# CODE QUEST

## WITH PYTHON

**GAMES | ART | MUSIC | AI**



A no-install introduction for curious coders ages 8-12

# This book belongs to

Coder name:

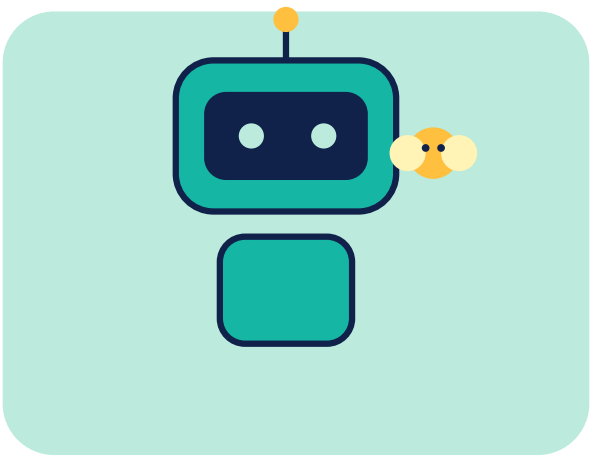
_____

Quest start date:

_____

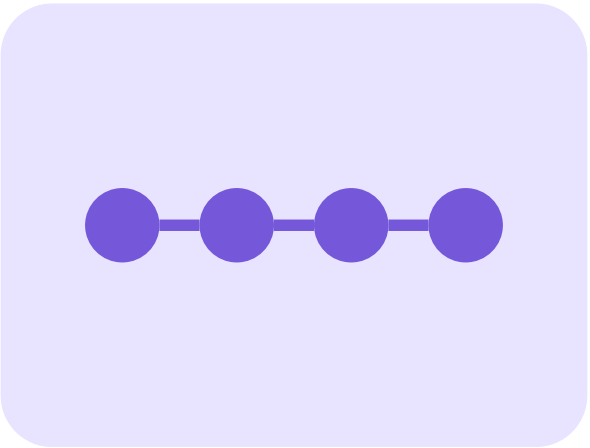
My project idea:

_____



# For grown-ups and guides

This is a short, hands-on introduction rather than a complete Python reference. A learner can finish it in 7-14 days at about 45-75 minutes per day.



Stay nearby for the first online-editor visit. The child should use invented personal information, avoid chat and sharing features, and ask before opening links. Let the learner type, predict, and debug. Helpful questions include: What did you expect? What changed? What is the smallest test you can try?

All projects use original characters, rules, text, and shapes.

Copyright 2026. All rights reserved. Text, code, characters, and vector diagrams were created for this edition. The cover hero is an original AI-generated illustration directed for this book. No endorsement by any named website is implied.

# Open the coding lab

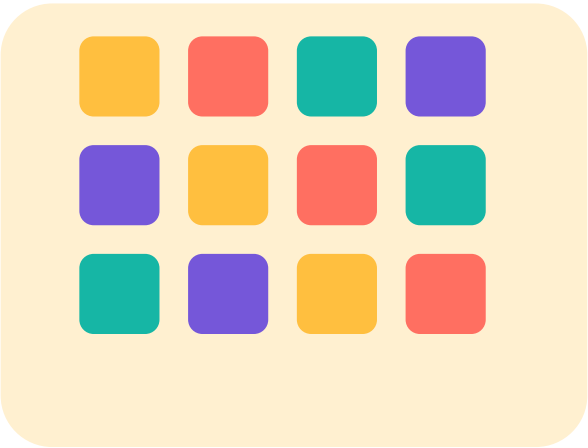
1. With an adult's permission, open play.pyroom.app in a modern browser.

2. Choose a Python console for text programs or Turtle for drawing projects.

3. Type the code in the dark code boxes. Press Run. Input questions may appear in the output area.

4. If Pyroom is unavailable, use another current Python 3 browser editor for text projects. For Turtle, try skulpt.org.

Websites can change. The lessons teach standard Python ideas and do not require an account.

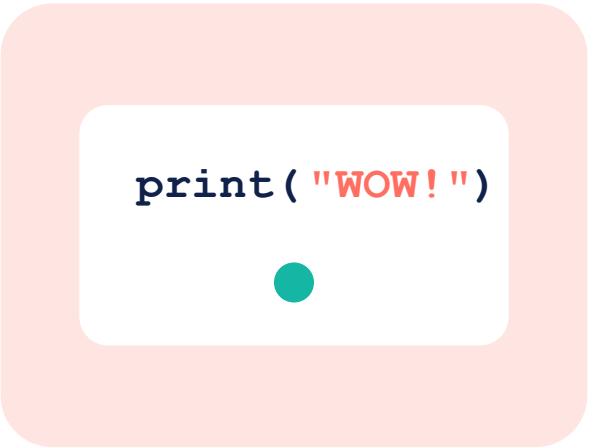


# Your safety shield

Ask an adult before visiting a new website, creating an account, downloading a file, turning on a camera or microphone, or sharing a project.

Use invented names, locations, passwords, and creature facts in exercises. Do not paste private information into a program or AI service.

Online editors are tools supplied by other organizations. This book is not sponsored by or connected with them. Their availability and features may change.



# Quest map

Quests 1-4: computers, algorithms, and the online editor

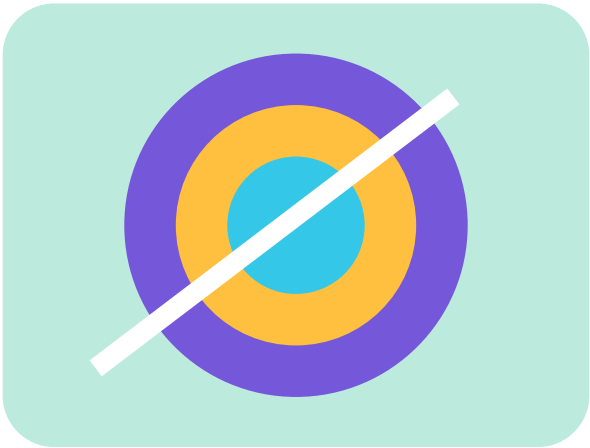
Quests 5-8: variables, input, decisions, and loops

Quests 9-10: randomness and functions

Quests 11-13: Turtle art, games, and music

Quest 14: AI and responsible use

Quest 15: plan, build, test, and share an original final project



# 01

YOUR FIRST MISSION

# Welcome to Code Quest



**Programming is giving a computer a set of clear instructions.**

**MISSION**

Read, run, change, and explain the code. Finish with the Boss Challenge.

LEARN

# Three ideas to unlock

1. A computer can help us draw, calculate, play sounds, tell stories, and make games. It cannot guess what we mean. It follows the instructions we give it.

2. A program is a saved set of instructions. Code is the way we write those instructions in a programming language. In this book, our language is Python.

3. You do not need to be a math expert. You need curiosity, patience, and a willingness to try tiny changes.

Pause and explain each idea in your own words.



SEE IT

# Read the code like a story

Predict first. Then run it in the online editor.

```
print("Hello, coder!")
print("My quest begins now.")
```

### ONE POSSIBLE OUTPUT

```
Hello, coder!
My quest begins now.
```

### CODE DETECTIVE

Point to the part that can change. Point to the part that produces output. What do you think each punctuation mark is doing?

# Try it: small changes, big discoveries

1. Change coder to your first name or a made-up hero name.



2. Add a third print line with a goal for your quest.

3. Predict the output before you press Run.

## MY PREDICTION

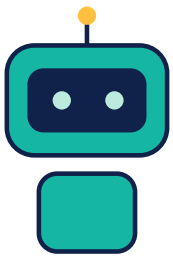
**SAFETY SHIELD**

Use invented names and information. Ask a trusted adult before visiting a new website or sharing a project link.

BUG HUNT

# Pip's program has a bug!

Run the buggy version. Read the error message or strange output. Circle the smallest part you think should change.



```
print("Hello, coder!)
```

YOUR FIX

## CHECK AFTER YOU TRY

```
print("Hello, coder!")
```

Debugging pattern: observe -> guess -> change one thing -> run again.

# Save point

Three words worth keeping in your coding backpack

| | |
|---|---|
| **program** | a set of computer instructions |
| **code** | instructions written in a programming language |
| **run** | ask the computer to follow the code |

## BOSS CHALLENGE

**Make a five-line opening scene for an original game.**

When it works, add your initials here: _____

## REFLECT

- What did I discover?
- What did I create?
- What would I change next?

# 02

INPUT, PROCESS, OUTPUT

# What Is a Computer?



**A computer takes input, processes it, stores data, and produces output.**

**MISSION**

Read, run, change, and explain the code. Finish with the Boss Challenge.

# Three ideas to unlock

1. A keyboard, mouse, microphone, camera, and game controller can provide input. Input is information sent into a computer.

2. The processor follows instructions. Memory holds items being used right now. Storage keeps files for later, even after the power is off.

3. A screen, speaker, printer, and vibrating controller can produce output. One device can be both input and output: a touchscreen shows pictures and senses taps.

Pause and explain each idea in your own words.



SEE IT

# Read the code like a story

Predict first. Then run it in the online editor.

```
player = input("Hero name: ")
print("Welcome,", player)
```

### ONE POSSIBLE OUTPUT

```
Hero name: Nova
Welcome, Nova
```

### CODE DETECTIVE

Point to the part that can change. Point to the part that produces output. What do you think each punctuation mark is doing?

# Try it: small changes, big discoveries



1. Name two inputs used by a game.

2. Name two outputs used by a music app.

3. Draw arrows showing input -> process -> output.

## MY PREDICTION

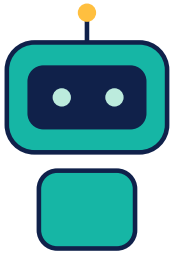
**SAFETY SHIELD**

Use invented names and information. Ask a trusted adult before visiting a new website or sharing a project link.

BUG HUNT

# Pip's program has a bug!

Run the buggy version. Read the error message or strange output. Circle the smallest part you think should change.



```
player = input("Hero name: ")
print("Welcome," player)
```

YOUR FIX

## CHECK AFTER YOU TRY

```
player = input("Hero name: ")
print("Welcome,", player)
```

Debugging pattern: observe -> guess -> change one thing -> run again.

# Save point

Three words worth keeping in your coding backpack

**hardware** — physical parts you can touch

**software** — programs and their data

**memory** — temporary working space

BOSS CHALLENGE

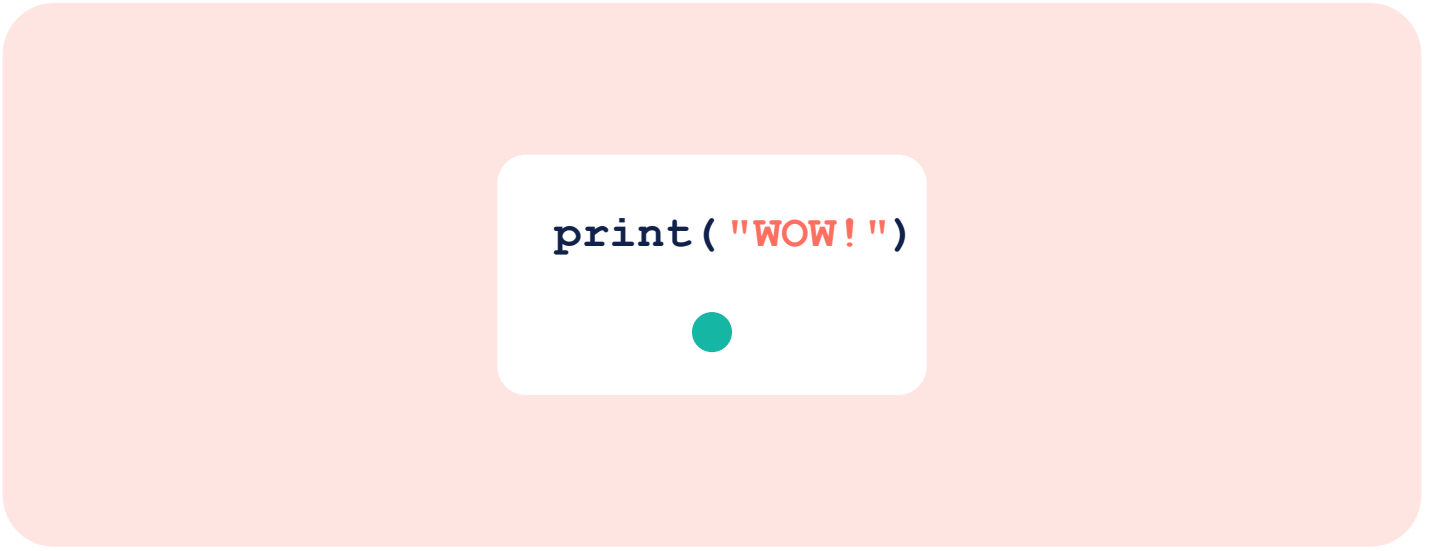
**Invent a computer hidden inside a backpack. List its input, process, storage, and output.**

When it works, add your initials here: _____

REFLECT

- What did I discover?
- What did I create?
- What would I change next?

# 03 THINK LIKE A ROBOT
# Algorithms and Bugs




**An algorithm is an ordered plan for solving a problem.**

**MISSION**

Read, run, change, and explain the code. Finish with the Boss Challenge.

# Three ideas to unlock

1. Imagine telling a robot how to draw a square. 'Draw a square' is not precise enough. The robot needs steps: move, turn, move, turn, and so on.

2. A bug is a mistake in code or in the plan. Debugging means finding the cause, changing one thing, and testing again.

3. Good programmers do not avoid every bug. They learn to read clues, make a small change, and run the program again.

Pause and explain each idea in your own words.



SEE IT

# Read the code like a story

Predict first. Then run it in the online editor.

```
step = 1
print("Step", step, ": open the map")
step = step + 1
print("Step", step, ": choose a path")
```

### ONE POSSIBLE OUTPUT

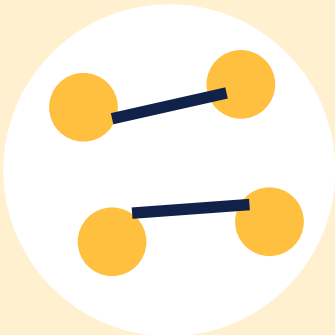
```
Step 1 : open the map
Step 2 : choose a path
```

### CODE DETECTIVE

Point to the part that can change. Point to the part that produces output. What do you think each punctuation mark is doing?

# Try it: small changes, big discoveries

1. Write steps for brushing teeth without using the word 'etcetera'.



2. Have someone follow your steps exactly.

3. Circle any step that needs more detail.

## MY PREDICTION

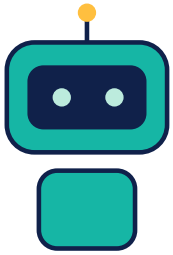
**SAFETY SHIELD**

Use invented names and information. Ask a trusted adult before visiting a new website or sharing a project link.

BUG HUNT

# Pip's program has a bug!

Run the buggy version. Read the error message or strange output. Circle the smallest part you think should change.



```
print("First")
print("Third")
print("Second")
```

YOUR FIX

## CHECK AFTER YOU TRY

```
print("First")
print("Second")
print("Third")
```

Debugging pattern: observe -> guess -> change one thing -> run again.

# Save point

Three words worth keeping in your coding backpack

**algorithm** — an ordered plan

**bug** — a mistake that causes a problem

**debug** — find and repair the cause of a problem

BOSS CHALLENGE

**Write an algorithm that guides Pip through a four-square maze.**

When it works, add your initials here: _____

REFLECT

- What did I discover?
- What did I create?
- What would I change next?