrapositive.

Example 2.7

Form the converse, inverse, and contrapositive of the proposition "if x = 3 then $x^2 = 9$," and investigate whether the converse is true.

With pen and paper Apply the patterns of Table 2.2. The converse is "if $x^2 = 9$ then x = 3." The inverse is "if $x \neq 3$ then $x^2 \neq 9$." The contrapositive is "if $x^2 \neq 9$ then $x \neq 3$." The original proposition is true — but what about

the converse? 3 is not the only number whose square is 9: $x = -3$ also squares to 9. So the converse is false. An example like this, showing that a proposition is false, is called a **counterexample**. Finding just one counterexample is enough.

Check it with Python Hunting for counterexamples is what computers do best. We check every integer from −10 to 10, looking for a number with $x^2 = 9$ but $x \neq 3$.

```
for x in range(-10, 11):        # check -10 through 10
    # any x with x**2 == 9 but x != 3?
    if x**2 == 9 and x != 3:
        print(x)                # print it if found
```

Output ▸

```
-3
```

The counterexample $x = -3$ was found. and means "and (both hold)," and != means "not equal." Combining conditions to hunt for counterexamples is a pattern we will use again and again.

Example 2.8

For the true proposition "if x is a multiple of 4, then x is even," confirm over the integers 1 to 100 that the contrapositive "if x is odd, then x is not a multiple of 4" is also true.

With pen and paper Original: a multiple of 4 can be written $4 \times (\text{integer}) = 2 \times (2 \times \text{integer})$, so it is always even. True. Contrapositive: an odd number is not divisible by 2, so it certainly is not divisible by 4. True. The truth values do match.

Check it with Python "$p \Rightarrow q$ is true" means there is no counterexample — no case of "p holds but q does not." Our plan: check 1 through 100, and flip a variable to False if a counterexample turns up.

```
ok1 = True
for x in range(1, 101):
    if x % 4 == 0 and not (x % 2 == 0):   # counterexample
        ok1 = False
ok2 = True
for x in range(1, 101):
    if x % 2 != 0 and not (x % 4 != 0):   # contrapositive
        ok2 = False
print(ok1)
print(ok2)
```

Output ▸

```
True
True
```

not expresses negation (¬). No counterexample for either one — both are confirmed true. When you meet a proposition later that is hard to prove directly, the standard move will be to ask, "is the contrapositive easier to show?"

2.5 Necessary and Sufficient Conditions

When the proposition p ⇒ q is true, we give names to the relationship between the two conditions: p is a **sufficient condition** for q, and q is a **necessary condition** for p.

Term **Sufficient and necessary conditions** When p ⇒ q is true, p is called a sufficient condition for q, and q a necessary condition for p. When both p ⇒ q and q ⇒ p are true, p is called a necessary and sufficient condition for q.

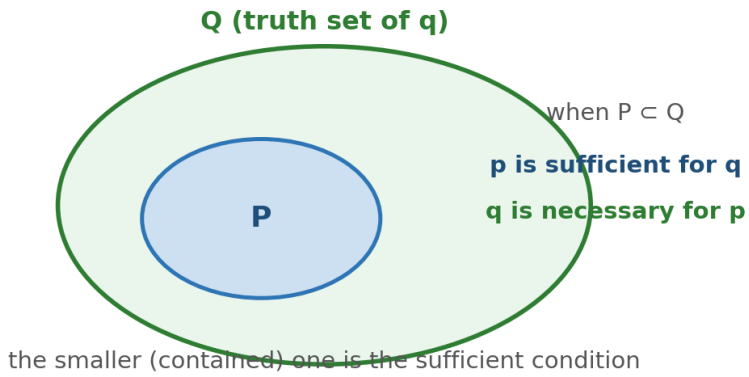


Figure 2.7 Necessary and sufficient conditions as a Venn diagram. When $P \subset Q$, p is sufficient and q is necessary.

The quickest way to grasp these words is the Venn diagram of truth sets (Figure 2.7). When $P \subset Q$: being in P is enough to be in Q, so p is sufficient. Being in Q is the minimum required for being in P, so q is necessary. Remember: **the smaller (contained) one is the sufficient condition**.

Example 2.9

For the conditions p: $x = 2$ and q: $x^2 = 4$, what kind of condition is p for q?

With pen and paper $p \Rightarrow q$: if $x = 2$ then $x^2 = 4$. True. $q \Rightarrow p$: if $x^2 = 4$ then $x = 2$. The counterexample $x = -2$ makes this false. So p is a sufficient condition for q, but not a necessary one. In truth sets, $P = \{2\}$ and $Q = \{2, -2\}$, and indeed $P \subset Q$ — the smaller one, p, is the sufficient condition.

Check it with Python The function all tests "are they all True?" in one stroke. Writing all(condition for x in xs if hypothesis) checks whether the condition is True for every x satisfying the hypothesis — in other words, it decides an "if–then."

```
xs = range(-10, 11)
print(all(x**2 == 4 for x in xs if x == 2))   # p implies q
print(all(x == 2 for x in xs if x**2 == 4))   # q implies p
```

Output ▸

```
True
False
```

The first line is true and the second false. Only $p \Rightarrow q$ holds, so p is judged

sufficient (and not necessary).

The arrow as a memory aid When $p \Rightarrow q$ is true, the tail of the arrow, p, is the sufficient condition, and the head, q, is the necessary one. Picture it as: being able to shoot the arrow is sufficient; the side that must catch it is necessary.

2.6 The Idea of Proof by Contradiction

The finale of this chapter is a powerful proof tool: **proof by contradiction**. Even when a claim is hard to prove directly, we can conclude it is true by showing that "assuming it is false leads to something absurd."

Term **Proof by contradiction** A method of proof in which we assume the negation of the desired conclusion and derive a contradiction, thereby showing the original conclusion is true.

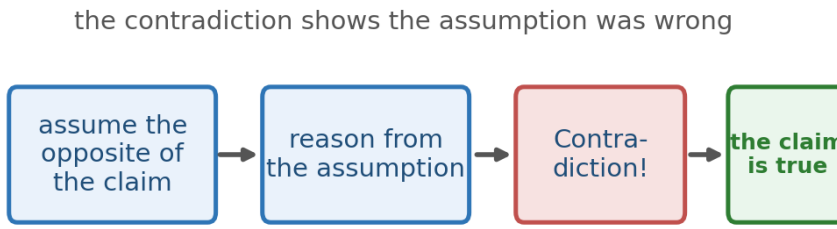


Figure 2.8 The flow of proof by contradiction: assume the opposite, reach a contradiction, conclude the claim is true.

Figure 2.8 shows the flow. A contradiction appearing means the assumption we placed at the start — the negation of the conclusion — was wrong; in other words, the original conclusion was right after all. The discovery of Hippasus introduced in the Chapter 1 column — that $\sqrt{2}$ is irrational — is proved by exactly this method.

Example 2.10

Follow the proof by contradiction that $\sqrt{2}$ is irrational, and check its key equation with Python.

With pen and paper Negate the conclusion "$\sqrt{2}$ is irrational" and assume instead that $\sqrt{2}$ is rational. Then $\sqrt{2}$ could be written as a fraction in lowest terms, $\sqrt{2} = \frac{p}{q}$. Squaring both sides and clearing the denominator gives

$$p^2 = 2q^2$$

The right side is 2 × (integer), so p^2 is even.

Why does p^2 being even force p itself to be even? If p were odd, we could write p = 2m + 1. Then $p^2 = (2m + 1)^2 = 4m^2 + 4m + 1 = 2(2m^2 + 2m) + 1$, which is odd. So if p^2 is even, p cannot be odd — it must be even.

So we write p = 2k and substitute, getting $4k^2 = 2q^2$, that is, $q^2 = 2k^2$ — and by the same reasoning, q is even too. But if p and q are both even, the fraction could be reduced by 2, contradicting our premise that it was in lowest terms. A contradiction has appeared, so the assumption was wrong: $\sqrt{2}$ is irrational.

Check it with Python The key to the proof was that "no natural numbers p, q satisfy $p^2 = 2q^2$." Let us check denominators q up to 1000 exhaustively and feel for ourselves that not a single pair turns up. Note that $\sqrt{2}$ is less than 2, so if $\frac{p}{q} = \sqrt{2}$ then p < 2q. It is therefore enough to check numerators p from 1 up to (but not including) 2q.

```
found = False
for q in range(1, 1001):           # q from 1 to 1000
    for p in range(1, 2 * q):      # numerators (p < 2q)
        if p * p == 2 * q * q:     # p**2 == 2*q*q ?
            found = True
print(found)             # False if none is found
```

Output ▸

```
False
```

Roughly a million combinations were checked, and not one pair with $p^2 = 2q^2$ was found. But be careful: this code only confirmed "none exists up to 1000." The right to say "none exists, no matter how large" belongs to the logic of proof by contradiction. Exhaustive search by computer, and proof by mathematics — this example shows clearly what each is best at.

Column The Logic of 0 and 1: George Boole and the Computer

True and false; "and," "or," and "not" — the machinery for calculating with these as if they were numbers was created by the 19th-century English mathematician George Boole. In 1847 he showed that if true is written as 1 and false as 0, logic can be computed like algebra. At the time it was purely theoretical research, but about 90 years later this "Boolean algebra" turned out to fit the design of electric circuits perfectly, and it became the foundation of the computer. Inside a modern computer, countless AND, OR, and NOT circuits compute exactly the logic of this chapter, billions of times every second. The name of Python's bool type (True / False) is a tribute to Boole. A single line of code using and, or, or not connects to 170 years of logic.

Chapter Summary

- A set is a collection in which membership is completely decided. Membership is written $\in$, and subsets are written $\subset$.
- The intersection $A \cap B$ is the set of elements in both sets; the union $A \cup B$ is the set of elements in at least one of them. The complement is "everything else" within the universal set.
- A proposition is a statement whose truth value is definite. A condition becomes true or false depending on the value of its letter, and the set of values making it true is its truth set.
- From $p \Rightarrow q$ we can form the converse $q \Rightarrow p$, the inverse $\neg p \Rightarrow \neg q$, and the contrapositive $\neg q \Rightarrow \neg p$. The original and the contrapositive always have the same truth value.
- That a proposition is false can be proved by exhibiting a single counterexample.
- When $p \Rightarrow q$ is true, p is sufficient and q is necessary. In truth sets, the smaller (contained) one is the sufficient condition.
- Proof by contradiction assumes the negation of the conclusion and derives a contradiction. The proof that $\sqrt{2}$ is irrational is the classic example.

Table 2.3 Python tools used in this chapter

Python tool	What it does
`{1, 2, 3} / set()`	build a set (set() is the empty set)
`x in A`	membership test (∈)
`A <= B`	subset test (⊂)
`A & B / A | B / U - A`	intersection, union, complement
`{x for x in ... if cond}`	build a truth set (set comprehension)
`and / or / not`	logical and, or, negation
`all(...)`	"all True?" = testing an if–then
`sorted(A)`	display a set as a sorted list

Practice Problems

Solve each problem with pen and paper first, then write Python code in an empty Colab cell to check your answer. Represent sets with the set type and truth values with the bool type.

Problem 1

Write the set A of divisors of 12 in curly-brace form, and give the truth values of 4 ∈ A and 8 ∈ A.

Problem 2

Decide whether {2, 4} is a subset of {1, 2, 3, 4, 6, 12}.

Problem 3

Let A be the set of even numbers up to 10 and B the set of multiples of 3 up to 10. Find A ∩ B and A ∪ B.

Problem 4

Within the universal set U = {1, 2, …, 10}, find the complement of the set A of multiples of 3.

Problem 5

Decide whether each statement is true or false. (1) 7 is a prime number (2) 9 is a prime number (3) 3 + 4 is greater than 6

Problem 6

Let x be a natural number up to 20. Find the truth set of the condition "x is a multiple of 4."

Problem 7

Form the converse, inverse, and contrapositive of the proposition "if x = 5 then $x^2 = 25$," and find a counterexample to the converse.

Problem 8

Form the contrapositive of the true proposition "if x is a multiple of 6, then x is a multiple of 3," and confirm over the integers 1 to 100 that it has no counterexample.

Problem 9

For the conditions p: x is a multiple of 4 and q: x is a multiple of 2 (x a natural number up to 100), what kind of condition is p for q?

Problem 10

$\sqrt{3}$ is also irrational. Check the key fact of its proof — that no natural numbers p, q satisfy $p^2 = 3q^2$ — for denominators up to 1000.

Answers and Explanations

Answer 1

The divisors of 12 are 1, 2, 3, 4, 6, 12, so A = {1, 2, 3, 4, 6, 12}. $4 \in A$ is true; 8 does not divide 12, so $8 \notin A$ (and $8 \in A$ is false).

```
A = {1, 2, 3, 4, 6, 12}
print(4 in A)
print(8 in A)
```

Output ▸

```
True
False
```

Answer 2

The elements 2 and 4 of {2, 4} are both in {1, 2, 3, 4, 6, 12}, so it is a subset.

```
print({2, 4} <= {1, 2, 3, 4, 6, 12})
```

Output ▸

```
True
```

Answer 3

A = {2, 4, 6, 8, 10} and B = {3, 6, 9}. The only number in both is 6, so $A \cap B = \{6\}$. Combining them, $A \cup B = \{2, 3, 4, 6, 8, 9, 10\}$.

```
A = {2, 4, 6, 8, 10}
B = {3, 6, 9}
print(sorted(A & B))
print(sorted(A | B))
```

Output ▸
```
[6]
[2, 3, 4, 6, 8, 9, 10]
```

6 is "even and a multiple of 3" — that is, a multiple of 6. You can see the intersection acting as the word "and."

Answer 4

A = {3, 6, 9}. Removing these from U leaves {1, 2, 4, 5, 7, 8, 10}.

```
U = set(range(1, 11))
A = {3, 6, 9}
print(sorted(U - A))
```

Output ▸
```
[1, 2, 4, 5, 7, 8, 10]
```

Answer 5

(1) The only divisors of 7 are 1 and 7, so it is prime. True. (2) 9 = 3 × 3, so it is not prime. False. (3) 3 + 4 = 7 > 6. True.

```
print(7 in {2, 3, 5, 7})     # is 7 prime?
print(9 in {2, 3, 5, 7})     # is 9 prime?
print(3 + 4 > 6)
```

Output ▸
```
True
False
True
```

We turned the primality question into a set question: "is it in the set of primes up to 10?" — sets and propositions working together.

Answer 6

The multiples of 4 up to 20: {4, 8, 12, 16, 20}.

```
S = {x for x in range(1, 21) if x % 4 == 0}
print(sorted(S))
```

Output ▸
```
[4, 8, 12, 16, 20]
```

Answer 7

Converse: "if $x^2 = 25$ then $x = 5$." Inverse: "if $x \neq 5$ then $x^2 \neq 25$." Contrapositive: "if $x^2 \neq 25$ then $x \neq 5$." The counterexample to the converse is $x = -5$ (its square is 25, yet it is not 5).

```
for x in range(-10, 11):
    if x**2 == 25 and x != 5:
        print(x)
```

Output ▸
```
-5
```

Answer 8

The contrapositive is "if x is not a multiple of 3, then x is not a multiple of 6." A number not divisible by 3 cannot be divisible by 6, so it is true.

```
ok = True
for x in range(1, 101):
    if x % 3 != 0 and not (x % 6 != 0):   # contrapositive
        ok = False
print(ok)
```

Output ▸
```
True
```

No counterexample appeared and ok stayed True — the contrapositive is confirmed true as well.

Answer 9

$p \Rightarrow q$: a multiple of 4 can be written $4k = 2 \times 2k$, so it is a multiple of 2. True. $q \Rightarrow p$: a multiple of 2 is not always a multiple of 4 (counterexample $x = 2$). False. So p is a sufficient condition for q, but not a necessary one.

```
xs = range(1, 101)
# p implies q
print(all(x % 2 == 0 for x in xs if x % 4 == 0))
# q implies p
print(all(x % 4 == 0 for x in xs if x % 2 == 0))
```

Output ▸
```
True
False
```

Answer 10

Just change the 2 of Example 2.10 to a 3. No pair with $p^2 = 3q^2$ turns up, and you can feel that $\sqrt{3}$ cannot be written as a fraction either (the proof itself runs by the same contradiction argument as the example: p and q would both be multiples of 3 — a contradiction).

```
found = False
for q in range(1, 1001):
    for p in range(1, 2 * q):
        if p * p == 3 * q * q:
            found = True
print(found)
```

Output ▸
```
False
```

Chapter 3
Building a Solid Foundation for Functions: Quadratic Functions

The idea of a function

Graphs of quadratic functions

Completing the square

Maximum and minimum values

The link to quadratic equations

Quadratic inequalities

Chapter 3 Building a Solid Foundation for Functions: Quadratic Functions

What You Will Learn

Chapter 1 organized numbers and expressions; Chapter 2 organized logic. From this chapter on, we enter the world of one of the main characters of Level 2 mathematics: the **function**. In Level 1B you studied the linear function $y = ax + b$ and the function $y = ax^2$, where y is proportional to the square of x. This chapter looks at them again from one step higher and makes you fully comfortable with the **quadratic function** $y = ax^2 + bx + c$.

The key technique is **completing the square**. Just by changing the form of the expression, you can read off the vertex of the graph, find maximum and minimum values, and even solve quadratic equations and quadratic inequalities with a single graph. Expressions, graphs, equations, and inequalities all join into one connected line — that is the view from this chapter.

The way we work stays the same: first understand the ideas with pen and paper, then run the same ideas in Python to check them. For drawing graphs, matplotlib — a star of Level 1B — makes its return.

By the end of this chapter, you will be able to:

- explain the meaning of the symbol f(x) and compute the value of a function
- read off the vertex and the axis of $y = a(x - p)^2 + q$ and draw its graph
- rewrite $y = ax^2 + bx + c$, by completing the square, into a form that shows the vertex
- find the maximum and minimum values of a quadratic function on a given domain
- use the discriminant D to count the points the graph shares with the x-axis
- solve quadratic inequalities using graphs
- check calculations and graphs with Python function definitions, SymPy, and matplotlib

3.1 The Idea of a Function: Naming the Rule of Correspondence

Recap (Level 1B, Chapters 4 and 6) When choosing a value of x determines exactly one value of y, we say y is a function of x. The graph of the linear function y = ax + b is a straight line, and the graph of $y = ax^2$, where y is proportional to the square of x, is a parabola.

In Level 2 we give functions names. We write a function with the symbol f, and the value determined by x as **f(x)** (read "f of x"). For example, writing

$$f(x) = x^2 - 2x + 3$$

means "f is the function that takes x and returns $x^2 - 2x + 3$." The value at x = 2 is found by substituting 2 for x: $f(2) = 4 - 4 + 3 = 3$. We call this "the value of the function at x = 2."

Term Function and f(x) When choosing a value of x determines exactly one value of y, y is called a function of x. The value of a function f at x is written f(x). The statement y = f(x) means "y is determined from x by the function f."

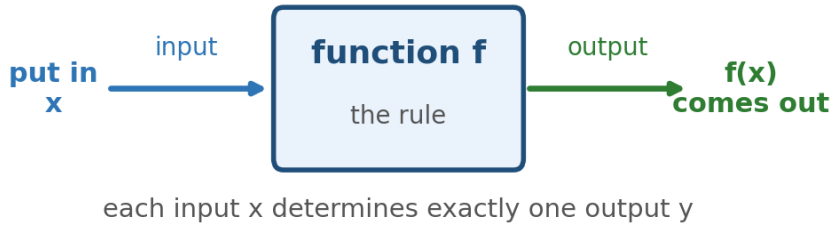


Figure 3.1 A function is a "rule of correspondence": a box that takes x in and puts exactly one f(x) out.

As Figure 3.1 shows, a function becomes easy to picture as "a box that receives an input x and returns an output f(x) according to a rule." What matters is that one input determines **exactly one** output.

Example 3.1

For the function $f(x) = x^2 - 2x + 3$, find f(2) and f(−1).

With pen and paper Substitute the values for x. $f(2) = 2^2 - 2 \times 2 + 3 =$

$4 - 4 + 3 = 3$. $f(-1) = (-1)^2 - 2 \times (-1) + 3 = 1 + 2 + 3 = 6$. When substituting a negative number, the trick is to keep the parentheses: $(-1)^2 = 1$ (just as in Level 1A, Chapter 10).

Check it with Python Recap (Python Basics part of Level 1B): with def you can define your own function. The mathematical f(x) can be written in almost exactly the same shape as a Python function definition.

```
def f(x):                    # define the function f
    return x**2 - 2*x + 3    # the rule
print(f(2))                  # value at x = 2
print(f(-1))                 # value at x = -1
```

Output ▸

```
3
6
```

Both match the hand calculations. The mathematical "f(x) = rule" and the Python "def f(x): return rule" correspond idea for idea. It is no coincidence that the word function is used in both mathematics and programming.

3.2 The Graph of a Quadratic Function: Mastering the Parabola

A function in which y is expressed by a quadratic (second-degree) expression in x,

$y = ax^2 + bx + c \quad (a \neq 0)$,

is called a **quadratic function**. Its graph has the same **parabola** shape as $y = ax^2$ from Level 1B, Chapter 6. First, let us recall the role of the coefficient a.

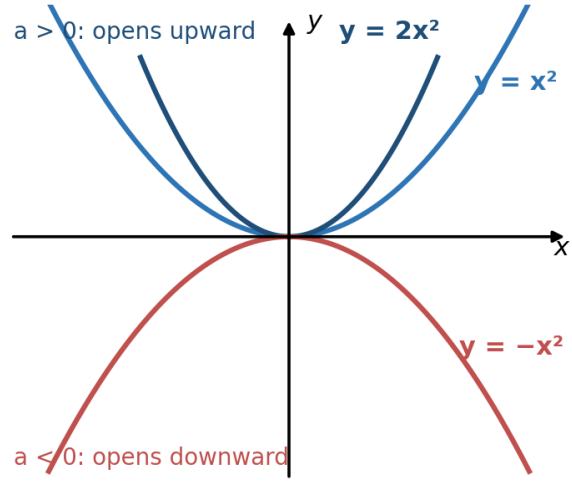


Figure 3.2 The role of the coefficient a. If a > 0 the parabola opens upward; if a < 0 it opens downward. The larger |a| is, the narrower the parabola.

As Figure 3.2 shows, when a > 0 the graph opens upward (a valley shape), and when a < 0 it opens downward (a mountain shape). Also, the larger |a| is, the narrower the opening.

Example 3.2

Find the quadratic function whose graph has the form $y = ax^2$ and passes through the point (2, 12).

With pen and paper Passing through (2, 12) means $y = 12$ when $x = 2$. Substituting, $12 = a \times 2^2 = 4a$. Dividing both sides by 4 gives $a = 3$. So $y = 3x^2$.

Check it with Python We solve the equation $4a = 12$ with SymPy's solve. Recall that Eq(left, right) expresses an equation (Level 1B, Chapter 3).

```
from sympy import symbols, Eq, solve
a = symbols('a')
print(solve(Eq(a * 2**2, 12), a))   # solve 4a = 12
```

Output ▸

```
[3]
```

We get $a = 3$. The flow "passes through a point → substitute to get an equation → solve it" is a basic pattern we will use again and again in later chapters.

The Form That Shows the Vertex: $y = a(x - p)^2 + q$

The key to handling the graph of a quadratic function freely is the following form.

$$y = a(x - p)^2 + q$$

The graph of this form is the parabola obtained by translating the graph of $y = ax^2$ by p in the x direction and q in the y direction. Its **vertex** is the point (p, q), and its **axis** is the line $x = p$. Because the vertex and the axis can be read directly from the expression, we will call this the vertex form.

> **Term** **Vertex and axis** The lowest point of a parabola (the highest point if it opens downward) is called its vertex. The vertical line through the vertex that splits the parabola into two mirror-image halves is called its axis. For $y = a(x - p)^2 + q$, the vertex is (p, q) and the axis is $x = p$.

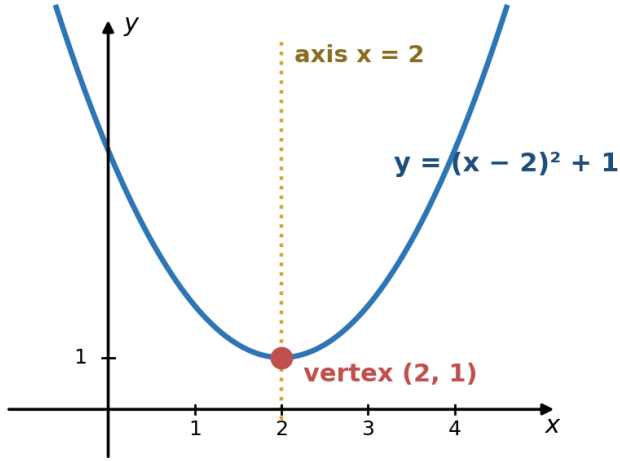


Figure 3.3 The graph of $y = (x - 2)^2 + 1$. The vertex is (2, 1) and the axis is $x = 2$.

Example 3.3

State the vertex and the axis of the quadratic function $y = (x - 2)^2 + 1$, and draw its graph.

With pen and paper In the form $y = a(x - p)^2 + q$ we have $a = 1$, $p = 2$, $q = 1$. So the vertex is (2, 1) and the axis is the line $x = 2$. Watch the sign in $x - 2$: when the parentheses contain $x - p$, the x-coordinate of the vertex is p (the sign looks reversed, so be careful). The graph is as in Figure 3.3.

Check it with Python Let us draw the graph with matplotlib (a recap of Level 1B, Chapter 6). We prepare many values of x, compute y for each, and

connect them with plot.

```
import matplotlib.pyplot as plt
# x from -1.0 to 5.0 in steps of 0.1
xs = [x / 10 for x in range(-10, 51)]
ys = [(x - 2)**2 + 1 for x in xs]
plt.plot(xs, ys)               # draw the parabola
plt.plot([2], [1], marker='o') # mark the vertex (2, 1)
plt.grid(True)                 # show the grid
plt.show()
```

Running this displays a parabola like Figure 3.3. Check with your own eyes that the vertex (2, 1) is the lowest point, and that the graph is symmetric about the vertical line x = 2.

3.3 Completing the Square: Making the Vertex Appear

In the general form $y = ax^2 + bx + c$, you cannot see where the vertex is. So we transform the expression into the vertex form $a(x - p)^2 + q$. This transformation is called **completing the square** — literally, finishing off a square (a second power).

> **Term** **Completing the square** Rewriting a quadratic expression $ax^2 + bx + c$ into the form $a(x - p)^2 + q$. The vertex (p, q) of the graph can then be read off from the expression.

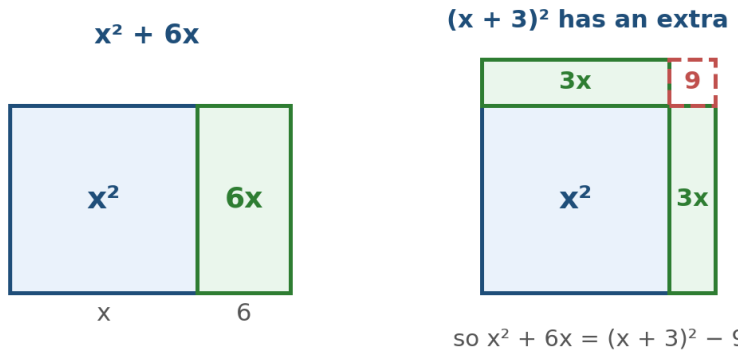


Figure 3.4 Completing the square, as a picture. Adding 9 to $x^2 + 6x$ completes the square $(x + 3)^2$, so $x^2 + 6x = (x + 3)^2 - 9$.

Figure 3.4 is the "picture" of completing the square. To rearrange the area $x^2 + 6x$ into a square, split 6x into two halves (3x and 3x) and attach them to the right side and the top of the square. Then a gap of $3 \times 3 = 9$ opens at

the top right corner. In other words, $(x + 3)^2$ has an extra 9. Therefore

$$x^2 + 6x = (x + 3)^2 - 9$$

As a procedure, remember it as: "use half of the coefficient of x — half of 6 is 3 — to build $(x + 3)^2$, then subtract the extra $3^2 = 9$."

Example 3.4

Complete the square for $y = x^2 + 6x + 5$ and find the vertex of its graph.

With pen and paper First complete the square for the $x^2 + 6x$ part. Half of the coefficient 6 is 3.

$$y = (x + 3)^2 - 9 + 5 = (x + 3)^2 - 4$$

We have the vertex form. Since $x + 3 = x - (-3)$, the vertex is $(-3, -4)$.

Check it with Python The reliable way to check completing the square is to expand the result and see whether it returns to the original. Time for expand from Chapter 1.

```
from sympy import symbols, expand
x = symbols('x')
print(expand((x + 3)**2 - 4))   # expand to check
```

Output ▸

```
x**2 + 6*x + 5
```

It returned to the original expression. Completing the square and expanding are transformations in opposite directions — just as with factoring, expand makes a dependable checker.

Example 3.5

Complete the square for $y = 2x^2 - 8x + 3$ and find the vertex.

With pen and paper The coefficient of x^2 is 2, so first factor 2 out of the x^2 and x terms only.

$$y = 2(x^2 - 4x) + 3$$

Complete the square inside the parentheses. Half of the coefficient -4 is -2, so $x^2 - 4x = (x - 2)^2 - 4$. Putting this back,

$$y = 2\{(x - 2)^2 - 4\} + 3 = 2(x - 2)^2 - 8 + 3 = 2(x - 2)^2 - 5$$

The vertex is $(2, -5)$. The easy place to slip is the -4 inside the factor of 2: when the parentheses are removed it becomes $2 \times (-4) = -8$.

Check it with Python

```
from sympy import symbols, expand
x = symbols('x')
print(expand(2*(x - 2)**2 - 5))
```

Output ▸

```
2*x**2 - 8*x + 3
```

This one also returned to the original. Even with a leading coefficient, completing the square is safe once you check by expanding.

3.4 Maximum and Minimum: The Highest and Lowest Points

The vertex of a quadratic function is the lowest point of the graph (when $a > 0$) or the highest point (when $a < 0$). So once you know the vertex, you know the **minimum value** or the **maximum value** of the function. For example, $y = (x - 1)^2 + 2$ has its lowest point at the vertex $(1, 2)$, so it takes the minimum value 2 at $x = 1$.

In real problems, however, the range over which x may move is often restricted. This range is called the **domain**. Once a domain is attached, the picture of maximum and minimum values changes.

Term **Domain** For a function, the range of values that x may take is called the domain — for example, $0 \le x \le 4$, written as an inequality. When no restriction is given, x is considered to move over all real numbers.

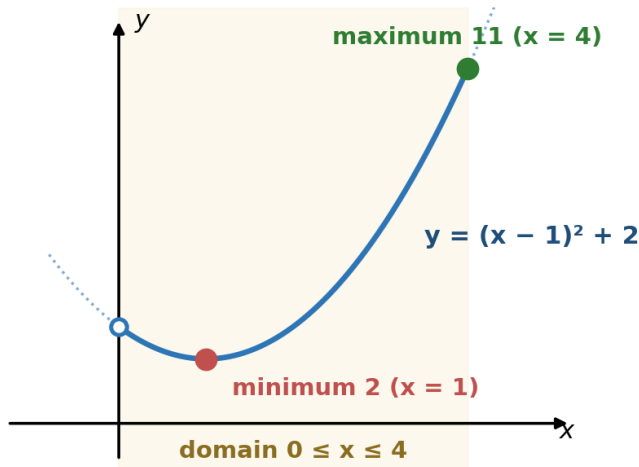


Figure 3.5 $y = (x - 1)^2 + 2$ on the domain $0 \le x \le 4$. The minimum is at the vertex; the maximum is at the end farther from the vertex.

Look at Figure 3.5. On the domain $0 \le x \le 4$, the minimum is taken at the vertex $x = 1$, but the maximum is taken at one of the **ends of the domain**. The farther an end is from the vertex, the larger the value of y, so comparing the two ends $x = 0$ and $x = 4$ decides the maximum.

Example 3.6

Find the maximum and minimum values of the quadratic function $y = (x - 1)^2 + 2$ on the domain $0 \le x \le 4$.

With pen and paper The vertex is (1, 2), and $x = 1$ lies inside the domain $0 \le x \le 4$. The graph opens upward, so the minimum value is 2, at $x = 1$. For the maximum, compare the two ends: at $x = 0$, $y = (0 - 1)^2 + 2 = 3$; at $x = 4$, $y = (4 - 1)^2 + 2 = 9 + 2 = 11$. The larger is 11, so the maximum value is 11, at $x = 4$.

Check it with Python Define the function with def and print its values at the vertex and the two ends.

```
def f(x):
    return (x - 1)**2 + 2
print(f(1))                  # vertex value = minimum
print(f(0))                  # left end of the domain
print(f(4))                  # right end of the domain
```

Output ▸

```
2
3
11
```

The minimum 2 ($x = 1$) and the maximum 11 ($x = 4$) are confirmed. "Minimum at the vertex, maximum at an end" — but when the vertex lies outside the domain, the minimum is also taken at an end. Make it a habit to sketch the graph and check where the vertex sits.

3.5 The Link to Quadratic Equations: Where the Graph Crosses the x-axis

Recap (Level 1B, Chapter 3) A quadratic equation $ax^2 + bx + c = 0$ can be solved with the quadratic formula. The expression under the square root, $b^2 - 4ac$, is called the discriminant D, and the sign of D decides the number of solutions.

At a point where the graph of $y = ax^2 + bx + c$ meets the x-axis, we have $y = 0$. In other words, **the x-coordinates of the points the graph shares with the x-axis are exactly the solutions of the quadratic equation $ax^2 + bx + c = 0$**. Here, graphs of functions and equations join hands.

Example 3.7

Find the x-coordinates of the points that the graph of $y = x^2 - 2x - 3$ shares with the x-axis.

With pen and paper Set $y = 0$ and solve the quadratic equation $x^2 - 2x - 3 = 0$. Factoring the left side gives $(x + 1)(x - 3) = 0$, so $x = -1, 3$. The graph crosses the x-axis at the two points $(-1, 0)$ and $(3, 0)$.

Check it with Python Recall that for an equation of the form = 0, solve accepts just the expression.

```
from sympy import symbols, solve
x = symbols('x')
print(solve(x**2 - 2*x - 3, x))   # solutions of = 0
```

Output ▸

```
[-1, 3]
```

The solutions are -1 and 3 — the two places where the graph crosses the x-axis. This same parabola will play the lead role again in the quadratic inequalities of the next section.

The Number of Shared Points and the Discriminant

The graph does not always share two points with the x-axis. What decides the number of solutions is the **discriminant** $D = b^2 - 4ac$. Let us pin down the correspondence with a figure and a table.

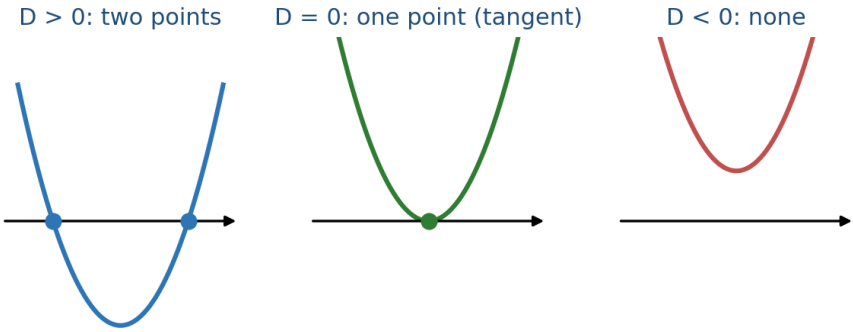


Figure 3.6 The discriminant D and the number of shared points: two if D > 0, one (tangent) if D = 0, none if D < 0.

Table 3.1 The discriminant D and the shared points with the x-axis

Discriminant	Shared points	How the graph looks
$D > 0$	2	crosses the x-axis at two points
$D = 0$	1	tangent to the x-axis (vertex on it)
$D < 0$	0	does not meet the x-axis

Term Discriminant For a quadratic equation $ax^2 + bx + c = 0$, the discriminant is $D = b^2 - 4ac$. Its sign tells the number of solutions — that is, the number of points the graph shares with the x-axis.

Example 3.8

Using the discriminant, decide how many points each graph shares with the x-axis.

(1) $y = x^2 - 2x - 3$ (2) $y = x^2 - 4x + 4$ (3) $y = x^2 - 4x + 5$

With pen and paper (1) $D = (-2)^2 - 4 \times 1 \times (-3) = 4 + 12 = 16 > 0$, so two points. (2) $D = (-4)^2 - 4 \times 1 \times 4 = 16 - 16 = 0$, so one point (tangent). (3) $D = (-4)^2 - 4 \times 1 \times 5 = 16 - 20 = -4 < 0$, so none.

Check it with Python SymPy has a function called discriminant that computes the discriminant. Writing discriminant(expression, letter) returns the discriminant of that quadratic.

```
from sympy import symbols, discriminant
x = symbols('x')
print(discriminant(x**2 - 2*x - 3, x))   # (1)
print(discriminant(x**2 - 4*x + 4, x))   # (2)
print(discriminant(x**2 - 4*x + 5, x))   # (3)
```

Output ▸

```
16
0
-4
```

Out come 16, 0, and −4 — the same D as by hand. Reading the signs, the shared points number two, one, and none.

3.6 Quadratic Inequalities: Solving with Graphs

In Chapter 1 we studied linear inequalities. Now comes

$$x^2 - 2x - 3 > 0,$$

a **quadratic inequality**. Quadratic inequalities can be solved at a glance using a graph. There is only one idea:

over which range of x is the graph of y = (left side) above the x-axis?

Read "> 0" as "the graph is above the x-axis" and "< 0" as "the graph is below the x-axis."

> **Term Quadratic inequality** An inequality of the form (quadratic expression) > 0 or (quadratic expression) < 0. The solution range can be read from how the graph of the left side sits relative to the x-axis.

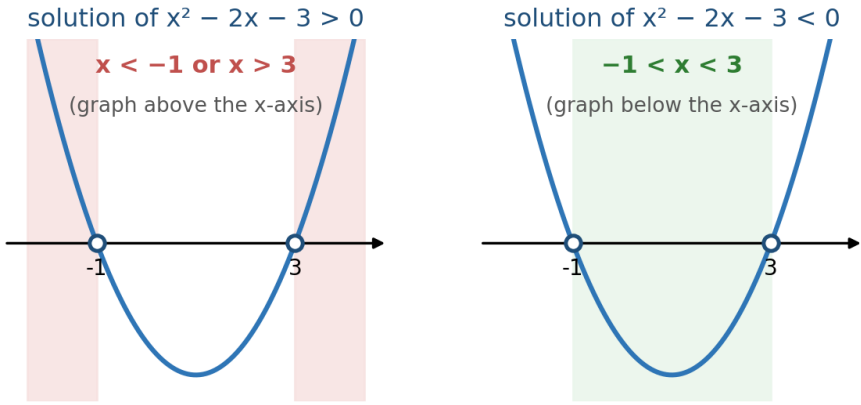


Figure 3.7 Solving quadratic inequalities. The solution of $x^2 - 2x - 3 > 0$ is where the graph is above the x-axis; the solution of < 0 is where it is below.

Example 3.9

Solve the quadratic inequalities $x^2 - 2x - 3 > 0$ and $x^2 - 2x - 3 < 0$.

With pen and paper As we saw in Example 3.7, the graph of $y = x^2 - 2x - 3$ is a parabola opening upward that crosses the x-axis at $x = -1$ and 3 (Figure 3.7). The graph is above the x-axis outside the crossing points, so the solution of > 0 is $x < -1$ or $x > 3$. The graph is below the x-axis between the crossing points, so the solution of < 0 is $-1 < x < 3$.

Check it with Python solve handles quadratic inequalities directly too. To the oo (infinity) and & (and) we met in Chapter 1, this time | (or) is added.

```
from sympy import symbols, solve
x = symbols('x')
print(solve(x**2 - 2*x - 3 > 0, x))
print(solve(x**2 - 2*x - 3 < 0, x))
```

Output ▸

```
((-oo < x) & (x < -1)) | ((3 < x) & (x < oo))
(-1 < x) & (x < 3)
```

The first line says "(x is less than −1) or (x is greater than 3)," that is, $x < -1$ or $x > 3$. The second line is $-1 < x < 3$. The symbol | is the same "or" we used for unions in Chapter 2; it joins the pieces whenever the solution splits into two ranges. Both agree with the graph reading by hand.

Example 3.10

A rectangular enclosure is to be made with a rope 20 m long. Let the width

be x m. Find the x that makes the area S as large as possible, and the area at that point.

With pen and paper The rope is 20 m around, so width + height = 10 m. The height is therefore (10 − x) m (Figure 3.8, left). The area is

$$S = x(10 - x) = -x^2 + 10x.$$

S has become a quadratic function of x. Completing the square,

$$S = -(x - 5)^2 + 25.$$

Since a = −1 < 0 the parabola opens downward, so the vertex is the highest point: the maximum value is 25, at x = 5. That is, the area is largest — 25 m² — when the enclosure is a 5 m × 5 m square (Figure 3.8, right). Since x is a width, keep the domain 0 < x < 10 in mind as well.

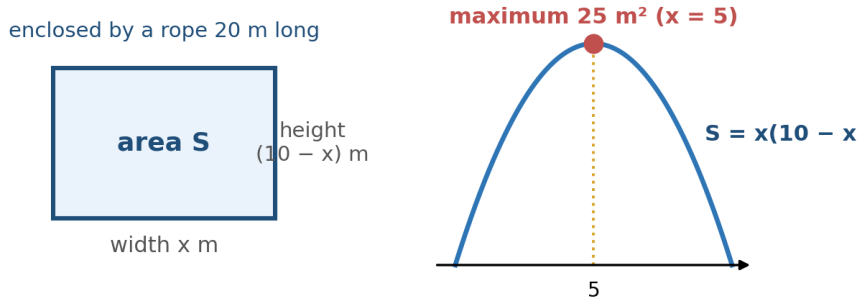


Figure 3.8 The area of the rectangular enclosure. The graph of S = x(10 − x) opens downward and takes its maximum value 25 at x = 5.

Check it with Python We check the completed square and compute the area at x = 5.

```
from sympy import symbols, expand
x = symbols('x')
print(expand(-(x - 5)**2 + 25))   # expand to check
def S(x):
    return x * (10 - x)
print(S(5))                       # area at x = 5
```

Output ▸

```
-x**2 + 10*x
25
```

Expanding gives −x² + 10x, the same expression as S = x(10 − x), and the largest area is confirmed as 25 m². "Express the quantity you want to

maximize or minimize as a quadratic function, then complete the square" — this is the classic pattern by which quadratic functions become tools for real problems.

Column Parabolas Are Everywhere

The Japanese word for parabola means "the curve of a thrown object" — and the English word comes from the Greek for "throwing alongside." Throw a ball at an angle and, ignoring air resistance, its path is exactly a parabola opening downward. The first person to show this mathematically was Galileo Galilei in 17th-century Italy. The arcs of water in a park fountain are parabolas for the same reason. Parabolas also have a marvelous property: light or radio waves arriving parallel to the axis are all collected at a single point, the focus. The dish of a satellite antenna is exactly a parabola spun around its axis — which is why it is called a parabolic antenna. Even the faint radio waves from deep space can be received because the parabola gathers them to one point. The curve $y = ax^2$ you studied in this chapter is hiding not only in textbooks, but in the sky and in antennas.

Chapter Summary

- A function f(x) is "a rule of correspondence that assigns exactly one value to each x." f(2) is the value of the function at x = 2.
- The graph of a quadratic function $y = ax^2 + bx + c$ is a parabola: opening upward if $a > 0$, downward if $a < 0$.
- The vertex form $y = a(x - p)^2 + q$ has vertex (p, q) and axis x = p.
- Completing the square builds the squared form using "half the coefficient of x" and subtracts the extra. It can be checked by expanding.
- Maximum and minimum values are decided by the vertex and the ends of the domain. Sketch the graph and check where the vertex sits.
- The x-coordinates of the shared points with the x-axis are the solutions of the quadratic equation; their number is decided by the sign of the discriminant $D = b^2 - 4ac$.
- A quadratic inequality is solved with a graph, read as "the range where the graph is above (below) the x-axis."

Table 3.2 Python tools used in this chapter

Python tool	What it does
`def f(x): return ...`	define the function f(x)
`solve(Eq(lhs, rhs), a)`	solve an equation (find a coefficient)
`expand(expr)`	expand to check completing the square
`solve(expr, x)`	solve an equation of the form expr = 0
`discriminant(expr, x)`	compute the discriminant D
`solve(inequality, x)`	solve a quadratic inequality (| is "or")
`plot from matplotlib`	draw the graph of a function

Practice Problems

Solve each problem with pen and paper first, then write Python code in an empty Colab cell to check your answer. For problems that involve graphs, drawing them with matplotlib is also a good check.

Problem 1

For the function $f(x) = x^2 + 3x$, find f(2) and f(−3).

Problem 2

Find the quadratic function whose graph has the form $y = ax^2$ and passes through the point (3, 18).

Problem 3

State the vertex and the axis of the quadratic function $y = (x + 1)^2 - 4$. Also confirm that the value at the vertex is smaller than the values at its neighbors on both sides (x = 0 and x = −2).

Problem 4

Complete the square for $y = x^2 - 4x + 1$ and find the vertex.

Problem 5

Complete the square for $y = 3x^2 + 6x - 2$ and find the vertex.

Problem 6

Find the maximum and minimum values of the quadratic function $y = (x - 3)^2 - 1$ on the domain $1 \le x \le 5$.

Problem 7

Find the x-coordinates of the points that the graph of $y = x^2 - x - 6$ shares with the x-axis.

Problem 8

Using the discriminant, decide how many points the graph of $y = x^2 + 2x + 3$ shares with the x-axis.

Problem 9

Solve the quadratic inequality $x^2 - x - 6 < 0$.

Problem 10

A rectangular enclosure is to be made with a rope 24 m long. Let the width be x m. Express the area S in terms of x, and find the x that makes S largest, together with the area at that point.

Answers and Explanations

Answer 1

f(2) = 4 + 6 = 10. f(−3) = 9 − 9 = 0. Substitute negative numbers with parentheses: (−3)² = 9.

```
def f(x):
    return x**2 + 3*x
print(f(2))
print(f(-3))
```

Output ▸

```
10
0
```

Answer 2

Substituting x = 3, y = 18 gives 18 = 9a, so a = 2 and y = 2x².

```
from sympy import symbols, Eq, solve
a = symbols('a')
print(solve(Eq(a * 3**2, 18), a))
```

Output ▸

```
[2]
```

Answer 3

Since x + 1 = x − (−1), the vertex is (−1, −4) and the axis is the line x = −1. The vertex value f(−1) = −4 is indeed smaller than the neighboring values f(0) = −3 and f(−2) = −3.

```
def f(x):
    return (x + 1)**2 - 4
print(f(-1))
print(f(0))
print(f(-2))
```

Output ▸

```
-4
-3
-3
```

Answer 4

Half of the coefficient -4 is -2. $y = (x - 2)^2 - 4 + 1 = (x - 2)^2 - 3$, so the vertex is $(2, -3)$.

```
from sympy import symbols, expand
x = symbols('x')
print(expand((x - 2)**2 - 3))
```

Output ▸

```
x**2 - 4*x + 1
```

Expanding returns the original expression, so the completed square is correct.

Answer 5

First factor out 3: $y = 3(x^2 + 2x) - 2$. Inside, $x^2 + 2x = (x + 1)^2 - 1$, so

$$y = 3\{(x + 1)^2 - 1\} - 2 = 3(x + 1)^2 - 5.$$

The vertex is $(-1, -5)$. Watch the step where $3 \times (-1) = -3$ combines with -2 to give -5.

```
from sympy import symbols, expand
x = symbols('x')
print(expand(3*(x + 1)**2 - 5))
```

Output ▸

```
3*x**2 + 6*x - 2
```

Answer 6

The vertex is $(3, -1)$, and $x = 3$ lies inside the domain $1 \le x \le 5$. The minimum is -1, at $x = 3$. For the maximum, the two ends give $f(1) = 4 - 1 = 3$ and $f(5) = 4 - 1 = 3$ — both 3 (the vertex sits exactly in the middle of the domain, so the two ends are equal). The maximum value is 3.

```
def f(x):
    return (x - 3)**2 - 1
print(f(3))                    # minimum
print(f(1))                    # left end
print(f(5))                    # right end
```

Output ▸

```
-1
3
3
```

Answer 7

Factoring $x^2 - x - 6 = 0$ gives $(x + 2)(x - 3) = 0$, so $x = -2, 3$.

```
from sympy import symbols, solve
x = symbols('x')
print(solve(x**2 - x - 6, x))
```

Output ▸

```
[-2, 3]
```

Answer 8

$D = 2^2 - 4 \times 1 \times 3 = 4 - 12 = -8 < 0$. So there are no shared points (the graph does not meet the x-axis).

```
from sympy import symbols, discriminant
x = symbols('x')
print(discriminant(x**2 + 2*x + 3, x))
```

Output ▸

```
-8
```

Answer 9

The graph of $y = x^2 - x - 6$ is a parabola opening upward that crosses the x-axis at $x = -2$ and 3 (Answer 7). The solution of < 0 is the range where the graph is below the x-axis — between the crossing points: $-2 < x < 3$.

```
from sympy import symbols, solve
x = symbols('x')
print(solve(x**2 - x - 6 < 0, x))
```

Output ▸

```
(-2 < x) & (x < 3)
```

Answer 10

Width + height = 12 m, so the height is (12 – x) m. The area is $S = x(12 - x) = -x^2 + 12x$. Completing the square, $S = -(x - 6)^2 + 36$. The parabola opens downward, so the maximum value is 36, at x = 6: a 6 m × 6 m square gives the largest area, 36 m² (just as in Example 3.10, the square wins).

```
from sympy import symbols, expand
x = symbols('x')
print(expand(-(x - 6)**2 + 36))
def S(x):
    return x * (12 - x)
print(S(6))
```

Output ▸

```
-x**2 + 12*x
36
```

Chapter 4
Measuring Figures with Numbers: Trigonometric Ratios

sin, cos, and tan

Relations among trigonometric ratios

Trigonometric ratios of obtuse angles

The law of sines

The law of cosines

The area formula

Measuring figures

Chapter 4 Measuring Figures with Numbers: Trigonometric Ratios

What You Will Learn

People have long used angles to measure things that are hard to measure directly: the distance to a tree across a river, the height of a mountain, or the distance to a faraway building. The tool for the job is the star of this chapter: the **trigonometric ratios** (sin, cos, and tan).

In the first half, we define the trigonometric ratios as ratios of the sides of a right triangle and actually compute the height of a tree. In the second half, we extend the ratios to obtuse angles and then acquire two great theorems that work for any triangle — the **law of sines** and the **law of cosines** — together with the area formula. Once a few of a triangle's three sides and three angles are known, all the rest can be computed: that is the goal of this chapter.

The way we work stays the same: first understand the ideas with pen and paper, then run the same ideas in Python to check them. For values we use SymPy (exact values) and the math module (decimal values), each where it fits best.

By the end of this chapter, you will be able to:

- explain sin, cos, and tan as ratios of the sides of a right triangle
- use tan to compute the height of a tree or a building
- use the relations such as $\sin^2\theta + \cos^2\theta = 1$ to find the other ratios from one
- use the unit circle to find trigonometric ratios of obtuse angles (angles beyond 90°)
- compute sides and angles of a triangle with the law of sines and the law of cosines
- find the area of a triangle with the formula $S = (1/2)bc \sin A$
- check trigonometric calculations with SymPy and the math module

4.1 What Trigonometric Ratios Are: sin, cos, and tan

Recap (Level 1B, Chapter 9) Two triangles are similar when all their angles are equal, and then the ratios of corresponding sides are all equal. So the shape of a right triangle is decided by just one angle besides the right angle — and its side ratios are decided by that angle alone.

Giving names to this property — "fix the angle and the side ratios are fixed" — is exactly what the trigonometric ratios do. As in Figure 4.1, in a right triangle with an angle θ (theta), the side facing θ is the **opposite side**, the side next to θ is the **adjacent side**, and the longest side, facing the right angle, is the **hypotenuse**. We define three ratios as follows.

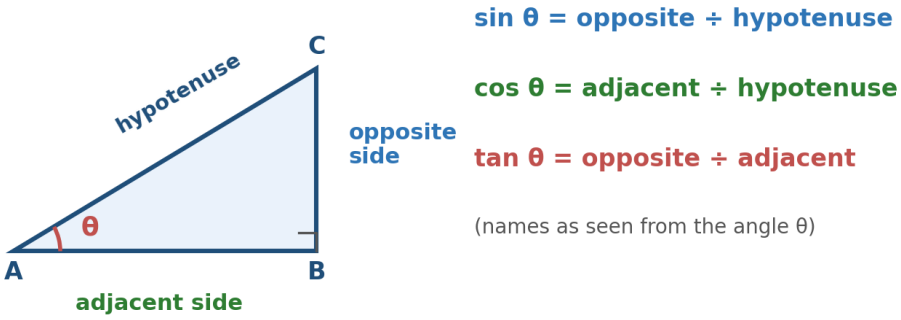


Figure 4.1 Defining the trigonometric ratios. Seen from the angle θ, the facing side is opposite and the neighboring side is adjacent.

Term Trigonometric ratios (sin, cos, tan) In a right triangle, for an angle θ we define sin θ = opposite ÷ hypotenuse, cos θ = adjacent ÷ hypotenuse, and tan θ = opposite ÷ adjacent (read sine, cosine, tangent). Similar triangles have equal ratios, so these values do not depend on the size of the triangle — only on the angle θ.

Example 4.1

A right triangle has sides 3, 4, and 5 (hypotenuse 5). Let A be the angle between the side of length 4 and the hypotenuse. Find sin A, cos A, and tan A.

With pen and paper Seen from the angle A, the opposite side is 3, the adjacent side is 4, and the hypotenuse is 5. So $\sin A = \frac{3}{5}$, $\cos A = \frac{4}{5}$, and $\tan A = \frac{3}{4}$. By the way, 3, 4, 5 really does make a right triangle because the Pythagorean theorem holds: $3^2 + 4^2 = 5^2$ (Level 1B, Chapter 10).

Check it with Python First confirm the hypotenuse with the Pythagorean theorem, then express the three ratios with Rational (fractions).

```
from sympy import sqrt, Rational
print(sqrt(3**2 + 4**2))   # hypotenuse by Pythagoras
print(Rational(3, 5))      # sin A
print(Rational(4, 5))      # cos A
print(Rational(3, 4))      # tan A
```

Output ▸

```
5
3/5
4/5
3/4
```

The hypotenuse is indeed 5. Trigonometric ratios are "just ratios of sides," so they can be written exactly as fractions. These plain ratios are about to work hard, from surveying all the way to the great theorems.

Example 4.2 Measuring the Height of a Tree

You stand 30 m from a tree and look up at its top; the angle of elevation is 35°. Taking your eye height to be 1.5 m, find the height of the tree (Figure 4.2).

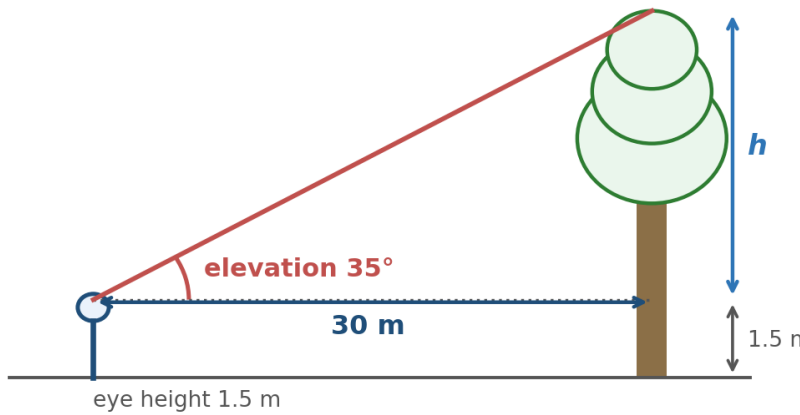


Figure 4.2 Measuring a tree with tan. From the elevation 35° and the distance 30 m, the height can be computed.

With pen and paper Let h be the height from eye level to the treetop. In the right triangle of Figure 4.2, tan 35° = h ÷ 30, so h = 30 × tan 35°. The

value of tan 35° must come from a table or a calculator — here, we leave it to Python. Adding the eye height 1.5 m at the end gives the height of the tree.

Check it with Python We compute with math.tan from the math module. One caution: a computer's trigonometric functions take angles in a unit called radians (you will study it in Level 2B). Writing math.radians(35) converts 35° into radians. round(h, 1) rounds h to one decimal place.

```
import math
h = 30 * math.tan(math.radians(35))   # to the treetop
print(round(h, 1))
print(round(h + 1.5, 1))      # add eye height 1.5 m
```

Output ▸

```
21.0
22.5
```

The tree is about 22.5 m tall. Without climbing the tree, a tape measure and an angle measure are all it takes — a calculation true to this chapter's title, "measuring figures with numbers."

Trigonometric Ratios of the Famous Angles

For 30°, 45°, and 60°, exact values come from the two set-square triangles of Figure 4.3. The side ratio of the 30°–60°–90° triangle is 1 : 2 : √3, and that of the 45°–45°–90° triangle is 1 : 1 : √2 (Level 1B, Chapter 10).

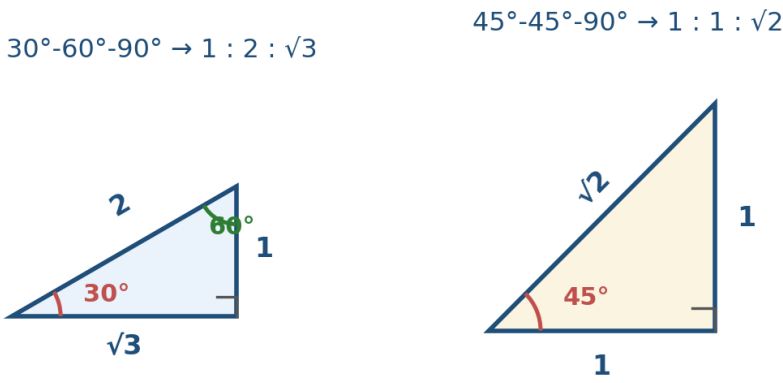


Figure 4.3 The two set-square triangles. The side ratios give exact trigonometric ratios for the famous angles.

Table 4.1 Trigonometric ratios of the famous angles

θ	30°	45°	60°
$sin\ \theta$	$\frac{1}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{3}}{2}$
$cos\ \theta$	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{1}{2}$
$tan\ \theta$	$\frac{1}{\sqrt{3}}$	1	$\sqrt{3}$

SymPy provides sin, cos, and tan, and returns these values in exact form. rad is SymPy's "degrees to radians" function (the SymPy version of math.radians).

```
from sympy import sin, cos, tan, rad
print(sin(rad(30)))
print(cos(rad(45)))
print(tan(rad(60)))
```

Output ▸

```
1/2
sqrt(2)/2
sqrt(3)
```

They match Table 4.1 (note that $\frac{\sqrt{2}}{2}$ is $\frac{1}{\sqrt{2}}$ with its denominator rationalized — rationalizing from Chapter 1 makes a comeback here).

4.2 Relations Among the Trigonometric Ratios: Know One, Know Them All

sin, cos, and tan are not three unrelated objects. In the right triangle of Figure 4.1, write opposite = a, adjacent = b, hypotenuse = c. The Pythagorean theorem gives $a^2 + b^2 = c^2$; dividing both sides by c^2,

$$(\frac{a}{c})^2 + (\frac{b}{c})^2 = 1$$

The two parentheses on the left are exactly sin θ and cos θ. Also, tan θ = a ÷ b becomes sin θ ÷ cos θ when numerator and denominator are divided by c.

> **Term** **Relations among the trigonometric ratios** $sin^2\theta + cos^2\theta = 1$ and $tan\,\theta = \frac{sin\,\theta}{cos\,\theta}$ always hold. Here sin²θ means (sin θ)².

With these relations, once one trigonometric ratio is known, the other two can be computed.

Example 4.3

Let θ be an acute angle with $\sin\theta = \frac{3}{5}$. Find cos θ and tan θ.

With pen and paper Substituting into $\sin^2\theta + \cos^2\theta = 1$: $\cos^2\theta = 1 - (\frac{3}{5})^2 = 1 - \frac{9}{25} = \frac{16}{25}$. Since θ is acute, cos θ > 0, so $\cos\theta = \frac{4}{5}$. Then $\tan\theta = \sin\theta \div \cos\theta = \frac{3}{5} \div \frac{4}{5} = \frac{3}{4}$. The same values as the 3 : 4 : 5 triangle of Example 4.1 have appeared.

Check it with Python We turn the flow of the calculation directly into code.

```
from sympy import sqrt, Rational
cosA = sqrt(1 - Rational(3, 5)**2)   # from the relation
print(cosA)
print(Rational(3, 5) / cosA)         # tan = sin / cos
```

Output ▸

```
4/5
3/4
```

Both match the hand calculation. The flow "one ratio → the relations → all the rest" will return again and again as the warm-up step for using this chapter's theorems.

4.3 Trigonometric Ratios of Obtuse Angles: Beyond the 90° Wall

So far, our trigonometric ratios covered only angles inside a right triangle — acute angles from 0° to 90°. But triangles can also contain an **obtuse angle**, one beyond 90°. Let us widen the definition so that obtuse angles can use trigonometric ratios too.

The tool is the **unit circle** (a circle of radius 1) in Figure 4.4. Take a point P on the upper half of the circle of radius 1 centered at the origin, and let θ be the angle measured from the positive x-axis. We then decide:

the x-coordinate of P is cos θ, and the y-coordinate of P is sin θ.

When θ is acute, this agrees with the earlier definition (think of a right triangle with hypotenuse 1: opposite ÷ 1 = the y-coordinate, adjacent ÷ 1 = the x-coordinate). And even when θ passes 90°, the coordinates of P are still perfectly well defined — so the trigonometric ratios now work from 0° to 180°.

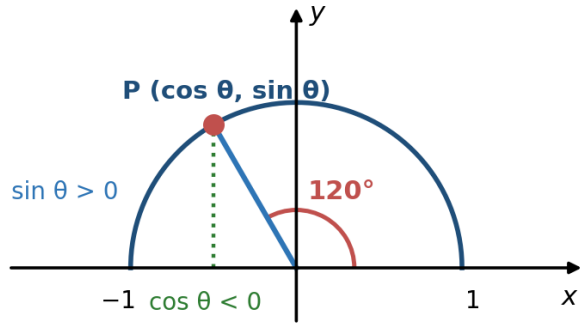


Figure 4.4 The unit-circle definition. The x-coordinate of P is cos θ and the y-coordinate is sin θ. For obtuse angles, cos θ becomes negative.

Look at θ = 120° in Figure 4.4. P lies to the left of the y-axis, so its x-coordinate — that is, **cos θ — is negative**. Meanwhile P is above the x-axis, so sin θ stays positive. tan θ = sin θ ÷ cos θ also becomes negative. For obtuse angles, remember: "cos and tan pick up a minus sign."

The relation with 180° − θ In Figure 4.4, the point for 120° and the point for 60° sit in mirror positions about the y-axis. So sin 120° = sin 60° and cos 120° = −cos 60°. In general, sin(180° − θ) = sin θ, cos(180° − θ) = −cos θ, and tan(180° − θ) = −tan θ — every obtuse ratio converts to an acute one.

Example 4.4

Find the values of sin 120°, cos 135°, and tan 150°.

With pen and paper Convert to acute angles with the 180° − θ relation. $\sin 120° = \sin 60° = \frac{\sqrt{3}}{2}$. $\cos 135° = -\cos 45° = -\frac{\sqrt{2}}{2}$. $\tan 150° = -\tan 30° = -\frac{1}{\sqrt{3}}$ (rationalized: $-\frac{\sqrt{3}}{3}$).

Check it with Python SymPy's sin, cos, and tan handle obtuse angles directly.

```
from sympy import sin, cos, tan, rad
print(sin(rad(120)))
print(cos(rad(135)))
print(tan(rad(150)))
```

Output ▸

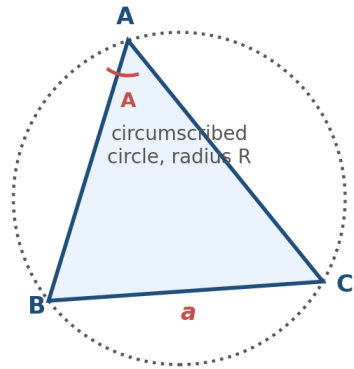
```
sqrt(3)/2
-sqrt(2)/2
-sqrt(3)/3
```

All match the hand calculations (tan 150° appears in rationalized form). You can also see cos and tan picking up their minus signs. With obtuse ratios in hand, from the next section we can treat triangles without any right angle with confidence.

4.4 The Law of Sines: A Bridge Between Angles and Sides (Part 1)

From here on we handle a general triangle ABC, with no right angle guaranteed. By convention, the sides facing the angles A, B, C are written a, b, c. The first great theorem ties each side to the sine of its facing angle.

Term **Law of sines** Let R be the radius of the circumscribed circle of triangle ABC (the circle through its three vertices). Then $\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C} = 2R$ holds. "Side ÷ sine of the facing angle" is the same value for every pair, and that value equals the diameter of the circumscribed circle.



Law of sines: a ÷ sin A = 2R (the same for every side-angle pair)

Figure 4.5 The law of sines. The circle through the three vertices is the circumscribed circle; a ÷ sin A equals its diameter 2R.

As in Figure 4.5, every triangle has exactly one circle through its three vertices — its circumscribed circle. The law of sines says: one side–angle pair fixes the size of that circle, and through it, all the other sides and angles connect. The moment to use it is **when a facing side–angle pair is in view**.

Example 4.5

In triangle ABC, a = 4, A = 30°, and B = 45°. Find the side b.

With pen and paper Use the law of sines with the pairs a, A and b, B: $\frac{4}{\sin 30^\circ} = \frac{b}{\sin 45^\circ}$. So $b = \frac{4 \sin 45^\circ}{\sin 30^\circ} = 4 \times \frac{\sqrt{2}}{2} \div \frac{1}{2} = 4\sqrt{2}$.

Check it with Python Enter the expression as it stands.

```
from sympy import sin, rad
b = 4 * sin(rad(45)) / sin(rad(30))   # law of sines
print(b)
```

Output ▸

```
4*sqrt(2)
```

We get b = 4√2 (about 5.66). One side–angle pair plus one more angle, and the facing side can be computed.

Example 4.6

In triangle ABC, a = 6 and A = 30°. Find the radius R of the circumscribed circle.

With pen and paper Use $\frac{a}{\sin A} = 2R$ from the law of sines: $\frac{6}{\sin 30°} = \frac{6}{1/2} = 12$ is the diameter 2R, so R = 6.

Check it with Python

```
from sympy import sin, rad
R = 6 / (2 * sin(rad(30)))   # from a / sin A = 2R
print(R)
```

Output ▸

```
6
```

All three vertices of this triangle sit on a circle of radius 6. From side-and-angle data to the size of an unseen circle — that is the charm of the law of sines.

4.5 The Law of Cosines: A Bridge Between Angles and Sides (Part 2)

The law of sines needed a facing side–angle pair. But what if we know two sides and their **included angle** — the angle between them? Enter the second great theorem: the law of cosines.

Term **Law of cosines** In triangle ABC, $a^2 = b^2 + c^2 - 2bc\cos A$ holds (from b, c, and the included angle A, the facing side a follows). Equations of the same form hold for B and C.

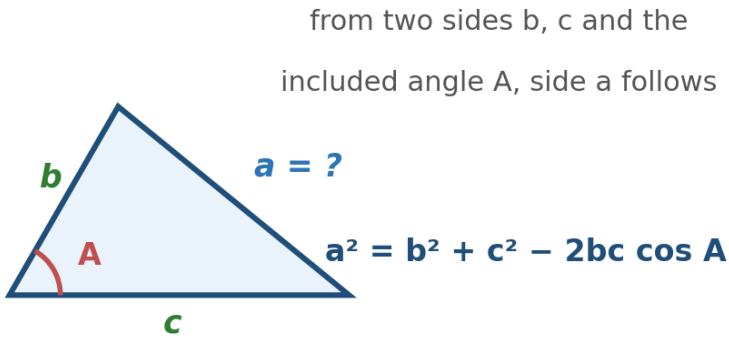


Figure 4.6 The law of cosines. From two sides b, c and the included angle A, the facing side a is found.

Look closely at this formula: when A = 90° we have cos A = 0, so $a^2 = b^2 + c^2$ — the Pythagorean theorem itself. The law of cosines is **the Pythagorean theorem extended to triangles without a right angle**. The term $-2bc \cos A$ reads as the correction for how far the angle is from a right angle.

Example 4.7

In triangle ABC, b = 5, c = 8, and A = 60°. Find the side a.

With pen and paper Substitute into the law of cosines: $a^2 = 5^2 + 8^2 - 2 \times 5 \times 8 \times \cos 60° = 25 + 64 - 80 \times \frac{1}{2} = 89 - 40 = 49$. Since a > 0, a = 7.

Check it with Python Enter the expression as it stands, taking the square root at the end.

```
from sympy import sqrt, cos, rad
a = sqrt(5**2 + 8**2 - 2*5*8*cos(rad(60)))
print(a)
```

Output ▸

```
7
```

We get a = 7. From two sides and the included angle, the remaining side of the triangle is decided in one stroke.

Example 4.8

A triangle has the three sides a = 7, b = 5, c = 8. Find the size of the angle A.

With pen and paper Use the law of cosines solved for cos A: $\cos A = \frac{b^2 + c^2 - a^2}{2bc}$. Substituting, $\cos A = \frac{25 + 64 - 49}{2 \times 5 \times 8} = \frac{40}{80} = \frac{1}{2}$. The angle with cos A $= \frac{1}{2}$ is A = 60° (the same triangle as Example 4.7 — we made the round trip from sides back to the angle).

Check it with Python To get the angle from the value of cos, use acos (the reverse computation of cos). The result comes back in radians, so we convert to degrees with deg (the reverse of rad).

```
from sympy import acos, deg, Rational
cosA = Rational(5**2 + 8**2 - 7**2, 2 * 5 * 8)
print(cosA)
print(deg(acos(cosA)))
```

Output ▸
```
1/2
60
```

A = 60°. With just the three sides, the angle can be computed — no protractor required. The law of cosines works dependably in both directions.

4.6 The Area of a Triangle and Measuring Figures

The finishing touch is area. A triangle's area is "base × height ÷ 2." In Figure 4.7, take the base to be c; then the height h can be written, from the right triangle made with the side b and the angle A, as h = b sin A. Therefore:

Term **Area formula for a triangle** The area S of triangle ABC is $S = \frac{1}{2}\,bc\,\sin A$ (two sides and their included angle decide the area).

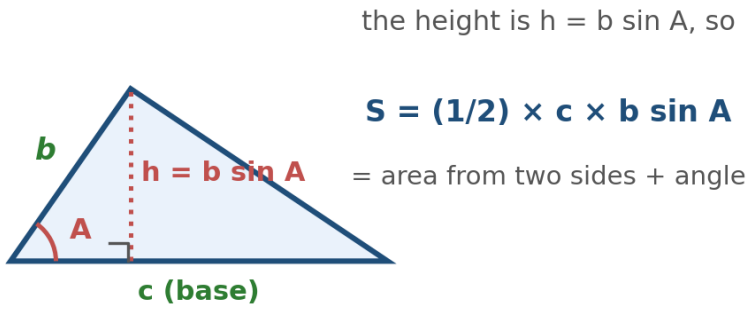


Figure 4.7 The area formula. Since the height is h = b sin A, we get S = (1/2) × base c × height b sin A.

Example 4.9

Find the area S of the triangle with b = 5, c = 8, and A = 60° (the same triangle as Example 4.7).

With pen and paper Substitute into the area formula: $S = \frac{1}{2} \times 5 \times 8 \times \sin 60° = 20 \times \frac{\sqrt{3}}{2} = 10\sqrt{3}$.

Check it with Python We print both the exact value and a decimal

approximation (evalf).

```
from sympy import sin, rad, Rational
S = Rational(1, 2) * 5 * 8 * sin(rad(60))
print(S)
print(S.evalf(4))   # to 4 digits
```

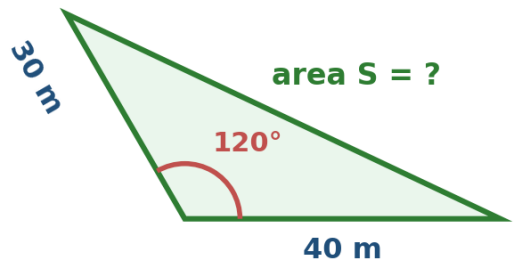
Output ▸

```
10*sqrt(3)
17.32
```

The area is $10\sqrt{3} \approx 17.32$. Without ever measuring the height, two sides and the included angle were enough.

Example 4.10 Measuring the Area of a Plot of Land

A park has a triangular lawn. Two of its sides are 30 m and 40 m, and the included angle is 120° (Figure 4.8). Find the area of the lawn.



a triangular plot in a park (two sides and the angle known)

Figure 4.8 A triangular plot in a park. With two sides and the included angle, the area formula gives the area.

With pen and paper Even when the included angle is obtuse, the area formula applies as it stands (thanks to extending the trigonometric ratios to obtuse angles in Section 4.3). $S = \frac{1}{2} \times 30 \times 40 \times \sin 120° = 600 \times \frac{\sqrt{3}}{2} = 300\sqrt{3}$.

Check it with Python

```
from sympy import sin, rad, Rational
S = Rational(1, 2) * 30 * 40 * sin(rad(120))
print(S)
print(S.evalf(4))
```

Output ▸

```
300*sqrt(3)
519.6
```

The lawn covers about 519.6 m². From the height of a tree (Example 4.2) to the area of a plot of land — measure a few angles and lengths, and the trigonometric ratios compute all the rest. The triangle is the mightiest measuring stick for figures.

Column The Man Who Mapped Japan with Triangles: Ino Tadataka and Triangulation

Computing the rest from sides and angles — the skill of this chapter goes by the name **triangulation**, and it has supported mapmaking for centuries. Measure one baseline accurately; after that, measuring only the angles to distant targets lets the laws of sines and cosines compute the distances. Even the distance to a mountaintop or an island you cannot walk to follows from a chain of triangles. In Edo-period Japan, Ino Tadataka began walking the country at age 55 and, over 17 years, accumulated angle measurements into an astonishingly accurate map of Japan. To this day, survey reference points (triangulation stations) sit on mountain summits, and the idea of triangulation lives on in satellite positioning. sin and cos left the classroom long ago — they are at work today in maps and in space.

Chapter Summary

- Trigonometric ratios are ratios of the sides of a right triangle: sin θ = opposite ÷ hypotenuse, cos θ = adjacent ÷ hypotenuse, tan θ = opposite ÷ adjacent.
- With tan, a height can be computed from a distance and an angle of elevation — the basis of surveying.
- The relations $\sin^2\theta + \cos^2\theta = 1$ and tan θ = sin θ ÷ cos θ give the other ratios from one.
- With the unit circle (x-coordinate of P is cos θ, y-coordinate is sin θ), the ratios extend from 0° to 180°. For obtuse angles, cos and tan are negative.

- Law of sines: $a \div \sin A = 2R$. Use it when a facing side–angle pair is given.
- Law of cosines: $a^2 = b^2 + c^2 - 2bc \cos A$. Use it when two sides and the included angle are given, or when all three sides are given. It extends the Pythagorean theorem.
- Area formula: $S = (1/2)bc \sin A$. Two sides and the included angle decide the area.

Table 4.2 Python tools used in this chapter

Python tool	What it does
`math.tan(math.radians(35))`	trigonometric ratios as decimals
`sin / cos / tan (SymPy)`	trigonometric ratios as exact values
`rad(60) / deg(...)`	degrees to radians / radians to degrees
`acos(value)`	the angle from a cosine value (reverse)
`sqrt / Rational`	exact radicals and fractions
`S.evalf(4)`	an exact value as a decimal (n digits)

Practice Problems

Solve each problem with pen and paper first, then write Python code in an empty Colab cell to check your answer. SymPy is handy for exact values, and the math module for decimals.

Problem 1

A right triangle has sides 5, 12, and 13 (hypotenuse 13). Let A be the angle between the side of length 12 and the hypotenuse. Find sin A, cos A, and tan A.

Problem 2

From a point 50 m away from a building, the angle of elevation to its top is 40°. Taking the eye height to be 1.5 m, find the height of the building to one decimal place.

Problem 3

Let θ be an acute angle with $\sin \theta = \frac{5}{13}$. Find $\cos \theta$ and $\tan \theta$.

Problem 4

Find the values of sin 150°, cos 120°, and tan 135°.

Problem 5

In triangle ABC, a = 6, A = 45°, and B = 60°. Find the side b.

Problem 6

In triangle ABC, a = 5 and A = 150°. Find the radius R of the circumscribed circle.

Problem 7

In triangle ABC, b = 5, c = 3, and A = 120°. Find the side a.

Problem 8

A triangle has the three sides a = 6, b = 10, c = 14. Find the size of its largest angle (the angle C facing c).

Problem 9

Find the area S of the triangle with b = 6, c = 10, and A = 30°.

Problem 10

Two sides of a triangular plot of land are 25 m and 32 m, and the included angle is 60°. Find the area of the plot, both as an exact value and as a decimal (4 digits).

Answers and Explanations

Answer 1

Seen from A, the opposite side is 5, the adjacent side is 12, and the hypotenuse is 13. So $\sin A = \frac{5}{13}$, $\cos A = \frac{12}{13}$, and $\tan A = \frac{5}{12}$.

```
from sympy import sqrt, Rational
print(sqrt(5**2 + 12**2))   # check the hypotenuse
print(Rational(5, 13))      # sin A
print(Rational(12, 13))     # cos A
print(Rational(5, 12))      # tan A
```

Output ▸

```
13
5/13
12/13
5/12
```

Answer 2

Let h be the height from eye level to the top: h = 50 × tan 40°. Adding the eye height, the building is about 43.5 m tall.

```
import math
h = 50 * math.tan(math.radians(40))
print(round(h, 1))
print(round(h + 1.5, 1))
```

Output ▸
```
42.0
43.5
```

Answer 3

$\cos^2\theta = 1 - \frac{25}{169} = \frac{144}{169}$. Acute, so $\cos\theta = \frac{12}{13}$. $\tan\theta = \frac{5}{13} \div \frac{12}{13} = \frac{5}{12}$ (the same values as the 5 : 12 : 13 triangle of Problem 1).

```
from sympy import sqrt, Rational
cosA = sqrt(1 - Rational(5, 13)**2)
print(cosA)
print(Rational(5, 13) / cosA)
```

Output ▸
```
12/13
5/12
```

Answer 4

Convert to acute angles with $180° - \theta$. $\sin 150° = \sin 30° = \frac{1}{2}$. $\cos 120° = -\cos 60° = -\frac{1}{2}$. $\tan 135° = -\tan 45° = -1$.

```
from sympy import sin, cos, tan, rad
print(sin(rad(150)))
print(cos(rad(120)))
print(tan(rad(135)))
```

Output ▸
```
1/2
-1/2
-1
```

Answer 5

From the law of sines $\frac{6}{\sin 45°} = \frac{b}{\sin 60°}$: $b = \frac{6\sin 60°}{\sin 45°} = 6 \times \frac{\sqrt{3}}{2} \div \frac{\sqrt{2}}{2} = \frac{6\sqrt{3}}{\sqrt{2}} = 3\sqrt{6}$ (tidied by rationalizing the denominator).

```
from sympy import sin, rad
b = 6 * sin(rad(60)) / sin(rad(45))
print(b)
```

Output ▸
```
3*sqrt(6)
```

Answer 6

The law of sines works for obtuse angles as it stands. $\sin 150° = \frac{1}{2}$, so $2R = \frac{5}{1/2} = 10$ and $R = 5$.

```
from sympy import sin, rad
R = 5 / (2 * sin(rad(150)))
print(R)
```

Output ▸
```
5
```

Answer 7

Substitute into the law of cosines. $\cos 120° = -\frac{1}{2}$, so $a^2 = 25 + 9 - 2 \times 5 \times 3 \times (-\frac{1}{2}) = 34 + 15 = 49$, and $a = 7$. Note that for an obtuse angle, the term $-2bc \cos A$ turns into an addition.

```
from sympy import sqrt, cos, rad
a = sqrt(5**2 + 3**2 - 2*5*3*cos(rad(120)))
print(a)
```

Output ▸
```
7
```

Answer 8

The largest angle faces the longest side. From the law of cosines, $\cos C = \frac{36 + 100 - 196}{2 \times 6 \times 10} = \frac{-60}{120} = -\frac{1}{2}$. The angle with $\cos C = -\frac{1}{2}$ is $C = 120°$ (a negative cosine was our sign of an obtuse angle).

```
from sympy import acos, deg, Rational
cosC = Rational(6**2 + 10**2 - 14**2, 2 * 6 * 10)
print(cosC)
print(deg(acos(cosC)))
```

```
Output ▸
-1/2
120
```

Answer 9

Substitute into the area formula: $S = \frac{1}{2} \times 6 \times 10 \times \sin 30° = 30 \times \frac{1}{2} = 15$.

```
from sympy import sin, rad, Rational
S = Rational(1, 2) * 6 * 10 * sin(rad(30))
print(S)
```

```
Output ▸
15
```

Answer 10

$S = \frac{1}{2} \times 25 \times 32 \times \sin 60° = 400 \times \frac{\sqrt{3}}{2} = 200\sqrt{3}$ — about 346.4 m² as a decimal.

```
from sympy import sin, rad, Rational
S = Rational(1, 2) * 25 * 32 * sin(rad(60))
print(S)
print(S.evalf(4))
```

```
Output ▸
200*sqrt(3)
346.4
```

Chapter 5
Making Counting Systematic: The Number of Cases

The principle of counting

The addition rule

The multiplication rule

Permutations

Combinations

Counting with repetition

Chapter 5 Making Counting Systematic: The Number of Cases

What You Will Learn

The outcomes of coin tosses, outfits to wear, committees to pick, the total number of passwords — situations that ask "how many ways?" are everywhere around us. This "how many ways" is called the **number of cases**. Small numbers can be counted on your fingers, but as the numbers grow they quickly slip out of reach. So in this chapter, we make counting itself systematic.

The toolkit is simple. The foundation is the principle of counting: no omissions, no double counting. On it stand two pillars, the **addition rule** and the **multiplication rule**, and from them we derive two powerful formulas: **permutations** (counting arrangements) and **combinations** (counting selections). Finally, we extend to counting with identical objects and with repetition.

This chapter's Python brings a strong ally. With the toolbox called itertools, the computer can generate every single case and actually count them. The answer from a formula and the answer from listing everything agree exactly — you will experience that moment again and again.

By the end of this chapter, you will be able to:

- count without omissions or double counting, using tree diagrams and tables
- use the addition rule and the multiplication rule, each in the right situation
- compute the number of arrangements with permutations ${}_{\mathrm{n}}P_{\mathrm{r}}$ and factorials n!
- compute the number of selections with combinations ${}_{\mathrm{n}}C_{\mathrm{r}}$
- count permutations with identical objects, and arrangements with repetition
- check counts by full enumeration with itertools and the math module

5.1 The Principle of Counting: No Omissions, No Double Counting

Recap (Level 1B, Chapter 12) Probability expresses "how likely" as a number. The base of that calculation is the number of cases: "how many ways in all" and "how many of them." If you train your counting in this chapter, the probability in Chapter 6 becomes much easier to handle.

The total number of ways something can happen is called the **number of cases**. When counting, there is only one motto:

no omissions, no double counting.

The classic way to count with no omissions and no double counting is the **tree diagram**. Following the branches in order, every case appears exactly once.

Term **Number of cases** The total number of ways something can happen. The golden rule of counting is "no omissions, no double counting." Tree diagrams and tables let you count mechanically.

Example 5.1

A coin is tossed 3 times. How many sequences of heads and tails are there in all? Count with a tree diagram.

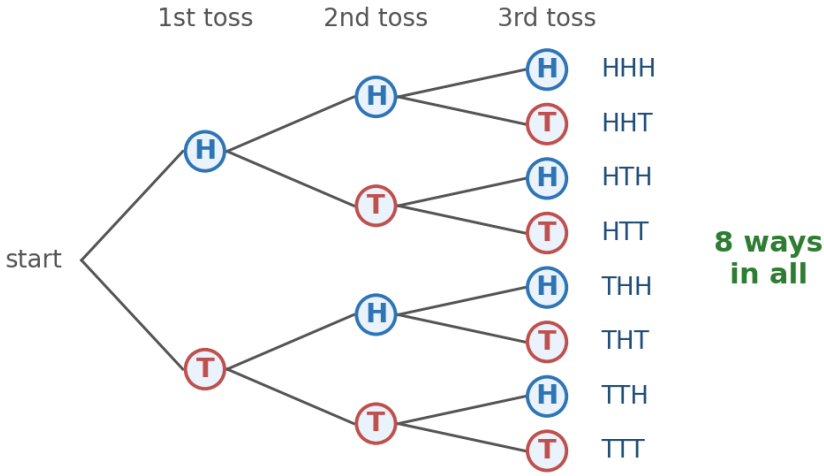


Figure 5.1 The tree diagram for 3 coin tosses. Following the branches, all 8 ways appear with no omissions and no double counting.

With pen and paper Follow the tree in Figure 5.1. The 1st toss gives two branches, heads (H) or tails (T). Each branch splits into two again at the 2nd toss, and again at the 3rd. The branches end in $2 \times 2 \times 2 = 8$ tips, so there are 8 ways in all.

Check it with Python Here comes a powerful toolbox: **itertools**, a standard library for "generating every case." Its product('HT', repeat=3) builds every sequence of 3 symbols drawn from 'H' and 'T' with repeats allowed — that is, it turns all the branch tips of the tree diagram into a list.

```
from itertools import product
patterns = list(product('HT', repeat=3))
print(' '.join(''.join(p) for p in patterns))
print(len(patterns))              # how many ways
```

Output ▸

```
HHH HHT HTH HTT THH THT TTH TTT
8
```

The same 8 ways as the tree diagram, in the same order. len counts them: 8. **"Check an answer counted by formula or diagram against a full enumeration"** — that is how Python is used in this chapter. The harder a count is by hand, the more dependable the computer's enumeration becomes.

5.2 The Addition Rule: Split into Cases, Then Add

When what we want to count splits cleanly into "cases where A happens" and "cases where B happens," and **A and B can never happen at the same time**, the total number of cases is found by adding. This is the addition rule.

> **Term Addition rule** When two cases A and B cannot happen at the same time (no overlap), the number of cases where A or B happens is (the number of cases for A) + (the number of cases for B).

Example 5.2

Two dice are rolled. In how many ways can the sum of the two faces be 5 or 10?

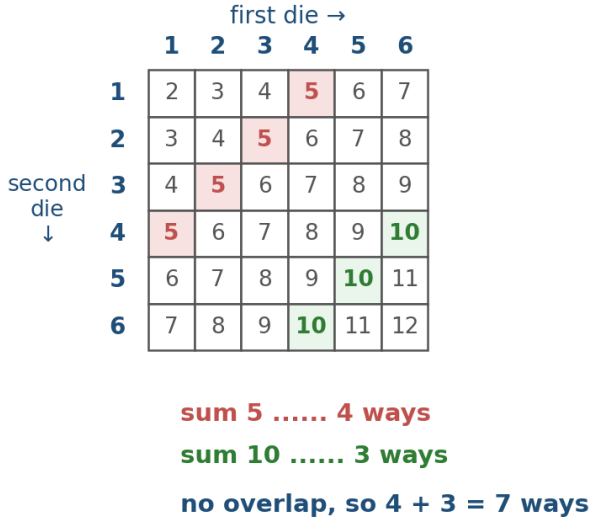


Figure 5.2 The table of sums. The cells for sum 5 and the cells for sum 10 do not overlap, so they may be added.

With pen and paper Count in the table of Figure 5.2. The sum is 5 for (1, 4), (2, 3), (3, 2), (4, 1) — 4 ways. The sum is 10 for (4, 6), (5, 5), (6, 4) — 3 ways. A sum cannot be 5 and 10 at the same time, so by the addition rule there are 4 + 3 = 7 ways.

Check it with Python We build all 36 pairs of faces with product, then pick out the ones that match each condition using list comprehensions (the truth-set tool from Chapter 2).

```
from itertools import product
pairs = list(product(range(1, 7), repeat=2))
sum5 = [p for p in pairs if sum(p) == 5]
sum10 = [p for p in pairs if sum(p) == 10]
print(len(sum5))
print(len(sum10))
print(len(sum5) + len(sum10))   # the addition rule
```

Output ▸

```
4
3
7
```

4 ways and 3 ways, 7 in total — matching the table. sum(p) computes the total of a pair. A caution: if the two cases do overlap, adding them double-counts. Checking "is there no overlap?" first is the proper manner of the

addition rule.

5.3 The Multiplication Rule: Split into Stages, Then Multiply

Now for "choosing in sequence." When A can happen in m ways and, **for each of those**, B can happen in n ways, the number of cases where both A and B happen is m × n. This is the multiplication rule. The 2 × 2 × 2 = 8 branches of the tree in Example 5.1 were exactly the multiplication rule at work.

> **Term** **Multiplication rule** When A can happen in m ways and, for each of those, B can happen in n ways, the number of cases where A and B both happen is m × n. With three or more stages, keep multiplying in the same way.

Example 5.3

There are 3 shirts (white, blue, black) and 2 pairs of pants (long, short). How many shirt-and-pants outfits are there?

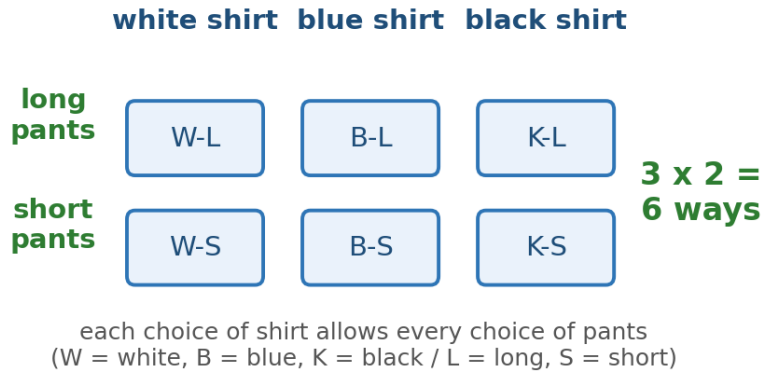


Figure 5.3 The multiplication rule. Each of the 3 choices of shirt allows 2 choices of pants.

With pen and paper As in Figure 5.3, there are 3 ways to choose the shirt, and whichever you choose, there are 2 ways to choose the pants. By the multiplication rule, 3 × 2 = 6 outfits.

Check it with Python product can also build "every pair" from two lists: product(list of shirts, list of pants) enumerates all the outfits.

```
from itertools import product
shirts = ['white', 'blue', 'black']
pants = ['long', 'short']
combos = list(product(shirts, pants))
for k in combos:
    print('-'.join(k))
print(len(combos))            # 3 x 2: multiplication rule
```

Output ▸

```
white-long
white-short
blue-long
blue-short
black-long
black-short
6
```

All 6 outfits are listed — the same grid as Figure 5.3. The multiplication rule applies when "whichever you choose, the number of next choices is the same." That condition, too, can be checked by eye once everything is enumerated.

Example 5.4

There are four number cards: 1, 2, 3, 4, one of each. Three of them are arranged to form a 3-digit integer. How many integers can be made?

With pen and paper Decide the hundreds, tens, and ones places in order. The hundreds place can be any of the 4 cards: 4 ways. The tens place takes one of the remaining 3 cards: 3 ways. The ones place: 2 ways. By the multiplication rule, $4 \times 3 \times 2 = 24$ integers. The point is that each card can be used only once, so the number of choices drops by 1 at each stage.

Check it with Python itertools' **permutations** builds every arrangement "without using the same object twice." permutations('1234', 3) hands us every pattern of choosing 3 of the 4 cards and arranging them.

```
from itertools import permutations
nums = list(permutations('1234', 3))
print(len(nums))
print(''.join(nums[0]), ''.join(nums[-1]))
```

```
Output ▸
24
123 432
```

24 — a match. The first pattern is 123, the last 432. This way of counting, "arrange what remains," has a name of its own: the permutation, the star of the next section.

5.4 Permutations: Choose and Arrange

An arrangement of r objects chosen from n different objects in a row is called a **permutation** of n objects taken r at a time, and the total number is written ${}_nP_r$ (P for permutation). As Example 5.4 showed, the multiplication rule computes it:

${}_nP_r = n(n-1)(n-2)\ \ldots\ (n-r+1)$ (multiply r factors, decreasing from n by 1 each time).

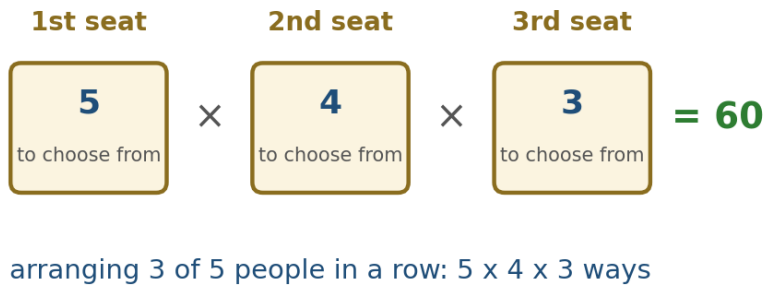


Figure 5.4 How permutations work. Each time a seat is filled, the number of people to choose from drops by 1.

In particular, the permutation of all n objects, ${}_nP_n = n \times (n-1) \times \ldots \times 2 \times 1$, has a symbol of its own: it is called **n factorial** and written n!.

> **Term Permutations and factorials** The number of permutations of n different objects taken r at a time is ${}_nP_r = n(n-1)\ \ldots\ (n-r+1)$. The number of arrangements of all n objects is $n! = n \times (n-1) \times \ldots \times 2 \times 1$ (n factorial).

Example 5.5

In how many ways can 3 of 5 people be chosen and arranged in a row? This

is ${}_5P_3$.

With pen and paper As in Figure 5.4, the 1st seat can be filled from 5 people, the 2nd from the remaining 4, and the 3rd from the remaining 3. By the multiplication rule, ${}_5P_3 = 5 \times 4 \times 3 = 60$ ways.

Check it with Python The math module has **math.perm(n, r)**, which computes a permutation in one stroke.

```
import math
print(5 * 4 * 3)          # the multiplication rule
print(math.perm(5, 3))    # the permutation function
```

Output ▸
```
60
60
```

Both give 60. The symbol ${}_5P_3$ may be new, but inside it is nothing other than the familiar multiplication rule.

Example 5.6

In how many ways can all 5 people A, B, C, D, E be arranged in a row?

With pen and paper All 5 people are arranged, so $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$ ways.

Check it with Python Writing permutations('ABCDE') without a count builds every arrangement of all five. The factorial is computed with **math.factorial(n)**.

```
from itertools import permutations
import math
# count by full enumeration
print(len(list(permutations('ABCDE'))))
print(math.factorial(5))        # compute 5!
```

Output ▸
```
120
120
```

The 120 from full enumeration and the factorial agree exactly. Five people give 120 ways — with ten people it would be 10! = 3,628,800. Factorials grow astonishingly fast (the column at the end of the chapter shows a surprising use of this "explosive power").

5.5 Combinations: Just Choosing — Order Does Not Matter

A permutation "chooses and arranges." But when picking committee members or forming a team, often you **only choose, and order does not matter**. A selection of r objects from n different objects, with order ignored, is called a **combination**, and the total number is written ${}_nC_r$ (C for combination).

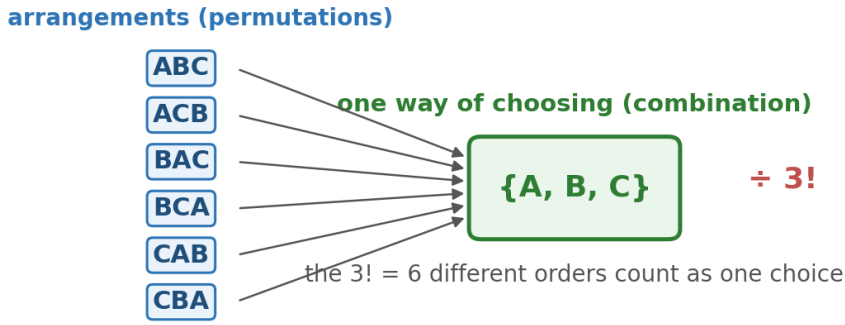


Figure 5.5 Permutations versus combinations. The 3! = 6 arrangements count as one and the same choice.

Look at Figure 5.5. When the three people A, B, C are chosen, there are 3! = 6 arrangements — ABC, ACB, ..., CBA — but as "the three people chosen" they are all the same one set {A, B, C}. So dividing the number of permutations by the r! rearrangements gives the number of combinations.

> **Term Combinations** The number of ways to choose r objects from n different objects, ignoring order, is ${}_nC_r = \frac{{}_nP_r}{r!}$ (divide the permutations by the r! rearrangements).

Example 5.7

In how many ways can 3 cleaning monitors be chosen from 5 people? This is ${}_5C_3$. The monitors have no order.

With pen and paper Counting arrangements first gives ${}_5P_3 = 60$ ways. Monitors have no order, so each group of 3 is counted 3! = 6 times. Therefore ${}_5C_3 = 60 \div 6 = 10$ ways.

Check it with Python Combinations are computed with **math.comb(n, r)** and enumerated with itertools' **combinations**. We

also check the division ${}_5P_3 \div 3!$.

```
import math
from itertools import combinations
# comb function, enumeration, and 5P3 // 3!
print(math.comb(5, 3))
print(len(list(combinations('ABCDE', 3))))
print(math.perm(5, 3) // math.factorial(3))
```

Output ▸

```
10
10
10
```

All three methods agree on 10. The split between permutations and combinations comes down to one point: **whether order matters**. Arranging → P; just choosing → C.

Example 5.8

From a group of 4 boys and 3 girls, a committee of 2 boys and 1 girl is chosen. How many ways are there?

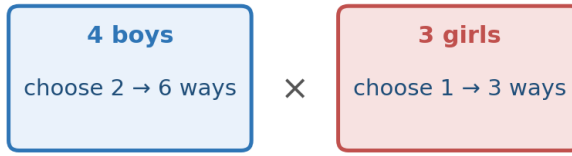


Figure 5.6 Count each group with combinations, then join them with the multiplication rule.

With pen and paper The boys can be chosen in ${}_4C_2 = 6$ ways and the girls in ${}_3C_1 = 3$ ways. For each choice of boys there is every choice of girls, so by the multiplication rule, $6 \times 3 = 18$ ways (Figure 5.6).

Check it with Python

```
import math
print(math.comb(4, 2))      # ways to choose the boys
print(math.comb(3, 1))      # ways to choose the girls
print(math.comb(4, 2) * math.comb(3, 1))   # multiply
```

Output ▸

```
6
3
18
```

18 ways. Building parts with combinations and assembling them with the multiplication rule — a classic pattern where this chapter's tools join into one.

5.6 Counting with Repetition

To finish, we handle two situations where the formulas so far cannot be used as they stand. In both, "repetition" is the key.

Permutations with Identical Objects

When arranging 5 letters that **include identical ones**, such as A, A, B, B, C, the naive count 5! = 120 is too many: swapping the two A's gives an arrangement that looks exactly the same (Figure 5.7).

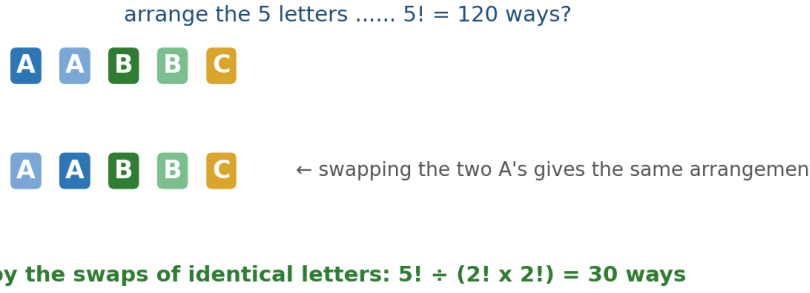


Figure 5.7 Permutations with identical objects. Swapping identical letters counts as the same arrangement.

Term **Permutations with identical objects** When n objects include p identical ones, q identical ones, and so on, the number of arrangements is $\frac{n!}{p! \times q! \times \ldots}$ (divide out the rearrangements of identical objects).

Example 5.9

In how many ways can the 5 letters A, A, B, B, C be arranged in a row?

With pen and paper Divide the total 5! by the 2! swaps of the A's and the 2! swaps of the B's:

$$\frac{5!}{2! \times 2!} = \frac{120}{4} = 30 \text{ ways.}$$

Check it with Python Full enumeration confirms it. permutations('AABBC') distinguishes the positions of the letters and builds 120 arrangements, but putting them into a **set** (Chapter 2) merges the ones that look the same into one.

```
import math
from itertools import permutations
numer = math.factorial(5)                  # numerator 5!
denom = math.factorial(2) * math.factorial(2)
print(numer // denom)
# remove duplicates with set, then count
print(len(set(permutations('AABBC'))))
```

Output ▸

```
30
30
```

The formula's 30 and the duplicate-free enumeration's 30 agree. The set property "identical things count once" (Chapter 2) earns its keep here as a counting checker.

Arrangements with Repetition

Now the opposite case: arrangements where **the same object may be chosen any number of times**. Unlike the cards of Example 5.4, the choices never run out, so the same number is multiplied each time. Arranging r symbols from n kinds gives $n \times n \times \ldots \times n = n^r$ ways. These are called **arrangements (permutations) with repetition**.

Example 5.10

A quiz has 4 questions, each answered True or False. How many ways are there to answer the whole quiz?

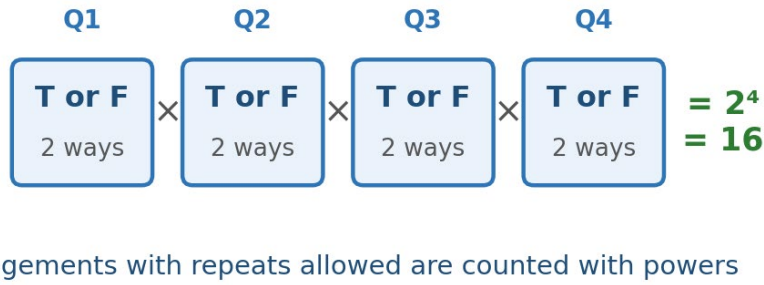


Figure 5.8 Arrangements with repetition. Each question has 2 choices, so multiply 2 four times.

With pen and paper Each question has 2 answers, T or F, and the choices never run out. By the multiplication rule, $2 \times 2 \times 2 \times 2 = 2^4 = 16$ ways (Figure 5.8).

Check it with Python Powers are computed with the familiar **. For enumeration, the repeat of product — the very tool of Example 5.1 — is exactly arrangements with repetition.

```
from itertools import product
# 2 to the 4th, and full enumeration with product
print(2 ** 4)
print(len(list(product('TF', repeat=4))))
```

Output ▸

```
16
16
```

16 — a match. In fact, the 3 coin tosses of Example 5.1 ($2^3 = 8$ ways) were arrangements with repetition too. The tree diagram that opened the chapter receives its name at the very end.

Column Passwords and Combinatorial Explosion

A 4-digit PIN is an arrangement with repetition of the digits 0–9: $10^4 = 10{,}000$ ways — a number a computer can try exhaustively in an instant. What about an 8-character password using uppercase letters, lowercase letters, and digits (62 kinds in all)? 62^8 is about 218 trillion — even at 100 million attempts per second, more than 25 days. This phenomenon, where the total swells multiplicatively with every added character, is called combinatorial explosion. Usually it is the nuisance that makes things "too

many to count" — but in the world of passwords, that explosive power is the very shield protecting our information. Behind every request to "please use a longer password" hides the arrangement-with-repetition calculation you learned in this chapter.

Chapter Summary

- Count the number of cases with "no omissions, no double counting." Tree diagrams and tables are the basic tools.
- Addition rule: cases with no overlap are added.
- Multiplication rule: when choosing in stages, multiply the number of cases at each stage.
- A permutation ${}_nP_r$ counts "choose and arrange": multiply r factors decreasing from n by 1. Arranging all n objects gives the factorial n!.
- A combination ${}_nC_r$ counts "just choosing": divide the permutations by r!. Whether order matters is the split between P and C.
- For identical objects, divide n! by the rearrangements of the identical ones. For repeats allowed, the count is n^r.
- Every formula's answer can be checked against a full enumeration with itertools.

Table 5.1 Python tools used in this chapter

Python tool	What it does
`product(..., repeat=r)`	build every sequence (repetition allowed)
`permutations(s, r)`	build every permutation (arrangements)
`combinations(s, r)`	build every combination (selections)
`math.factorial(n)`	compute the factorial n!
`math.perm(n, r)`	compute the number of permutations
`math.comb(n, r)`	compute the number of combinations
`len(...) / set(...)`	count items / count without duplicates

Practice Problems

Solve each problem with pen and paper first, then write Python code in an empty Colab cell to check your answer. Trying both the formula and the full enumeration with itertools is recommended.

Problem 1

Two people play one round of rock-paper-scissors. In how many ways can the two hands be shown? (There are 3 hands: rock, paper, scissors.)

Problem 2

Two dice are rolled. In how many ways can the sum of the faces be 4 or 11?

Problem 3

There are 2 hats, 3 shirts, and 2 pairs of shoes. In how many ways can one of each be chosen?

Problem 4

From the number cards 1, 2, 3, 4, 5, three are arranged to form a 3-digit integer. How many integers can be made?

Problem 5

Compute ${}_7P_2$. Also confirm that it equals $\frac{7!}{5!}$.

Problem 6

In how many ways can all 6 people be arranged in a row?

Problem 7

In how many ways can 3 representatives be chosen from 8 people? This is ${}_8C_3$.

Problem 8

From 5 boys and 4 girls, a committee of 2 boys and 2 girls is chosen. How many ways are there?

Problem 9

In how many ways can the 5 letters A, A, A, B, B be arranged in a row?

Problem 10

Three kinds of stamps, A, B, C, are arranged in a row of 4, with repeats allowed. How many arrangements are there?

Answers and Explanations

Answer 1

The first person's hand: 3 ways; for each, the second person's hand: 3 ways. By the multiplication rule, $3 \times 3 = 9$ ways.

```
from itertools import product
hands = list(product('RPS', repeat=2))
print(len(hands))
```

Output ▸
```
9
```

Answer 2

The sum is 4 for (1, 3), (2, 2), (3, 1): 3 ways. The sum is 11 for (5, 6), (6, 5): 2 ways. No overlap, so by the addition rule, 3 + 2 = 5 ways.

```
from itertools import product
pairs = list(product(range(1, 7), repeat=2))
print(len([p for p in pairs if sum(p) == 4]))
print(len([p for p in pairs if sum(p) == 11]))
print(3 + 2)
```

Output ▸
```
3
2
5
```

Answer 3

By the multiplication rule, 2 × 3 × 2 = 12 ways. product can build every combination from three or more lists as well.

```
from itertools import product
print(2 * 3 * 2)
print(len(list(product('AB', 'XYZ', 'PQ'))))
```

Output ▸
```
12
12
```

Answer 4

Hundreds place: 5 ways × tens place: 4 ways × ones place: 3 ways, that is, ${}_5P_3 = 60$ integers.

```
import math
from itertools import permutations
print(math.perm(5, 3))
print(len(list(permutations('12345', 3))))
```

```
Output ▸
60
60
```

Answer 5

${}_7P_2 = 7 \times 6 = 42$. Also $\frac{7!}{5!} = \frac{7 \times 6 \times 5!}{5!} = 7 \times 6$, so they are indeed equal (the 5! in numerator and denominator cancels).

```
import math
print(math.perm(7, 2))
print(math.factorial(7) // math.factorial(5))
```

```
Output ▸
42
42
```

Answer 6

A permutation of all 6 people: 6! = 720 ways.

```
import math
from itertools import permutations
print(math.factorial(6))
print(len(list(permutations('ABCDEF'))))
```

```
Output ▸
720
720
```

Answer 7

Representatives have no order, so this is a combination: ${}_8C_3 = \frac{8 \times 7 \times 6}{3!} = \frac{336}{6} = 56$ ways.

```
import math
from itertools import combinations
print(math.comb(8, 3))
print(len(list(combinations('ABCDEFGH', 3))))
```

```
Output ▸
56
56
```

Answer 8

The boys can be chosen in ${}_5C_2 = 10$ ways and the girls in ${}_4C_2 = 6$ ways. By

the multiplication rule, $10 \times 6 = 60$ ways.

```
import math
print(math.comb(5, 2))
print(math.comb(4, 2))
print(math.comb(5, 2) * math.comb(4, 2))
```

Output ▸

```
10
6
60
```

Answer 9

Permutations with identical objects: $\frac{5!}{3! \times 2!} = \frac{120}{12} = 10$ ways.

```
import math
from itertools import permutations
numer = math.factorial(5)
denom = math.factorial(3) * math.factorial(2)
print(numer // denom)
print(len(set(permutations('AAABB'))))
```

Output ▸

```
10
10
```

Answer 10

Repeats are allowed, so these are arrangements with repetition: $3^4 = 81$ ways.

```
from itertools import product
print(3 ** 4)
print(len(list(product('ABC', repeat=4))))
```

Output ▸

```
81
81
```

Chapter 6
Handling Chance with Formulas: Probability

The basics of probability

Complementary events

Mutually exclusive events

Independent trials

Repeated trials

Conditional probability

Expected value

Chapter 6 Handling Chance with Formulas: Probability

What You Will Learn

Tomorrow's chance of rain, the odds of winning a lottery, the likelihood of getting the item you want in a game — nobody knows the future. And yet, "how likely" can be expressed as a number, compared, and computed. That is **probability**.

In fact, the preparation is already done: the counting you trained in the previous chapter is the very heart of probability. Once you can count "how many ways in all" and "how many of them," probability is nothing more than their ratio. In this chapter we start from the basics and add tools one by one — **complementary events**, **mutually exclusive events**, **independent trials**, **repeated trials**, and **conditional probability** — arriving at last at the **expected value**, which tells us "how much on average."

This chapter's Python gains a new companion: random. Rolling a die 100,000 times is the work of an instant for a computer. Watch with your own eyes as the probability computed by theory and the result of the experiment land right on top of each other.

By the end of this chapter, you will be able to:

- compute probabilities as ratios of numbers of cases, based on "equally likely"
- use complements to find "at least one" probabilities quickly
- add probabilities of mutually exclusive events with the addition rule
- multiply probabilities of independent trials with the multiplication rule
- compute repeated-trial probabilities using ${}_{\mathrm{n}}C_{\mathrm{r}}$
- handle situations where information changes probability, with conditional probability
- compute expected values and check them with random simulations

6.1 The Basics of Probability: Measuring Likelihood with Numbers

Recap (Level 1B, Chapter 12) Probability expresses likelihood as a number from 0 to 1. We also learned the view that the relative frequency (the proportion) over many repetitions approaches the probability. This chapter systematizes how to compute that probability by counting.

No face of a die comes up more easily than another. When every case can be expected to occur to the same degree like this, the cases are said to be **equally likely**. Then the probability of an event A (the outcome we care about) is defined as a ratio of numbers of cases.

Term Probability When all possible cases are equally likely, the probability of an event A is $P(A) = \frac{number\ of\ cases\ for\ A}{number\ of\ all\ possible\ cases}$. A probability is always a number between 0 and 1.

Example 6.1

A die is rolled once. Find the probability that an even face comes up.

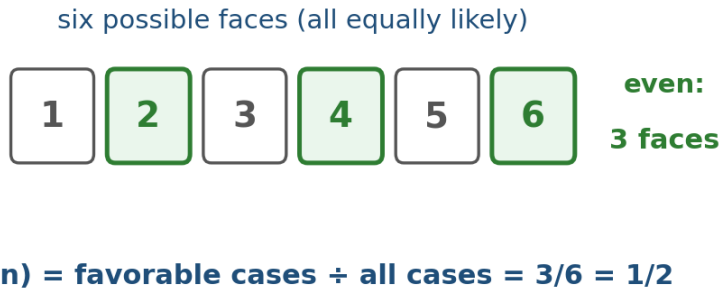


Figure 6.1 Probability is a ratio of cases. Of the 6 faces, 3 are even, so 3/6 = 1/2.

With pen and paper The possible faces are 1 to 6 — six equally likely cases. Of these, the even ones are 2, 4, 6: three cases (Figure 6.1). So P(even) = $\frac{3}{6} = \frac{1}{2}$.

Check it with Python First we compute the theoretical value; then a new tool, **random**, runs the experiment "does it really come up about half the time?" random is the standard library for random numbers:

random.randint(1, 6) returns a random integer from 1 to 6, just like a die. A computer's random numbers are pseudo-random — made by calculation — so fixing the "seed" with random.seed(0) reproduces the same sequence every time (we fix the seed so your output matches the book's).

```
from sympy import Rational
import random
faces = [1, 2, 3, 4, 5, 6]
evens = [x for x in faces if x % 2 == 0]
print(Rational(len(evens), len(faces)))   # theory
random.seed(0)            # fix the random seed
n = 100000
count = 0
for _ in range(n):
    if random.randint(1, 6) % 2 == 0:
        count += 1
print(count / n)          # 100,000-roll experiment
```

Output ▸

```
1/2
0.4991
```

Theory says 1/2; the 100,000-roll experiment gives 0.4991 — almost a perfect match. As Figure 6.2 shows, the more rolls, the more the experimental proportion is drawn toward the theoretical value. The "probability found by counting" and the "proportion in experiments" point to the same thing — that is what the number called probability means.

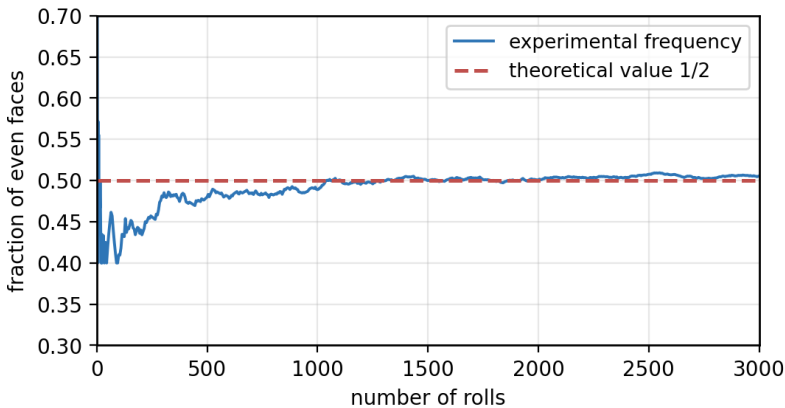


Figure 6.2 Number of rolls versus the fraction of even faces. As the rolls increase, it approaches the theoretical value 1/2.

Example 6.2

Two dice are rolled. Find the probability that the sum of the faces is 5.

With pen and paper The possible pairs of faces number 6 × 6 = 36 (the multiplication rule of Chapter 5). The sum is 5 for (1, 4), (2, 3), (3, 2), (4, 1) — 4 cases (Chapter 5, Example 5.2). So P(sum is 5) = $\frac{4}{36} = \frac{1}{9}$.

Check it with Python Chapter 5's "enumerate and count" becomes, as it stands, the computation of a probability: itertools counts both the numerator and the denominator.

```
from itertools import product
from sympy import Rational
pairs = list(product(range(1, 7), repeat=2))
hits = [p for p in pairs if sum(p) == 5]
print(len(hits), len(pairs))
print(Rational(len(hits), len(pairs)))
```

```
Output ▸
4 36
1/9
```

4 ÷ 36 = 1/9. **"Count, then divide"** — the counting of the previous chapter has become the numerator and denominator of a probability. Every tool from Chapter 5 can be used again in this chapter.

6.2 Complements and Mutually Exclusive Events: Probability by Subtracting and Adding

Complements: Attack from the "Not" Side

For an event A, the event "A does not happen" is called the **complement** of A (in symbols, it is written with a bar drawn over the A). Exactly one of A and "not A" always happens, so their probabilities add to 1 — and a handy formula is born (Figure 6.3).

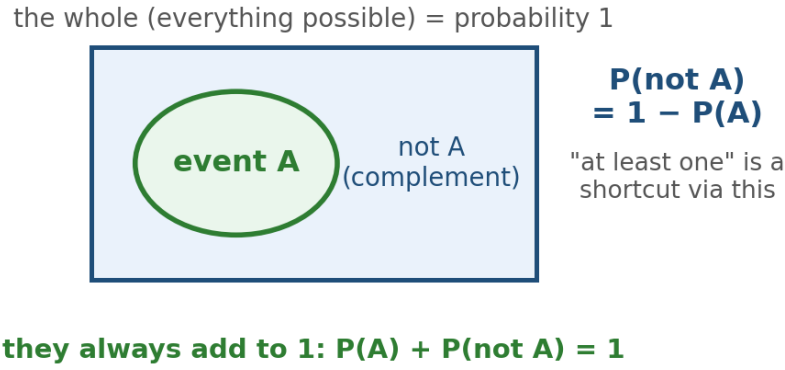


Figure 6.3 The complement. Subtracting P(A) from the total probability 1 gives the probability of "not A."

Term **Complementary events** The probability of the complement of A (the event that A does not happen) is $P(not\ A) = 1 - P(A)$. For "at least one ..." probabilities, the complement "not even once ..." is the shortcut.

Example 6.3

A coin is tossed 3 times. Find the probability that heads comes up at least once.

With pen and paper Counting "at least one heads" head-on needs cases for 1, 2, and 3 heads. So we use the complement: the opposite of "at least one heads" is "no heads at all" = "tails all 3 times," which is the single case (T, T, T). The total is $2^3 = 8$ cases, so P(all tails) = $\frac{1}{8}$. The probability we want is $1 - \frac{1}{8} = \frac{7}{8}$.

Check it with Python Enumerate, count "all tails," and subtract from 1.

```
from itertools import product
from sympy import Rational
allp = list(product('HT', repeat=3))
tails3 = [p for p in allp if p == ('T', 'T', 'T')]
print(Rational(len(tails3), len(allp)))     # P(all tails)
print(1 - Rational(len(tails3), len(allp)))  # complement
```

Output ▸

```
1/8
7/8
```

7/8. Instead of counting three separate cases, one subtraction. When you hear "at least," reach for the complement — a standard move in probability.

Mutually Exclusive Events: If They Cannot Overlap, Add

When two events A and B can never happen at the same time, A and B are said to be **mutually exclusive**. For mutually exclusive events, the probability of "A or B" is found by simple addition (Figure 6.4) — the probability version of Chapter 5's addition rule.

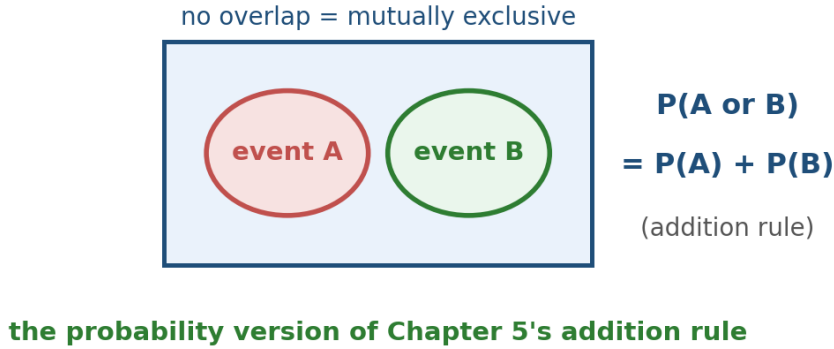


Figure 6.4 Mutually exclusive events. With no overlap, probabilities may simply be added (the addition rule).

Term **Mutually exclusive events and the addition rule** When events A and B are mutually exclusive (cannot happen at the same time), $P(A \text{ or } B) = P(A) + P(B)$ holds. When they do overlap, adding as-is double-counts — beware.

Example 6.4

Two dice are rolled. Find the probability that the sum of the faces is 5 or 10.

With pen and paper The sum is 5 in 4 cases and 10 in 3 cases (Chapter 5, Example 5.2). A sum cannot be 5 and 10 at the same time, so the events are mutually exclusive. By the addition rule, $P = \frac{4}{36} + \frac{3}{36} = \frac{7}{36}$.

Check it with Python

```
from itertools import product
from sympy import Rational
pairs = list(product(range(1, 7), repeat=2))
sum5 = [p for p in pairs if sum(p) == 5]
sum10 = [p for p in pairs if sum(p) == 10]
print(Rational(len(sum5), 36) + Rational(len(sum10), 36))
```

Output ▸
```
7/36
```

7/36. The "4 + 3 = 7 ways" we counted with Chapter 5's addition rule, divided by 36, has become the addition rule for probabilities.

6.3 Independent Trials: Probabilities Joined by Multiplication

Tossing a coin, rolling a die — an experiment or observation whose result is decided by chance is called a **trial**. When the result of one trial has no influence at all on the result of another, the two trials are said to be **independent**. For independent trials, probabilities join by multiplication.

Term **Independent trials and the multiplication rule** When two trials are independent, the probability that A happens in trial 1 and B happens in trial 2 is $P(A \text{ and } B) = P(A) \times P(B)$ — the probability version of Chapter 5's multiplication rule.

Example 6.5

A coin is tossed once and a die is rolled once. Find the probability that the coin shows heads and the die shows 6.

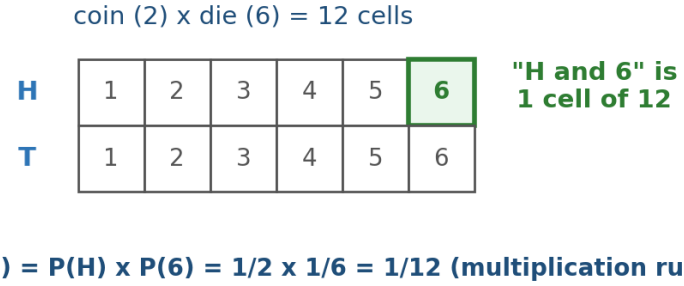


Figure 6.5 Independent trials. Of the 12 cells of coin × die, "H and 6" is exactly 1 cell.

With pen and paper The coin's result does not affect the die, so the two trials are independent. By the multiplication rule, $P = \frac{1}{2} \times \frac{1}{6} = \frac{1}{12}$. In the 12-cell table of Figure 6.5, "H and 6" takes up exactly one cell.

Check it with Python We check both ways: full enumeration (build all 12 cells) and the multiplication.

```
from itertools import product
from sympy import Rational
combos = list(product('HT', [1, 2, 3, 4, 5, 6]))
hits = [k for k in combos if k == ('H', 6)]
print(Rational(len(hits), len(combos)))
print(Rational(1, 2) * Rational(1, 6))
```

Output ▸

```
1/12
1/12
```

Both give 1/12. One multiplication does the job — the power of the multiplication rule — but never forget the condition: only when the trials are independent. When they influence each other, conditional probability (Section 6.5) takes the stage.

6.4 Repeated Trials: Trying the Same Thing Again and Again

Tossing a coin 5 times, rolling a die 3 times — repeating the same trial independently several times is called **repeated trials**. In "the probability of succeeding exactly r times out of n," Chapter 5's combination ${}_nC_r$ makes its appearance.

Consider tossing a coin 5 times and getting exactly 3 heads (Figure 6.6). HHHTT, HTHTH, ... — every path has the same probability $(\frac{1}{2})^3 \times (\frac{1}{2})^2$. And the number of paths is the number of ways to choose the positions of the 3 heads among the 5 tosses: ${}_5C_3 = 10$. So the probability is "the probability of one path × the number of paths."

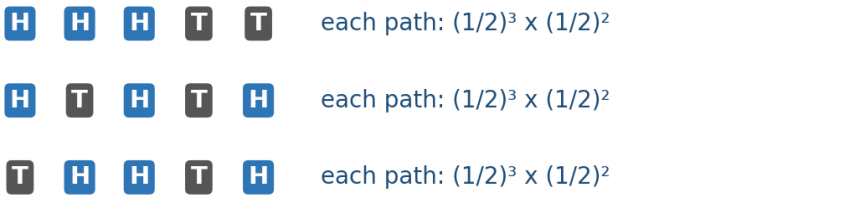


Figure 6.6 Repeated trials. Every path has the same probability, so multiply by the number of paths (a combination).

Term **Probability of repeated trials** Let p be the probability that event A happens in one trial. When the trial is repeated n times, the probability that A happens exactly r times is ${}_nC_r\, p^r\, (1-p)^{n-r}$ (the probability of one path times the number of paths).

Example 6.6

A coin is tossed 5 times. Find the probability that heads comes up exactly 3 times.

With pen and paper Put $p = \frac{1}{2}$, $n = 5$, $r = 3$ into the formula: $P = {}_5C_3 \times (\frac{1}{2})^3 \times (\frac{1}{2})^2 = 10 \times \frac{1}{32} = \frac{10}{32} = \frac{5}{16}$.

Check it with Python We check both the formula and the full enumeration (count the sequences with 3 heads among all 32). A tuple's count tallies the 'H's, just like the letter counting of Chapter 5.

```
import math
from itertools import product
from sympy import Rational
paths = math.comb(5, 3)              # number of paths
p = paths * Rational(1, 2)**3 * Rational(1, 2)**2
print(p)
allp = list(product('HT', repeat=5))
heads3 = [x for x in allp if x.count('H') == 3]
print(Rational(len(heads3), len(allp)))
```

```
Output ▸
5/16
5/16
```

Formula and enumeration agree on 5/16. Combinations (Chapter 5) × the multiplication rule (the previous section) — repeated trials are these two tools working together.

Example 6.7

A die is rolled 3 times. Find the probability that the face 1 comes up exactly once.

With pen and paper In one trial, the face 1 comes up with probability p = $\frac{1}{6}$ and does not with probability $\frac{5}{6}$. With n = 3, r = 1: P = ${}_3C_1 \times \frac{1}{6} \times (\frac{5}{6})^2 = 3 \times \frac{25}{216} = \frac{75}{216} = \frac{25}{72}$.

Check it with Python

```
import math
from itertools import product
from sympy import Rational
p = math.comb(3, 1) * Rational(1, 6) * Rational(5, 6)**2
print(p)
allp = list(product(range(1, 7), repeat=3))
ones1 = [x for x in allp if x.count(1) == 1]
print(Rational(len(ones1), len(allp)))
```

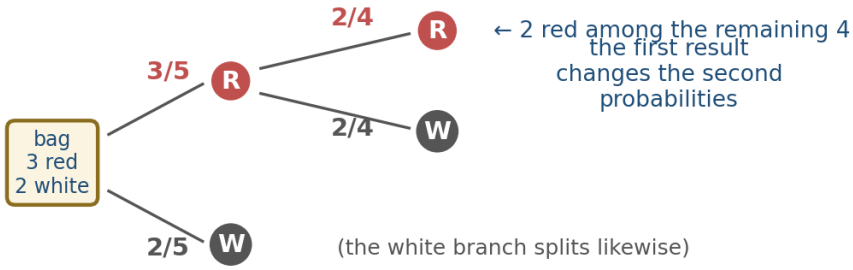
```
Output ▸
25/72
25/72
```

25/72 (about 0.35). The full enumeration of all 216 cases gives the same value, confirming the formula before our eyes. Even when success and failure have different probabilities (1/6 and 5/6), the formula applies as it stands.

6.5 Conditional Probability: Information Changes Probability

A bag holds 3 red balls and 2 white balls. One ball is drawn and, **without putting it back**, another is drawn. The probability for the second ball depends on the first result: if the first was red, the bag now holds 2 red and

2 white. The probability of an event B "given that an event A is known to have happened" is called the **conditional probability** of B given A, written P(B | A).



P(both red) = P(1st red) x P(2nd red | 1st red) = 3/5 x 2/4 = 3/10

Figure 6.7 Conditional probability. The first result changes the probabilities on the second branches.

Term Conditional probability and the multiplication rule Write P(B | A) for the conditional probability of B given that A has happened. Then $P(A \text{ and } B) = P(A) \times P(B \mid A)$ holds (the multiplication rule when trials influence each other). Independent trials are the special case where P(B | A) = P(B).

Example 6.8

A bag holds 3 red balls and 2 white balls. One ball is drawn and, without replacement, another is drawn. Find (1) the conditional probability that the second ball is red given that the first was red, and (2) the probability that both balls are red.

With pen and paper (1) If the first is red, 4 balls remain: 2 red and 2 white. So P(2nd red | 1st red) = $\frac{2}{4} = \frac{1}{2}$. (2) By the multiplication rule, P(both red) = P(1st red) × P(2nd red | 1st red) = $\frac{3}{5} \times \frac{2}{4} = \frac{3}{10}$ (the computation that walks the branches of the tree in Figure 6.7).

Check it with Python We number the balls to tell them apart and enumerate all 20 ordered pairs (1st, 2nd) with permutations. The expression k[0][0] takes the first character of the first ball — its color.

```
from itertools import permutations
from sympy import Rational
balls = ['R1', 'R2', 'R3', 'W1', 'W2']
pairs2 = list(permutations(balls, 2))
red1 = [k for k in pairs2 if k[0][0] == 'R']
red2 = [k for k in red1 if k[1][0] == 'R']
print(Rational(len(red2), len(red1)))     # conditional
print(Rational(len(red2), len(pairs2)))   # both red
```

Output ▸

```
1/2
3/10
```

(1) is 1/2 and (2) is 3/10. Conditional probability is "a probability whose denominator is shrunk to just the cases matching the condition" — you can read that straight from the code, where red1 (the 12 cases with a red first ball) serves as the denominator.

6.6 Expected Value: How Much on Average?

The finishing touch of the probability chapter is the **expected value**. Draw one lottery ticket — how much do you receive on average? The "average value" of a quantity decided by chance is the expected value, computed by adding up (value × its probability) over all values.

Term Expected value When the possible values are $x_1, x_2, \ldots, x_n$ with probabilities $p_1, p_2, \ldots, p_n$, the expected value is $E = x_1p_1 + x_2p_2 + \ldots + x_np_n$ (the sum of value × probability).

Example 6.9

Find the expected value of the face shown when a die is rolled once.

With pen and paper Every face has probability $\frac{1}{6}$, so $E = 1 \times \frac{1}{6} + 2 \times \frac{1}{6} + \ldots + 6 \times \frac{1}{6} = \frac{21}{6} = \frac{7}{2} = 3.5$. No face shows 3.5, of course — the meaning is that the average over many rolls approaches 3.5.

Check it with Python We add up value × probability with a for loop, then compare with the average of 100,000 rolls using random.

```
from sympy import Rational
import random
E = 0
for x in [1, 2, 3, 4, 5, 6]:
    E += x * Rational(1, 6)
print(E)
random.seed(1)
n = 100000
total = 0
for _ in range(n):
    total += random.randint(1, 6)
print(total / n)   # average of 100,000 rolls
```

Output ▸

```
7/2
3.4995
```

Against the theoretical 7/2 = 3.5, the experimental average is 3.4995. That the expected value is "the average in the long run" is confirmed by the experiment as well.

Example 6.10 The Expected Value of a Lottery

There are 100 lottery tickets: the 1st prize of 1000 dollars on 2 tickets, the 2nd prize of 100 dollars on 10 tickets, and the remaining 88 tickets win nothing (0 dollars). Find the expected value of the prize when one ticket is drawn (Figure 6.8).

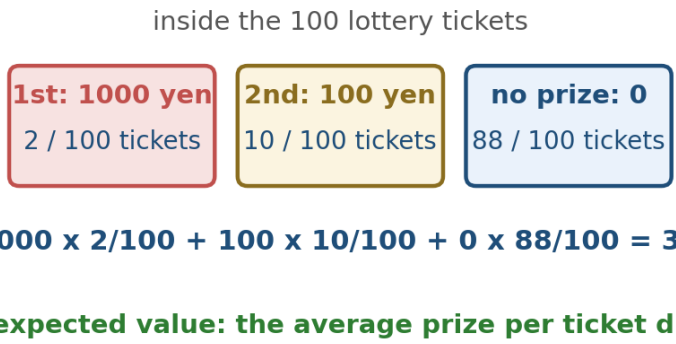


Figure 6.8 The expected value of a lottery. Adding prize × probability gives the average amount per ticket.

With pen and paper $E = 1000 \times \frac{2}{100} + 100 \times \frac{10}{100} + 0 \times \frac{88}{100} = 20 + 10 + 0 = 30$ dollars.

Check it with Python

```
from sympy import Rational
E = 1000 * Rational(2, 100) + 100 * Rational(10, 100)
print(E)
```

Output ▸

```
30
```

The expected value is 30 dollars. If this lottery sold tickets at 50 dollars each, every draw would lose 20 dollars on average. The expected value is a measuring stick for comparing choices involving chance with a cool head.

Column Lotteries and Insurance: A Society Designed on Expected Values

Lottery prizes are designed so that their expected value is less than half the ticket price (the payout rate of Japanese lotteries is just under 50%). In other words, the more you buy, the more you lose on average — and yet people still buy dreams, perhaps because there is a joy that expected values cannot measure. Insurance is the same machinery pointed the other way: insurers design premiums so that their expected value slightly exceeds the expected value of the payouts. Policyholders lose a little on average — and in exchange buy the security of avoiding a single catastrophic loss. Beneath both lotteries and insurance lies what you learned in this chapter: the sum of value × probability. A person who can compute expected values holds a numerical lens for seeing through the chance-driven machinery of the world.

Chapter Summary

- When cases are equally likely, probability = "cases for the event ÷ all cases." Chapter 5's counting becomes numerator and denominator.
- Complement: P(not A) = 1 − P(A). Attack "at least" problems from the complement.
- Mutually exclusive events add: P(A or B) = P(A) + P(B) (the addition rule).

- Independent trials multiply: P(A and B) = P(A) × P(B) (the multiplication rule).
- Repeated trials: the probability of exactly r successes in n trials is ${}_nC_r\, p^r\, (1 - p)^{n-r}$ — one path's probability × the number of paths.
- Conditional probability P(B | A) is "the probability of B given that A is known to have happened." P(A and B) = P(A) × P(B | A).
- The expected value is the sum of value × probability: the long-run average, checkable with random simulations.

Table 6.1 Python tools used in this chapter

Python tool	What it does
`Rational(numer, denom)`	compute probabilities exactly as fractions
`product / permutations`	enumerate to count numerator and denominator
`x.count(value)`	count occurrences of a value in a tuple
`math.comb(n, r)`	count the paths of repeated trials
`random.randint(1, 6)`	a random integer from 1 to 6 (a die)
`random.seed(0)`	fix the seed to reproduce results
`float(E)`	turn a fraction into a decimal

Practice Problems

Solve each problem with pen and paper first, then write Python code in an empty Colab cell to check your answer. Checking from both sides — exact values with Rational and enumeration, experiments with random — is recommended.

Problem 1

A die is rolled once. Find the probability that a multiple of 3 comes up.

Problem 2

Two dice are rolled. Find the probability that the sum of the faces is 8.

Problem 3

A coin is tossed twice. Using the complement, find the probability that heads comes up at least once.

Problem 4

Two dice are rolled. Find the probability that the sum of the faces is 3 or 11.

Problem 5

A die is rolled twice. Using the multiplication rule, find the probability that 6 comes up both times.

Problem 6

A coin is tossed 4 times. Find the probability that heads comes up exactly twice.

Problem 7

A die is rolled 4 times. Find the probability that the face 6 comes up exactly twice.

Problem 8

A bag holds 2 red balls and 3 white balls. One ball is drawn and, without replacement, another is drawn. Find (1) the conditional probability that the second ball is white given that the first was white, and (2) the probability that both balls are white.

Problem 9

Find the expected value of the number of heads when a coin is tossed 3 times.

Problem 10

There are 200 lottery tickets: the 1st prize of 500 dollars on 4 tickets, the 2nd prize of 50 dollars on 20 tickets, and the rest win nothing (0 dollars). Find the expected value of the prize when one ticket is drawn.

Answers and Explanations

Answer 1

The multiples of 3 are 3 and 6: two cases. $\mathrm{P} = \frac{2}{6} = \frac{1}{3}$.

```
from sympy import Rational
faces = [1, 2, 3, 4, 5, 6]
mult3 = [x for x in faces if x % 3 == 0]
print(Rational(len(mult3), len(faces)))
```

Output ▸

```
1/3
```

Answer 2

The sum is 8 for (2, 6), (3, 5), (4, 4), (5, 3), (6, 2): five cases. $\mathrm{P} = \frac{5}{36}$.

```
from itertools import product
from sympy import Rational
pairs = list(product(range(1, 7), repeat=2))
sum8 = [p for p in pairs if sum(p) == 8]
print(Rational(len(sum8), len(pairs)))
```

Output ▸
```
5/36
```

Answer 3

The complement is "tails both times," and P(both tails) = $\frac{1}{4}$. So P(at least one heads) = $1 - \frac{1}{4} = \frac{3}{4}$.

```
from itertools import product
from sympy import Rational
allp = list(product('HT', repeat=2))
tails2 = [p for p in allp if p == ('T', 'T')]
print(1 - Rational(len(tails2), len(allp)))
```

Output ▸
```
3/4
```

Answer 4

The sum is 3 in 2 cases and 11 in 2 cases — mutually exclusive. By the addition rule, P = $\frac{2}{36} + \frac{2}{36} = \frac{4}{36} = \frac{1}{9}$.

```
from itertools import product
from sympy import Rational
pairs = list(product(range(1, 7), repeat=2))
sum3 = [p for p in pairs if sum(p) == 3]
sum11 = [p for p in pairs if sum(p) == 11]
print(Rational(len(sum3), 36) + Rational(len(sum11), 36))
```

Output ▸
```
1/9
```

Answer 5

The two rolls are independent, so by the multiplication rule, P = $\frac{1}{6} \times \frac{1}{6} = \frac{1}{36}$.

```
from sympy import Rational
print(Rational(1, 6) * Rational(1, 6))
```

```
Output ▸
1/36
```

Answer 6

By the repeated-trials formula, $\mathrm{P} = {}_4C_2 \times (\tfrac{1}{2})^2 \times (\tfrac{1}{2})^2 = 6 \times \tfrac{1}{16} = \tfrac{3}{8}$.

```
import math
from sympy import Rational
paths = math.comb(4, 2)
print(paths * Rational(1, 2)**2 * Rational(1, 2)**2)
```

```
Output ▸
3/8
```

Answer 7

With $\mathrm{p} = \tfrac{1}{6}$, $\mathrm{n} = 4$, $\mathrm{r} = 2$: $\mathrm{P} = {}_4C_2 \times (\tfrac{1}{6})^2 \times (\tfrac{5}{6})^2 = 6 \times \tfrac{25}{1296} = \tfrac{25}{216}$.

```
import math
from sympy import Rational
paths = math.comb(4, 2)
print(paths * Rational(1, 6)**2 * Rational(5, 6)**2)
```

```
Output ▸
25/216
```

Answer 8

(1) If the first is white, 4 balls remain: 2 red and 2 white. So P(2nd white | 1st white) = $\tfrac{2}{4} = \tfrac{1}{2}$. (2) By the multiplication rule, P(both white) = $\tfrac{3}{5} \times \tfrac{2}{4} = \tfrac{3}{10}$.

```
from itertools import permutations
from sympy import Rational
balls = ['R1', 'R2', 'W1', 'W2', 'W3']
pairs2 = list(permutations(balls, 2))
white1 = [k for k in pairs2 if k[0][0] == 'W']
white2 = [k for k in white1 if k[1][0] == 'W']
print(Rational(len(white2), len(white1)))
print(Rational(len(white2), len(pairs2)))
```

```
Output ▸
1/2
3/10
```

Answer 9

The number of heads is 0, 1, 2, or 3, with probabilities $\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}$ (proportions counted from all 8 cases). $E = 0 \times \frac{1}{8} + 1 \times \frac{3}{8} + 2 \times \frac{3}{8} + 3 \times \frac{1}{8} = \frac{12}{8} = \frac{3}{2} = 1.5$ heads.

```
from itertools import product
from sympy import Rational
allp = list(product('HT', repeat=3))
E = 0
for heads in [0, 1, 2, 3]:
    hits = [p for p in allp if p.count('H') == heads]
    E += heads * Rational(len(hits), len(allp))
print(E)
print(float(E))
```

Output ▸

```
3/2
1.5
```

Answer 10

$E = 500 \times \frac{4}{200} + 50 \times \frac{20}{200} = 10 + 5 = 15$ dollars.

```
from sympy import Rational
E = 500 * Rational(4, 200) + 50 * Rational(20, 200)
print(E)
```

Output ▸

```
15
```

Chapter 7
Deepening the Properties of Shapes: Plane and Solid Geometry

Properties of triangles

Properties of circles

Constructions

Ceva's theorem

Menelaus's theorem

Basics of solid geometry

Viewing geometry logically

Chapter 7 Deepening the Properties of Shapes: Plane and Solid Geometry

What You Will Learn

Constructions, congruence, similarity, inscribed angles, the Pythagorean theorem — you covered the basics of geometry in Level 1B. In this chapter we dig one level deeper into that world: special points hiding inside every triangle, powerful theorems that circles obey, and a beautiful condition for three lines to pass through a single point. None of these can be spotted by appearance alone — they are true **properties** of shapes.

The first half is plane geometry. Starting from the ratio in which an **angle bisector** splits the opposite side, we meet the **centroid, circumcenter, and incenter** of a triangle, the **inscribed angle theorem** and the **power of a point** for circles, and then **Ceva's theorem** and **Menelaus's theorem**. The second half steps into space, where **Euler's polyhedron formula** ties together the vertices, edges, and faces of any polyhedron. We finish by viewing geometry logically: looking at the theorems again through the eyes of Chapter 2's statements and converses.

This chapter's Python gains a new companion: sympy.geometry. Point, Triangle, Line, Circle — once a figure is expressed in coordinates, every distance, intersection, and angle can be checked by computation. Ruler and compass on one side, coordinates and computation on the other: we will travel back and forth between the two worlds.

By the end of this chapter, you will be able to:

- find segment lengths using the ratio in which an angle bisector splits the opposite side
- explain the centroid, circumcenter, and incenter, and verify them with coordinates
- compute angles and lengths with the inscribed angle theorem and the power of a point
- explain perpendicular-bisector constructions as "sets of equidistant points"
- find the remaining ratio with Ceva's and Menelaus's theorems

- relate the vertices, edges, and faces of a polyhedron with Euler's formula
- distinguish a theorem from its converse and check both with Python and coordinates

7.1 Properties of Triangles: Special Lines and Special Points

Recap (Level 1B, Chapters 7–10) Constructing perpendicular bisectors and angle bisectors (Chapter 7), congruence and proof (Chapter 8), similarity and "parallel lines and ratios" (Chapter 9), the inscribed angle theorem and the Pythagorean theorem (Chapter 10). This chapter puts all of those Level 1B tools to full use.

Triangles come with "special lines" that arise naturally from their vertices and sides. First, the **angle bisector**: the line that splits a vertex angle exactly in half. In what ratio does it cut the opposite side? The answer is surprisingly clean (Figure 7.1).

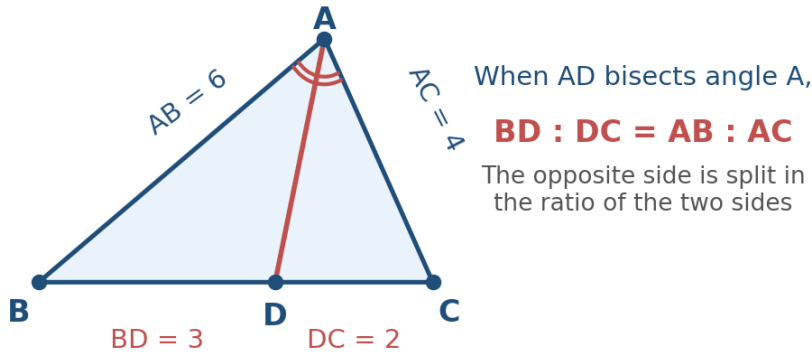


Figure 7.1 The angle bisector splits the opposite side in the ratio of the two adjacent sides, AB : AC.

Term Angle bisector and ratio In triangle ABC, if the bisector of ∠A meets side BC at D, then BD : DC = AB : AC . The ratio of the two adjacent sides reappears on the opposite side (provable with similarity).

Example 7.1

In triangle ABC with AB = 6, AC = 4, and BC = 5, the bisector of ∠A meets side BC at D. Find the lengths BD and DC.

With pen and paper BD : DC = AB : AC = 6 : 4 = 3 : 2. Splitting BC = 5 in the ratio 3 : 2 gives $BD = 5 \times \frac{3}{5} = 3$ and $DC = 5 \times \frac{2}{5} = 2$.

Check it with Python Ratio arithmetic is a job for Rational (fractions), our reliable friend from the probability chapter.

```
from sympy import Rational
AB, AC, BC = 6, 4, 5
BD = BC * Rational(AB, AB + AC)
DC = BC - BD
print(BD, DC)
```

Output ▸

```
3 2
```

BD = 3 and DC = 2. Splitting a whole into "3 and 2 out of 3 + 2 = 5" — ratio computations turn into fraction multiplications.

Next come the "special points." A segment joining a vertex to the midpoint of the opposite side is called a **median**. There are three medians, and they always meet at a single point: the **centroid**. Likewise, the three perpendicular bisectors of the sides meet at one point, the **circumcenter**, and the three angle bisectors meet at one point, the **incenter** (Figure 7.2).

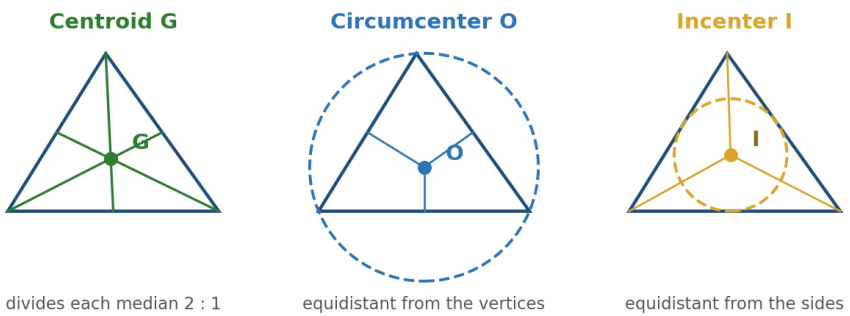


Figure 7.2 Three centers of a triangle. Each is born from three lines that always meet at a single point.

Term **Centroid, circumcenter, incenter** The centroid G is the intersection of the medians and divides each median 2 : 1 from the vertex. The circumcenter O is the intersection of the perpendicular bisectors, equidistant from the three vertices (center of the circumscribed circle). The incenter I is the intersection of the angle bisectors, equidistant from the three sides (center of the inscribed circle).

Example 7.2

Find the centroid G of the triangle with vertices A(0, 0), B(6, 0), and C(3, 6), and check that G divides the median AM (where M is the midpoint of side BC) in the ratio 2 : 1.

With pen and paper The centroid is the average of the three vertices: G = $(\frac{0+6+3}{3}, \frac{0+0+6}{3})$ = (3, 2). M is the midpoint of BC, so M = $(\frac{9}{2}, 3)$. Computing AG and GM with the Pythagorean theorem and comparing them gives exactly 2 : 1.

Check it with Python Enter our new tool, **sympy.geometry**. Point(x, y) is a point in the coordinate plane and Triangle builds a triangle from 3 points. tri.centroid returns the centroid in one stroke, and A.distance(G) computes the distance between two points exactly (radicals and all). A midpoint is an average of coordinates, so (B + C) / 2 builds it.

```
from sympy import Point, Triangle
A, B, C = Point(0, 0), Point(6, 0), Point(3, 6)
tri = Triangle(A, B, C)
G = tri.centroid
M = (B + C) / 2          # midpoint of side BC
print(G)
print(A.distance(G) / G.distance(M))
```

Output ▸

```
Point2D(3, 2)
2
```

The centroid is (3, 2), and AG ÷ GM = 2, that is, AG : GM = 2 : 1. "The three medians always meet at one point, and that point divides each median 2 : 1" — a property of every triangle, confirmed effortlessly by coordinate computation.

7.2 Properties of Circles: Inscribed Angles and the Power of a Point

You met the inscribed angle theorem in Level 1B, Chapter 10: all inscribed angles on the same arc are equal, and each is half the central angle — the most famous property a circle has (Figure 7.3). Let us first re-verify this theorem from the side of coordinates and computation.

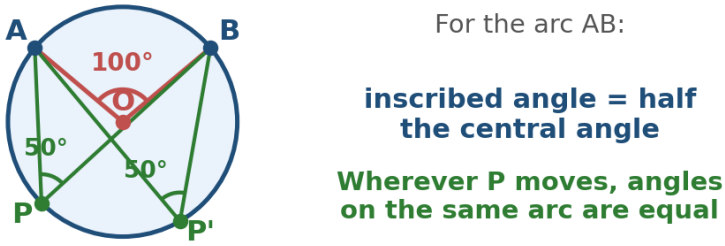


Figure 7.3 The inscribed angle theorem. Angles on the same arc are all equal — half the central angle.

Term **Inscribed angle theorem** The inscribed angles on one arc all have the same size: half of the central angle on that arc.

Example 7.3

On a circle of radius 1 (center O), place point A at the 140° position and point B at the 40° position. Compute the central angle ∠AOB for the arc AB and the inscribed angle ∠APB from three positions on the circle (200°, 270°, 320°), and confirm the inscribed angle theorem.

With pen and paper The central angle is just the difference of positions: ∠AOB = 140° − 40° = 100°. By the inscribed angle theorem, every inscribed angle should be half of that, 50° — no matter where P is.

Check it with Python We write our own angle-measuring function, powered by Chapter 4's law of cosines. Once we know the three side lengths of triangle APB, $\cos\angle APB = \frac{PA^2 + PB^2 - AB^2}{2 \times PA \times PB}$ gives the cosine of the angle, and math.acos turns it back into an angle. math.dist is the distance between two points, and pt(deg) returns the point on the circle at angle deg.

```
import math
def angle(P, A, B):     # angle APB via the law of cosines
    PA = math.dist(P, A)
    PB = math.dist(P, B)
    AB = math.dist(A, B)
    cos_v = (PA**2 + PB**2 - AB**2) / (2 * PA * PB)
    return math.degrees(math.acos(cos_v))
def pt(deg):            # point on the circle at angle deg
    r = math.radians(deg)
    return (math.cos(r), math.sin(r))
O = (0, 0)
A, B = pt(140), pt(40)
print(round(angle(O, A, B), 1))      # central angle
for deg in [200, 270, 320]:
    P = pt(deg)
    print(round(angle(P, A, B), 1))  # inscribed angle
```

Output ▸

```
100.0
50.0
50.0
50.0
```

The central angle is 100°; the inscribed angle is 50.0° at all three positions. Wherever P moves, the angle subtending the same arc never changes. Chapter 4's law of cosines served as our verification machine for a circle theorem.

Next, a circle theorem that is new in Level 2. Take a point P outside the circle and draw a line through it meeting the circle at two points A and B. Then the product PA × PB **does not depend on which line you draw** (Figure 7.4). This is the theorem of the **power of a point**.

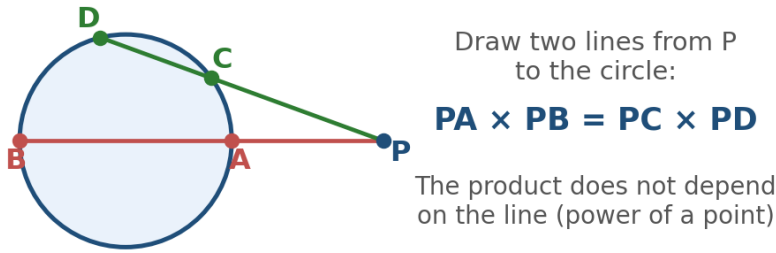


Figure 7.4 The power of a point. However the line through P is drawn, the value of PA × PB never changes.

Term Power of a point When a line through point P meets a circle at two points A and B, the product PA × PB is the same for every such line. It also holds when P is inside the circle (the key to the proof: the inscribed angle theorem and similarity).

Example 7.4

A circle of radius 5 is centered at the origin, and P(7, 0) lies outside it. Compute PA × PB for three lines through P and confirm that the value is constant.

With pen and paper The easiest line is the one through the center: the distances to the circle are 7 − 5 = 2 and 7 + 5 = 12, so PA × PB = 2 × 12 = 24. If the theorem is right, every line should give 24.

Check it with Python We use Circle and Line from sympy.geometry. circ.intersection(...) returns the intersection points of the circle and a line, so we try three lines in different directions.

```
from sympy import Point, Circle, Line
circ = Circle(Point(0, 0), 5)
P = Point(7, 0)
for Q in [Point(0, 0), Point(0, 5), Point(3, 4)]:
    A, B = circ.intersection(Line(P, Q))
    prod = P.distance(A) * P.distance(B)
    print(prod.simplify())
```

Output ▸

```
24
24
24
```

All three lines give 24. The intersection coordinates are full of radicals, yet the products line up at 24 as if by magic. Once the circle and the point are fixed, this product is fixed too — that is why it is called the "power" of the point.

7.3 Constructions: Seen as "Sets of Equidistant Points"

Drawing figures with only a ruler and a compass is called **construction**.

You practiced constructing perpendicular bisectors and angle bisectors by hand in Level 1B, Chapter 7. The Level 2 question is: **why do those steps produce the right line?** The key is the language of sets from Chapter 2.

The **perpendicular bisector** of segment AB is "the set of all points equidistant from the two points A and B." When you draw compass arcs of the same radius from A and from B, both intersection points are equidistant from A and B — so joining them gives the perpendicular bisector. Likewise, an **angle bisector** is "the set of points equidistant from the two sides of the angle." Seeing one line as "the collection of points satisfying a condition" is the foundation for loci in Chapter 9, Geometry and Equations.

This view also explains the secret of the circumcenter from Section 7.1. A point on the perpendicular bisector of AB is equidistant from A and B; a point on that of BC is equidistant from B and C. Their intersection O satisfies both, so OA = OB and OB = OC, hence OA = OB = OC. But then O is also equidistant from A and C, so **it automatically lies on the perpendicular bisector of CA as well**. The third line passing through the same point is not a coincidence — it is a logical necessity. The same argument with "equidistant from the sides" works for the incenter.

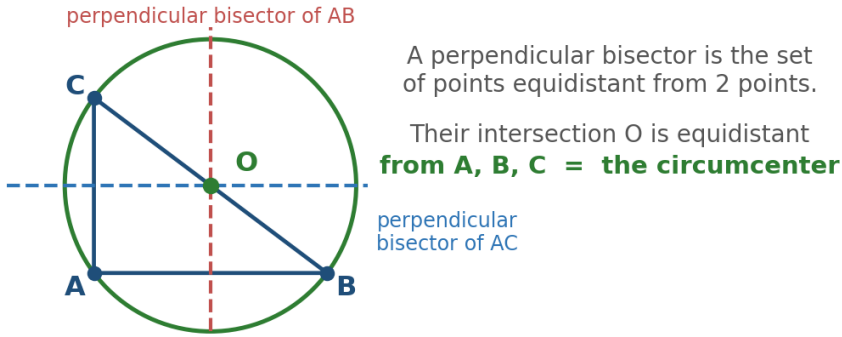


Figure 7.5 Perpendicular bisectors and the circumcenter. The intersection O of two bisectors is equidistant from all three vertices, so the third bisector must pass through O too.

Term **Basic constructions** A perpendicular bisector is "the set of points equidistant from 2 points"; an angle bisector is "the set of points equidistant from 2 sides." The construction steps realize these properties with a compass.

Example 7.5

A triangle has vertices A(0, 0), B(4, 0), and C(0, 3). Find the circumcenter O as the intersection of the perpendicular bisectors of sides AB and AC, and check that O is equidistant from the three vertices.

With pen and paper This triangle has a right angle at A. The perpendicular bisector of AB is the line x = 2 and that of AC is the line y = 1.5, so they meet at O(2, 1.5) — the midpoint of the hypotenuse BC. The circumcenter of a right triangle lies at the midpoint of the hypotenuse: this is the Level 1B, Chapter 10 fact "an inscribed angle on a diameter is 90°," turned inside out.

Check it with Python Segment in sympy.geometry comes with perpendicular_bisector(), which returns the perpendicular bisector of a segment — the construction steps turned directly into code.

```
from sympy import Point, Segment
A, B, C = Point(0, 0), Point(4, 0), Point(0, 3)
L1 = Segment(A, B).perpendicular_bisector()
L2 = Segment(A, C).perpendicular_bisector()
O = L1.intersection(L2)[0]
print(O)
print(O.distance(A), O.distance(B), O.distance(C))
```

Output ▸

```
Point2D(2, 3/2)
5/2 5/2 5/2
```

The intersection is O(2, 3/2), and the distances to the three vertices all come out 5/2. The hypotenuse BC has length 5 by the Pythagorean theorem, so the distance to its midpoint, 5/2, matches exactly. The "set of equidistant points" view is confirmed right there in the code.

7.4 Ceva's Theorem and Menelaus's Theorem

This section presents two celebrated theorems about triangles and ratios. Both share the same striking shape: "the product of three ratios equals 1." First, **Ceva's theorem**. Take a point P inside triangle ABC, and let the lines joining each vertex to P meet the opposite sides at D, E, and F (Figure 7.6, left). Then the six segments created on the sides obey the following relation.

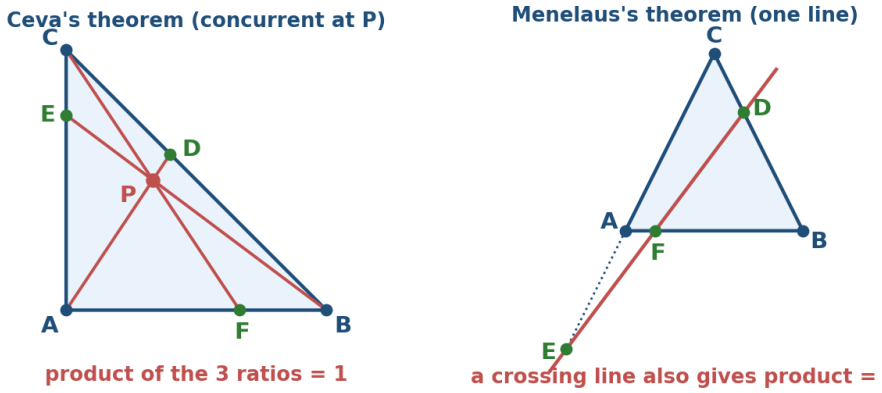


Figure 7.6 Ceva's theorem (left) and Menelaus's theorem (right). In both, the product of the three ratios is 1.

Term **Ceva's theorem** If the lines joining the vertices of triangle ABC to an interior point P meet the opposite sides at D, E, F, then $\frac{AF}{FB} \times \frac{BD}{DC} \times \frac{CE}{EA} = 1$. Conversely, if this product is 1, the three lines meet at a single point.

Start at vertex A and walk around the sides, A → F → B → D → C → E → A, picking up the ratios as you go — an easy way to remember the formula. If two of the three ratios are known, the third comes from solving an equation.

Example 7.6

For an interior point P of triangle ABC, we know AF : FB = 2 : 1 and BD : DC = 3 : 2. Use Ceva's theorem to find CE : EA. Then place coordinates and verify that the three lines really do meet at one point.

With pen and paper Let CE/EA = x. Ceva's theorem gives $\frac{2}{1} \times \frac{3}{2} \times x = 1$. Solving, x = 1/3, so CE : EA = 1 : 3.

Check it with Python The first half solves the equation with solve. The second half places A(0, 0), B(5, 0), C(0, 5), takes F, D, E according to the ratios, finds the intersection P of lines AD and BE, and asks contains whether the third line CF passes through P.

```
from sympy import symbols, solve, Rational, Point, Line
x = symbols('x')
r = solve(Rational(2, 1) * Rational(3, 2) * x - 1, x)[0]
print(r)
A, B, C = Point(0, 0), Point(5, 0), Point(0, 5)
F = Point(Rational(10, 3), 0)   # AF:FB = 2:1
D = Point(2, 3)                 # BD:DC = 3:2
E = Point(0, Rational(15, 4))   # CE:EA = 1:3
P = Line(A, D).intersection(Line(B, E))[0]
print(P, Line(C, F).contains(P))
```

Output ▸

```
1/3
Point2D(5/3, 5/2) True
```

The ratio is 1/3, and the line CF does pass through the intersection P(5/3, 5/2): True. The **converse** of Ceva's theorem — if the product of the ratios is 1, the three lines are concurrent — is visibly at work in the coordinates.

Now for **Menelaus's theorem**. This time the star is not a point but a single line. Suppose one line crosses the three sides of triangle ABC (or their extensions) at F, D, and E (Figure 7.6, right). Then exactly the same equation as Ceva's holds.

Term **Menelaus's theorem** If one line meets the sides AB, BC, CA of triangle ABC (or their extensions) at F, D, E, then $\frac{AF}{FB} \times \frac{BD}{DC} \times \frac{CE}{EA} = 1$. At least one of the three points lies on an extension of a side.

Same formula, completely different picture. Here is the mnemonic: **meeting at a point — Ceva; crossed by a line — Menelaus**. Note that when a line crosses a triangle, at least one of the three intersection points must jump outside onto an extension of a side.

Example 7.7

A line crosses the sides of triangle ABC with AF : FB = 1 : 5 and BD : DC = 2 : 1. Use Menelaus's theorem to find CE : EA. Then find the three intersection points with coordinates and confirm that the product of the ratios is 1.

With pen and paper Let CE/EA = x. Menelaus's theorem gives $\frac{1}{5} \times \frac{2}{1} \times x = 1$. Solving, x = 5/2, so CE : EA = 5 : 2.

Check it with Python We place A(0, 0), B(6, 0), C(3, 6) and the line L through the two points (1, 0) and (4, 4). intersection finds where L meets the lines containing the three sides, and the distances give the product of the ratios.

```
from sympy import symbols, solve, Rational, Point, Line
x = symbols('x')
r = solve(Rational(1, 5) * Rational(2, 1) * x - 1, x)[0]
print(r)
A, B, C = Point(0, 0), Point(6, 0), Point(3, 6)
L = Line(Point(1, 0), Point(4, 4))
F = L.intersection(Line(A, B))[0]
D = L.intersection(Line(B, C))[0]
E = L.intersection(Line(C, A))[0]
print(F, D, E)
prod = (A.distance(F) / F.distance(B)
        * B.distance(D) / D.distance(C)
        * C.distance(E) / E.distance(A))
print(prod.simplify())
```

Output ▸

```
5/2
Point2D(1, 0) Point2D(4, 4) Point2D(-2, -4)
1
```

The ratio is 5/2. The intersection E(−2, −4) jumps out onto the extension of side CA, yet the product of the ratios is exactly 1. Ceva and Menelaus — two look-alike theorems that are in fact siblings: each can be used to prove the other.

7.5 Basics of Solid Geometry: Polyhedra and Measurement

Recap (Level 1B, Chapter 11) Solid figures — prisms, pyramids, cylinders, cones — and the computation of surface areas and volumes were covered in Level 1B, Chapter 11. Recall the 1/3 that appears in the volume of a pyramid or cone, and the use of nets for surface areas.

Now we move the stage into space. A solid bounded only by flat faces is a **polyhedron**. A polyhedron whose faces are all congruent regular polygons, with the same number of faces meeting at every vertex, is a

regular polyhedron. There are exactly five: the regular tetrahedron, hexahedron (cube), octahedron, dodecahedron, and icosahedron — a beautiful fact known since ancient Greece.

Write V for the number of vertices of a polyhedron, E for the number of edges, and F for the number of faces. Then an astonishing relation holds for every convex polyhedron: **Euler's polyhedron formula** (Figure 7.7).

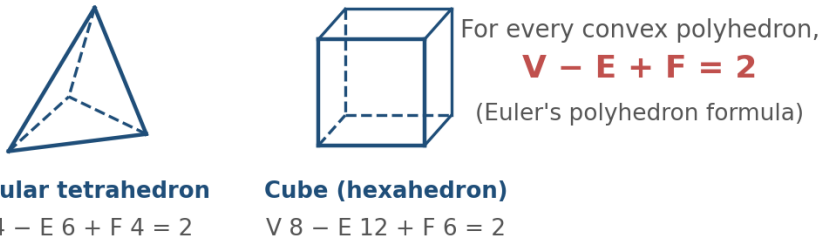


Figure 7.7 Euler's polyhedron formula. For the tetrahedron and the cube alike, V − E + F = 2.

Term **Euler's polyhedron formula** For the numbers of vertices V, edges E, and faces F of any convex polyhedron, $V - E + F = 2$. The relation depends only on the "skeleton" of the polyhedron, not on its shape or size.

Example 7.8

A convex polyhedron has 20 vertices and 12 faces. Find the number of edges. Also compute V − E + F for all five regular polyhedra and confirm that it really is 2.

With pen and paper Substituting V = 20 and F = 12 into V − E + F = 2 gives 20 − E + 12 = 2, so E = 30. In fact, 20 vertices, 30 edges, and 12 faces are the numbers of the regular dodecahedron.

Check it with Python The first half solves the equation with our old friend solve from Chapter 3. The second half stores the (V, E, F) of the five regular polyhedra as data and computes V − E + F for all of them in a loop.

```
from sympy import symbols, solve
E = symbols('E')
print(solve(20 - E + 12 - 2, E))
data = [('tetrahedron', 4, 6, 4),
        ('hexahedron', 8, 12, 6),
        ('octahedron', 6, 12, 8),
        ('dodecahedron', 20, 30, 12),
        ('icosahedron', 12, 30, 20)]
for name, v, e, f in data:
    print(name, v - e + f)
```

Output ▸

```
[30]
tetrahedron 2
hexahedron 2
octahedron 2
dodecahedron 2
icosahedron 2
```

The number of edges is 30, and all five regular polyhedra line up at V − E + F = 2. An invariant relation hiding among "things you can count" — vertices, edges, faces. Euler's formula is also a seed that later grew into topology, a major field of modern mathematics.

The other lead role in solid geometry is **measurement**: lengths, areas, and volumes. The weapon is the same as in the plane — the Pythagorean theorem. As an example, consider the height of a regular tetrahedron with edge a. The foot of the perpendicular from the apex to the base is the **circumcenter** of the base triangle (the point equidistant from its vertices). In an equilateral triangle the circumcenter and the centroid coincide, and the distance to each vertex is known to be $R = \frac{a}{\sqrt{3}}$. The height h then comes from a right triangle via the Pythagorean theorem: $h^2 = a^2 - R^2$.

Example 7.9

Find the height and the volume of a regular tetrahedron with edge length 6.

With pen and paper $R = 6/\sqrt{3} = 2\sqrt{3}$, so $h^2 = 36 - 12 = 24$ and $h = 2\sqrt{6}$. The volume is "base area × height × $\frac{1}{3}$." The base is an equilateral triangle with side 6, whose area is $9\sqrt{3}$.

Check it with Python With sympy's sqrt, values containing radicals stay

exact, and simplify tidies up the final expressions.

```
from sympy import sqrt, Rational, simplify
a = 6
area = sqrt(3) / 4 * a**2     # area of the base triangle
R = a / sqrt(3)               # circumradius of the base
h = sqrt(a**2 - R**2)         # height
V = Rational(1, 3) * area * h
print(simplify(h))
print(simplify(V))
```

Output ▸

```
2*sqrt(6)
18*sqrt(2)
```

The height is $2\sqrt{6}$ and the volume is $18\sqrt{2}$. Even in space, the basic move is the same as in the plane: find a right triangle and bring in the Pythagorean theorem. Notice also that the circumcenter from Section 7.1 reappeared here as the foothold of a spatial computation.

7.6 Viewing Geometry Logically: A Theorem and Its Converse

> **Recap (Level 2, Chapter 2)** For a statement "if p then q," swapping p and q gives the converse, "if q then p." Even when the original statement is true, the converse need not be — a single counterexample makes it false — as we learned in Chapter 2.

The finishing touch of this chapter is to look at geometric theorems through **the eyes of logic**. The Pythagorean theorem, for instance, is the statement "if a triangle is right-angled, then $a^2 + b^2 = c^2$." What about its **converse** — "if $a^2 + b^2 = c^2$, then the triangle is right-angled"? It turns out to be true as well, and it is called the **converse of the Pythagorean theorem**. You can certify a right angle by measuring three lengths, without ever measuring an angle — extremely handy in practice (Figure 7.8).

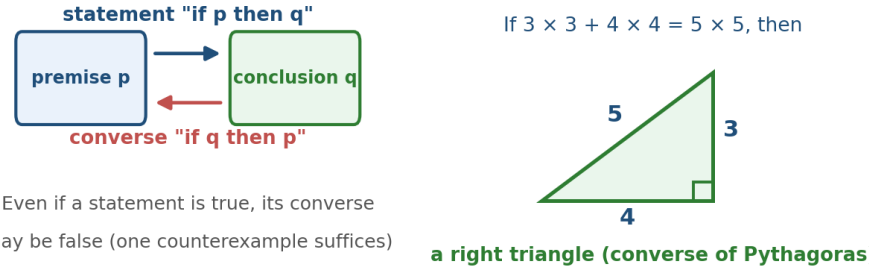


Figure 7.8 A theorem and its converse. If the sides are 3, 4, 5, the angle is right — no protractor needed.

Term **Converse of a theorem** The converse "if q then p" of a theorem "if p then q" is a different statement, and whether it holds must be checked separately. The Pythagorean theorem is a fortunate example whose converse also holds.

"The converse also holds" is never automatic. For example, "if a triangle is equilateral, then it is isosceles" is true, but the converse — "if isosceles, then equilateral" — is false: an isosceles triangle whose base has a different length is a counterexample. Ceva's theorem, on the other hand, does have a true converse, and Example 7.6 used it to confirm that three lines are concurrent. Whenever you use a theorem, ask yourself: am I using the theorem itself, or its converse? That is what it means to view geometry logically.

Example 7.10

A triangle has vertices A(0, 0), B(5, 0), and C(9/5, 12/5). Find the three side lengths and use the converse of the Pythagorean theorem to decide whether $\angle C$ is a right angle. In Python, compute the angle directly to check.

With pen and paper The distance formula gives AC = 3, BC = 4, AB = 5 (the coordinates of C come from solving the simultaneous distance equations AC = 3 and BC = 4). Since $3^2 + 4^2 = 9 + 16 = 25 = 5^2$, the converse of the Pythagorean theorem says this is a right triangle with hypotenuse AB, that is, $\angle C = 90°$.

Check it with Python sympy's Triangle has angles, which returns the angle at each vertex. Let us check directly whether $\angle C$ really is $\pi/2$ (= 90°).

```
from sympy import Point, Triangle, Rational, pi
C = Point(Rational(9, 5), Rational(12, 5))
tri = Triangle(Point(0, 0), Point(5, 0), C)
print(tri.angles[C])
print(tri.angles[C] == pi / 2)
```

Output ▸

```
pi/2
True
```

∠C is exactly π/2. From "numbers" — side lengths — we extracted "shape" information: a right angle. That is the power of a converse. Throughout this chapter we kept placing figures on coordinates and checking them by computation. In Chapter 9, Geometry and Equations, this "coordinates × figures" idea itself takes the lead, as we describe lines and circles by equations.

Column The Three Classical Construction Problems: Proving "It Cannot Be Done"

The mathematicians of ancient Greece left behind three notorious challenges to be solved with ruler and compass alone: trisecting an arbitrary angle, doubling the cube (constructing a cube of exactly twice the volume), and squaring the circle (constructing a square with the same area as a circle). Many brilliant minds attacked them in vain, and the verdict arrived only in the 19th century: thanks to the development of algebra, all three were proved **impossible** with ruler and compass alone. Meanwhile, the young Gauss discovered that the regular 17-gon **can** be constructed — a discovery said to have settled his resolve to become a mathematician. Finding what can be done, and proving what cannot: both are genuine achievements of mathematics. This 2,000-year story shows just how far the power of logic can reach.

Chapter Summary

- An angle bisector splits the opposite side in the ratio AB : AC.
- The centroid is the intersection of the medians (2 : 1 from each vertex); the circumcenter is equidistant from the vertices (perpendicular bisectors); the incenter is equidistant from the sides (angle bisectors).

- An inscribed angle is half the central angle on the same arc, regardless of the position of P.
- Power of a point: when a line through P meets a circle at A and B, the product PA × PB does not depend on the line.
- Ceva (three lines through one point) and Menelaus (one line across three sides): the product of the three ratios is 1.
- For polyhedra, V − E + F = 2 (Euler's formula). Spatial measurement: find a right triangle and use the Pythagorean theorem.
- The converse of a theorem is a different statement; place coordinates and check it with Python.

Table 7.1 Python tools used in this chapter

Python tool	What it does
`Point(x, y)`	a point given by coordinates
`A.distance(B)`	exact distance between two points
`Triangle(A, B, C)`	a triangle from 3 points
`tri.centroid / tri.angles`	centroid / vertex angles of a triangle
`Segment(A, B)`	the segment joining two points
`.perpendicular_bisector()`	perpendicular bisector of a segment
`Line(A, B)`	the line through two points
`L.intersection(M)`	intersection points of two objects
`L.contains(P)`	whether point P lies on object L

Practice Problems

Solve each problem with pen and paper first, then write Python code in the empty Colab cells to check your answers. For geometry, placing coordinates with sympy.geometry is a great way to verify your work.

Problem 1

In triangle ABC, AB = 8, AC = 6, and BC = 7. The bisector of ∠A meets side BC at D. Find the lengths BD and DC.

Problem 2

Find the centroid of the triangle with vertices A(0, 0), B(6, 0), and C(0, 9).

Problem 3

On a circle of radius 1 (center O), place point A at the 160° position and point B at the 20° position. Find the central angle $\angle AOB$ and the inscribed angle $\angle APB$ seen from the point P at the 270° position.

Problem 4

A circle of radius 4 is centered at the origin, and P(6, 0) lies outside it. Compute $PA \times PB$ for the line through P and the center, and for the line through P and the point (0, 4), and confirm that the two values agree.

Problem 5

Find the circumcenter and the circumradius of the triangle with vertices A(0, 0), B(6, 0), and C(0, 8).

Problem 6

For an interior point P of triangle ABC, we know AF : FB = 1 : 2 and BD : DC = 4 : 3. Use Ceva's theorem to find CE : EA.

Problem 7

A line crosses the sides of triangle ABC with AF : FB = 1 : 3 and BD : DC = 3 : 1. Use Menelaus's theorem to find CE : EA.

Problem 8

A convex polyhedron has 12 vertices and 20 faces. Find the number of edges.

Problem 9

Find the height and the volume of a regular tetrahedron with edge length 3.

Problem 10

A triangle has vertices A(0, 0), B(13, 0), and C(25/13, 60/13). Find the three side lengths and use the converse of the Pythagorean theorem to decide whether $\angle C$ is a right angle.

Answers and Explanations

Answer 1

By the angle-bisector property, BD : DC = AB : AC = 8 : 6 = 4 : 3. Splitting BC = 7 in the ratio 4 : 3 gives BD = 4 and DC = 3.

```
from sympy import Rational
AB, AC, BC = 8, 6, 7
BD = BC * Rational(AB, AB + AC)
print(BD, BC - BD)
```

Output ▸

```
4 3
```

Answer 2

The centroid averages the coordinates: the x-coordinate is $\frac{0+6+0}{3} = 2$ and the y-coordinate is $\frac{0+0+9}{3} = 3$, so the centroid is G(2, 3).

```
from sympy import Point, Triangle
tri = Triangle(Point(0, 0), Point(6, 0), Point(0, 9))
print(tri.centroid)
```

Output ▸

```
Point2D(2, 3)
```

Answer 3

The central angle is the difference of positions: $\angle AOB = 160° - 20° = 140°$. The inscribed angle is half of that, 70°.

```
import math
def angle(P, A, B):
    PA = math.dist(P, A)
    PB = math.dist(P, B)
    AB = math.dist(A, B)
    cos_v = (PA**2 + PB**2 - AB**2) / (2 * PA * PB)
    return math.degrees(math.acos(cos_v))
def pt(deg):
    r = math.radians(deg)
    return (math.cos(r), math.sin(r))
A, B = pt(160), pt(20)
print(round(angle((0, 0), A, B), 1))
print(round(angle(pt(270), A, B), 1))
```

Output ▸

```
140.0
70.0
```

Answer 4

On the line through the center, PA × PB = (6 − 4) × (6 + 4) = 2 × 10 = 20.

By the power of a point, the other line should give 20 as well.

```
from sympy import Point, Circle, Line
circ = Circle(Point(0, 0), 4)
P = Point(6, 0)
for Q in [Point(0, 0), Point(0, 4)]:
    A, B = circ.intersection(Line(P, Q))
    print((P.distance(A) * P.distance(B)).simplify())
```

Output ▸

```
20
20
```

Answer 5

The triangle has a right angle at A, so the circumcenter is the midpoint of the hypotenuse BC: O(3, 4). The radius is OA = $\sqrt{3^2 + 4^2} = 5$, which also equals half of the hypotenuse BC = 10.

```
from sympy import Point, Segment
A, B, C = Point(0, 0), Point(6, 0), Point(0, 8)
L1 = Segment(A, B).perpendicular_bisector()
L2 = Segment(A, C).perpendicular_bisector()
O = L1.intersection(L2)[0]
print(O, O.distance(A))
```

Output ▸

```
Point2D(3, 4) 5
```

Answer 6

Let CE/EA = x. Ceva's theorem gives $\frac{1}{2} \times \frac{4}{3} \times x = 1$. Solving, x = 3/2, so CE : EA = 3 : 2.

```
from sympy import symbols, solve, Rational
x = symbols('x')
print(solve(Rational(1, 2) * Rational(4, 3) * x - 1, x))
```

Output ▸

```
[3/2]
```

Answer 7

Let CE/EA = x. Menelaus's theorem gives $\frac{1}{3} \times \frac{3}{1} \times x = 1$. Solving, x = 1, so CE : EA = 1 : 1.

```
from sympy import symbols, solve, Rational
x = symbols('x')
print(solve(Rational(1, 3) * Rational(3, 1) * x - 1, x))
```

Output ▸

```
[1]
```

Answer 8

Substituting V = 12 and F = 20 into Euler's formula V − E + F = 2 gives 12 − E + 20 = 2, so E = 30 (12 vertices, 30 edges, and 20 faces are the numbers of the regular icosahedron).

```
from sympy import symbols, solve
E = symbols('E')
print(solve(12 - E + 20 - 2, E))
```

Output ▸

```
[30]
```

Answer 9

The circumradius of the base is $R = 3/\sqrt{3} = \sqrt{3}$. The height is $h = \sqrt{9 - 3} = \sqrt{6}$. The base area is $\frac{9\sqrt{3}}{4}$, so the volume is $\frac{1}{3} \times \frac{9\sqrt{3}}{4} \times \sqrt{6} = \frac{9\sqrt{2}}{4}$.

```
from sympy import sqrt, Rational, simplify
a = 3
area = sqrt(3) / 4 * a**2
h = sqrt(a**2 - (a / sqrt(3))**2)
V = Rational(1, 3) * area * h
print(simplify(h), simplify(V))
```

Output ▸

```
sqrt(6) 9*sqrt(2)/4
```

Answer 10

The distance formula gives AC = 5, BC = 12, AB = 13. Since $5^2 + 12^2 = 25 + 144 = 169 = 13^2$, the converse of the Pythagorean theorem says the triangle is right-angled with $\angle C = 90°$.

```
from sympy import Point, Triangle, Rational, pi
C = Point(Rational(25, 13), Rational(60, 13))
tri = Triangle(Point(0, 0), Point(13, 0), C)
print(tri.angles[C] == pi / 2)
```

Output ▸

```
True
```

Chapter 8
Extending Expressions: Complex Numbers and Higher-Degree Equations

Computing with polynomials

Algebraic fractions

The binomial theorem

Complex numbers

Solutions of quadratic equations

Relation between roots and coefficients

Higher-degree equations

Chapter 8 Extending Expressions: Complex Numbers and Higher-Degree Equations

What You Will Learn

In Chapter 1 you learned to expand and factor expressions. This chapter extends that "world of expressions" in two directions. One is an **extension of computation** — division of polynomials, algebraic fractions, and the binomial theorem, which finishes an expansion in one stroke. The other is an **extension of numbers** — we welcome a new number i whose square is −1, widening the world of numbers all the way to the complex numbers.

Why extend numbers at all? In Chapter 3 we learned that a quadratic equation whose graph misses the x-axis has "no real solutions." But once the world is widened to the complex numbers, we can declare that **every quadratic equation has solutions**. And with the **relation between roots and coefficients** and the **factor theorem** in hand, you will be able to solve **higher-degree equations** — cubics and beyond — on your own.

The Python stars of this chapter are sympy's expression-manipulation tools. Polynomial division is div, factoring is factor, and the imaginary unit i is I — sympy will double-check, one by one, every operation you learn by hand.

By the end of this chapter, you will be able to:

- carry out polynomial division and write the result as $A = B \times Q + R$
- use the remainder theorem to find a remainder without dividing
- reduce and combine algebraic fractions
- find expansion coefficients with the binomial theorem and Pascal's triangle
- do arithmetic with complex numbers and handle division using the conjugate
- classify solutions with the discriminant and evaluate symmetric expressions with the relation between roots and coefficients
- solve higher-degree equations using the factor theorem

8.1 Computing with Polynomials: Division and the Remainder Theorem

Recap (Level 2, Chapter 1) Addition, subtraction, and multiplication (expansion) of polynomials in x, together with factoring, were covered in Chapter 1. The member joining the family in this chapter is the last of the four operations: division.

The integer division 17 ÷ 5 was the computation "17 = 5 × 3 + 2" — finding the quotient 3 and the remainder 2. Polynomial division has exactly the same shape. We divide the dividend A by the divisor B to obtain

A = B × Q + R (the degree of R is lower than the degree of B)

that is, a quotient Q and a remainder R (Figure 8.1). The long-division steps, too, mirror those for numbers: we eliminate the highest-degree term first and work downward.

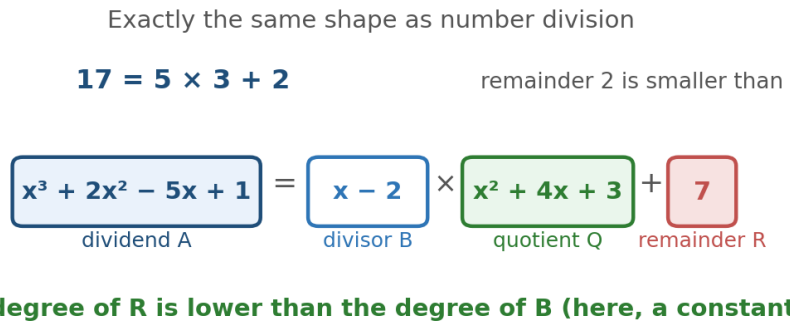


Figure 8.1 Polynomial division. The same "A = B × Q + R" shape as the number division 17 = 5 × 3 + 2.

Term Polynomial division To divide a polynomial A by a polynomial B is to find a quotient Q and a remainder R with A = B × Q + R, where the degree of R is lower than the degree of B. There is exactly one such pair.

Example 8.1

Divide $x^3 + 2x^2 - 5x + 1$ by $x - 2$, and find the quotient and the remainder.

With pen and paper Long division, eliminating from the top: $x^3 \div x = x^2$, so the quotient starts with x^2. Subtracting $x^2(x - 2) = x^3 - 2x^2$ leaves $4x^2 - 5x + 1$, and continuing the same way gives quotient $x^2 + 4x + 3$ and remainder 7. The check is A = B × Q + R: expanding $(x - 2)(x^2 + 4x + 3) +$

7 should restore the original.

Check it with Python sympy's div(A, B, x) returns the quotient and the remainder together. At the end we expand B × Q + R and confirm that it returns to the dividend.

```
from sympy import symbols, div, expand
x = symbols('x')
A = x**3 + 2*x**2 - 5*x + 1
B = x - 2
q, r = div(A, B, x)
print(q)
print(r)
print(expand(B * q + r))   # check
```

Output ▸

```
x**2 + 4*x + 3
7
x**3 + 2*x**2 - 5*x + 1
```

The quotient is $x^2 + 4x + 3$ and the remainder is 7 — and B × Q + R lands right back on the original expression. This "A = B × Q + R" shape is the foundation for the remainder theorem next, and for the factor theorem in Section 8.6.

Now for a curious shortcut. If all you want is the remainder on dividing by x − 2, **you can get it without dividing at all**. Substitute x = 2 into A = B × Q + R: the B = x − 2 part becomes 0, Q vanishes entirely, and only P(2) = R remains (Figure 8.2).

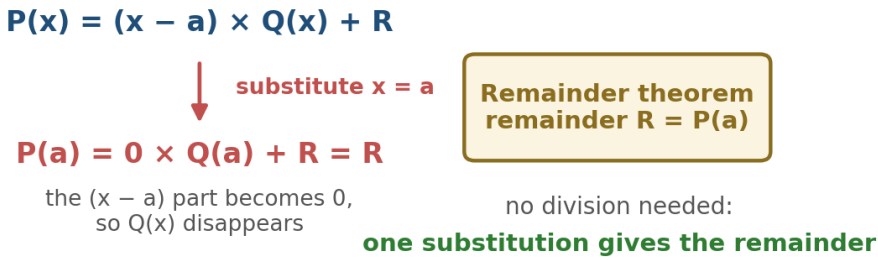


Figure 8.2 The remainder theorem. Substituting x = a wipes out (x − a) × Q(x), and only the remainder shows its face.

Term **Remainder theorem** The remainder when a polynomial P(x) is divided by x − a equals P(a). One substitution finds the remainder — no division required.

Example 8.2

Use the remainder theorem to find the remainder when $P(x) = x^3 + 2x^2 - 5x + 1$ is divided by x − 2, and the remainder when it is divided by x − 1.

With pen and paper By the remainder theorem, the remainder on dividing by x − 2 is $P(2) = 8 + 8 - 10 + 1 = 7$, matching the division in Example 8.1. The remainder on dividing by x − 1 is $P(1) = 1 + 2 - 5 + 1 = -1$.

Check it with Python rem(P, B, x) returns only the remainder of the division. P.subs(x, 2) substitutes a value into an expression — the "P(2)" of the remainder theorem, computed directly. Let us see whether the two agree.

```
from sympy import symbols, rem
x = symbols('x')
P = x**3 + 2*x**2 - 5*x + 1
print(rem(P, x - 2, x))   # remainder from the division
print(P.subs(x, 2))       # P(2)
print(rem(P, x - 1, x))
print(P.subs(x, 1))       # P(1)
```

Output ▸

```
7
7
-1
-1
```

The division remainder and the substituted value agree in both pairs, exactly. "If you only need the remainder, substitution is enough" — in Section 8.6, this little theorem transforms into the master key for higher-degree equations. What can you conclude if you find an a with P(a) = 0? Keep that question in the back of your mind.

8.2 Algebraic Fractions: Fractions in the World of Expressions

Just as numbers have fractions, so do expressions. An **algebraic fraction** is an expression with a variable in its denominator. The rules are the same

as for numerical fractions: to reduce, "cancel the common factor of numerator and denominator"; to combine, "make the denominators match, then add or subtract" (Figure 8.3). For expressions, though, finding the common factor requires **factoring first**. Chapter 1's factoring takes the leading role here as well.

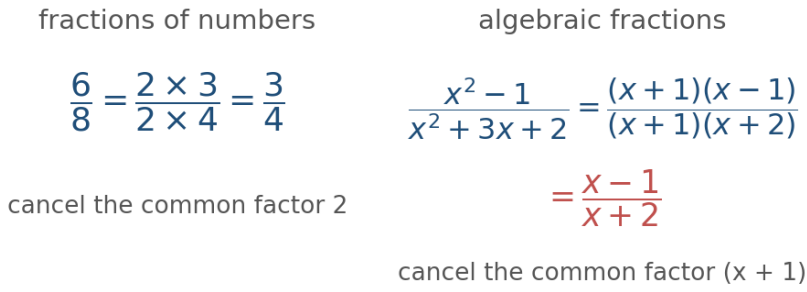


Figure 8.3 Reducing an algebraic fraction. Factor first, then cancel the common factor — exactly the same rule as for numerical fractions.

Term **Algebraic fraction** An expression with a variable in the denominator. Reducing (cancelling common factors) and combining (matching denominators) follow the same rules as numerical fractions. The key to finding common factors is factoring.

Example 8.3

(1) Reduce $\frac{x^2 - 1}{x^2 + 3x + 2}$. (2) Combine $\frac{1}{x-1} + \frac{1}{x+1}$ into a single algebraic fraction.

With pen and paper (1) The numerator factors as (x + 1)(x – 1) and the denominator as (x + 1)(x + 2), so cancelling the common factor x + 1 leaves $\frac{x-1}{x+2}$. (2) Matching the denominators gives $\frac{(x+1)+(x-1)}{(x-1)(x+1)} = \frac{2x}{(x-1)(x+1)}$.

Check it with Python simplify handles the reduction, and together does the combining.

```
from sympy import symbols, simplify, together
x = symbols('x')
f = (x**2 - 1) / (x**2 + 3*x + 2)
print(simplify(f))
g = 1 / (x - 1) + 1 / (x + 1)
print(together(g))
```

Output ▸

```
(x - 1)/(x + 2)
2*x/((x - 1)*(x + 1))
```

Both the reduction and the combination match the hand computation. Algebraic fractions are long-lived tools that reappear in trigonometric functions and calculus from Chapter 10 onward. Note also that the expressions are undefined at values that make a denominator 0 (here $x = \pm 1$ and $x = -2$).

8.3 The Binomial Theorem: Finding Expansion Coefficients at Once

$(a + b)^2 = a^2 + 2ab + b^2$ is an old friend from Chapter 1. But what about $(a + b)^5$? You could multiply it out step by step, yet the expansion coefficients follow a beautiful pattern — revealed by **Pascal's triangle** (Figure 8.4). Each entry is the sum of the two neighbors just above it. And its true identity is none other than the combination ${}_nC_r$ from Chapter 5.

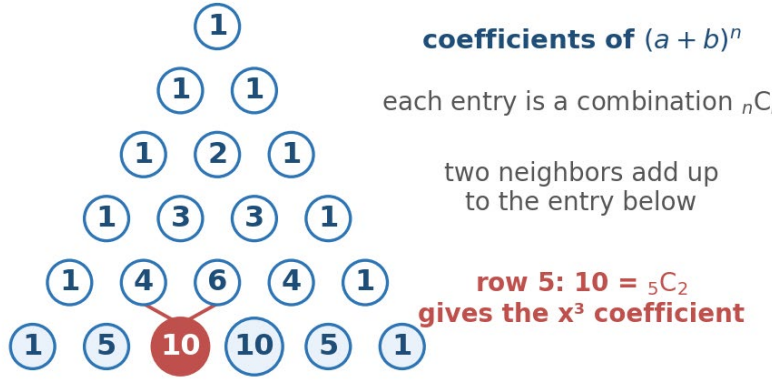


Figure 8.4 Pascal's triangle. Each row lists the expansion coefficients of a power of (a + b), and the entries are combination numbers.

Why do combinations appear? $(a + b)^5$ is a product of 5 copies of (a + b).

An a^3b^2 term arises whenever we decide **which 2 of the 5 parentheses contribute a b**. There are ${}_5C_2 = 10$ such choices, so the coefficient is 10. Generalizing this observation gives the binomial theorem.

> **Term** **Binomial theorem** In the expansion of the n-th power of (a + b), the coefficient of the $a^{n-r}b^r$ term is ${}_nC_r$. Every coefficient of the expansion can be written as a combination number.

Example 8.4

Expand $(x + 2)^5$. Also find the coefficient of the x^3 term directly with the binomial theorem.

With pen and paper The x^3 term arises by choosing the "2" from 2 of the 5 parentheses: ${}_5C_2 \times x^3 \times 2^2 = 10 \times 4 \times x^3 = 40x^3$, so the coefficient is 40. For the full expansion, take row 5 of Pascal's triangle, "1, 5, 10, 10, 5, 1," and attach the powers of 2 as you go.

Check it with Python expand performs the expansion, and the binomial coefficient ${}_nC_r$ is binomial(n, r) — the sympy version of Chapter 5's math.comb, returning the same values.

```
from sympy import symbols, expand, binomial
x = symbols('x')
print(expand((x + 2)**5))
print(binomial(5, 2) * 2**2)   # coefficient of x³
```

Output ▸

```
x**5 + 10*x**4 + 40*x**3 + 80*x**2 + 80*x + 32
40
```

The coefficient of x^3 in the expansion is indeed 40. No tedious multiplication needed — the binomial theorem lets you snipe exactly the coefficient you want. Chapter 5's combinations have reappeared in an unexpected place: the expansion of expressions.

8.4 Complex Numbers: Extending the World of Numbers

Recap (Level 2, Chapter 3) When the graph of a quadratic function misses the x-axis, we said the quadratic equation has "no real solutions." For example, no real number satisfies $x^2 = -1$. In this chapter we step over that wall.

The world of numbers has grown whenever the need arose (Figure 8.5): negative numbers to make subtraction free, fractions to make division free, irrational numbers to solve $x^2 = 2$. Then why not **invent a new number** to solve $x^2 = -1$? That is exactly how the **imaginary unit i** was born. We agree that i is a number satisfying $i^2 = -1$.

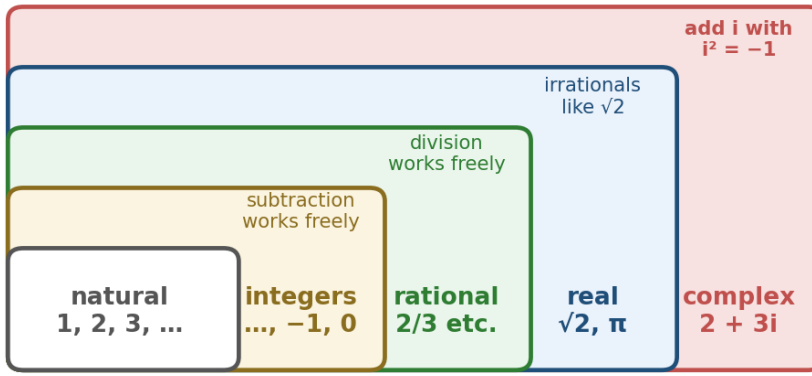


Figure 8.5 Extending the world of numbers. The complex numbers are the outermost layer, built by adjoining i to the reals.

Term **Complex number** A number that can be written as a + bi using real numbers a, b and the imaginary unit i ($i^2 = -1$). a is the real part and b the imaginary part. When b = 0 it is a real number, so the complex numbers contain the reals.

Computing with complex numbers comes down to one rule: **treat i like a variable, and replace i^2 by −1 whenever it appears**. Addition and subtraction work real part with real part, imaginary part with imaginary part; multiplication is expand-then-tidy.

Example 8.5

For $\alpha = 3 + 2i$ and $\beta = 1 - 4i$, compute the sum $\alpha + \beta$ and the product $\alpha\beta$.

With pen and paper The sum, part by part: $(3 + 1) + (2 - 4)i = 4 - 2i$. The product, by expanding: $\alpha\beta = 3 - 12i + 2i - 8i^2 = 3 - 10i + 8 = 11 - 10i$

(the key step is $i^2 = -1$).

Check it with Python In sympy the imaginary unit is written as a capital **I**. Expansion uses the same expand as for real numbers, and I**2 is replaced by −1 automatically.

```
from sympy import I, expand
a = 3 + 2*I
b = 1 - 4*I
print(a + b)
print(expand(a * b))
```

Output ▸

```
4 - 2*I
11 - 10*I
```

The sum is 4 − 2i and the product is 11 − 10i. The hand rule — treat i like a variable — is reproduced by sympy exactly.

Division remains. When i sits in a denominator, we use the **complex conjugate**: the conjugate of a + bi is a − bi, with the sign of the imaginary part flipped. Their product $(a + bi)(a - bi) = a^2 + b^2$ is **always a real number**, so multiplying numerator and denominator by the denominator's conjugate sweeps i out of the denominator — the same idea as rationalizing a denominator that contains a square root.

> **Term** **Complex conjugate** The conjugate of z = a + bi is a − bi. The product of z and its conjugate is $a^2 + b^2$, a real number — the tool for clearing i from a denominator.

Example 8.6

State the conjugate of z = 1 − 4i. Then write $\frac{3 + 2i}{1 - 4i}$ in the form a + bi.

With pen and paper The conjugate is 1 + 4i. For the division, multiply numerator and denominator by 1 + 4i: the denominator becomes $1^2 + 4^2 = 17$, and the numerator $(3 + 2i)(1 + 4i) = 3 + 12i + 2i - 8 = -5 + 14i$. Hence $-\frac{5}{17} + \frac{14}{17}i$.

Check it with Python conjugate returns the conjugate, and simplify takes care of tidying the division.

```
from sympy import I, conjugate, simplify
z = 1 - 4*I
print(conjugate(z))
w = (3 + 2*I) / (1 - 4*I)
print(simplify(w))
```

Output ▸

```
1 + 4*I
-5/17 + 14*I/17
```

It matches the hand computation. With this, all four operations on complex numbers are complete — in the complex world, you can divide by any nonzero complex number. Add, subtract, multiply, and divide by nonzero numbers: the complex numbers form a full-fledged number system with the same rights as the reals.

8.5 Quadratic Equations Revisited: The Discriminant and the Relation between Roots and Coefficients

With complex numbers in hand, let us look again at the quadratic equation $ax^2 + bx + c = 0$. In the quadratic formula $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$, the quantity under the radical, $D = b^2 - 4ac$, is called the **discriminant**. The sign of D splits the solutions cleanly into three cases (Figure 8.6). And when $D < 0$, "no solutions" is no longer the verdict: a negative number under the radical can be written with i, so the equation has exactly 2 solutions — as complex numbers.

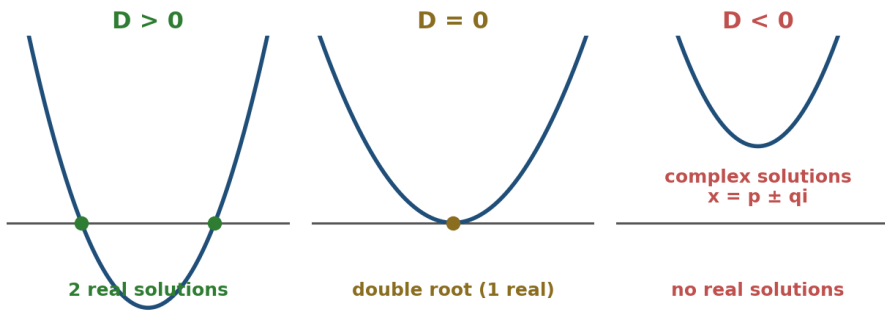


Figure 8.6 Classifying solutions by the discriminant. Even when $D < 0$ the graph does not vanish — the solutions simply become complex.

Term **Discriminant and classification of solutions** For $ax^2 + bx + c = 0$, the discriminant is $D = b^2 - 4ac$. If $D > 0$ there are two distinct real solutions; if $D = 0$, a repeated (double) solution; if $D < 0$, two distinct complex solutions (conjugates of each other).

Example 8.7

Compute the discriminant of $x^2 + 2x + 5 = 0$, and solve the equation over the complex numbers.

With pen and paper $D = 2^2 - 4 \times 1 \times 5 = 4 - 20 = -16 < 0$. The quadratic formula gives $x = \frac{-2 \pm \sqrt{-16}}{2} = \frac{-2 \pm 4i}{2} = -1 \pm 2i$. Rewriting $\sqrt{-16}$ as $4i$ is the new step.

Check it with Python discriminant computes D directly, and solve returns complex solutions as readily as real ones.

```
from sympy import symbols, solve, discriminant
x = symbols('x')
expr = x**2 + 2*x + 5
print(discriminant(expr, x))
print(solve(expr, x))
```

Output ▸

```
-16
[-1 - 2*I, -1 + 2*I]
```

The discriminant is -16 and the solutions are $-1 \pm 2i$ — two of them. The equation Chapter 3 called "unsolvable" has a perfectly good pair of solutions in the complex world. Notice that the two are conjugates: the complex solutions of an equation with real coefficients always arrive as a conjugate pair.

Here is one more powerful tool for quadratics. Write out the two solutions α, β with the quadratic formula and try adding and multiplying them: the radical parts cancel, leaving astonishingly simple expressions.

Term **Relation between roots and coefficients** If α and β are the solutions of $ax^2 + bx + c = 0$, then $\alpha + \beta = -\frac{b}{a}$, $\alpha\beta = \frac{c}{a}$. The sum and product of the solutions can be read off the coefficients — without solving.

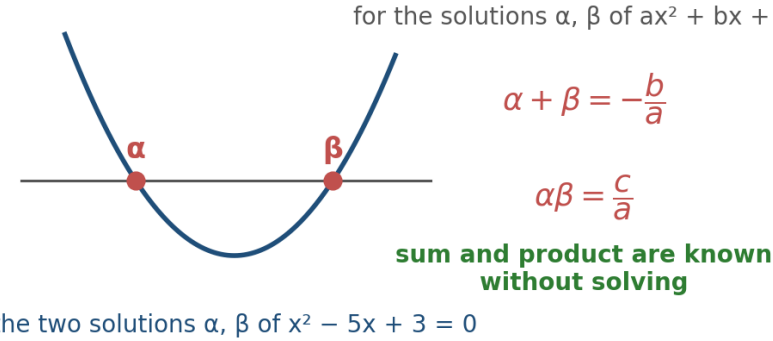


Figure 8.7 The relation between roots and coefficients. The two solutions may be complicated, but their sum and product can be seen in the coefficients alone.

Example 8.8

Let α and β be the two solutions of $x^2 - 5x + 3 = 0$. Find α + β, αβ, and $\alpha^2 + \beta^2$.

With pen and paper By the relation between roots and coefficients, α + β = 5 and αβ = 3. For $\alpha^2 + \beta^2$, use the classic symmetric-expression identity $\alpha^2 + \beta^2 = (\alpha + \beta)^2 - 2\alpha\beta = 25 - 6 = 19$. We never needed the solutions themselves ($\frac{5 \pm \sqrt{13}}{2}$) — the sum and product were enough.

Check it with Python This time we deliberately solve for the roots and check directly that the sum is 5, the product 3, and the sum of squares 19. simplify tidies the computations with radicals.

```
from sympy import symbols, solve, simplify
x = symbols('x')
roots = solve(x**2 - 5*x + 3, x)
a, b = roots
print(simplify(a + b))
print(simplify(a * b))
print(simplify(a**2 + b**2))
```

Output ▸

```
5
3
19
```

Sum 5, product 3, sum of squares 19 — exactly as the relation predicts.

Even though the solutions are the irrational numbers $\frac{5 \pm \sqrt{13}}{2}$, the symmetric

expression comes out as a clean integer. "Talking about the solutions without finding them" — that is the real power of this relation.

8.6 Higher-Degree Equations: Lowering the Degree with the Factor Theorem

At last we take on **higher-degree equations** — degree 3 and beyond. The key is Section 8.1's remainder theorem: the remainder when P(x) is divided by x − a is P(a). So what if **P(a) = 0**? The remainder is 0 — that is, P(x) is **divisible** by x − a, so x − a is a factor of P(x). This is the **factor theorem**.

> **Term** **Factor theorem** If P(a) = 0, then P(x) has x − a as a factor (is divisible by x − a). The converse also holds. The breakthrough tool for factoring higher-degree polynomials.

Here is how to use it. To solve P(x) = 0, first hunt for a value a with P(a) = 0 (the standard tactic: substitute the divisors of the constant term, ±1, ±2, ... in turn). Once found, divide by x − a to lower the degree by one; if what remains is quadratic, finish with the quadratic formula (Figure 8.8).

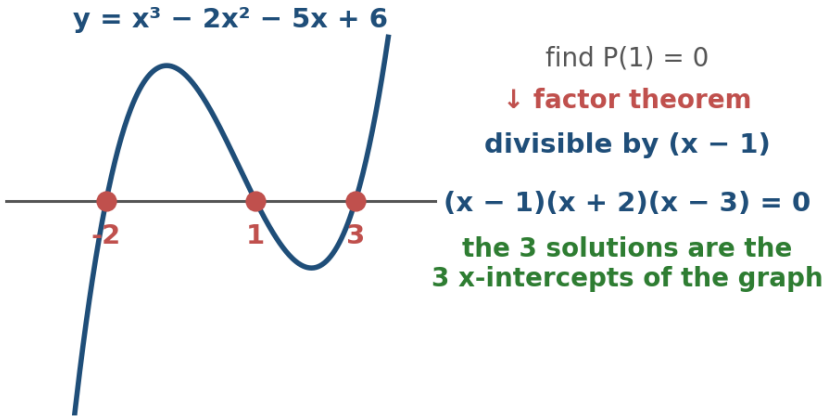


Figure 8.8 Solving a higher-degree equation with the factor theorem. Find P(1) = 0, then divide by x − 1 to lower the degree.

Example 8.9

Factor $P(x) = x^3 - 2x^2 - 5x + 6$.

With pen and paper $P(1) = 1 - 2 - 5 + 6 = 0$, so by the factor theorem x − 1 is a factor. Dividing gives $P(x) = (x - 1)(x^2 - x - 6)$, and factoring the quadratic part yields $(x - 1)(x + 2)(x - 3)$.

Check it with Python subs checks P(1) = 0, and factor — the same tool as in Chapter 1 — handles the factoring. It works on cubics just as well.

```
from sympy import symbols, factor
x = symbols('x')
P = x**3 - 2*x**2 - 5*x + 6
print(P.subs(x, 1))   # P(1) = 0 → x − 1 is a factor
print(factor(P))
```

Output ▸

```
0
(x - 3)*(x - 1)*(x + 2)
```

A complete factorization: $(x - 1)(x + 2)(x - 3)$. The equation $P(x) = 0$ therefore has the solutions $x = 1, -2, 3$ — matching, one for one, the three x-intercepts of the graph in Figure 8.8.

Example 8.10

Solve the equation $x^3 = 8$ over the complex numbers.

With pen and paper Rearrange to $x^3 - 8 = 0$. $P(2) = 0$, so $x - 2$ is a factor, and division gives $(x - 2)(x^2 + 2x + 4) = 0$. The quadratic part has $D = 4 - 16 = -12 < 0$, so its solutions are complex: $x = \frac{-2 \pm 2\sqrt{3}i}{2} = -1 \pm \sqrt{3}i$. The solutions are $x = 2$ and $-1 \pm \sqrt{3}i$ — three in all.

Check it with Python factor confirms the factored form, and solve confirms the three solutions.

```
from sympy import symbols, solve, factor
x = symbols('x')
print(factor(x**3 - 8))
print(solve(x**3 - 8, x))
```

Output ▸

```
(x - 2)*(x**2 + 2*x + 4)
[2, -1 - sqrt(3)*I, -1 + sqrt(3)*I]
```

Hearing "the solutions of $x^3 = 8$," you might picture only 2 — but in the complex world there are 3. In fact, an equation of degree n is known to have exactly n solutions over the complex numbers (counting repeated ones) — the fundamental theorem of algebra. As a reward for extending our numbers to the complex realm, the world of equations has become completely transparent.

Column Imaginary Numbers Were Born from Cubic Equations: The Age of Cardano

"A number whose square is −1" may sound like a mathematician's idle fantasy. History, however, ran the other way: imaginary numbers were born of necessity. The stage is 16th-century Italy, where mathematicians fought public calculation duels over the formula for solving cubic equations. Tartaglia found a method, and Cardano published it in his book — today it is called Cardano's formula. But the formula held a surprise: even for equations whose answers are all real, **square roots of negative numbers could appear midway through the computation**. If one closed one's eyes to the imaginary numbers and pressed on, the calculation would land, in the end, on the correct real answer. Through this strange experience, mathematicians began to accept imaginary numbers as "tools of computation," and complex numbers eventually earned full citizenship in mathematics. The new numbers were born not from fantasy, but from the stubborn desire to solve equations.

Chapter Summary

- Polynomial division: A = B × Q + R (the degree of R is lower than that of B) — the same shape as number division.
- Remainder theorem: the remainder of P(x) divided by x − a is P(a). One substitution finds it.
- Algebraic fractions follow the same rules as numerical ones; the key is factoring.
- Binomial theorem: the coefficients in the expansion of the n-th power of (a + b) are the combinations ${}_nC_r$ (Pascal's triangle).
- Complex numbers a + bi compute with $i^2 = -1$. To divide, multiply by the conjugate of the denominator.
- The discriminant D classifies the solutions; even when D < 0 there are 2 complex solutions. Roots and coefficients: α + β = −b/a, αβ = c/a.
- Factor theorem: if P(a) = 0 then x − a is a factor. Solve higher-degree equations by lowering the degree; an equation of degree n has n complex solutions.

Table 8.1 Python tools used in this chapter

Python tool	What it does
`div(A, B, x)`	polynomial quotient and remainder
`rem(P, B, x)`	only the remainder of the division
`P.subs(x, 2)`	substitute a value into an expression
`expand / factor`	expand or factor (from Chapter 1)
`simplify / together`	reduce / combine algebraic fractions
`binomial(n, r)`	binomial coefficient (combinations)
`I`	imaginary unit i (I**2 becomes –1)
`conjugate(z)`	the complex conjugate
`discriminant(expr, x)`	discriminant of a quadratic

Practice Problems

Solve each problem with pen and paper first, then write Python code in the empty Colab cells to check your answers. Mobilize the whole toolbox of this chapter: div for division, factor for factoring, and I for complex numbers.

Problem 1

Divide $x^3 - 3x^2 + 4x - 5$ by $x - 2$, and find the quotient and the remainder.

Problem 2

Use the remainder theorem to find the remainder when $P(x) = x^3 + 4x^2 - 3x + 2$ is divided by $x + 1$.

Problem 3

Reduce $\frac{x^2 - 4}{x^2 + 5x + 6}$.

Problem 4

Expand $(x + 3)^4$. Also find the coefficient of the x^2 term directly with the binomial theorem.

Problem 5

Compute the following. (1) $(2 + 3i)(2 - 3i)$ (2) $(1 + 2i)^2$

Problem 6

Write $\frac{5 + i}{2 - i}$ in the form $a + bi$.

Problem 7

Compute the discriminant of $x^2 - 4x + 7 = 0$, and solve the equation over

the complex numbers.

Problem 8

Let α and β be the two solutions of $x^2 + 3x - 2 = 0$. Find $\alpha + \beta$, $\alpha\beta$, and $\alpha^2 + \beta^2$.

Problem 9

Factor $P(x) = x^3 + 2x^2 - x - 2$ using the factor theorem.

Problem 10

Solve the equation $x^3 = 27$ over the complex numbers.

Answers and Explanations

Answer 1

By long division (or div): quotient $x^2 - x + 2$, remainder -1. Check: expanding $(x - 2)(x^2 - x + 2) - 1$ restores the original.

```
from sympy import symbols, div
x = symbols('x')
A = x**3 - 3*x**2 + 4*x - 5
q, r = div(A, x - 2, x)
print(q, r)
```

Output ▸

```
x**2 - x + 2 -1
```

Answer 2

Since $x + 1 = x - (-1)$, the remainder theorem gives the remainder $P(-1) = -1 + 4 + 3 + 2 = 8$.

```
from sympy import symbols, rem
x = symbols('x')
P = x**3 + 4*x**2 - 3*x + 2
print(rem(P, x + 1, x))
print(P.subs(x, -1))
```

Output ▸

```
8
8
```

Answer 3

The numerator is $(x + 2)(x - 2)$ and the denominator is $(x + 2)(x + 3)$. Cancelling the common factor $x + 2$ leaves $\frac{x - 2}{x + 3}$.

```
from sympy import symbols, simplify
x = symbols('x')
print(simplify((x**2 - 4) / (x**2 + 5*x + 6)))
```

Output ▸

```
(x - 2)/(x + 3)
```

Answer 4

Take row 4 of Pascal's triangle, "1, 4, 6, 4, 1," and attach the powers of 3. The coefficient of x^2 is ${}_4C_2 \times 3^2 = 6 \times 9 = 54$.

```
from sympy import symbols, expand, binomial
x = symbols('x')
print(expand((x + 3)**4))
print(binomial(4, 2) * 3**2)
```

Output ▸

```
x**4 + 12*x**3 + 54*x**2 + 108*x + 81
54
```

Answer 5

(1) A product of conjugates, so $2^2 + 3^2 = 13$ (a real number). (2) $(1 + 2i)^2 = 1 + 4i + 4i^2 = 1 + 4i - 4 = -3 + 4i$.

```
from sympy import I, expand
print(expand((2 + 3*I) * (2 - 3*I)))
print(expand((1 + 2*I)**2))
```

Output ▸

```
13
-3 + 4*I
```

Answer 6

Multiply numerator and denominator by the conjugate $2 + i$. The denominator becomes $2^2 + 1^2 = 5$, the numerator $(5 + i)(2 + i) = 10 + 5i + 2i - 1 = 9 + 7i$. Hence $\frac{9}{5} + \frac{7}{5}i$.

```
from sympy import I, simplify
print(simplify((5 + I) / (2 - I)))
```

Output ▸

```
9/5 + 7*I/5
```

Answer 7

$D = 16 - 28 = -12 < 0$. The quadratic formula gives $x = \frac{4 \pm 2\sqrt{3}i}{2} = 2 \pm \sqrt{3}i$.

```
from sympy import symbols, solve, discriminant
x = symbols('x')
expr = x**2 - 4*x + 7
print(discriminant(expr, x))
print(solve(expr, x))
```

Output ▸

```
-12
[2 - sqrt(3)*I, 2 + sqrt(3)*I]
```

Answer 8

By the relation between roots and coefficients, $\alpha + \beta = -3$ and $\alpha\beta = -2$. Then $\alpha^2 + \beta^2 = (\alpha + \beta)^2 - 2\alpha\beta = 9 + 4 = 13$.

```
from sympy import symbols, solve, simplify
x = symbols('x')
a, b = solve(x**2 + 3*x - 2, x)
print(simplify(a + b), simplify(a * b))
print(simplify(a**2 + b**2))
```

Output ▸

```
-3 -2
13
```

Answer 9

$P(1) = 1 + 2 - 1 - 2 = 0$, so $x - 1$ is a factor. Dividing gives $P(x) = (x - 1)(x^2 + 3x + 2) = (x - 1)(x + 1)(x + 2)$.

```
from sympy import symbols, factor
x = symbols('x')
P = x**3 + 2*x**2 - x - 2
print(P.subs(x, 1))
print(factor(P))
```

Output ▸

```
0
(x - 1)*(x + 1)*(x + 2)
```

Answer 10

$x^3 - 27 = 0$. From $P(3) = 0$ we get $(x - 3)(x^2 + 3x + 9) = 0$, and the

quadratic part has $D = 9 - 36 = -27 < 0$. The solutions are $x = 3$ and $x = \frac{-3 \pm 3\sqrt{3}i}{2}$ — three in all.

```
from sympy import symbols, solve, factor
x = symbols('x')
print(factor(x**3 - 27))
print(solve(x**3 - 27, x))
```

Output ▸

```
(x - 3)*(x**2 + 3*x + 9)
[3, -3/2 - 3*sqrt(3)*I/2, -3/2 + 3*sqrt(3)*I/2]
```

Chapter 9
Handling Shapes with Coordinates: Geometry and Equations

Points and lines

Equations of lines

Relations between two lines

Equations of circles

Loci

Regions

Regions defined by inequalities

Chapter 9 Handling Shapes with Coordinates: Geometry and Equations

What You Will Learn

In Chapter 7 we studied the properties of triangles and circles, placing figures on coordinates with sympy.geometry to verify our work. In this chapter, the coordinates themselves take center stage. A point is a pair of numbers (x, y); a line or a circle is an **equation** — we translate the whole world of shapes into the world of expressions.

The payoff of this translation is enormous. Distances come from the Pythagorean theorem, intersections become solutions of simultaneous equations, and a "set of points satisfying a condition" (a **locus**) reveals its identity through algebra. Turn the equations into inequalities, and lines and circles begin to describe **regions** — expanses of the plane. Geometry problems become computation problems: that is the idea of analytic geometry, which began with Descartes.

The Python star is sympy.geometry, carried over from Chapter 7, with one new trick: equation(). It extracts **the equation itself** from a Line or a Circle — sympy handles the "shape → expression" translation for us.

By the end of this chapter, you will be able to:

- compute the distance between two points, the midpoint, and internal division points
- build the equation of a line from slope and intercept, or from two points
- test for parallel and perpendicular lines using slopes, and compute the distance from a point to a line
- write the equation of a circle from its center and radius, and recover them from the general form
- classify the relation between a circle and a line by comparing a distance with the radius
- translate a condition into an equation and identify a locus
- read the region defined by inequalities and find maxima and minima over a region

9.1 Points and Distances: Basics of the Coordinate Plane

Recap (Level 2, Chapter 7) Point (a point in the coordinate plane) and A.distance(B) (the distance between two points) from sympy.geometry appeared in Chapter 7. Behind the distance is the Pythagorean theorem. Both remain key players in this chapter.

In the coordinate plane, a point is a pair of numbers (x, y). Let us start with the distance between two points. For A(x_1, y_1) and B(x_2, y_2), build a right triangle whose legs are the horizontal and vertical differences; the Pythagorean theorem then gives $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ (Figure 9.1). The midpoint M of segment AB has coordinates equal to the averages of the two endpoints.

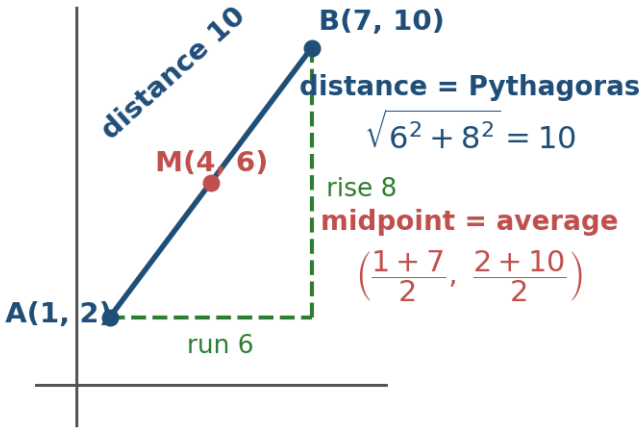


Figure 9.1 The distance between two points comes from the Pythagorean theorem; the midpoint is the average of the coordinates. The language of shapes becomes the language of computation.

Term Distance and midpoint Distance AB = $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. The midpoint M is $(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2})$ (the average of the coordinates).

Example 9.1

Find the distance between the points A(1, 2) and B(7, 10), and the coordinates of the midpoint of segment AB.

With pen and paper The horizontal difference is $7 - 1 = 6$ and the vertical difference is $10 - 2 = 8$. The distance is $\sqrt{6^2 + 8^2} = \sqrt{100} = 10$. The midpoint is the average, $(\frac{1+7}{2}, \frac{2+10}{2}) = (4, 6)$.

Check it with Python distance is familiar from Chapter 7. The midpoint is "the average of the coordinates," so it can be written as (A + B) / 2.

```
from sympy import Point
A = Point(1, 2)
B = Point(7, 10)
print(A.distance(B))
print((A + B) / 2)
```

Output ▸

```
10
Point2D(4, 6)
```

The distance is 10 and the midpoint is (4, 6). Run 6, rise 8, hypotenuse 10 — a scaled-up version of the 3 : 4 : 5 right triangle that closed Chapter 7.

Let us generalize the midpoint. The point P that splits segment AB so that AP : PB = m : n is called the **internal division point**. Its coordinates are a weighted version of the average, and the key is that **the nearer endpoint gets the larger weight** (Figure 9.2). Setting m = n recovers the midpoint formula.

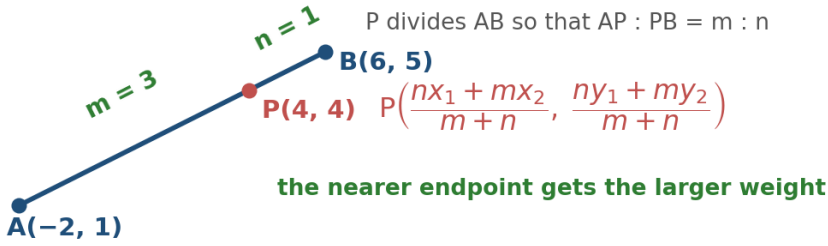


Figure 9.2 The internal division point. To divide in the ratio m : n, average the endpoints with the larger weight on the nearer one.

Term Internal division point The point P dividing segment AB internally in the ratio m : n has coordinates $(\frac{nx_1 + mx_2}{m+n}, \frac{ny_1 + my_2}{m+n})$. With m = n it is the midpoint.

Example 9.2

Find the coordinates of the point P that divides the segment joining A(−2, 1) and B(6, 5) internally in the ratio 3 : 1.

With pen and paper With m = 3 and n = 1, the x-coordinate is $\frac{1 \times (-2) + 3 \times 6}{3 + 1} = \frac{16}{4} = 4$ and the y-coordinate is $\frac{1 \times 1 + 3 \times 5}{3 + 1} = \frac{16}{4} = 4$, so P(4, 4).

Check it with Python The formula is "the average of A weighted by n and B weighted by m." A Point can be multiplied by numbers and added, so the formula goes straight into code.

```
from sympy import Point
A = Point(-2, 1)
B = Point(6, 5)
m, n = 3, 1
P = (A * n + B * m) / (m + n)
print(P)
```

Output ▸

```
Point2D(4, 4)
```

P(4, 4). When dividing 3 : 1, P sits closer to B, so B carries the larger weight 3 — the formula embodies "the nearer endpoint gets the larger weight."

9.2 Equations of Lines

> **Recap (Level 1B)** That the graph of a linear function y = ax + b is a straight line, with slope a and intercept b, was covered in Level 1B. In this chapter we recast the line: from "the graph of a function" to "the shape an equation describes."

Every point (x, y) on the line satisfies the relation y = ax + b; conversely, collecting all points that satisfy the relation reproduces the line. In other words, **y = ax + b is the equation of the line** (Figure 9.3). Given two points, the slope a is "the change in y divided by the change in x," and choosing b so that the line passes through one of the points completes the equation.

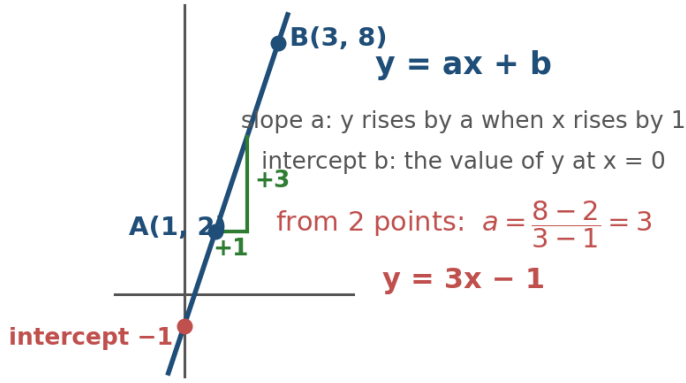


Figure 9.3 The equation of a line, y = ax + b. Once the slope a and the intercept b are fixed, the line is determined uniquely.

Term **Equation of a line** The line with slope a and intercept b is y = ax + b. The slope of the line through (x_1, y_1) and (x_2, y_2) is $a = \frac{y_2 - y_1}{x_2 - x_1}$. The general form ax + by + c = 0 also covers lines parallel to the y-axis.

Example 9.3

Find the equation of the line through the two points A(1, 2) and B(3, 8).

With pen and paper The slope is $a = \frac{8-2}{3-1} = \frac{6}{2} = 3$. Since y = 3x + b passes through A(1, 2), we get 2 = 3 + b, so b = −1. The equation is y = 3x − 1.

Check it with Python Line(A, B) builds the line through two points; slope gives the slope and equation(x, y) extracts the equation. equation returns "the left side of = 0," so the output −6x + 2y + 2 means the general form −6x + 2y + 2 = 0. Solving for y with solve brings back the familiar y = 3x − 1.

```
from sympy import Point, Line, symbols, solve
x, y = symbols('x y')
L = Line(Point(1, 2), Point(3, 8))
print(L.slope)
print(L.equation(x, y))
print(solve(L.equation(x, y), y)[0])
```

Output ▸
```
3
-6*x + 2*y + 2
3*x - 1
```

Slope 3, equation $-6x + 2y + 2 = 0$, and solving for y gives $y = 3x - 1$. One line, two faces: the general form and the y = form. equation() — extracting the expression from the shape — is this chapter's new trick.

9.3 Two Lines: Parallel, Perpendicular, and Distance

The relation between two lines can be read off their slopes at a glance. They are **parallel** when the slopes are equal, and **perpendicular** when the product of the slopes is −1 (Figure 9.4, left). For instance, a line perpendicular to one of slope 2 has slope −1/2: while one moves 1 right and 2 up, the other moves 2 right and 1 down — rotating by 90° swaps the run and the rise and flips a sign.

> **Term** **Parallel and perpendicular** For two lines $y = ax + b$ and $y = cx + d$: parallel ⇔ $a = c$ (equal slopes); perpendicular ⇔ $ac = -1$ (the product of the slopes is −1).

Example 9.4

Consider the line $y = 2x + 3$. (1) Decide whether the line through (1, 0) with slope 2 is parallel to it. (2) Find the equation of the line through (2, 1) perpendicular to it.

With pen and paper (1) Both slopes are 2, so the lines are parallel. (2) The perpendicular slope satisfies $2 \times a = -1$, so $a = -1/2$. Since $y = -\frac{1}{2}x + b$ passes through (2, 1), we get $1 = -1 + b$, $b = 2$. Hence $y = -\frac{1}{2}x + 2$.

Check it with Python is_parallel tests for parallelism, and perpendicular_line(P) builds "the perpendicular through P" in one stroke (a cousin of Chapter 7's perpendicular_bisector).

```
from sympy import Point, Line, symbols
x, y = symbols('x y')
L1 = Line(Point(0, 3), slope=2)    # y = 2x + 3
L3 = Line(Point(1, 0), slope=2)
print(L1.is_parallel(L3))
L2 = L1.perpendicular_line(Point(2, 1))
print(L2.slope)
print(L2.equation(x, y))
```

Output ▸

```
True
-1/2
-x - 2*y + 4
```

The parallel test returns True; the perpendicular has slope −1/2 and equation −x − 2y + 4 = 0. Solving for y gives y = −x/2 + 2, matching the hand computation.

Here is one more powerful formula. The **distance** from a point to a line — the length of the perpendicular dropped from the point — can be computed with a single substitution, using the general form ax + by + c = 0 (Figure 9.4, right).

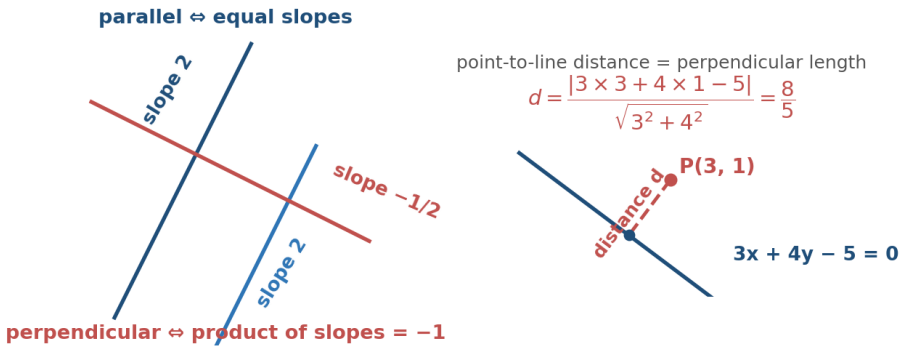


Figure 9.4 Parallel and perpendicular are decided by slopes (left). The distance from a point to a line is the length of the perpendicular, given by a single formula (right).

Term **Distance from a point to a line** The distance from the point (x₀, y₀) to the line ax + by + c = 0 is $d = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$. The numerator is the absolute value of the left side evaluated at the point.

Example 9.5

Find the distance between the point P(3, 1) and the line $3x + 4y - 5 = 0$.

With pen and paper Substituting into the formula: $d = \frac{|3 \times 3 + 4 \times 1 - 5|}{\sqrt{3^2 + 4^2}} = \frac{8}{5}$. The denominator is $\sqrt{25} = 5$.

Check it with Python Build the Line from two points on $3x + 4y - 5 = 0$, say $(-1, 2)$ and $(3, -1)$; then L.distance(P) returns the distance from the point to the line.

```
from sympy import Point, Line
L = Line(Point(-1, 2), Point(3, -1))   # 3x + 4y - 5 = 0
print(L.distance(Point(3, 1)))
```

Output ▸

```
8/5
```

The distance is 8/5. This point-to-line distance becomes the key for classifying the relation between a circle and a line in Section 9.4.

9.4 Equations of Circles

A circle is "the set of points at equal distance r from the center" (Chapter 7). With center (a, b), a point (x, y) on the circle satisfies "distance from (a, b) equals r," and squaring the distance formula gives **$(x - a)^2 + (y - b)^2 = r^2$** — the **equation of the circle** (Figure 9.5, left). To recover the center and radius from the expanded general form $x^2 + y^2 + lx + my + n = 0$, complete the square in x and in y. Chapter 3's completing the square stars again, this time for circles.

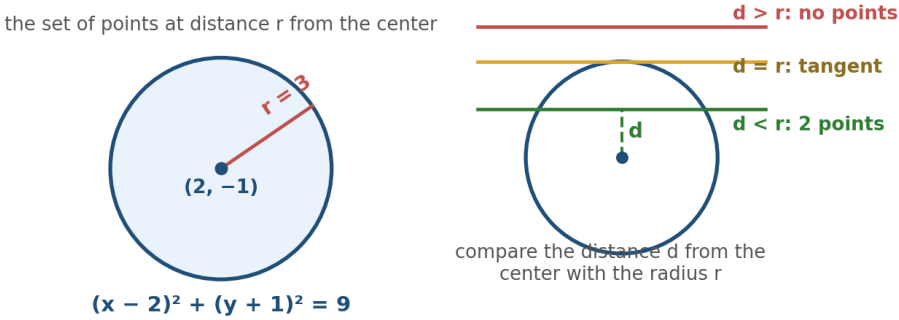


Figure 9.5 The equation of a circle (left), and the relation between a circle and a line (right): compare the distance d from the center to the line with the radius r.

Term **Equation of a circle** The circle with center (a, b) and radius r is $(x - a)^2 + (y - b)^2 = r^2$. From the general form $x^2 + y^2 + lx + my + n = 0$, read off the center and radius by completing the square.

Example 9.6

Write the equation of the circle with center (2, −1) and radius 3, and also its expanded general form.

With pen and paper $(x - 2)^2 + (y + 1)^2 = 9$. Expanding gives $x^2 - 4x + 4 + y^2 + 2y + 1 = 9$, which tidies up to $x^2 + y^2 - 4x + 2y - 4 = 0$.

Check it with Python Circle(center, radius) builds the circle, and equation(x, y) extracts its equation (returning "the left side of = 0," just as for lines). expand produces the general form.

```
from sympy import Point, Circle, symbols
x, y = symbols('x y')
C = Circle(Point(2, -1), 3)
print(C.equation(x, y))
print(C.equation(x, y).expand())
```

Output ▸

```
(x - 2)**2 + (y + 1)**2 - 9
x**2 - 4*x + y**2 + 2*y - 4
```

$(x - 2)^2 + (y + 1)^2 - 9 = 0$ and its expansion $x^2 + y^2 - 4x + 2y - 4 = 0$: two faces of the same circle. Starting from the general form, completing the square rewinds this process and reveals the center and radius.

How do we decide whether a circle and a line intersect? The key is Section 9.3's point-to-line distance. **Compare the distance d from the center to the line with the radius r**: if $d < r$ they meet at 2 points, if $d = r$ the line is tangent, and if $d > r$ they do not meet (Figure 9.5, right). The intersection points themselves come from solving the circle's and the line's equations simultaneously.

Term **Circle and line** Classify by the distance d from the center to the line and the radius r: $d < r \Leftrightarrow$ 2 intersection points, $d = r \Leftrightarrow$ tangent, $d > r \Leftrightarrow$ no common points.

Example 9.7

Find the common points of the circle $x^2 + y^2 = 25$ and the line $y = x + 1$.

Also compute the distance from the center to the line and confirm "2 intersection points" from the distance as well.

With pen and paper Substituting $y = x + 1$ into the circle: $x^2 + (x + 1)^2 = 25$, which tidies to $2x^2 + 2x - 24 = 0$, that is, $x^2 + x - 12 = (x + 4)(x - 3) = 0$. So $x = -4, 3$, and the common points are $(-4, -3)$ and $(3, 4)$. The distance from the center $(0, 0)$ to the line $x - y + 1 = 0$ is $\frac{|1|}{\sqrt{2}} = \frac{\sqrt{2}}{2}$, smaller than the radius 5, so the line indeed crosses at 2 points.

Check it with Python intersection, our intersection tool from Chapter 7, works for a circle-and-line pair as well. The distance is C.center.distance(L).

```
from sympy import Point, Circle, Line
C = Circle(Point(0, 0), 5)
L = Line(Point(0, 1), slope=1)   # y = x + 1
print(C.intersection(L))
print(C.center.distance(L))
```

Output ▸

```
[Point2D(-4, -3), Point2D(3, 4)]
sqrt(2)/2
```

The common points are $(-4, -3)$ and $(3, 4)$. The distance from the center, $\frac{\sqrt{2}}{2} \approx 0.71$, is smaller than the radius 5, agreeing with the classification. "Shapes intersecting" translates into "simultaneous equations having solutions" — the power of coordinates.

9.5 Loci: Collections of Points Satisfying a Condition

Recap (Level 2, Chapter 7) In Section 7.3 we learned to see a perpendicular bisector as "the set of points equidistant from 2 points" and a circle as "the set of points equidistant from the center." A locus carries out this "collection of points satisfying a condition" idea with coordinates.

When a point moves while satisfying some condition, the figure it traces is called a **locus**. With coordinates, locus-hunting becomes a mechanical procedure: call the moving point P(x, y), translate the condition into an expression in x and y, tidy up, and read off which figure the equation describes — that is all.

Here is a classic. What figure do the points trace whose **distances to two fixed points have a constant ratio**? If the ratio is 1 : 1, the answer is the perpendicular bisector — but for, say, 2 : 1, the answer turns out to be a circle (the **circle of Apollonius**, Figure 9.6).

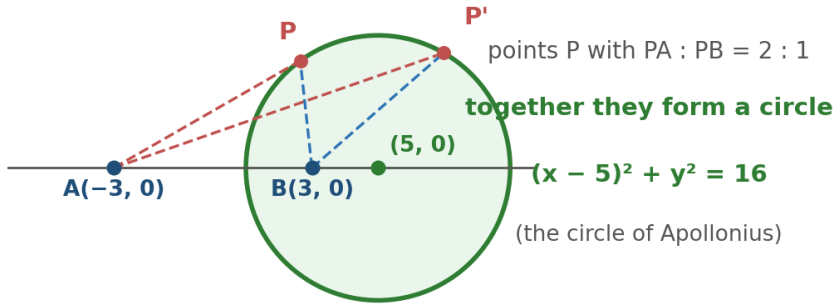


Figure 9.6 The locus of points P with PA : PB = 2 : 1. Together they form a circle (the circle of Apollonius).

Term **Locus** The figure traced by the points satisfying a condition. Method: call the moving point P(x, y) → translate the condition into an expression → tidy up and read off the figure.

Example 9.8

Find the locus of points P whose distances to A(−3, 0) and B(3, 0) satisfy PA : PB = 2 : 1.

With pen and paper Let P(x, y). Squaring both sides of PA = 2PB (the trick that avoids radicals) gives $PA^2 = 4PB^2$, that is, $(x + 3)^2 + y^2 = 4\{(x - 3)^2 + y^2\}$. Expanding and tidying: $-3x^2 + 30x - 3y^2 - 27 = 0$; dividing both sides by −3 gives $x^2 + y^2 - 10x + 9 = 0$. Completing the square: $(x - 5)^2 + y^2 = 16$ — the circle with center (5, 0) and radius 4.

Check it with Python Tidy the condition $PA^2 - 4PB^2$ with expand and divide by −3 to get the general form. Then build "the circle with center (5, 0) and radius 4" with Circle, expand its equation, and compare the two expressions side by side.

```
from sympy import symbols, Point, Circle
x, y = symbols('x y')
cond = (x + 3)**2 + y**2 - 4*((x - 3)**2 + y**2)
print((cond / -3).expand())
C = Circle(Point(5, 0), 4)
print(C.equation(x, y).expand())
```

Output ▸

```
x**2 - 10*x + y**2 + 9
x**2 - 10*x + y**2 + 9
```

The tidied condition and the expanded equation of the circle $(x - 5)^2 + y^2 = 16$ agree perfectly. The true identity of the condition "distance ratio 2 : 1" was a circle. The locus triple jump — condition → expression → figure — has been confirmed by sympy from both ends.

9.6 Regions: The World Described by Inequalities

Finally, let us replace the "=" in an equation with an inequality sign. $y = x + 1$ is a line, but $\mathbf{y > x + 1}$ is "all the points above the line" — an expanse of the plane, a **region** (Figure 9.7, left). Likewise $x^2 + y^2 < 25$ is the inside of a circle (Figure 9.7, right). Mind whether the boundary is included: ≦ and ≧ include it (draw a solid boundary), while < and > exclude it (draw a dashed one).

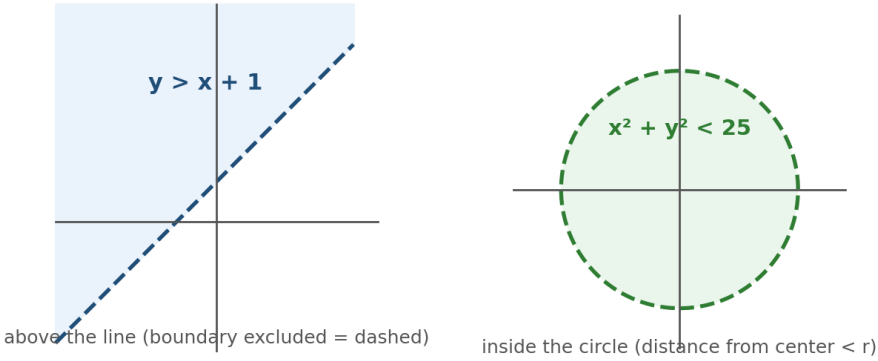


Figure 9.7 Regions described by inequalities: above a line (left) and inside a circle (right). Dash the boundary when it is excluded.

Term **Region described by an inequality** $y > f(x)$ is the region above the graph; $y < f(x)$ is below. For circles, $(x - a)^2 + (y - b)^2 < r^2$ is the inside and $> r^2$ the outside. ≦ and ≧ include the boundary.

Example 9.9

Decide whether each of the three points (1, 3), (4, 2), and (−2, 1) belongs to the region described by the simultaneous inequalities $y > x + 1$ and $x^2 + y^2 < 25$.

With pen and paper Substitute each point into both inequalities; the point is in the region exactly when both hold. (1, 3): $3 > 2$ ✓ and $1 + 9 = 10 < 25$ ✓ → inside. (4, 2): $2 > 5$ ✗ → outside. (−2, 1): $1 > -1$ ✓ and $4 + 1 = 5 < 25$ ✓ → inside.

Check it with Python Python's comparison operators > and < return True when the inequality holds and False otherwise. Joining two conditions with and expresses "both hold" — and the test function is done.

```
def inside(px, py):
    return py > px + 1 and px**2 + py**2 < 25
for p in [(1, 3), (4, 2), (-2, 1)]:
    print(p, inside(p[0], p[1]))
```

Output ▸

```
(1, 3) True
(4, 2) False
(-2, 1) True
```

The verdicts are True, False, True. "Belonging to the region" means "satisfying all the inequalities" — in the language of sets (Chapter 2), it is a membership test for the intersection of the two regions.

The most practical use of regions is **maximizing or minimizing over a region**. For example, over the region described by $x \geqq 0$, $y \geqq 0$, $x + y \leqq 4$, and $2x + y \leqq 6$ (the quadrilateral in Figure 9.8), how large can $x + 2y$ get? Setting $x + 2y = k$, this is a line of slope $-1/2$. As k grows the line slides upward, so **the last moment it can still touch the region happens at a corner**.

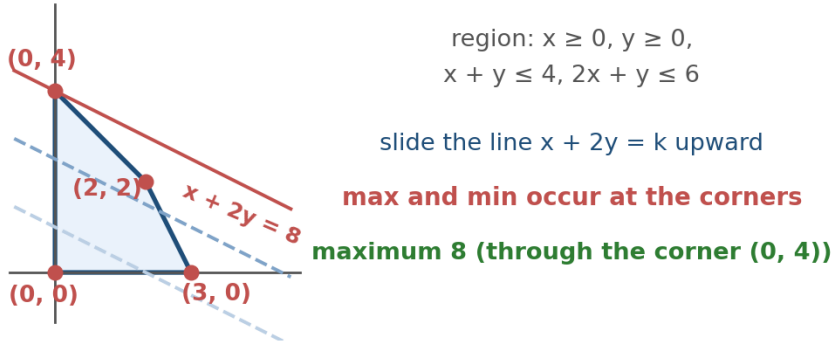


Figure 9.8 A region from simultaneous inequalities, and a maximum. Sliding the line x + 2y = k, the maximum and minimum occur at corners of the region.

Example 9.10

Over the region described by $x \geqq 0$, $y \geqq 0$, $x + y \leqq 4$, and $2x + y \leqq 6$, find the maximum value of $x + 2y$.

With pen and paper The region is a quadrilateral with corners (0, 0), (3, 0), (0, 4), and the intersection (2, 2) of the lines $x + y = 4$ and $2x + y = 6$. The values of $x + 2y$ there are 0, 3, 8, and 6, so the maximum is 8 (at the corner (0, 4)).

Check it with Python Three corners are obvious, so we list them; the remaining one comes from solving the two lines simultaneously with solve. Then evaluate $x + 2y$ at each corner.

```
from sympy import symbols, solve, Eq
x, y = symbols('x y')
corners = [(0, 0), (3, 0), (0, 4)]
P = solve([Eq(x + y, 4), Eq(2*x + y, 6)], [x, y])
corners.append((P[x], P[y]))
for cx, cy in corners:
    print((cx, cy), cx + 2*cy)
```

Output ▸

```
(0, 0) 0
(3, 0) 3
(0, 4) 8
(2, 2) 6
```

The maximum is 8, at the corner (0, 4). Maximizing profit under limited resources — this idea is called linear programming, and it is used everywhere in real decision-making, from factory production plans to

transport optimization. Regions of inequalities are mathematics with a direct line to the real world.

Column Descartes and the Invention of Coordinates: The Marriage of Geometry and Algebra

The coordinate plane we have used so casually in this chapter is credited to the 17th-century philosopher René Descartes (his contemporary Fermat reached the same idea independently). Before then, geometry problems were solved in the ancient Greek style of construction and argument — a world where victory hinged on a flash of insight about the right auxiliary line. Descartes's revolution was to represent points by pairs of numbers and translate figures into equations. Geometry problems thereby became algebra — **computations with a fixed procedure** — solvable by anyone, inspiration or not. Born of the marriage between geometry (shapes) and algebra (expressions), the method is called analytic geometry, and it laid the groundwork on which calculus would later be built. The pixels on a computer screen and the positions of game characters are all coordinates: Descartes's invention quietly supports our daily lives 400 years on.

Chapter Summary

- Distances come from the Pythagorean theorem. The midpoint is the average of coordinates; the internal division point is a weighted average with "the larger weight on the nearer endpoint."
- The equation of a line is $y = ax + b$. From two points, slope = change in y ÷ change in x. The general form is $ax + by + c = 0$.
- Parallel ⇔ equal slopes; perpendicular ⇔ product of slopes = -1. The point-to-line distance comes from a single formula.
- The equation of a circle is $(x - a)^2 + (y - b)^2 = r^2$. From the general form, complete the square to read off center and radius.
- Classify a circle and a line by comparing the distance d from the center to the line with the radius r.
- A locus is the triple jump "let P(x, y) → translate the condition → tidy up and read off the figure."
- Inequalities describe regions (solid boundary if included, dashed if not). For maxima and minima over a region, look at the corners.

Table 9.1 Python tools used in this chapter

Python tool	What it does
`Point(x, y)`	a point in the plane (from Chapter 7)
`A.distance(B)`	distance between two points
`Line(A, B)`	line through two points (or slope=)
`L.slope`	the slope of a line
`L.equation(x, y)`	extract the equation of a line / circle
`L.is_parallel(M)`	test whether two lines are parallel
`L.perpendicular_line(P)`	the perpendicular through point P
`L.distance(P)`	distance from a point to a line
`Circle(center, radius)`	a circle (intersection works too)
`solve([eq1, eq2])`	solve simultaneous equations

Practice Problems

Solve each problem with pen and paper first, then write Python code in the empty Colab cells to check your answers. Mobilize Point, Line, Circle, and equation() — the full "shape ⇄ expression" translation kit.

Problem 1

Find the distance between the points A(2, 3) and B(8, 11), and the coordinates of the midpoint of segment AB.

Problem 2

Find the coordinates of the point that divides the segment joining A(1, −2) and B(9, 6) internally in the ratio 1 : 3.

Problem 3

Find the slope and the equation of the line through the two points A(2, 1) and B(4, 7).

Problem 4

Find the slope and the equation of the line through (3, 2) perpendicular to the line $y = 3x - 2$.

Problem 5

Find the distance between the point P(1, 2) and the line $4x + 3y - 20 = 0$.

Problem 6

Write the equation of the circle with center (−1, 2) and radius 5, and also its

expanded general form.

Problem 7

Find the common points of the circle $x^2 + y^2 = 10$ and the line $y = 3x$.

Problem 8

Find the locus of points P whose distances to A(−5, 0) and B(5, 0) satisfy PA : PB = 3 : 2.

Problem 9

Decide whether each of the three points (2, 3), (0, 5), and (−1, −2) belongs to the region described by the simultaneous inequalities $y < 2x + 3$ and $(x - 1)^2 + y^2 < 16$.

Problem 10

Over the region described by $x \geqq 0$, $y \geqq 0$, $x + 2y \leqq 8$, and $3x + y \leqq 9$, find the maximum value of $x + y$.

Answers and Explanations

Answer 1

The horizontal difference is 6 and the vertical difference is 8, so the distance is $\sqrt{36 + 64} = 10$. The midpoint is the average, (5, 7).

```
from sympy import Point
A = Point(2, 3)
B = Point(8, 11)
print(A.distance(B))
print((A + B) / 2)
```

Output ▸

```
10
Point2D(5, 7)
```

Answer 2

With m = 1 and n = 3, the x-coordinate is $\frac{3 \times 1 + 1 \times 9}{4} = 3$ and the y-coordinate is $\frac{3 \times (-2) + 1 \times 6}{4} = 0$, so the point is (3, 0). Dividing closer to A, the larger weight 3 goes to A.

```
from sympy import Point
A = Point(1, -2)
B = Point(9, 6)
print((A * 3 + B * 1) / 4)
```

Output ▸
```
Point2D(3, 0)
```

Answer 3

The slope is $\frac{7-1}{4-2} = \frac{6}{2} = 3$. Since $y = 3x + b$ passes through A(2, 1), $b = -5$. The equation is $y = 3x - 5$.

```
from sympy import Point, Line, symbols, solve
x, y = symbols('x y')
L = Line(Point(2, 1), Point(4, 7))
print(L.slope)
print(solve(L.equation(x, y), y)[0])
```

Output ▸
```
3
3*x - 5
```

Answer 4

Perpendicularity gives $3 \times a = -1$, so $a = -1/3$. Since $y = -\frac{1}{3}x + b$ passes through (3, 2), we get $2 = -1 + b$, $b = 3$. Hence $y = -\frac{1}{3}x + 3$ (in general form, $x + 3y - 9 = 0$).

```
from sympy import Point, Line, symbols
x, y = symbols('x y')
L1 = Line(Point(0, -2), slope=3)   # y = 3x - 2
L2 = L1.perpendicular_line(Point(3, 2))
print(L2.slope)
print(L2.equation(x, y))
```

Output ▸
```
-1/3
-x - 3*y + 9
```

Answer 5

Substituting into the formula: $d = \frac{|4 \times 1 + 3 \times 2 - 20|}{\sqrt{4^2 + 3^2}} = \frac{10}{5} = 2$.

```
from sympy import Point, Line
L = Line(Point(2, 4), Point(5, 0))   # 4x + 3y - 20 = 0
print(L.distance(Point(1, 2)))
```

Output ▸
```
2
```

Answer 6

$(x + 1)^2 + (y - 2)^2 = 25$. Expanding and tidying gives $x^2 + y^2 + 2x - 4y - 20 = 0$.

```
from sympy import Point, Circle, symbols
x, y = symbols('x y')
C = Circle(Point(-1, 2), 5)
print(C.equation(x, y))
print(C.equation(x, y).expand())
```

Output ▸

```
(x + 1)**2 + (y - 2)**2 - 25
x**2 + 2*x + y**2 - 4*y - 20
```

Answer 7

Substituting $y = 3x$ gives $x^2 + 9x^2 = 10$, so $x^2 = 1$ and $x = \pm 1$. The common points are $(1, 3)$ and $(-1, -3)$.

```
from sympy import Point, Circle, Line, sqrt
C = Circle(Point(0, 0), sqrt(10))
L = Line(Point(0, 0), slope=3)   # y = 3x
print(C.intersection(L))
```

Output ▸

```
[Point2D(-1, -3), Point2D(1, 3)]
```

Answer 8

Let $P(x, y)$. Squaring both sides of $2PA = 3PB$ gives $4PA^2 = 9PB^2$. Tidying up yields $x^2 + y^2 - 26x + 25 = 0$, and completing the square gives $(x - 13)^2 + y^2 = 144$ — the circle with center $(13, 0)$ and radius 12 (a circle of Apollonius).

```
from sympy import symbols, Point, Circle
x, y = symbols('x y')
cond = 4*((x + 5)**2 + y**2) - 9*((x - 5)**2 + y**2)
print((cond / -5).expand())
C = Circle(Point(13, 0), 12)
print(C.equation(x, y).expand())
```

Output ▸

```
x**2 - 26*x + y**2 + 25
x**2 - 26*x + y**2 + 25
```

Answer 9

(2, 3): $3 < 7$ ✓ and $1 + 9 = 10 < 16$ ✓ → inside. (0, 5): $5 < 3$ ✗ → outside. (−1, −2): $-2 < 1$ ✓ and $4 + 4 = 8 < 16$ ✓ → inside.

```
def inside(px, py):
    return py < 2*px + 3 and (px - 1)**2 + py**2 < 16
for p in [(2, 3), (0, 5), (-1, -2)]:
    print(p, inside(p[0], p[1]))
```

Output ▸

```
(2, 3) True
(0, 5) False
(-1, -2) True
```

Answer 10

The corners are (0, 0), (3, 0), (0, 4), and the intersection (2, 3) of $x + 2y = 8$ and $3x + y = 9$. The values of $x + y$ there are 0, 3, 4, and 5, so the maximum is 5 (at the corner (2, 3)).

```
from sympy import symbols, solve, Eq
x, y = symbols('x y')
corners = [(0, 0), (3, 0), (0, 4)]
P = solve([Eq(x + 2*y, 8), Eq(3*x + y, 9)], [x, y])
corners.append((P[x], P[y]))
for cx, cy in corners:
    print((cx, cy), cx + cy)
```

Output ▸

```
(0, 0) 0
(3, 0) 3
(0, 4) 4
(2, 3) 5
```

Chapter 10
Reading Periodic Change: Trigonometric Functions

General angles

Radian measure

Graphs of trigonometric functions

Period

Addition formulas

Double-angle and half-angle formulas

Combining a sine and a cosine

Trigonometric equations and inequalities

Chapter 10 Reading Periodic Change: Trigonometric Functions

What You Will Learn

Day and night, the seasons, waves, pendulums, the beating of a heart — the world is full of change that repeats. The stars of this chapter are the trigonometric functions, the mathematical language of **periodic change**.

The trigonometric ratios sin, cos, and tan from Chapter 4 lived in the world of triangle angles — from 0° to 180°. In this chapter we widen the very notion of angle to **general angles** — direction and amount of rotation — and introduce a new unit, **radian measure**. The trigonometric ratios are then reborn as **functions** defined for all real numbers, and their graphs become beautiful waves.

The second half is an armory of computation. From the **addition formulas** to the **double-angle and half-angle formulas**, and on to **combining a sine and a cosine into one wave** — the formulas grow out of one another like a chain. The finale is **trigonometric equations and inequalities**: with the unit circle, even equations in the world of waves can be solved with a picture.

The Python stars are sympy's sin, cos, tan and the constant pi, joined by a newcomer, expand_trig, which expands expressions with the addition formulas. Let sympy double-check every formula you compute by hand.

By the end of this chapter, you will be able to:

- understand general angles as rotations, and convert between degrees and radians
- compute arc lengths and sector areas with radian measure
- find trigonometric values of any angle using the unit circle
- sketch the graphs and read off period, amplitude, and shift
- compute values and expressions with the addition, double-angle, and half-angle formulas
- combine a $\sin\theta + b\cos\theta$ into a single sine wave
- solve trigonometric equations and inequalities on the unit circle

10.1 General Angles and Radian Measure: Widening the World of Angles

Triangle angles run from 0° to 180°. But a Ferris-wheel gondola or a clock hand keeps turning past one full turn, and past two. So let us recast an angle not as "a corner of a shape" but as an **amount of rotation**: how far a half-line from the center (the **radius arm**) has turned from its starting position (the initial ray). Agreeing that counterclockwise is positive and clockwise is negative, angles like 390° and −60° become perfectly respectable. These are **general angles** (Figure 10.1).

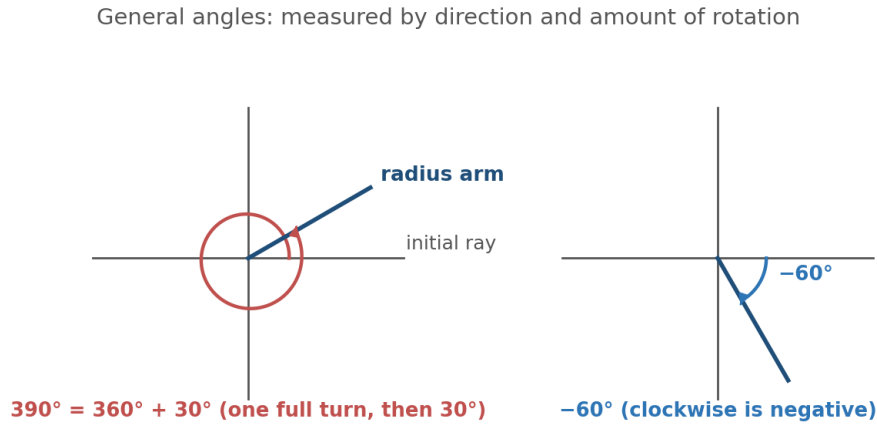


Figure 10.1 General angles: measured by the direction and amount of rotation, counterclockwise positive. 390° is "one full turn, then 30°."

We also introduce a new unit for angles. On a circle of radius 1, measure an angle by **the length of the arc it cuts** — this is **radian measure**. The circumference is 2π, so one full turn = 2π radians; that is, 180° = π radians. With this one correspondence, conversion is just a proportion (Figure 10.2). The payoff: the formulas for arc length and sector area become dramatically simple, and radians are the standard unit when calculus arrives in Chapter 13.

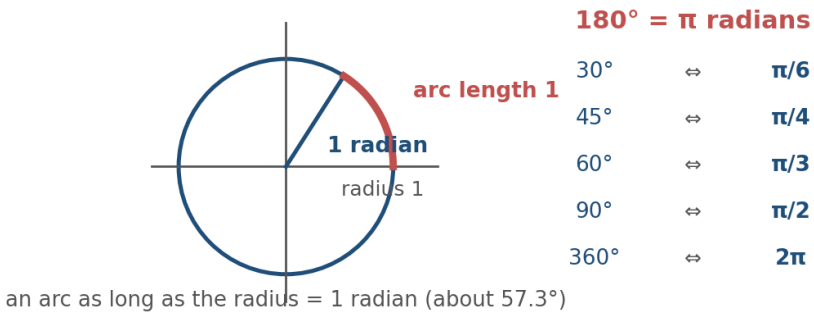


Figure 10.2 Radian measure. An arc as long as the radius spans 1 radian; convert with 180° = π radians.

Term **Radian measure** Measuring angles by arc length on a circle of radius 1. 180° = π radians. For radius r and central angle θ (in radians), the arc length is l = rθ and the sector area is $S = \frac{1}{2}r^2\theta$.

Example 10.1

(1) Express 60° and 225° in radians. (2) Find the arc length and the area of a sector with radius 6 and central angle $\frac{\pi}{3}$.

With pen and paper (1) Use 180° = π and proportions. 60° is one third of 180°, so $\frac{\pi}{3}$; and 225° is $\frac{225}{180}\pi = \frac{5}{4}\pi$. (2) The arc length is l = rθ = $6 \times \frac{\pi}{3} = 2\pi$, and the area is $S = \frac{1}{2} \times 6^2 \times \frac{\pi}{3} = 6\pi$.

Check it with Python sympy's pi keeps the circle constant exact and symbolic. Degrees convert to radians via "× pi / 180." The formulas l = rθ and $S = \frac{1}{2}r^2\theta$ go straight into code.

```
from sympy import pi
print(60 * pi / 180)        # 60° in radians
print(225 * pi / 180)       # 225° in radians
r = 6
theta = pi / 3
print(r * theta)            # arc length
print(r**2 * theta / 2)     # sector area
```

Output ▸
```
pi/3
5*pi/4
2*pi
6*pi
```

60° = π/3, 225° = 5π/4, arc length 2π, area 6π. sympy carries π as a symbol rather than a decimal, so the answers come out exactly as in a textbook. From here on, all angles in this chapter are in radians.

10.2 Trigonometric Functions: Ratios Set Free on the Unit Circle

Recap (Level 2, Chapter 4) We defined sin, cos, and tan as side ratios in a right triangle, then used coordinates to extend them from 0° to 180°. The viewpoint "the y-coordinate of P is sin, the x-coordinate is cos" now takes the leading role.

For general angles, the trigonometric functions are defined on the **unit circle** (center at the origin, radius 1). Let P be the point where the radius arm of angle θ meets the unit circle; then

cos θ = the x-coordinate of P, sin θ = the y-coordinate of P, tan θ = the slope of line OP

(Figure 10.3). The radius arm can turn through any angle, so sin and cos take values for every real θ. On 0°–180° this agrees exactly with the Chapter 4 definition, so it is a legitimate extension of the trigonometric ratios. And when P goes once around the circle, everything repeats — this "repetition" creates the waves of the next section.

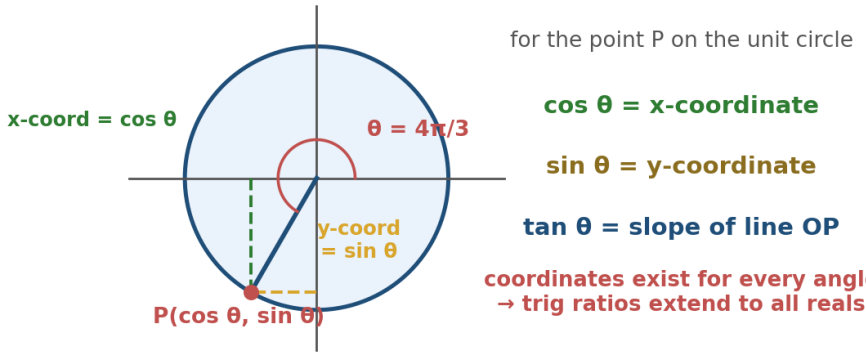


Figure 10.3 The unit-circle definition. cos θ is the x-coordinate and sin θ the y-coordinate — defined for every angle.

Term **Trigonometric functions** For the point P determined by angle θ on the unit circle: cos θ = x-coordinate, sin θ = y-coordinate, tan θ = $\frac{\sin\theta}{\cos\theta}$ (the slope of OP). The identity $\sin^2\theta + \cos^2\theta = 1$ is exactly the statement that P lies on the unit circle (the Pythagorean theorem).

Example 10.2

Find sin θ, cos θ, and tan θ for $\theta = \frac{4}{3}\pi$.

With pen and paper $\frac{4}{3}\pi = 240°$ is a third-quadrant angle. The radius arm meets the unit circle at $P(-\frac{1}{2}, -\frac{\sqrt{3}}{2})$, so cos θ = $-\frac{1}{2}$, sin θ = $-\frac{\sqrt{3}}{2}$, and tan θ = sin θ ÷ cos θ = $\sqrt{3}$.

Check it with Python sympy's sin, cos, and tan take radians and, for the standard angles, return exact values with radicals.

```
from sympy import sin, cos, tan, pi
print(sin(4*pi/3))
print(cos(4*pi/3))
print(tan(4*pi/3))
```

Output ▸

```
-sqrt(3)/2
-1/2
sqrt(3)
```

sin is $-\frac{\sqrt{3}}{2}$, cos is $-\frac{1}{2}$, tan is $\sqrt{3}$. In the third quadrant both coordinates are negative, so sin and cos are negative, while the slope is positive, so tan is positive — the signs match the unit-circle picture.

Example 10.3

Suppose θ is a second-quadrant angle with sin θ = $\frac{3}{5}$. Find cos θ and tan θ.

With pen and paper From the identity $\sin^2\theta + \cos^2\theta = 1$ we get $\cos^2\theta = 1 - \frac{9}{25} = \frac{16}{25}$. In the second quadrant the x-coordinate is negative, so cos θ = $-\frac{4}{5}$, and tan θ = sin θ ÷ cos θ = $-\frac{3}{4}$.

Check it with Python Hold the fractions exactly with Rational and

compute the identity as written. Choosing the sign for the quadrant (negative square root here) is the human's job.

```
from sympy import Rational, sqrt
s = Rational(3, 5)
c = -sqrt(1 - s**2)   # quadrant 2, so cos is negative
print(c)
print(s / c)          # tan = sin ÷ cos
```

Output ▸

```
-4/5
-3/4
```

cos θ = −4/5 and tan θ = −3/4. The identity from Chapter 4 carries over to general angles unchanged: as long as P sits on the unit circle, the Pythagorean relation $sin^2\theta + cos^2\theta = 1$ never lets you down.

10.3 Graphs and Periods of Trigonometric Functions

Plot θ on the horizontal axis and y = sin θ on the vertical, and a smooth wave appears (Figure 10.4). Each time the radius arm makes one full turn (2π), the y-coordinate of P repeats the same motion, so the wave repeats every 2π. This "width of repetition" is called the **period**. The cos graph is the same wave shape — the sin graph shifted by −π/2 in the θ direction. tan has period π and a graph of its own.

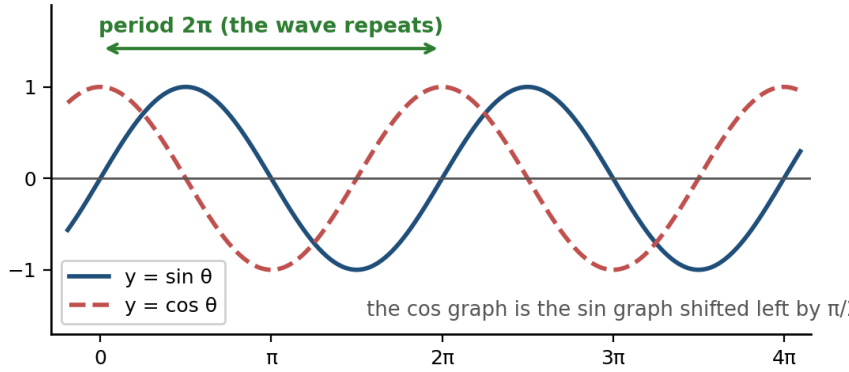


Figure 10.4 The graphs of y = sin θ and y = cos θ: waves of period 2π, with cos the sin graph shifted left by π/2.

Term Period If f(θ + T) = f(θ) holds for every θ, the smallest such positive T is the period. sin and cos have period 2π; tan has period π.

Example 10.4

Verify that $\sin(\theta + 2\pi) = \sin\theta$, $\sin 2(\theta + \pi) = \sin 2\theta$, and $\tan(\theta + \pi) = \tan\theta$, and state the periods of $y = \sin 2\theta$ and $y = \tan\theta$.

With pen and paper One full turn brings the radius arm back, so $\sin(\theta + 2\pi) = \sin\theta$. For $y = \sin 2\theta$, when θ advances by π, the inner 2θ advances by 2π, so the period is π — **doubling θ halves the period**. In general $y = \sin k\theta$ has period $\frac{2\pi}{k}$. And tan returns to the same slope after half a turn, so its period is π.

Check it with Python Check with simplify that "the expression shifted by one period" minus "the original" is 0.

```
from sympy import symbols, sin, tan, pi, simplify
t = symbols('t')
print(simplify(sin(t + 2*pi) - sin(t)))
print(simplify(sin(2*(t + pi)) - sin(2*t)))
print(simplify(tan(t + pi) - tan(t)))
```

Output ▸

```
0
0
0
```

All three differences are 0 — sin shifted by 2π, and $\sin 2\theta$ and tan shifted by π, coincide with the originals exactly. The period of $y = \sin 2\theta$ is π, and the period of $y = \tan\theta$ is π.

> **Recap (Level 2, Chapter 3)** The graph of $y = a(x - p)^2 + q$ is the parabola $y = ax^2$ shifted by p in x and q in y. The reading "a subtraction inside the formula means a shift" works for trigonometric graphs in exactly the same way.

The shape of a wave can be read straight off its formula: the graph of **$y = a \sin k(\theta - p)$** is a wave with amplitude (height) $|a|$, period $\frac{2\pi}{k}$, shifted by p in the θ direction (Figure 10.5).

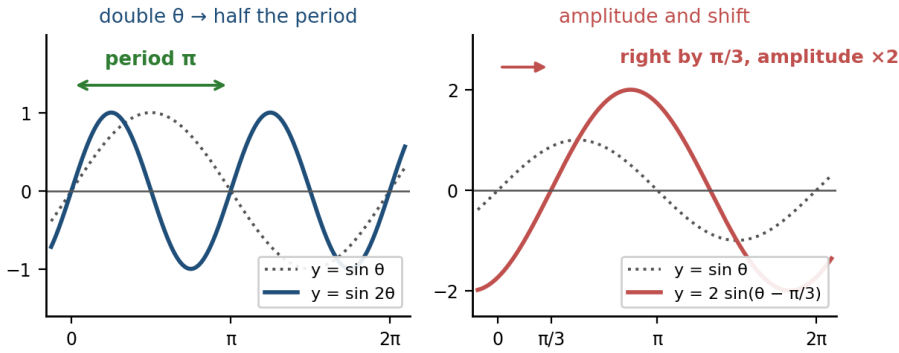


Figure 10.5 Doubling θ halves the period (left). The coefficient a sets the amplitude; θ − p is a shift (right).

Example 10.5

State the amplitude and period of $y = 2\sin(\theta - \frac{\pi}{3})$. Then find the values at $\theta = \frac{\pi}{3}, \frac{5}{6}\pi$, and $\frac{11}{6}\pi$.

With pen and paper Amplitude 2, period 2π: the wave $y = 2\sin\theta$ shifted right by $\frac{\pi}{3}$. At $\theta = \frac{\pi}{3}$ the inside is 0, so $y = 0$. At $\theta = \frac{5}{6}\pi$ the inside is $\frac{\pi}{2}$, so $y = 2$ (the maximum). At $\theta = \frac{11}{6}\pi$ the inside is $\frac{3}{2}\pi$, so $y = -2$ (the minimum).

Check it with Python Name the expression f and substitute the three values of θ in turn with subs.

```
from sympy import symbols, sin, pi
t = symbols('t')
f = 2 * sin(t - pi/3)
print(f.subs(t, pi/3))
print(f.subs(t, 5*pi/6))   # θ giving the maximum
print(f.subs(t, 11*pi/6))  # θ giving the minimum
```

Output ▸

```
0
2
-2
```

0, 2, −2 — the wave's starting point, crest, and trough match the computation exactly. The coefficient a sets the height; θ − p sets the horizontal position. When you can "see" the graph in the formula, this section has done its job.

10.4 The Addition Formulas: Adding Angles

Can you find sin 75°? The angle 75° is not in the standard table, but it splits as 45° + 30°. "Express the trigonometric values of a sum of angles using the original angles" — that is what the **addition formulas** do (Figure 10.6). They are the parent of the whole formula family: the double-angle, half-angle, and combination formulas that follow are all their children.

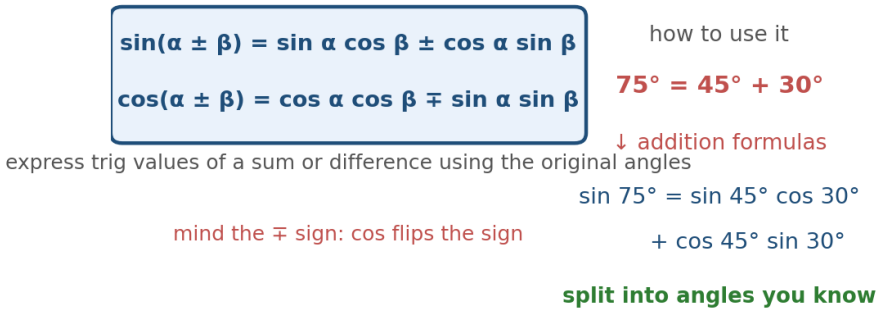


Figure 10.6 The addition formulas express trig values of a sum or difference of angles using the original angles. Mind the flipped sign for cos.

Term **Addition formulas** $\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta$, $\cos(\alpha \pm \beta) = \cos\alpha\cos\beta \mp \sin\alpha\sin\beta$. For cos, the sign on the right is flipped (signs correspond in order).

Example 10.6

Use the addition formulas to find sin 75°.

With pen and paper Split $75° = 45° + 30°$: $\sin 75° = \sin 45°\cos 30° + \cos 45°\sin 30° = \frac{\sqrt{2}}{2}\times\frac{\sqrt{3}}{2} + \frac{\sqrt{2}}{2}\times\frac{1}{2} = \frac{\sqrt{6}+\sqrt{2}}{4}$.

Check it with Python Convert 75° to radians and hand it to sin: sympy returns the exact value. We also check that its difference from the hand computation is 0.

```
from sympy import sin, pi, sqrt, simplify
angle = 75 * pi / 180
print(simplify(sin(angle)))
hand = (sqrt(6) + sqrt(2)) / 4   # the hand computation
print(simplify(sin(angle) - hand))
```

Output ▸
```
sqrt(2)/4 + sqrt(6)/4
0
```

sympy's answer $\frac{\sqrt{2}}{4} + \frac{\sqrt{6}}{4}$ is the same number as the hand result $\frac{\sqrt{6}+\sqrt{2}}{4}$ (their difference is 0). Angles missing from the table become computable once the addition formulas split them into angles you know.

Example 10.7

Let α and β be first-quadrant angles with $\sin\alpha = \frac{3}{5}$ and $\cos\beta = \frac{5}{13}$. Find $\cos(\alpha+\beta)$ and $\sin(\alpha+\beta)$.

With pen and paper First fill in the partners: in the first quadrant, $\cos\alpha = \frac{4}{5}$ and $\sin\beta = \frac{12}{13}$ (from the identity). Then $\cos(\alpha+\beta) = \frac{4}{5}\times\frac{5}{13} - \frac{3}{5}\times\frac{12}{13} = -\frac{16}{65}$ and $\sin(\alpha+\beta) = \frac{3}{5}\times\frac{5}{13} + \frac{4}{5}\times\frac{12}{13} = \frac{63}{65}$.

Check it with Python Hold the four values with Rational and compute the right-hand sides as written.

```
from sympy import Rational
sa, ca = Rational(3, 5), Rational(4, 5)
sb, cb = Rational(12, 13), Rational(5, 13)
print(ca*cb - sa*sb)   # cos(α+β)
print(sa*cb + ca*sb)   # sin(α+β)
```

Output ▸
```
-16/65
63/65
```

$\cos(\alpha+\beta) = -16/65$ and $\sin(\alpha+\beta) = 63/65$. The cosine came out negative because $\alpha+\beta$ has passed 90° into the second quadrant. Without knowing the angles themselves, the sin and cos values alone determine the trigonometric values of their sum — the power of the addition formulas.

10.5 Double-Angle, Half-Angle, and Combining Waves

Set $\alpha = \beta$ in the addition formulas: $\sin(\alpha+\alpha) = \sin\alpha\cos\alpha + \cos\alpha\sin\alpha$, that is, **$\sin 2\alpha = 2\sin\alpha\cos\alpha$**. Likewise $\cos 2\alpha = \cos^2\alpha - \sin^2\alpha$, which the identity rewrites as $1 - 2\sin^2\alpha$ or $2\cos^2\alpha - 1$. These are the **double-angle formulas**. Reading the cos double-angle formula backwards yields formulas for half an angle — the **half-angle formulas**.

Term **Double-angle and half-angle formulas** $\sin 2\alpha = 2\sin\alpha\cos\alpha$, $\cos 2\alpha = \cos^2\alpha - \sin^2\alpha = 1 - 2\sin^2\alpha = 2\cos^2\alpha - 1$. Solving the cos formula backwards gives the half-angle formulas $\sin^2\frac{\alpha}{2} = \frac{1-\cos\alpha}{2}$, $\cos^2\frac{\alpha}{2} = \frac{1+\cos\alpha}{2}$. All of them are children of the addition formulas.

Example 10.8

Let θ be a first-quadrant angle with $\sin\theta = \frac{3}{5}$. Find $\sin 2\theta$ and $\cos 2\theta$.

With pen and paper In the first quadrant, $\cos\theta = \frac{4}{5}$. The double-angle formulas give $\sin 2\theta = 2 \times \frac{3}{5} \times \frac{4}{5} = \frac{24}{25}$ and $\cos 2\theta = 1 - 2 \times \frac{9}{25} = \frac{7}{25}$.

Check it with Python Let us also see that the two forms of the cosine double angle ($1 - 2\sin^2\theta$ and $\cos^2\theta - \sin^2\theta$) give the same value.

```
from sympy import Rational
s, c = Rational(3, 5), Rational(4, 5)
print(2*s*c)         # sin2θ
print(1 - 2*s**2)    # cos2θ
print(c**2 - s**2)   # another form of cos2θ
```

Output ▸

```
24/25
7/25
7/25
```

$\sin 2\theta = 24/25$ and $\cos 2\theta = 7/25$, whichever form we compute. Even after doubling the angle, the sin and cos of the original angle are all we need — true children of the addition formulas.

The finale of this section is **combining a sine and a cosine**. The "sum of two waves" $a\sin\theta + b\cos\theta$ collapses, in fact, into a single sine wave. Plot the point (a, b), call its distance from the origin r and its angle from the x-axis α; then

$$a\sin\theta + b\cos\theta = r\sin(\theta + \alpha) \quad (r = \sqrt{a^2 + b^2})$$

(Figure 10.7). Once the waves merge into one, the maximum and minimum can be read off at a glance: ±r.

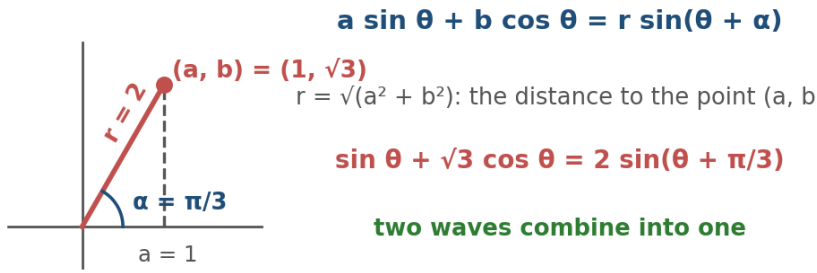


Figure 10.7 Combining waves. Read the distance r and angle α of the point (a, b), and two waves merge into a single sine.

Term **Combining a sine and a cosine** $a \sin\theta + b \cos\theta = r \sin(\theta + \alpha)$, where $r = \sqrt{a^2 + b^2}$ and α is the angle of the point (a, b). The maximum is r and the minimum is −r.

Example 10.9

Combine $\sin\theta + \sqrt{3}\cos\theta$ into the form $r \sin(\theta + \alpha)$, and state the maximum value.

With pen and paper Plot the point $(1, \sqrt{3})$: then $r = \sqrt{1 + 3} = 2$ and the angle is $\alpha = \frac{\pi}{3}$. Hence $\sin\theta + \sqrt{3}\cos\theta = 2\sin(\theta + \frac{\pi}{3})$, and the maximum value is 2.

Check it with Python Check with simplify that the difference between the two sides is 0, then open the combined form with expand_trig and watch it return to the original.

```
from sympy import symbols, sin, cos, sqrt, pi
from sympy import simplify, expand_trig
t = symbols('t')
left = sin(t) + sqrt(3) * cos(t)
right = 2 * sin(t + pi/3)
print(simplify(left - right))
print(expand_trig(right))
```

Output ▸

```
0
sin(t) + sqrt(3)*cos(t)
```

The difference is 0, and expand_trig opens the combined form right back into $\sin\theta + \sqrt{3}\cos\theta$. Combination is the addition formulas played in

reverse: two scattered waves become one, and the maximum value 2 can be read directly from the formula.

10.6 Trigonometric Equations and Inequalities: Solving on the Unit Circle

The finishing touch: equations and inequalities containing trigonometric functions. Take $\sin\theta = \frac{1}{2}$ $(0 \leqq \theta < 2\pi)$. We already own the right tool — the **unit circle**. Since $\sin\theta$ is the y-coordinate, hunt for the intersections of the unit circle with the line $y = \frac{1}{2}$, and the radius arms of the solutions appear (Figure 10.8, left). For an inequality, read off the **arc** on which the condition holds (Figure 10.8, right).

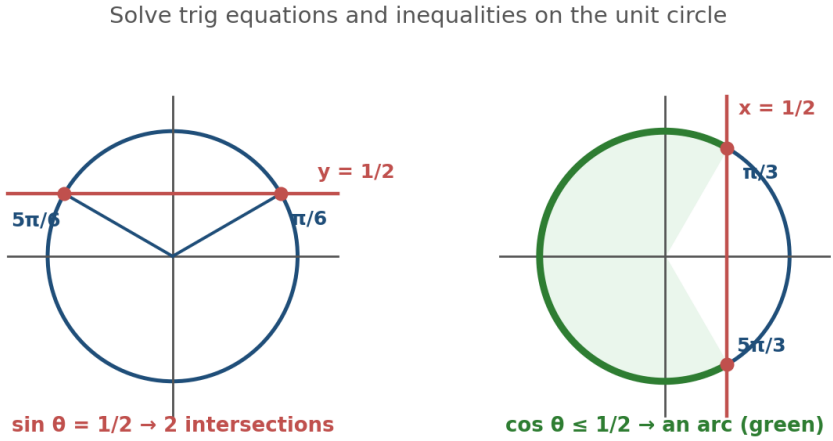


Figure 10.8 Solve trigonometric equations and inequalities on the unit circle: intersections with a line for equations, an arc for inequalities.

Term Trigonometric equations and inequalities Solve on the unit circle: $\sin\theta = k$ means intersections with the line $y = k$; $\cos\theta = k$, with the line $x = k$. For inequalities, read the arc on which the condition holds. Because of periodicity, mind the specified range (such as $0 \leqq \theta < 2\pi$).

Example 10.10

For $0 \leqq \theta < 2\pi$, solve the following. (1) $\sin\theta = \frac{1}{2}$ (2) the inequality $\cos\theta \leqq \frac{1}{2}$

With pen and paper (1) The unit circle meets $y = \frac{1}{2}$ at two points: $\theta = \frac{\pi}{6}$

and $\frac{5}{6}\pi$. (2) $\cos\theta = \frac{1}{2}$ at $\theta = \frac{\pi}{3}$ and $\frac{5}{3}\pi$. The x-coordinate is at most $\frac{1}{2}$ on the arc between them, so $\frac{\pi}{3} \leqq \theta \leqq \frac{5}{3}\pi$.

Check it with Python solve returns the solutions of a trigonometric equation. For (2), solve the boundary equation $\cos\theta = \frac{1}{2}$ and read the range off the unit-circle picture.

```
from sympy import symbols, sin, cos, solve, Rational
t = symbols('t')
print(solve(sin(t) - Rational(1, 2), t))
print(solve(cos(t) - Rational(1, 2), t))
```

Output ▸

```
[pi/6, 5*pi/6]
[pi/3, 5*pi/3]
```

The solutions of (1) are $\pi/6$ and $5\pi/6$; the boundary of (2) is $\pi/3$ and $5\pi/3$, giving $\frac{\pi}{3} \leqq \theta \leqq \frac{5}{3}\pi$. The unit-circle picture and solve agree. Because trigonometric functions are periodic, an equation has infinitely many solutions unless a range is specified — make it a habit to check "over which range am I solving?"

Column Trigonometric Functions Are the Language of Waves: Sound, Electricity, Smartphones

It is no accident that the graphs of trigonometric functions look like waves — the waves of the world really do have this shape. The vibration of air from a tuning fork is a pure sine wave, and the alternating voltage in a household outlet is a sin curve repeating 50 or 60 times per second. What about the complex tone of a musical instrument? In the 19th century the mathematician Fourier discovered that **every periodic wave can be expressed as a sum of simple sine waves**. This "Fourier analysis" is the beating heart of modern digital technology. When you listen to music on a smartphone, save a photo as a JPEG, or communicate by radio waves, an enormous number of sines and cosines are being added together behind the scenes. The combining of waves you learned in this chapter — two waves into one — is the doorway into that grand world.

Chapter Summary

- General angles measure direction and amount of rotation. In radian measure, 180° = π radians; arc length l = rθ, sector area S = $\frac{1}{2}r^2\theta$.
- Trigonometric functions are defined on the unit circle: cos θ = x-coordinate, sin θ = y-coordinate, tan θ = slope. The 0°–180° ratios extend to all real numbers.
- sin and cos have period 2π; tan has period π. y = a sin k(θ − p) has amplitude |a|, period $\frac{2\pi}{k}$, and a shift of p in θ.
- Addition formulas: sin(α ± β) = sin α cos β ± cos α sin β, cos(α ± β) = cos α cos β ∓ sin α sin β.
- Double angles come from setting α = β in the addition formulas; solving the cos formula backwards gives the half-angle formulas.
- Combining: a sin θ + b cos θ = r sin(θ + α) with r = $\sqrt{a^2 + b^2}$. Maximum r, minimum −r at a glance.
- Solve trigonometric equations and inequalities on the unit circle: intersections with a line, or an arc. Mind the specified range.

Table 10.1 Python tools used in this chapter

Python tool	What it does
`sin(x) / cos(x) / tan(x)`	trig functions (angles in radians)
`pi`	the constant π, kept exact
`deg * pi / 180`	convert degrees to radians
`Rational(3, 5)`	exact fractions (from Chapter 8)
`sqrt(3)`	exact square roots
`simplify(expr)`	tidy a trigonometric expression
`expand_trig(expr)`	expand with the addition formulas
`f.subs(t, value)`	substitute a value (from Chapter 8)
`solve(expr, t)`	solve a trigonometric equation

Practice Problems

Solve each problem with pen and paper first, then write Python code in the empty Colab cells to check your answers. Angles in radians, values exact

with Rational and sqrt — mobilize the whole toolbox of this chapter.

Problem 1

Express 150° in radians. Also find the arc length and the area of a sector with radius 4 and central angle $\frac{\pi}{4}$.

Problem 2

Find sin θ, cos θ, and tan θ for $\theta = \frac{5}{6}\pi$.

Problem 3

Suppose θ is a fourth-quadrant angle with $\cos\theta = \frac{5}{13}$. Find sin θ and tan θ.

Problem 4

State the period of $y = \cos 3\theta$. Then compute the difference from the original for the expression shifted by π and for the expression shifted by $\frac{2}{3}\pi$, and confirm.

Problem 5

State the amplitude and period of $y = 3\sin(\theta + \frac{\pi}{6})$. Then find the values at $\theta = 0$ and $\theta = \frac{\pi}{3}$.

Problem 6

Use the addition formulas to find cos 15°.

Problem 7

Let α and β be first-quadrant angles with $\sin\alpha = \frac{4}{5}$ and $\cos\beta = \frac{12}{13}$. Find $\sin(\alpha - \beta)$ and $\cos(\alpha - \beta)$.

Problem 8

Let θ be a first-quadrant angle with $\cos\theta = \frac{1}{3}$. Find sin 2θ and cos 2θ.

Problem 9

Combine $\sqrt{3}\sin\theta - \cos\theta$ into the form $r\sin(\theta + \alpha)$, and state the maximum and minimum values.

Problem 10

For $0 \leqq \theta < 2\pi$, solve the following equations. (1) $\cos\theta = -\frac{1}{2}$ (2) $\tan\theta = 1$

Answers and Explanations

Answer 1

$150° = \frac{150}{180}\pi = \frac{5}{6}\pi$. The arc length is $l = 4 \times \frac{\pi}{4} = \pi$, and the area is $S = \frac{1}{2} \times 16 \times \frac{\pi}{4} = 2\pi$.

```
from sympy import pi
print(150 * pi / 180)
r = 4
theta = pi / 4
print(r * theta)
print(r**2 * theta / 2)
```

Output ▸

```
5*pi/6
pi
2*pi
```

Answer 2

$\frac{5}{6}\pi = 150°$ is a second-quadrant angle. The point on the unit circle is $(-\frac{\sqrt{3}}{2}, \frac{1}{2})$, so $\sin\theta = \frac{1}{2}$, $\cos\theta = -\frac{\sqrt{3}}{2}$, and $\tan\theta = -\frac{1}{\sqrt{3}} = -\frac{\sqrt{3}}{3}$.

```
from sympy import sin, cos, tan, pi
print(sin(5*pi/6))
print(cos(5*pi/6))
print(tan(5*pi/6))
```

Output ▸

```
1/2
-sqrt(3)/2
-sqrt(3)/3
```

Answer 3

From the identity, $sin^2\theta = 1 - \frac{25}{169} = \frac{144}{169}$. In the fourth quadrant the y-coordinate is negative, so $\sin\theta = -\frac{12}{13}$ and $\tan\theta = -\frac{12}{5}$.

```
from sympy import Rational, sqrt
c = Rational(5, 13)
s = -sqrt(1 - c**2)   # quadrant 4, so sin is negative
print(s)
print(s / c)
```

```
Output ▸
-12/13
-12/5
```

Answer 4

y = cos kθ has period $\frac{2\pi}{k}$, so the period is $\frac{2}{3}\pi$. Shifting by π does not bring the expression back (a difference remains), while shifting by $\frac{2}{3}\pi$ makes the difference 0.

```
from sympy import symbols, cos, pi, simplify
t = symbols('t')
print(simplify(cos(3*(t + pi)) - cos(3*t)))
print(simplify(cos(3*(t + 2*pi/3)) - cos(3*t)))
```

```
Output ▸
-2*cos(3*t)
0
```

Answer 5

Amplitude 3, period 2π. At θ = 0 the value is $3 \sin \frac{\pi}{6} = \frac{3}{2}$; at θ = $\frac{\pi}{3}$ the inside is $\frac{\pi}{2}$, so the value is 3 (the maximum).

```
from sympy import symbols, sin, pi
t = symbols('t')
f = 3 * sin(t + pi/6)
print(f.subs(t, 0))
print(f.subs(t, pi/3))   # θ giving the maximum
```

```
Output ▸
3/2
3
```

Answer 6

Split 15° = 45° − 30°: $\cos 15° = \cos 45° \cos 30° + \sin 45° \sin 30° = \frac{\sqrt{6}+\sqrt{2}}{4}$. (It equals $\sin 75°$ because 15° and 75° are complementary angles.)

```
from sympy import cos, pi, sqrt, simplify
angle = 15 * pi / 180
print(simplify(cos(angle)))
hand = (sqrt(6) + sqrt(2)) / 4
print(simplify(cos(angle) - hand))
```

Output ▸
```
sqrt(2)/4 + sqrt(6)/4
0
```

Answer 7

In the first quadrant, $\cos\alpha = \frac{3}{5}$ and $\sin\beta = \frac{5}{13}$. The subtraction forms give $\sin(\alpha - \beta) = \frac{4}{5} \times \frac{12}{13} - \frac{3}{5} \times \frac{5}{13} = \frac{33}{65}$ and $\cos(\alpha - \beta) = \frac{3}{5} \times \frac{12}{13} + \frac{4}{5} \times \frac{5}{13} = \frac{56}{65}$.

```
from sympy import Rational
sa, ca = Rational(4, 5), Rational(3, 5)
sb, cb = Rational(5, 13), Rational(12, 13)
print(sa*cb - ca*sb)   # sin(α−β)
print(ca*cb + sa*sb)   # cos(α−β)
```

Output ▸
```
33/65
56/65
```

Answer 8

In the first quadrant, $\sin\theta = \sqrt{1 - \frac{1}{9}} = \frac{2\sqrt{2}}{3}$. The double-angle formulas give $\sin 2\theta = 2 \times \frac{2\sqrt{2}}{3} \times \frac{1}{3} = \frac{4\sqrt{2}}{9}$ and $\cos 2\theta = 2 \times \frac{1}{9} - 1 = -\frac{7}{9}$.

```
from sympy import Rational, sqrt
c = Rational(1, 3)
s = sqrt(1 - c**2)   # quadrant 1, so sin is positive
print(s)
print(2*s*c)         # sin2θ
print(2*c**2 - 1)    # cos2θ
```

Output ▸
```
2*sqrt(2)/3
4*sqrt(2)/9
-7/9
```

Answer 9

Plot the point $(\sqrt{3}, -1)$: then $r = \sqrt{3 + 1} = 2$ and the angle is $\alpha = -\frac{\pi}{6}$. Hence $\sqrt{3}\sin\theta - \cos\theta = 2\sin(\theta - \frac{\pi}{6})$, with maximum 2 and minimum −2.

```
from sympy import symbols, sin, cos, sqrt, pi
from sympy import simplify, expand_trig
t = symbols('t')
left = sqrt(3) * sin(t) - cos(t)
right = 2 * sin(t - pi/6)
print(simplify(left - right))
print(expand_trig(right))
```

Output ▸

```
0
sqrt(3)*sin(t) - cos(t)
```

Answer 10

(1) From the intersections with the line $x = -\frac{1}{2}$: $\theta = \frac{2}{3}\pi$ and $\frac{4}{3}\pi$. (2) $\tan\theta = 1$ means a radius arm of slope 1, so $\theta = \frac{\pi}{4}$ and $\frac{5}{4}\pi$ (tan has period π, so adding π to solve's answer $\pi/4$ gives the other solution in range).

```
from sympy import symbols, cos, tan, solve, Rational
t = symbols('t')
print(solve(cos(t) + Rational(1, 2), t))
print(solve(tan(t) - 1, t))
```

Output ▸

```
[2*pi/3, 4*pi/3]
[pi/4]
```

Chapter 11
Handling Rapid Change: Exponential and Logarithmic Functions

Extending exponents

Laws of exponents

Exponential functions

Logarithms

Laws of logarithms

Logarithmic functions

Common logarithms

Exponential and logarithmic equations

Chapter 11 Handling Rapid Change: Exponential and Logarithmic Functions

What You Will Learn

Bacteria double with every division, savings grow by a few percent of compound interest a year, and infections snowball — the world is full of change that grows by multiplication. The stars of this chapter are the exponential functions that describe such **rapid change**, and their partners, the logarithmic functions.

The first half is the world of exponents. We widen the **exponent** — the 3 in 2^3 — to zero, negative numbers, and even fractions (**extending exponents**). The guiding motto: extend so that the **laws of exponents** are preserved. Out of this grows the **exponential function** $y = 2^x$, whose graph climbs explosively (or plunges).

The second half is the world of logarithms. The **logarithm** $\log_a M$ asks "to what power must a be raised to give M?" — an exponent read backwards. We cover the **laws of logarithms** that turn multiplication into addition, the graph of the **logarithmic function** running inverse to the exponential, the **common logarithm** that names the number of digits, and finally **exponential and logarithmic equations** — all in one sweep.

The Python stars are sympy's log(M, a) (the logarithm with base a) and Rational (for fractional exponents). simplify will double-check every exponent law and logarithm law along the way.

By the end of this chapter, you will be able to:

- compute zeroth, negative, and fractional powers
- tidy expressions using the laws of exponents
- describe the graph of an exponential function (increase/decrease, the point (0, 1), the asymptote)
- translate between exponential and logarithmic form, and evaluate logarithms
- compute with the laws of logarithms and the change-of-base formula

- find the number of digits of a huge number using common logarithms
- solve exponential and logarithmic equations by matching bases (including checking that arguments are positive)

11.1 Extending Exponents: Zeroth, Negative, and Fractional Powers

Recap (Level 2, Chapter 1) Laws of exponents such as $a^2 \times a^3 = a^5$ were learned with natural-number exponents. Read as "how many times to multiply," 2^0 and 2^{-1} make no sense at all. In this section we decide their meaning — so that the laws stay unbroken.

$2^3 = 8$, $2^2 = 4$, $2^1 = 2$ — each time the exponent drops by 1, the value halves. Let us simply continue the rule: 2^0 is $2 \div 2 = 1$, 2^{-1} is $1 \div 2 = \frac{1}{2}$, 2^{-2} is $\frac{1}{4}$ ... (Figure 11.1). In other words, if we decide **so that the laws of exponents are preserved**, there is only one possible answer.

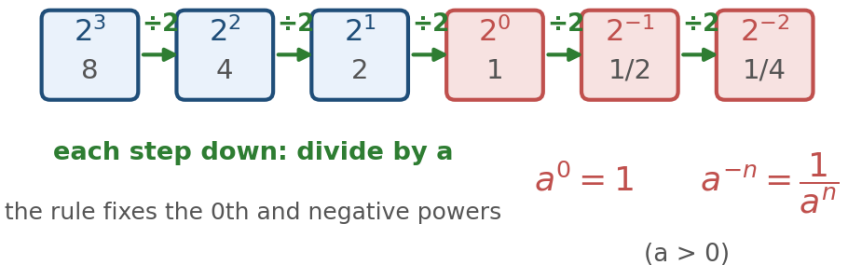


Figure 11.1 Each step down in the exponent divides by a. Continuing the rule fixes the meaning of the zeroth and negative powers naturally.

Term Zeroth and negative powers For $a > 0$: $a^0 = 1$, $a^{-n} = \frac{1}{a^n}$. Defined so that the law $a^m \times a^n = a^{m+n}$ holds for all integers.

On to fractional exponents. Since $a^{\frac{1}{2}} \times a^{\frac{1}{2}} = a^1$ ought to hold, $a^{\frac{1}{2}}$ must be "the number that gives a when squared" — that is, $\sqrt{a}$ (Figure 11.2). In

general $a^{\frac{m}{n}}$ is the n-th root of a^m. The exponent has now leapt from the integers all the way to fractions (rational numbers).

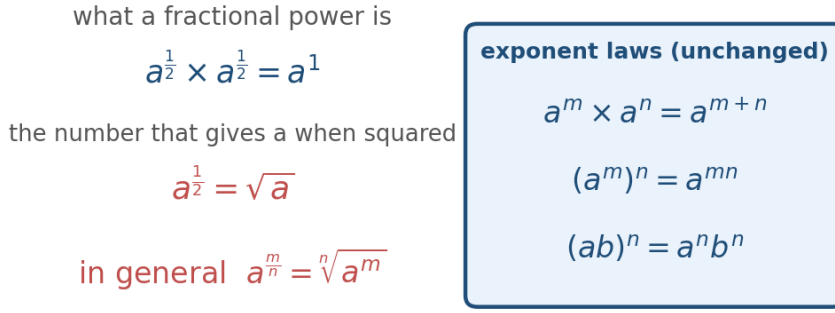


Figure 11.2 A fractional power is a root. We define it so that the laws of exponents (right) remain valid after the extension.

Term **Fractional powers and the laws of exponents** For $a > 0$: $a^{\frac{1}{n}}$ = the n-th root of a, $a^{\frac{m}{n}}$ = the n-th root of a^m. The laws $a^m a^n = a^{m+n}$, $(a^m)^n = a^{mn}$, $(ab)^n = a^n b^n$ remain valid after the extension.

Example 11.1

Find the following values. (1) 2^{-3} (2) $8^{\frac{1}{3}}$ (3) $27^{\frac{2}{3}}$

With pen and paper (1) $\frac{1}{2^3} = \frac{1}{8}$. (2) The number whose cube is 8, namely 2. (3) The cube root of 27 is 3, so $3^2 = 9$.

Check it with Python Pass the exponent as a Rational and sympy computes fractional powers exactly (the decimal 1/3 carries rounding error, so Rational is the safe choice).

```
from sympy import Rational
print(2**Rational(-3))
print(8**Rational(1, 3))
print(27**Rational(2, 3))
```

Output ▸

```
1/8
2
9
```

1/8, 2, and 9. A negative exponent means "reciprocal," a fractional exponent means "root" — both meanings fell out naturally from the laws of exponents.

Example 11.2

For a > 0, simplify the following using the laws of exponents. (1) $a^2 \times a^3$ (2) $(a^2)^3$ (3) $a^{\frac{1}{2}} \times a^{\frac{1}{3}}$

With pen and paper (1) Add the exponents: a^5. (2) Multiply the exponents: a^6. (3) $\frac{1}{2} + \frac{1}{3} = \frac{5}{6}$, so $a^{\frac{5}{6}}$. The laws work for fractional exponents exactly as before.

Check it with Python The trick is symbols('a', positive=True), declaring "a is a positive number." Exponent arithmetic is safest on positive numbers, and we pass that assumption on to sympy.

```
from sympy import symbols, simplify, Rational
a = symbols('a', positive=True)
print(simplify(a**2 * a**3))
print(simplify((a**2)**3))
print(simplify(a**Rational(1, 2) * a**Rational(1, 3)))
```

Output ▸

```
a**5
a**6
a**(5/6)
```

a^5, a^6, and $a^{\frac{5}{6}}$. Integer or fraction, the laws of exponents wear a single face — the payoff of "extending so the laws are preserved."

11.2 Exponential Functions: Graphs of Explosive Change

> **Recap (Level 2, Chapter 3)** With quadratics, we read the shape of the graph straight off the formula $y = a(x - p)^2 + q$. The habit of "see the formula, see the graph" carries over to exponential functions unchanged.

With exponents extended to all real numbers, we can put x itself in the exponent: $\mathbf{y = a^x}$ ($a > 0$, $a \neq 1$) — the **exponential function**. For $a > 1$ the values climb explosively as x grows; for $0 < a < 1$ they plunge. Either way the graph always passes through (0, 1) (since $a^0 = 1$), and the x-axis is an

asymptote — the values approach 0 without limit but never reach it (Figure 11.3).

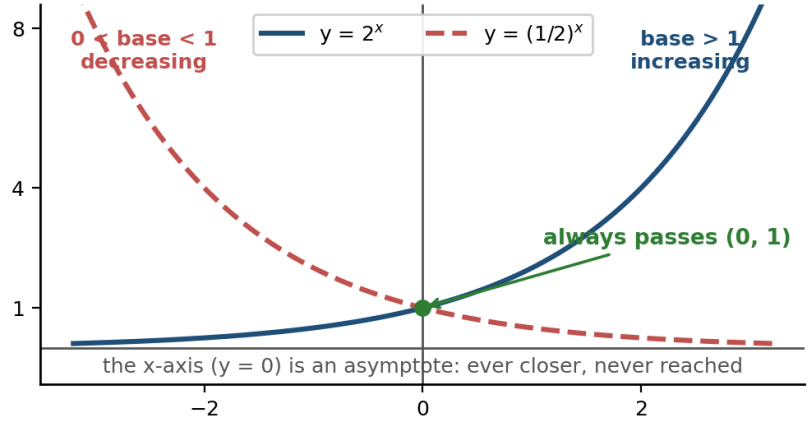


Figure 11.3 The exponential functions $y = 2^x$ and $y = (1/2)^x$: always through (0, 1), with the x-axis as asymptote. The base decides increase or decrease.

> **Term** **Exponential function** $y = a^x$ ($a > 0$, $a \neq 1$). Increasing for $a > 1$, decreasing for $0 < a < 1$. Always passes through (0, 1); the x-axis ($y = 0$) is an asymptote. The values are always positive.

Example 11.3

For the function $y = 2^x$, find the values of y at $x = -2$, 0, and 3.

With pen and paper $2^{-2} = \frac{1}{4}$, $2^0 = 1$, and $2^3 = 8$. While x rises by 5, y multiplies 32-fold, from $\frac{1}{4}$ to 8 — swelling by multiplication, not addition: that is the exponential function.

Check it with Python Keep 2**x as an expression and substitute the values with subs.

```
from sympy import symbols
x = symbols('x')
f = 2**x
print(f.subs(x, -2))
print(f.subs(x, 0))
print(f.subs(x, 3))
```

```
Output ▸
1/4
1
8
```

1/4, 1, and 8. Each unit step in x doubles y — this constant ratio is the essence of the exponential. Dividing bacteria and compound interest both grow in exactly this shape.

11.3 Logarithms: Exponents Read Backwards

"2 cubed is 8" — read backwards, this becomes "to what power must 2 be raised to give 8? Answer: 3." The number answering "what power" is the **logarithm**. In symbols we write **$\log_2 8 = 3$**, calling 2 the **base** and 8 the **argument**. A logarithm, in other words, is just another name for an exponent: the exponential statement and the logarithmic statement say one and the same thing (Figure 11.4).

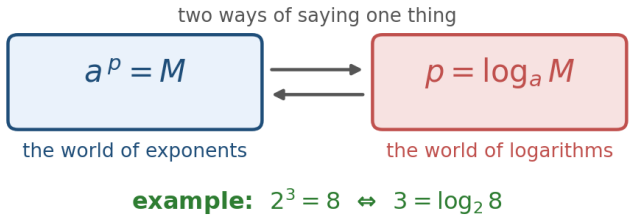


Figure 11.4 Exponential and logarithmic form are two ways of saying one thing. The logarithm $\log_a M$ is another name for the exponent: "to what power must a be raised to give M?"

Term **Logarithm** For $a > 0$, $a \neq 1$, $M > 0$: $a^p = M \quad \Leftrightarrow \quad p = \log_a M$. We call a the base and M the argument. The argument is always positive (because a^p is always positive).

Example 11.4

Find the following values. (1) $\log_2 8$ (2) $\log_3 \frac{1}{9}$ (3) $\log_5 \sqrt{5}$

With pen and paper Ask "to what power must the base be raised to give the argument?" (1) $2^3 = 8$, so the answer is 3. (2) $3^{-2} = \frac{1}{9}$, so -2. (3) $5^{\frac{1}{2}} =$

$\sqrt{5}$, so $\frac{1}{2}$. The negative and fractional exponents of Section 11.1 are pulling their weight here.

Check it with Python sympy's log(M, a) is the logarithm with base a. For the cases that do not collapse on their own, adding simplify makes sure of the value.

```
from sympy import log, Rational, sqrt, simplify
print(log(8, 2))
print(log(Rational(1, 9), 3))
print(simplify(log(sqrt(5), 5)))
```

Output ▸

```
3
-2
1/2
```

3, −2, and 1/2. Evaluating a logarithm is the same as answering a "guess the exponent" quiz — and since exponents have been extended, logarithms freely take negative and fractional values too.

11.4 Laws of Logarithms: Turning Products into Sums

Translate the exponent law $a^m \times a^n = a^{m+n}$ into the language of logarithms, and a remarkable property appears: a **product** of arguments becomes a **sum** of logarithms. Likewise quotients become differences, and powers drop down to coefficients (Figure 11.5). The laws of exponents and the laws of logarithms are two sides of the same law.

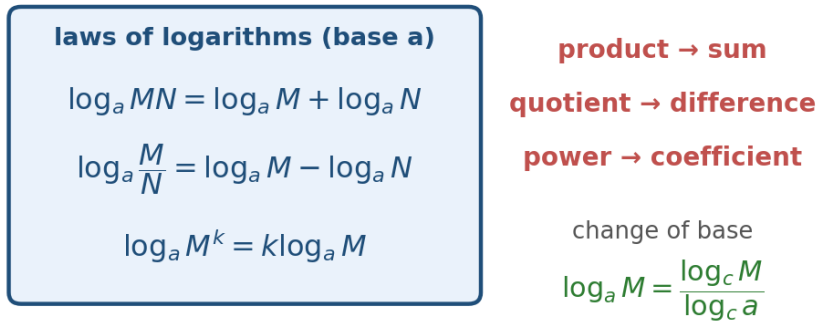


Figure 11.5 The laws of logarithms: products become sums, quotients become differences, powers become coefficients — plus the change-of-base formula.

Term **Laws of logarithms** $\log_a MN = \log_a M + \log_a N$, $\log_a \frac{M}{N} = \log_a M - \log_a N$, $\log_a M^k = k \log_a M$. Change of base: $\log_a M = \frac{\log_c M}{\log_c a}$ (any convenient base c will do).

Example 11.5

Find the following values. (1) $\log_{10} 2 + \log_{10} 5$ (2) $\log_2 12 - \log_2 3$

With pen and paper (1) A sum becomes a product of arguments: $\log_{10}(2 \times 5) = \log_{10} 10 = 1$. (2) A difference becomes a quotient: $\log_2 \frac{12}{3} = \log_2 4 = 2$. Scattered logarithms snap together under the laws and compute in one stroke.

Check it with Python Run a sum or difference of logs through simplify: it merges them by the laws of logarithms and produces the value.

```
from sympy import log, simplify
print(simplify(log(2, 10) + log(5, 10)))
print(simplify(log(12, 2) - log(3, 2)))
```

Output ▸

```
1
2
```

1 and 2. Each of $\log_{10} 2$ and $\log_{10} 5$ is an awkward irrational on its own, yet their sum is exactly 1. Turning multiplication into addition — this property is why logarithms have stayed in service for 400 years (see the column at the end of this chapter).

Example 11.6

Find the following values. (1) $\log_2 3 \times \log_3 4$ (2) $\log_4 8$

With pen and paper (1) Change both bases to 2: $\log_3 4 = \frac{\log_2 4}{\log_2 3} = \frac{2}{\log_2 3}$, so the product is 2. (2) With base 2: $\frac{\log_2 8}{\log_2 4} = \frac{3}{2}$. When the bases differ, the change-of-base formula puts everything on common ground.

Check it with Python Internally sympy rewrites every logarithm in terms of the natural logarithm, so a single simplify takes care of the base change too.

```
from sympy import log, simplify
print(simplify(log(3, 2) * log(4, 3)))
print(simplify(log(8, 4)))
```

Output ▸

```
2
3/2
```

2 and 3/2. In (1) the awkward number $\log_2 3$ cancelled cleanly in mid-multiplication. The change-of-base formula is a certificate of freedom: any logarithm can be rewritten in whatever base you like.

11.5 Logarithmic Functions and Common Logarithms: Naming the Digits

The function assigning to x its logarithm, $\mathbf{y = \log_a x}$, is the **logarithmic function**. It is the exponential function $y = a^x$ with x and y swapped — its inverse — so the graph is the mirror image of $y = a^x$ in the line y = x. It passes through (1, 0), and its domain is x > 0 (the argument is positive) (Figure 11.6). However large x grows, y creeps up only slowly — the "ultra-slow" counterpart to the exponential's explosion.

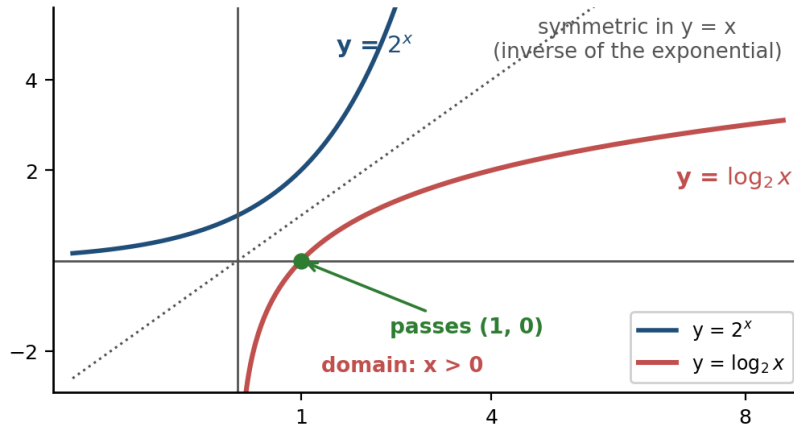


Figure 11.6 The logarithmic function $y = \log_2 x$ is the inverse of $y = 2^x$: symmetric in the line y = x, through (1, 0), with domain x > 0.

Term **Logarithmic function** $y = \log_a x$ ($a > 0$, $a \neq 1$). The inverse of the exponential $y = a^x$, with graph symmetric in y = x. Passes through (1, 0); domain x > 0. Increasing for a > 1, decreasing for 0 < a < 1.

Example 11.7

For the function $y = \log_2 x$, find the values of y at $x = \frac{1}{2}$, 1, and 8.

With pen and paper $\log_2 \frac{1}{2} = -1$, $\log_2 1 = 0$, and $\log_2 8 = 3$. While x multiplies 16-fold from $\frac{1}{2}$ to 8, y rises by only 4, from −1 to 3 — the slow growth that mirrors the exponential exactly in reverse.

Check it with Python Hand the three arguments to log(M, a) in turn.

```
from sympy import log, Rational
print(log(Rational(1, 2), 2))
print(log(1, 2))
print(log(8, 2))
```

Output ▸

```
-1
0
3
```

−1, 0, and 3. "How many times x multiplied" is translated into "how much y increased" — the logarithmic function measures the world of multiplication with an additive ruler.

The logarithm with base 10, **$\log_{10}$**, is called the **common logarithm**. Since we write numbers in base ten, the common logarithm connects directly to the **number of digits**: for example, $1000 = 10^3$ has 4 digits, so "digits = integer part of $\log_{10}$ + 1." Even for a number far too large to write out, the common logarithm names its digit count (Figure 11.7).

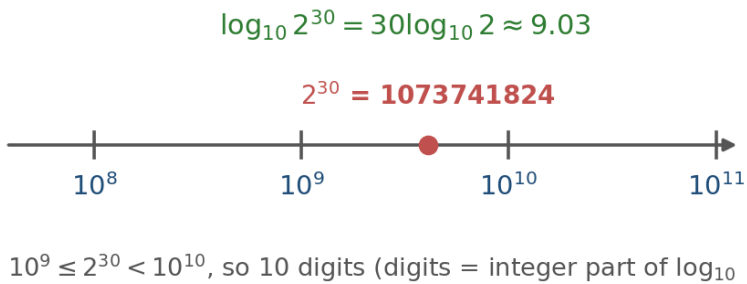


Figure 11.7 Common logarithms and digit counts. From $\log_{10} 2^{30} \approx 9.03$ we get $10^9 \leqq 2^{30} < 10^{10}$ — that is, 10 digits.

Term **Common logarithm and digit count** The base-10 logarithm $\log_{10} M$ is called the common logarithm. For a positive integer N, if the integer part of $\log_{10} N$ is k, then N has k + 1 digits ($10^k \leqq N < 10^{k+1}$).

Example 11.8

Taking $\log_{10} 2 = 0.3010$, find the number of digits of 2^{30}.

With pen and paper Drop the power down with the logarithm law: $\log_{10} 2^{30} = 30 \log_{10} 2 = 30 \times 0.3010 = 9.03$. The integer part is 9, so $10^9 \leqq 2^{30} < 10^{10}$ — that is, **10 digits**.

Check it with Python N(expr, digits) returns a numerical approximation. And let us compute 2**30 directly, to count the digits with our own eyes.

```
from sympy import log, N
digits = 30 * log(2, 10)
print(N(digits, 4))   # the value of log10
print(2**30)          # the actual value
```

Output ▸

```
9.031
1073741824
```

$\log_{10} 2^{30} \approx 9.031$, and $2^{30} = 1073741824$ does indeed have 10 digits. Knowing the digit count without computing the value — a powerful weapon when astronomical numbers are on the table.

11.6 Exponential and Logarithmic Equations: Match the Bases

The finishing touch: equations involving exponents or logarithms. The motto is "**match the bases**." For an exponential equation, write both sides as powers of the same base and **compare the exponents**; for a logarithmic equation, rewrite in exponential form (or merge the logs into one and compare the arguments) (Figure 11.8). With logarithms, always end by checking that **every argument is positive**.

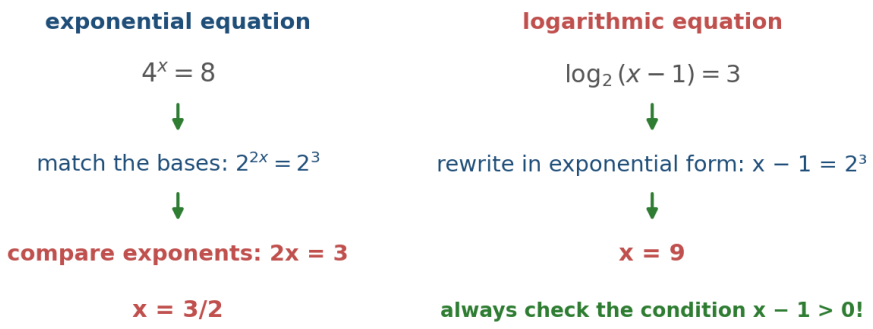


Figure 11.8 For exponential equations, match the bases and compare exponents (left). For logarithmic equations, rewrite in exponential form — and always check that the arguments are positive (right).

Term **Exponential and logarithmic equations** Exponential: write both sides over the same base and compare exponents. Logarithmic: rewrite with $a^p = M \Leftrightarrow p = \log_a M$, or merge the logs by the laws and compare arguments. Never skip the check that every argument is positive.

Example 11.9

Solve the following equations. (1) $4^x = 8$ (2) $2^{x+1} = 16$

With pen and paper (1) Matching the bases at 2 gives $2^{2x} = 2^3$. Comparing exponents, $2x = 3$, so $x = \frac{3}{2}$. (2) From $2^{x+1} = 2^4$ we get $x + 1 = 4$, so $x = 3$.

Check it with Python solve handles exponential equations too. Declaring x with real=True keeps the answers to real solutions only.

```
from sympy import symbols, solve
x = symbols('x', real=True)
print(solve(4**x - 8, x))
print(solve(2**(x + 1) - 16, x))
```

Output ▸

```
[3/2]
[3]
```

x = 3/2 and x = 3. "Match the bases, compare the exponents" — an exponential function is monotonic (always rising, or always falling), so equal values mean equal exponents. That is exactly what licenses the comparison.

Example 11.10

Solve the following equations. (1) $\log_2(x-1)=3$ (2) $\log_3 x+\log_3(x-2)=1$

With pen and paper (1) In exponential form, $x-1=2^3=8$, so $x=9$; the condition $x-1>0$ holds. (2) Merge the left side: $\log_3 x(x-2)=1$, so $x(x-2)=3$. Tidying, $x^2-2x-3=(x-3)(x+1)=0$, giving $x=3,-1$ — but the arguments must be positive ($x>0$ and $x-2>0$, i.e. $x>2$), so the answer is **$x=3$ only**.

Check it with Python For logarithmic equations, solve takes the domain into account and returns only the valid solutions. Note that $x=-1$ never appears in (2).

```
from sympy import symbols, solve, log
x = symbols('x')
print(solve(log(x - 1, 2) - 3, x))
print(solve(log(x, 3) + log(x - 2, 3) - 1, x))
```

Output ▸

```
[9]
[3]
```

$x=9$ and $x=3$. In (2) the quadratic's root $x=-1$ was rejected by the positivity condition, and solve returned only $x=3$. With logarithmic equations, a solution produced by algebra is not yet a true solution — make the positivity check a habit.

Column Thinking in Orders of Magnitude: Magnitude, Decibels, pH

When an earthquake's magnitude rises by 1, the energy grows about 32-fold. Between magnitude 7 and 9 — a difference of just 2 — lies a factor of roughly 1000. The magnitude scale is in fact built on the **common logarithm** of the energy. Decibels for loudness, pH for acidity, stellar magnitudes for brightness — all are logarithmic scales. Why is the world's measuring tape so full of logarithms? Because the quantities we deal with span ranges that differ by orders of magnitude. A whisper and a jet engine differ in energy by a factor of a trillion, yet on a logarithmic scale they fit on a ruler from 0 to 120 decibels. By converting the world of multiplication into an additive scale, logarithms shrink scale-spanning realities down to human size. Remarkably, human senses themselves respond roughly to the logarithm of a stimulus (the Weber–Fechner law) — logarithms are etched

into our own bodies.

Chapter Summary

- Exponents extend to zero, negatives, and fractions: $a^0 = 1$, $a^{-n} = \frac{1}{a^n}$, $a^{\frac{m}{n}}$ = the n-th root of a^m. The motto: preserve the laws of exponents.
- Laws of exponents: $a^m a^n = a^{m+n}$, $(a^m)^n = a^{mn}$, $(ab)^n = a^n b^n$.
- The exponential $y = a^x$ increases for $a > 1$ and decreases for $0 < a < 1$. Always through $(0, 1)$; the x-axis is an asymptote.
- The logarithm $\log_a M$ asks "to what power must a be raised to give M?" $a^p = M \Leftrightarrow p = \log_a M$. The argument M is positive.
- Laws of logarithms: products → sums, quotients → differences, powers → coefficients. The change-of-base formula aligns the bases.
- The logarithmic function $y = \log_a x$ is the inverse of the exponential (symmetric in $y = x$). The common logarithm $\log_{10}$ names the digit count: digits = integer part + 1.
- Solve exponential and logarithmic equations by matching bases and comparing exponents or arguments. With logarithms, never skip the positivity check.

Table 11.1 Python tools used in this chapter

Python tool	What it does
`Rational(1, 3)`	exact fractional exponents (Ch. 8)
`symbols('a', positive=True)`	declare a symbol as positive
`symbols('x', real=True)`	declare a symbol as real
`2**x`	powers and exponential functions
`log(M, a)`	the logarithm $\log_a M$ with base a
`simplify(expr)`	tidy via exponent and logarithm laws
`N(expr, digits)`	numerical approximation
`f.subs(x, value)`	substitute a value (from Chapter 8)
`solve(expr, x)`	solve exponential / log equations

Practice Problems

Solve each problem with pen and paper first, then write Python code in the empty Colab cells to check your answers. Fractional exponents with Rational, logarithms with log(M, a) — mobilize the whole toolbox of this chapter.

Problem 1

Find the following values. (1) 3^{-2} (2) $16^{\frac{1}{4}}$ (3) $8^{\frac{2}{3}}$

Problem 2

For a > 0, simplify the following using the laws of exponents. (1) $a^3 \times a^4$ (2) $a^{\frac{3}{4}} \times a^{\frac{1}{4}}$ (3) $(a^6)^{\frac{1}{3}}$

Problem 3

For the function $y = 3^x$, find the values of y at x = −1, 0, and 2.

Problem 4

Find the following values. (1) $\log_2 16$ (2) $\log_5 \frac{1}{25}$ (3) $\log_7 \sqrt{7}$

Problem 5

Find the following values. (1) $\log_6 2 + \log_6 18$ (2) $\log_5 50 - \log_5 2$

Problem 6

Find the following values. (1) $\log_3 5 \times \log_5 9$ (2) $\log_9 27$

Problem 7

For the function $y = \log_3 x$, find the values of y at $x = \frac{1}{9}$, 1, and 27.

Problem 8

Taking $\log_{10} 3 = 0.4771$, find the number of digits of 3^{20}.

Problem 9

Solve the following equations. (1) $9^x = 27$ (2) $5^{x-1} = 125$

Problem 10

Solve the following equations. (1) $\log_3 (x + 2) = 2$ (2) $\log_2 x + \log_2 (x - 2) = 3$

Answers and Explanations

Answer 1

(1) $\frac{1}{3^2} = \frac{1}{9}$. (2) The number whose fourth power is 16, namely 2. (3) The cube root of 8 is 2, so $2^2 = 4$.

```
from sympy import Rational
print(3**Rational(-2))
print(16**Rational(1, 4))
print(8**Rational(2, 3))
```

Output ▸

```
1/9
2
4
```

Answer 2

(1) Add the exponents: a^7. (2) $\frac{3}{4} + \frac{1}{4} = 1$, so the answer is a. (3) Multiply the exponents: a^2.

```
from sympy import symbols, simplify, Rational
a = symbols('a', positive=True)
print(simplify(a**3 * a**4))
print(simplify(a**Rational(3, 4) * a**Rational(1, 4)))
print(simplify((a**6)**Rational(1, 3)))
```

Output ▸
```
a**7
a
a**2
```

Answer 3

$3^{-1} = \frac{1}{3}$, $3^0 = 1$, and $3^2 = 9$. Each unit step in x triples y.

```
from sympy import symbols
x = symbols('x')
f = 3**x
print(f.subs(x, -1))
print(f.subs(x, 0))
print(f.subs(x, 2))
```

Output ▸
```
1/3
1
9
```

Answer 4

(1) $2^4 = 16$, so 4. (2) $5^{-2} = \frac{1}{25}$, so −2. (3) $7^{\frac{1}{2}} = \sqrt{7}$, so $\frac{1}{2}$.

```
from sympy import log, Rational, sqrt, simplify
print(log(16, 2))
print(log(Rational(1, 25), 5))
print(simplify(log(sqrt(7), 7)))
```

Output ▸
```
4
-2
1/2
```

Answer 5

(1) A sum becomes a product: $\log_6(2 \times 18) = \log_6 36 = 2$. (2) A difference becomes a quotient: $log_5 \frac{50}{2} = \log_5 25 = 2$.

```
from sympy import log, simplify
print(simplify(log(2, 6) + log(18, 6)))
print(simplify(log(50, 5) - log(2, 5)))
```

Output ▸
```
2
2
```

Answer 6

(1) Changing the base: $\log_5 9 = \frac{\log_3 9}{\log_3 5} = \frac{2}{\log_3 5}$, so the product is 2. (2) With base 3: $\frac{\log_3 27}{\log_3 9} = \frac{3}{2}$.

```
from sympy import log, simplify
print(simplify(log(5, 3) * log(9, 5)))
print(simplify(log(27, 9)))
```

Output ▸
```
2
3/2
```

Answer 7

$\log_3 \frac{1}{9} = -2$, $\log_3 1 = 0$, and $\log_3 27 = 3$. Each time x multiplies by 9, y rises by 2.

```
from sympy import log, Rational
print(log(Rational(1, 9), 3))
print(log(1, 3))
print(log(27, 3))
```

Output ▸
```
-2
0
3
```

Answer 8

$\log_{10} 3^{20} = 20 \log_{10} 3 = 20 \times 0.4771 = 9.542$. The integer part is 9, so $10^9 \leqq 3^{20} < 10^{10}$ — 10 digits. The actual value 3486784401 does indeed have 10 digits.

```
from sympy import log, N
digits = 20 * log(3, 10)
print(N(digits, 4))
print(3**20)
```

```
Output ▸
9.542
3486784401
```

Answer 9

(1) Matching the bases at 3 gives $3^{2x} = 3^3$, so 2x = 3 and $x = \frac{3}{2}$. (2) From $5^{x-1} = 5^3$ we get x – 1 = 3, so x = 4.

```
from sympy import symbols, solve
x = symbols('x', real=True)
print(solve(9**x - 27, x))
print(solve(5**(x - 1) - 125, x))
```

```
Output ▸
[3/2]
[4]
```

Answer 10

(1) In exponential form, $x + 2 = 3^2 = 9$, so x = 7 (and the condition x + 2 > 0 holds). (2) Merging the logs: $x(x - 2) = 2^3 = 8$. Tidying, $x^2 - 2x - 8 = (x - 4)(x + 2) = 0$, giving x = 4, −2 — but the positivity condition x > 2 leaves only x = 4.

```
from sympy import symbols, solve, log
x = symbols('x')
print(solve(log(x + 2, 3) - 2, x))
print(solve(log(x, 2) + log(x - 2, 2) - 3, x))
```

```
Output ▸
[7]
[4]
```

Appendix 1
A Summary of Python Basics

Running Python

Arithmetic

Variables

Conditionals

Loops

Working with collections

Functions

Drawing graphs

Creating classes

Appendix 1　A Summary of Python Basics

How to Use This Appendix

This appendix gathers the Python syntax you learned in Levels 1 and 2 into a single "cheat sheet." Nothing here is new, so read on with confidence.

Use it however you like. When you are writing code in the main chapters and find yourself wondering "wait, how do I write that again?", look up the table and sample code in the matching section. For the full explanations, you can always return to the chapters of the earlier levels.

As you read this appendix, check that you can:

- recall the main Python syntax you have learned, at a glance
- find the right syntax starting from "what you want to do"
- confirm the role of each construct with a simple example

※ Each section gives a quick-reference table plus sample code with output and notes.

A1.1　Running Python

Table A1.1　Running Python

Syntax	Example	What it does
`print(...)`	`print("Hello")` → Hello	Shows text or the result of a calculation on the screen.
`#` (comment)	`# this is a note`	A note that is never executed; use it to leave explanations in your code.
running an expression	`3 + 4` → `7`	Write an expression in a cell and run it; the result comes back.

Sample code

```
# show text on the screen
print("Hello, Python!")
# results of calculations work too
print(3 + 4)
```

Output ▸

```
Hello, Python!
7
```

- A line starting with # is a "comment." It is never executed; use it to leave explanations in your code.
- print(...) is the command that displays whatever is inside the ( ).
- Put an expression inside, like print(3 + 4), and the calculation runs first — the result 7 is what gets displayed.

A1.2 Arithmetic in Python

Table A1.2 Arithmetic

Syntax	Example	What it does
`int / float`	`3` (integer) `3.14` (decimal)	Kinds (types) of numbers. Dividing with / always gives a decimal.
`type(...)`	`type(3)` → <class 'int'>	Looks up the kind (type) of a value.
`+ - * /`	`7 / 2` → 3.5	The four arithmetic operations.
`**`	`2 ** 3` → 8	Power (2 cubed).
`// %`	`7 // 2` → 3 `7 % 2` → 1	For positive integers: the quotient (integer part) and the remainder.
`( )`	`2 + 3 * 4` → 14 `(2 + 3) * 4` → 20	Multiplication and division come first; parentheses change the order.

Sample code

```
print(7 + 2)    # addition
print(7 - 2)    # subtraction
print(7 * 2)    # multiplication
print(7 / 2)    # division (the result is a decimal)
print(7 // 2)   # quotient (rounded down)
print(7 % 2)    # remainder
print(2 ** 3)   # power (2 cubed)
```

```
Output ▸
9
5
14
3.5
3
1
8
```

- + - * / are the four arithmetic operations. A result of / is always a decimal (such as 3.5), even when the division comes out even.
- For the positive integers used here, // gives the integer part of the quotient (rounding down), and % gives the remainder. For 7 ÷ 2, the quotient is 3 and the remainder is 1.
- ** is a power: 2 ** 3 multiplies 2 three times, giving 8.

A1.3 Variables

Table A1.3 Variables

Syntax	Example	What it does
= (assignment)	`price = 20`	Puts a value into a variable (a named box).
using a variable	`print(price)` → 20	Write the name to take the value back out.
" " (string)	`name = "Yamada"`	A collection of characters, wrapped in " " or ' '.
+ (joining)	`"Hello, " + name` → Hello, Yamada	Joins strings together with +.
`print(a, b)`	`print("Sum:", 7)` → Sum: 7	Shows numbers and text together (comma-separated).

Sample code

```
price = 20              # store a number
name = "Yamada"           # store a string
print("Hello, " + name)  # join strings
print("The price is", price, "dollars")
```

```
Output ▸
Hello, Yamada
The price is 20 dollars
```

- = is "assignment": it puts 20 into the box named price.
- Wrapping characters in " " makes a string. With + you can join strings together.
- List items in print( ) separated by commas, and they are displayed in a row with single spaces in between.

A1.4 Conditionals

Table A1.4 Conditionals

Syntax	Example	What it does
`if ...:`	`if x > 0:` `print("plus")`	Runs only when the condition holds (True). Colon at the end; the body is indented.
`else:`	`else:` `print("0 or less")`	Runs when the condition does not hold (False).
`elif ...:`	`elif x == 0:` `print("zero")`	Checks another condition in turn.
comparison operators	`==  !=` `<  >  <=  >=`	Equal / not equal / order. The result is True or False.
`and / or / not`	`x > 0 and x < 10` `not x < 0`	Combines conditions. not flips True / False.

Sample code

```
score = 75
if score >= 80:
    print("Pass")
elif score >= 60:
    print("Retest")
else:
    print("Fail")
```

Output ▸

```
Retest
```

- Only when the if condition holds (True) does the indented block below run.
- elif checks "another condition" in turn; else is for when none of them matched.

- score is 75 — not 80 or more, but at least 60 — so "Retest" is displayed.
- Do not forget the colon at the end of the line and the indentation below it.

A1.5 Repeating with Loops

Table A1.5 Loops

Syntax	Example	What it does
`for ... in ...:`	`for i in range(3):` `  print(i)`	Repeats a fixed number of times.
`range(...)`	`range(1, 5)` → 1, 2, 3, 4	Makes consecutive integers. The end value is not included.
`while ...:`	`while x < 5:` `  x = x + 1`	Repeats while the condition holds.
the running-total pattern	`total = 0` `total = total + s`	Adds up from 0 to build a sum.

Sample code

```
total = 0
for i in range(1, 6):   # 1, 2, 3, 4, 5
    total = total + i
print(total)
```

Output ▸

```
15
```

- range(1, 6) makes the integers from 1 to 5 (the end value 6 is not included).
- for puts those numbers into i one after another, repeating the block below each time.
- Adding i into total each time builds up 1+2+3+4+5 = 15 (this is the "running-total pattern").

A1.6 Working with Collections of Data

Table A1.6 Working with collections

Syntax	Example	What it does
[ ] (list)	`nums = [10, 20, 30]`	Gathers several values in order.
index	`nums[0]` → 10 `nums[-1]` → 30	Counted from 0; -1 is the last element.
`.append(...)`	`nums.append(40)`	Adds an element to the end of a list.
`del`	`del nums[0]`	Deletes the element at a given index.
( ) (tuple)	`p = (10, 20)`	A bundle that cannot be changed afterwards.
{ } (dictionary)	`d = {"math": 80}` `d["math"]` → 80	Looks up values by name (key).

Sample code

```
scores = [60, 70, 80]   # a list
scores.append(90)       # add to the end
print(scores[0])        # the first element
print(scores[-1])       # the last element
for s in scores:
    print(s)
```

Output ▸

```
60
90
60
70
80
90
```

- Several values gathered in order inside [ ] make a list. The numbering (index) starts at 0.
- .append( ) adds an element to the end of the list; scores[-1] means the last element.
- With for, you can take out and process the elements of a list one by one.

A1.7 Functions

Table A1.7 Functions

Syntax	Example	What it does
`def ...():`	`def hello():` `  print("Hi")`	Names a block of work, turning it into a reusable "part."
calling	`hello()`	Writing name() runs the work inside.
parameters	`def add(a, b):`	Pass values through ( ) to change the behavior.
`return`	`return a + b`	Returns the result (return value), usable in variables and calculations.

Sample code

```
def add(a, b):      # define a function
    return a + b    # return the result

answer = add(3, 5)  # call the function
print(answer)
```

Output ▸

```
8
```

- def names a block of work and turns it into a "part." Inside the ( ) are the parameters (values passed in).
- return sends back the result (the return value); you can use it in calculations or store it in a variable.
- Calling add(3, 5) computes with a=3 and b=5; the 8 that comes back goes into answer.

A1.8 Drawing Graphs

matplotlib (the graphing toolbox) is brought in with the single line below, and from then on we use it under the short name plt.

```
import matplotlib.pyplot as plt
```

Table A1.8 Drawing graphs

Syntax	Example	What it does
`plt.plot(...)`	`plt.plot([1, 4, 9])`	Draws a line graph.
`plt.scatter(...)`	`plt.scatter(x, y)`	Draws points (a scatter plot).
`plt.show()`	`plt.show()`	Displays the graph you drew.
`marker`	`plt.plot(x, y, marker="o")`	Marks each point on the line.
title / xlabel / ylabel / legend	`plt.title("...")` `plt.xlabel("x")` `plt.ylabel("y")` etc.	Adds a title and axis names. For a legend, set label="..." and then call plt.legend().

Sample code

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]
y = [1, 4, 9, 16]
plt.plot(x, y, marker="o")
plt.title("y = x * x")
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```

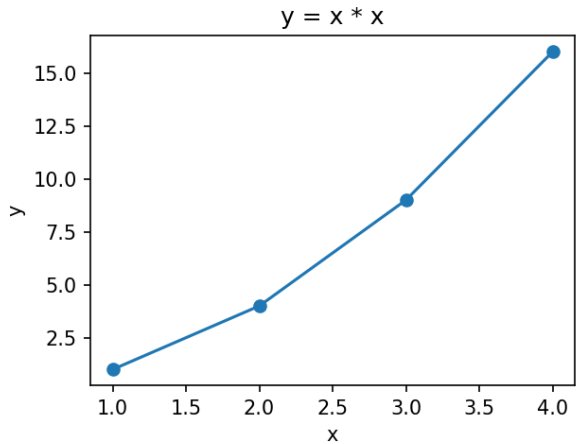


Figure A1.1 What the sample code produces (in Colab, this graph appears below the cell).

- import brings in matplotlib, which we use under the short name plt.

- plt.plot(x, y) joins the (x, y) pairs with a line; adding marker="o" marks each point as well.
- plt.title / xlabel / ylabel add a title and axis names.
- plt.show() displays the finished graph (in Colab, it appears below the cell).

A1.9 Creating Classes

A class is a "blueprint" that bundles data (attributes) with what that data can do (methods).

Table A1.9 Creating classes

Syntax	Example	What it does

`class`	`class Dog:`
	`def __init__(`
	`self, name):`
		Creates a class (blueprint). __init__ is called exactly once at creation.

ClassName(...)	`pochi = Dog("Pochi")`	Builds an object (instance) from the blueprint.
`self.attribute`	`self.name = name` `pochi.name`	Data the object owns (an attribute): save it via self, read it with .name.
`def method(self):`	`def bark(self):` `print(self.name)`	A function inside the class (a method); the first parameter is self.

Sample code

```
class Dog:
    def __init__(self, name):
        self.name = name          # save as an attribute

    def bark(self):
        print(self.name + " says Woof!")

pochi = Dog("Pochi")   # create an instance
pochi.bark()           # call the method
```

Output ▸

```
Pochi says Woof!
```

- class bundles data (attributes) and work (methods) into a "blueprint."

- __init__ is called exactly once when an object is built from the blueprint, saving the name into self.name.
- Dog("Pochi") builds an instance (a real object) from the blueprint.
- Calling pochi.bark() runs the method, which uses the saved name in its message.

Column　A Cheat Sheet Grows More Useful the More You Tend It

The tables in this appendix form a "cheat sheet" for looking back over the Python syntax you have learned so far. But the most useful cheat sheet of all is the one you build yourself. Whenever you learn a new construct, jot it down in your own notebook in the form "syntax — short example — one-line note." Summarizing as if explaining to someone else deepens your understanding remarkably, and noting where you stumbled keeps you from stumbling there twice. Even professional programmers do not memorize everything: when the need arises, they quickly consult their own notes or the official documentation — and that "skill of looking things up" is itself a fine programming skill. Let these tables be the starting point, and grow a cheat sheet of your own.

Appendix Summary

- We gathered the Python syntax from the earlier levels into quick-reference tables with sample code.
- We reviewed everything in one pass: running Python, arithmetic, variables, conditionals, loops, collections, functions, graphs, and classes.
- When you get stuck in the main chapters, come back and use these tables and samples as your cheat sheet.
- Rather than memorizing syntax, remember "what can be done" — and look up the exact spelling whenever you need it.

Appendix 2
Essential LaTeX (MathJax) Notation

Math mode basics

Superscripts and subscripts

Fractions and roots

Greek letters and symbols

Function names

Brackets, spacing, and text

Notation you will meet later

Appendix 2 Essential LaTeX (MathJax) Notation

How to Use This Appendix

In a Colab text cell (markdown cell), you can write mathematics using a notation called LaTeX. What you write is rendered as beautifully typeset formulas by a system called MathJax. Every formula in the companion Colab notebooks of this series is written in exactly this notation.

This appendix is a quick-reference guide to the most frequently used notation. Whenever you want to write a math-filled note of your own, or check how a formula in a Colab notebook is written, look up the table and the worked example in the matching section.

As you read this appendix, check that you can:

- understand the basics of math mode (wrapping formulas in $)
- find the right notation starting from "the formula you want to display"
- write your own math notes in a Colab text cell

※ Each section gives a quick-reference table plus a "what to type in the text cell" example and the displayed result. The displayed results are typeset with Word's equation engine and look almost exactly as they do in Colab.

A2.1 Math Mode Basics

Only the part **wrapped in $ signs** enters "math mode." A single $ on each side places the formula inside the flow of the text (inline); wrapping with $$ gives the formula a line of its own, displayed large and centered (display).

Table A2.1 Math mode basics

How to write (LaTeX)	Displayed as	What it does
`$ ... $`	inside the text	Embeds a formula in a sentence (inline).
`$$ ... $$`	centered on its own line	Gives the formula a line of its own (display).
half-width input	—	Inside math mode, type half-width (ASCII) letters and symbols.

What to type in the text cell

```
The area of a circle is $S = \pi r^2$.

$$S = \pi r^2$$
```

Displayed result

The area of a circle is $S = \pi r^2$.

$S = \pi r^2$ (this one appears large, centered on its own line)

- Wrapping with a single $ slips the formula into the flow of the sentence.
- Wrapping with $$ makes the formula the star of its own line — the better choice for important equations.
- What \pi and ^ mean is exactly what the following sections explain, one by one.

A2.2 Superscripts and Subscripts: Exponents and Indices

Table A2.2 Superscripts and subscripts

How to write (LaTeX)	Displayed as	What it does
`x^2`	x^2	^ makes a superscript (an exponent).
`x^{10}`	x^{10}	Two or more characters go inside { }.
`a_1`	a_1	_ makes a subscript (an index).
`a_{n+1}`	a_{n+1}	Likewise, wrap longer subscripts in { }.
`x_1^2`	$x_1{}^2$	Subscript and superscript can be combined.

What to type in the text cell

```
Consider the quadratic $y = ax^2 + bx + c$.
The distance between $P(x_1, y_1)$ and $Q(x_2, y_2)$ ...
```

Displayed result

Consider the quadratic $y = ax^2 + bx + c$. The distance between $P(x_1, y_1)$ and $Q(x_2, y_2)$ ...

- ^ for superscripts, _ for subscripts; a single character stands alone, two or more go inside { } — that is the basic rule.
- Writing x^10 instead of x^{10} raises only the 1 (giving x¹10), so take care.

A2.3 Fractions and Roots

Table A2.3 Fractions and roots

How to write (LaTeX)	Displayed as	What it does
`\dfrac{a}{b}`	$\dfrac{a}{b}$	A fraction (display size). The d in \dfrac stands for display.
`\frac{1}{2}`	$\frac{1}{2}$	A fraction (smaller, for inline use).
`\sqrt{a}`	$\sqrt{a}$	A square root.
`\sqrt[3]{a}`	$\sqrt[3]{a}$	A cube root; the index goes in [ ].

What to type in the text cell

```
$$x = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$
```

Displayed result

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- \dfrac{numerator}{denominator} builds a fraction; expressions and roots nest freely inside.
- An expression like b^2 - 4ac goes straight inside \sqrt{ } — the quadratic formula fits on a single line.
- \pm is the ± symbol (introduced in the next section).

A2.4 Greek Letters and Symbols

Table A2.4 Greek letters and symbols

How to write (LaTeX)	Displayed as	What it does
\pi \theta	π θ	π for the circle constant, θ for angles.
\alpha \beta	α β	α and β, also used for angles.
\times \div \pm	$\times$ $\div$ $\pm$	Times, divided by, plus-minus.
\le \ge \ne	$\le$ $\ge$ $\ne$	At most, at least, not equal.
\to	$\to$	An arrow (approaches, goes to).
\cdots \infty	$\cdots$ ∞	The dots "..." and infinity.

What to type in the text cell

```
For $0 \le \theta < 2\pi$, solve the equation
$\sin \theta = \dfrac{1}{2}$.
```

Displayed result

For $0 \le \theta < 2\pi$, solve the equation $\sin \theta = \frac{1}{2}$.

- Greek letters and symbols are written as \ plus a name. The names are simply the English readings, so saying them aloud helps them stick (pi, theta, times...).
- \le abbreviates less or equal, and \ge greater or equal.

A2.5 Function Names: sin, cos, log, and Friends

Function names such as sin and log are written with a leading \ . That renders them upright, as function names should be — instead of the slanted italics of variables (which would look like s × i × n).

Table A2.5 Function names

How to write (LaTeX)	Displayed as	What it does
`\sin \theta`	$sin\ \theta$	The leading \ makes it an upright function name.
`\cos \theta  \tan \theta`	$cos\ \theta \quad tan\ \theta$	Same for cos and tan.
`\sin^2 \theta`	$sin^2\ \theta$	The customary way to write "sin θ squared."
`\log_{10} 2`	$log_{10}\ 2$	A logarithm; the base is a subscript (_).
`\log_2 8 = 3`	$log_2\ 8 = 3$	A single-character base may drop the { }.

What to type in the text cell

```
$$\sin^2 \theta + \cos^2 \theta = 1$$
```

Displayed result

$$sin^2\ \theta + cos^2\ \theta = 1$$

- Compare \sin with plain sin (no \) and the difference between upright and italic jumps out. Always give function names their \ .
- A logarithm's base is just a subscript, so the _ of A2.2 applies as is — with { } around bases of two or more characters, as in \log_{10}.

A2.6 Brackets, Spacing, and Text

Table A2.6 Brackets, spacing, and text

How to write (LaTeX)	Displayed as	What it does
`\left( ... \right)`	$\left(\frac{a}{b}\right)$	Brackets that stretch to fit their contents.
`\,`	(a thin space)	Adds a small gap.
`\quad`	(a wide space)	Adds a wide gap, e.g. between formulas.
`\text{ }`	text inside math	Puts ordinary text inside math mode.

What to type in the text cell

```
$$y = \left( \dfrac{1}{2} \right)^x \quad
\text{(an exponential with base below 1)}$$
```

Displayed result

$$y = \left(\frac{1}{2}\right)^x \quad \text{(an exponential with base below 1)}$$

- Plain ( ) stays small even around a fraction. Pair \left(with \right) and the brackets stretch to match what is inside.
- \left and \right must always come as a pair — one without the other is an error.
- Inside \text{ } you can write ordinary words, spaces and all.

A2.7 Notation You Will Meet Later: Sums, Integrals, Limits

To finish, here are just three symbols that appear in the levels ahead. Their mechanics are nothing new — only combinations of the superscripts and subscripts above. There is no need to memorize them now; come back here when you meet them.

Table A2.7 Notation you will meet later

How to write (LaTeX)	Displayed as	What it does
`\sum_{k=1}^{n} k`	$\sum_{k=1}^{n} k$	The summation sign Σ (sigma); appears with sequences.
`\int_a^b f(x) dx`	$\int_a^b f(x)\,dx$	The integral sign, from lower end a to upper end b.
`\lim_{x \to \infty}`	$\lim_{x \to \infty}$	The limit sign: "let x grow without bound."

What to type in the text cell

```
$$\sum_{k=1}^{5} k = 1 + 2 + 3 + 4 + 5 = 15$$
```

Displayed result

$$\sum_{k=1}^{5} k = 1 + 2 + 3 + 4 + 5 = 15$$

- What sits below and above the Σ is written with the subscript _ and superscript ^ — not a new rule, just A2.2 in combination.
- This is the very sum — 1 through 5, totaling 15 — that Appendix 1 computed with a for loop in A1.5. The mathematical symbol and the program meet once again.

Column Writing Mathematics in Plain Characters: the Story of TeX and MathJax

The system behind this notation, TeX (pronounced "tech"), was created in 1978 by the computer scientist Donald Knuth. Unable to bear how poorly the formulas in his own books were being printed, he resolved to build a typesetting system himself — so the story goes. For more than forty years since, nearly every mathematics and physics paper and textbook in the world has been written in this notation. MathJax is the engine that brings TeX's formula notation to the web browser — and it is what renders the text cells in Colab. In other words, the notation in this appendix is the very "common language of formulas" that researchers use to write their papers. Now that you can write a line of mathematics in your own notes, you are standing at the entrance to that tradition.

Appendix Summary

- In a Colab text cell, wrap LaTeX notation in $ signs to write formulas (MathJax renders them).
- ^ for superscripts, _ for subscripts; two or more characters go inside { }.
- Fractions are \dfrac{ }{ }, roots are \sqrt{ }; symbols and Greek letters are \ plus a name.
- Function names (\sin, \log, …) take a leading \ to stand upright; a logarithm's base is a subscript.
- Whenever the notation slips your mind, come back to these tables and check.

Afterword

As technology grows fast and AI becomes a normal part of our lives, more and more people are thinking, "I want to learn the basics properly once again."

Mathematical thinking and programming skills are no longer just for experts. They are essential knowledge that everyone should have today. However, just hearing the word "math" might make many people remember how hard it was during their school days.

That is why this book is designed not to teach math and Python separately, but to connect them so you can "understand intuitively by seeing." By translating math formulas into code and running them, lists of symbols that once looked boring will suddenly come to life and make sense.

For the younger generation, this book will be a passport to the future. For working adults, it will be a tool to greatly expand your life's possibilities. It is completely okay if you couldn't keep up with math classes in the past, or if you never had the chance to learn.

I hope this book helps you get back the joy of learning at your own pace, and that you take your first step toward the endless search for truth.

Author / Hideaki Miyoshi.

The author founded an ISP in the late 1990s and went on to contribute to the development of numerous systems, including large-scale e-commerce platforms and B2B transaction systems. In recent years, he has focused on developing talent capable of adapting to the rapid evolution of AI.

I would like to express my heartfelt gratitude to Ms. Geraldine Angaga Lickas for her dedicated efforts in preparing the code and adapting the explanatory text to an ESL-friendly level for this book.

HIDEAKI MIYOSHI