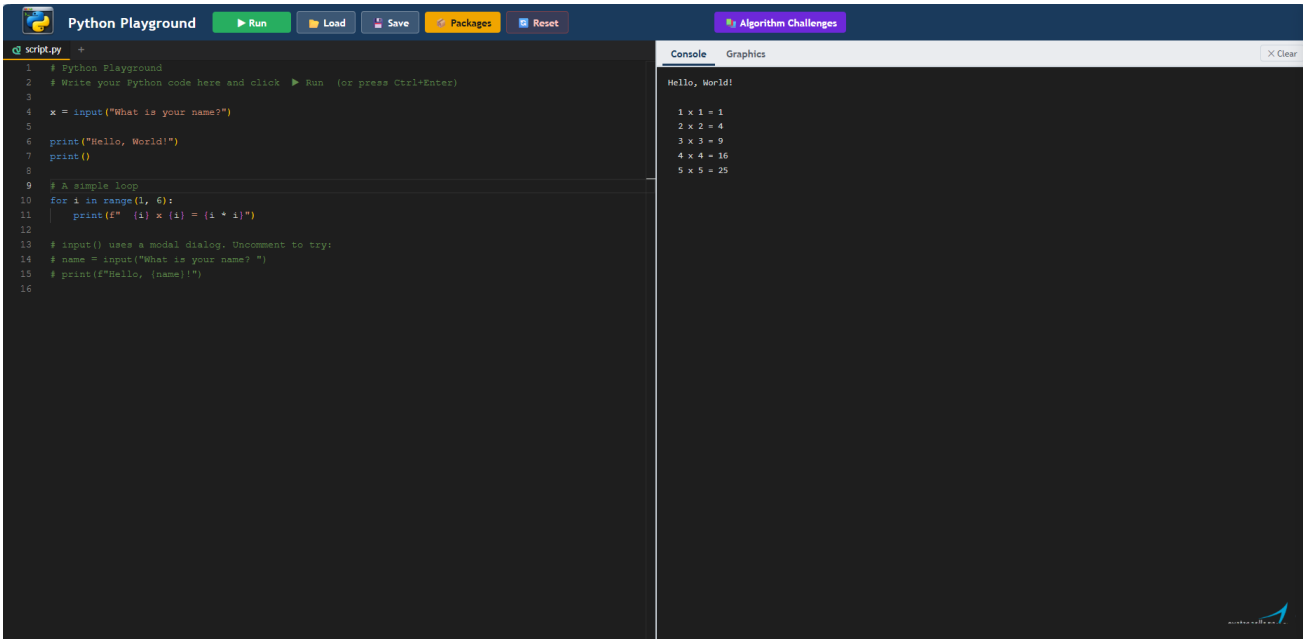
Python Playground

Student User Manual

Level 2 Edition

Write and run Python code — right in your browser

No installation needed. No account required.



Contents

Contents

1 What is Python Playground?

Python Playground is a website that lets you write and run Python programs straight in your web browser. You do not need to install anything — just open the page and start coding.

It is designed for students who are learning Python. You can:

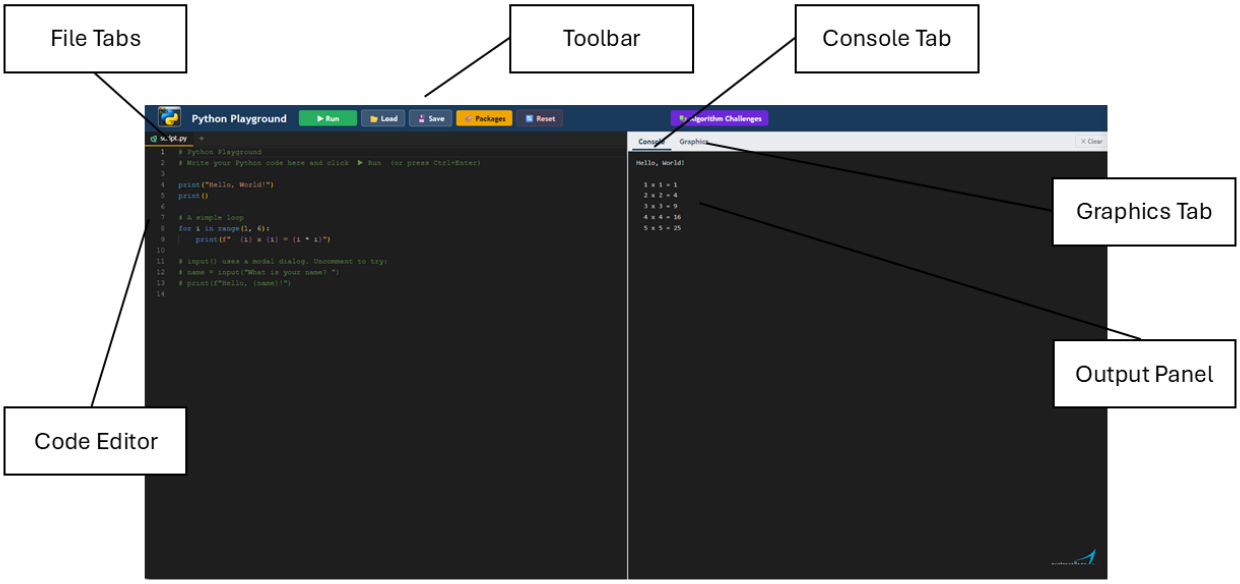
- Type and edit Python code in the editor on the left
- Click Run (or press Ctrl + Enter) to execute your program
- See text output and error messages in the panel on the right
- Draw shapes with Turtle graphics
- Create charts and graphs with Matplotlib
- Load a .py file from your computer to continue working on it
- Save your work back to your computer

Good to Know

Python Playground runs Python in your browser — it does not send your code anywhere else. Each time you reload the page, Python starts fresh. Any variables or installed libraries will need to be set up again.

2 The Screen Layout

When you open Python Playground, the screen is divided into areas. The diagram below shows what each part does.



Area	What it does
Toolbar (top)	Contains the Run, Load, Save, Packages and Reset buttons
File Tabs (below toolbar)	Shows the files in your workspace — click a tab to switch between them
Code Editor (left)	Where you type your Python code. Line numbers appear on the left
Output Panel (right)	Displays text output (Console tab) and graphics (Graphics tab)
Console tab	Shows anything your program prints, plus any error messages
Graphics tab	Shows Turtle drawings and Matplotlib charts

3 Writing Your First Program

When the page loads, a short example program is already in the editor. You can delete it and write your own, or edit it to experiment.

Here is a simple program to try:

```
# This is a comment — Python ignores it
name = input("What is your name? ")
print("Hello,", name)
print("Welcome to Python Playground!")
```

Comments

Any line starting with # is a comment. Comments are notes for you — Python does not run them. Use comments to explain what your code does.

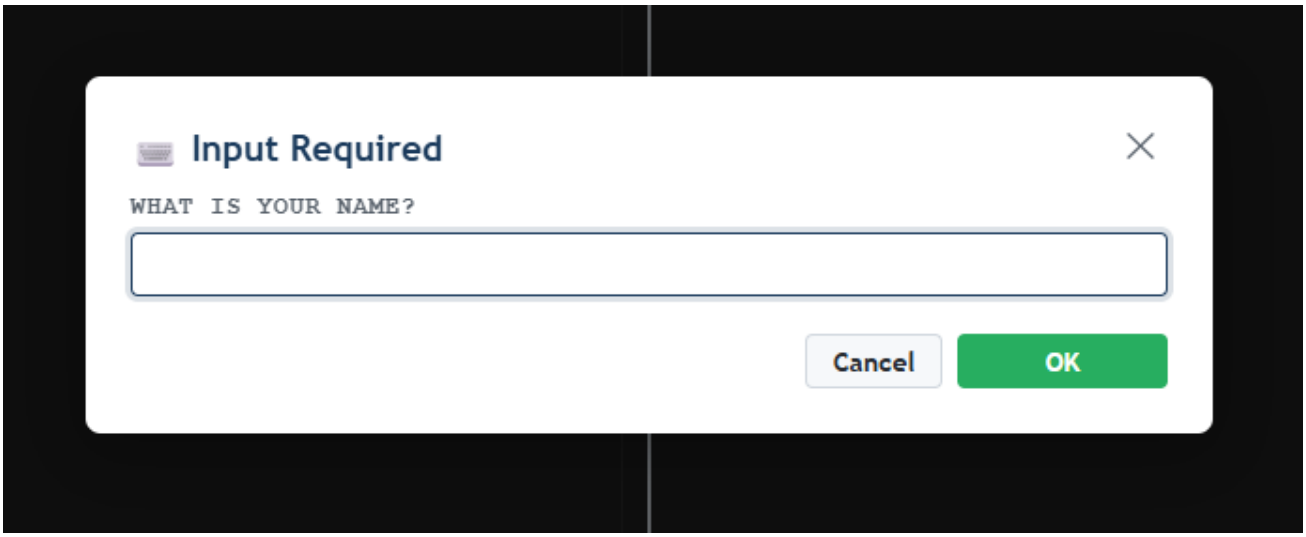
4 Running Your Code

Once you have written some code, you run it to see what it does.

How to run your program

1. Click the ▶ Run button in the toolbar at the top of the page.
2. **Or** press **Ctrl + Enter** on your keyboard (the quickest way).
3. The output panel on the right will show the result.

If your program uses `input()` — for example asking for your name — a pop-up box will appear on screen. Type your answer and press OK.



> ⚠ **If the Run button is grey**
> Python Playground is still loading the Python engine the first time you visit.
> Wait a few seconds until the button becomes coloured, then try again.

5 Reading the Output

Console tab

The Console tab shows:

- **White text** — the output from your print() statements
- **Red text** — error messages when something goes wrong

If you see a red error message, read it carefully — it usually tells you exactly which line has the problem. Look for the line number and a short description of the error.

```
Traceback (most recent call last):
  File "<string>", line 3, in <module>
NameError: name 'mesage' is not defined
```

The example above says the error is on line 3, and that 'mesage' is not a known variable (it is probably a spelling mistake — it should be 'message').

Graphics tab

Turtle drawings and Matplotlib charts appear in the Graphics tab. Python Playground will switch to this tab automatically when your program creates a drawing or chart.

Click the Clear button at the top of the output panel to erase previous output and start fresh.

6 Loading and Saving Files

Saving your work

Your code is not saved automatically. Always save before you close the browser.

1. Click the 💾 Save button in the toolbar.
2. A box will appear asking for a filename.
3. Type a name for your file (for example: my_program) and click Save.
4. Your browser will download the file as a .py file to your Downloads folder.

> **Tip**
> Move the downloaded file to your student folder straight away so you can find it later.

Loading a saved file

1. Click the 📂 Load button in the toolbar.
2. A file picker window will open.
3. Find your .py file and click Open.
4. Your code will appear in the editor ready to run.

> **⚠ Loading replaces what is in the editor**
> When you load a file, it replaces the code that is currently in the editor.
> If you have unsaved changes, save first!

Multiple files

You can have more than one file open at the same time. Each file appears as a tab above the editor.

1. Click the + button next to the tabs to load an extra file into your workspace.
2. Click a tab to switch to that file.
3. Click the × on a tab to close that file.
4. If you save when more than one file is open, all files are saved together as a single .zip archive.

7 Turtle Graphics

Turtle graphics lets you draw pictures and shapes using Python commands. You control a small turtle that moves around the screen drawing lines as it goes.

You do not need to install anything — just write import turtle at the top of your program.

Getting started

```
import turtle

turtle.forward(100)     # move forward 100 steps
turtle.right(90)        # turn right 90 degrees
turtle.forward(100)
turtle.right(90)
turtle.forward(100)
turtle.right(90)
turtle.forward(100)
# This draws a square!
```

The drawing appears in the Graphics tab on the right.

Common turtle commands

Command	What it does
`turtle.forward(n)`	Move forward n steps
`turtle.backward(n)`	Move backward n steps
`turtle.right(angle)`	Turn right by angle degrees
`turtle.left(angle)`	Turn left by angle degrees
`turtle.penup()`	Lift the pen — move without drawing
`turtle.pendown()`	Put the pen down — draw as you move
`turtle.pencolor(colour)`	Change the drawing colour
`turtle.fillcolor(colour)`	Set the fill colour
`turtle.begin_fill()`	Start filling a shape
`turtle.end_fill()`	Finish filling the shape
`turtle.circle()`	Draw a circle
`turtle.speed(speed)`	Set speed 1 (slow) to 10 (fast), 0 = instant
`turtle.bgcolor(colour)`	Change the background colour
`turtle.hideturtle()`	Make the turtle cursor invisible
`turtle.goto(x,y)`	Jump to a specific position on screen

Example — coloured star

```
import turtle

colours = ["red", "orange", "yellow", "green", "blue"]
turtle.speed(5)

for i in range(5):
    turtle.pencolor(colours[i])
    turtle.forward(150)
    turtle.right(144)

turtle.hideturtle()
```

8 Drawing Charts with Matplotlib

Matplotlib is a library for creating charts and graphs. Python Playground supports it — charts appear automatically in the Graphics tab after your program runs.

Bar chart example

```
import matplotlib.pyplot as plt

subjects = ["Maths", "English", "Science", "IT"]
scores   = [72, 85, 68, 91]

plt.bar(subjects, scores, color="steelblue")
plt.title("My Test Scores")
plt.ylabel("Score (%)")
plt.ylim(0, 100)
plt.show()
```

Line chart example

```
import matplotlib.pyplot as plt

months = ["Jan", "Feb", "Mar", "Apr", "May"]
temps  = [4.2, 5.1, 8.3, 11.7, 15.2]

plt.plot(months, temps, marker="o", color="tomato")
plt.title("Average Temperature")
plt.ylabel("Temperature (°C)")
plt.grid(True)
plt.show()
```

plt.show()
Always call plt.show() at the end of your chart code.
This tells Python Playground to capture and display the chart.

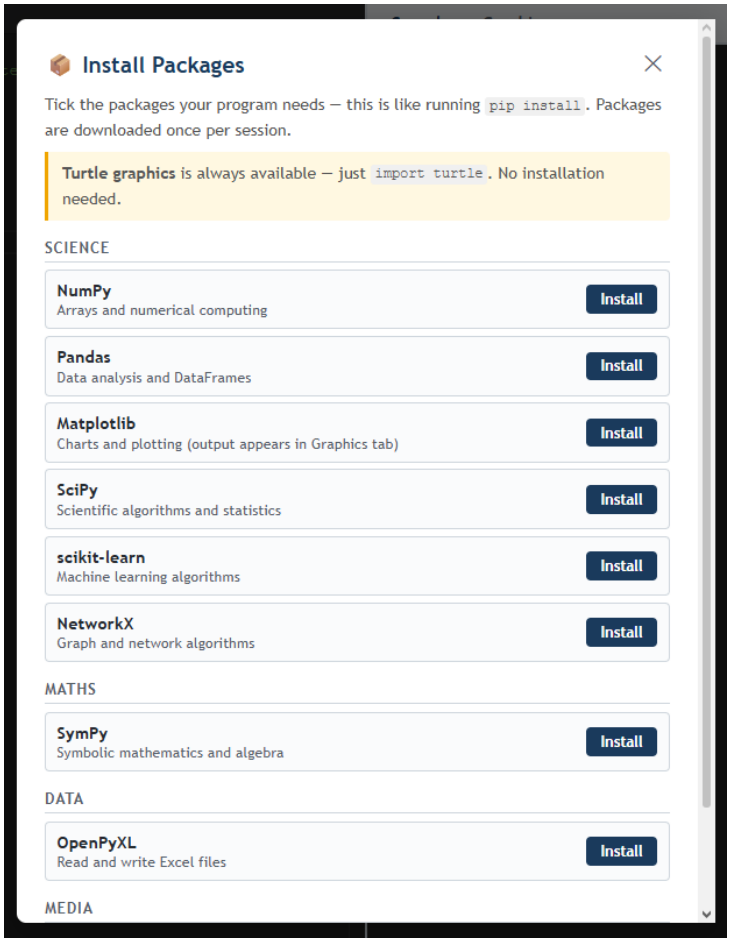
Common chart types

Chart type	Matplotlib command
Bar chart	plt.bar(x_values, y_values)
Horizontal bar	plt.barh(x_values, y_values)
Line chart	plt.plot(x_values, y_values)
Scatter plot	plt.scatter(x_values, y_values)
Pie chart	plt.pie(values, labels=labels)
Histogram	plt.hist(data, bins=10)

9 Installing Libraries

A library is a collection of extra Python code that adds new features to the language. For example, Pandas helps you work with tables of data, and Matplotlib creates charts.

Some libraries are available in Python Playground but need to be installed before you can use them. This takes about 30–60 seconds and only needs to happen once per session.



How to install a library

1. Click the Packages button in the toolbar.
2. The Library Manager window opens with a list of available libraries.
3. Tick the box next to the library you want.
4. Click Install.
5. Wait for the green tick to appear — the library is now ready to use.

> ⚠ **Libraries reset on page reload**
> If you reload the page, all installed libraries are removed.
> You will need to install them again.
> Your code is also lost unless you saved it first.

Available libraries

Library	What it does
`Turtle`	Draw shapes and pictures (always available — no installation needed)
`Matplotlib`	Create charts, graphs, and plots
`NumPy`	Work with numbers and arrays quickly
`Pandas`	Work with tables of data (like a spreadsheet in Python)
`SciPy`	Advanced maths and science calculations
`SymPy`	Algebra and symbolic maths (solve equations, factorise, etc.)
`Pillow`	Load and edit images
`OpenPyXL`	Read and write Excel spreadsheet files
`NetworkX`	Work with networks and graphs (nodes and edges)
`scikit-learn`	Machine learning algorithms

10 Using Libraries

Once a library is installed, you import it at the top of your program using the import keyword.

NumPy — working with numbers

NumPy makes it easy to do maths on large lists of numbers all at once.

```
import numpy as np

numbers = np.array([4, 8, 15, 16, 23, 42])
print("Sum:",  np.sum(numbers))
print("Mean:", np.mean(numbers))
print("Max:",  np.max(numbers))
```

Pandas — working with data tables

Pandas lets you create and work with tables of data called DataFrames — similar to a spreadsheet.

```
import pandas as pd

# Create a small table
data = {
    "Name":  ["Alice", "Bob", "Charlie"],
    "Score": [88, 72, 95],
}
```

```
df = pd.DataFrame(data)

print(df)
print()
print("Average score:", df["Score"].mean())
```

SymPy — solving maths problems

SymPy can solve equations, simplify expressions and do calculus.

```
from sympy import symbols, solve, expand, factor

x = symbols("x")

# Solve x² - 5x + 6 = 0
solutions = solve(x**2 - 5*x + 6, x)
print("Solutions:", solutions)

# Expand (x + 2)(x - 3)
print("Expanded:", expand((x + 2) * (x - 3)))
```

import as

Writing `import numpy as np` creates a short nickname.
Instead of typing `numpy.sum()` every time, you can write `np.sum()`.
This is just a shortcut — the code does exactly the same thing.

11 Working with CSV Files

What is a CSV file?

A CSV (Comma-Separated Values) file stores data in a plain text file. Each row of data is on its own line, and the values are separated by commas.

Example CSV content (saved as `students.csv`):

```
Name,Age,Score
Alice,16,88
Bob,17,72
Charlie,16,95
Diana,17,81
```

How to use a CSV file in Python Playground

1. Save your CSV file on your computer.
2. In Python Playground, click the + button next to the file tabs.
3. Select your CSV file from the file picker.
4. The file appears as a new tab in the workspace.
5. Your Python code can now open it using its filename.

Reading a CSV file with the csv module

```
import csv

# Open and read the CSV file
with open("students.csv") as f:
    reader = csv.DictReader(f)
    for row in reader:
        print(row["Name"], "scored", row["Score"])
```

csv.DictReader reads each row as a dictionary, using the first row (the header) as the keys.

Reading a CSV file with Pandas

If you have installed Pandas, this is even easier:

```
import pandas as pd

df = pd.read_csv("students.csv")
print(df)
print()
print("Average score:", df["Score"].mean())
print("Highest score:", df["Score"].max())
```

Creating a chart from CSV data

Combining Pandas and Matplotlib lets you visualise your CSV data:

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("students.csv")

plt.bar(df["Name"], df["Score"], color="steelblue")
plt.title("Student Scores")
plt.ylabel("Score")
plt.ylim(0, 100)
plt.show()
```

Useful Pandas commands for CSV data

Command	What it does
`df.head(5)`	Show the first 5 rows of the table
`df.tail(5)`	Show the last 5 rows of the table
`df.shape`	Show the number of rows and columns
`df.columns`	List the column headings
`df["Column"]`	Select one column

Command	What it does
`df.mean()`	Average of each numeric column
`df.max()`	Largest value in each numeric column
`df.min()`	Smallest value in each numeric column
`df.sort_values("Column")`	Sort the table by a column
`df[df["Col"] > 80]`	Filter rows where a column meets a condition

12 Keyboard Shortcuts

Shortcut	Action
`Ctrl + Enter`	Run your program
`Ctrl + Z`	Undo last change
`Ctrl + Y`	Redo
`Ctrl + A`	Select all code
`Ctrl + C`	Copy selected text
`Ctrl + V`	Paste
`Ctrl + /  (forward slash)`	Comment or uncomment a line
`Tab`	Indent selected lines
`Shift + Tab`	Remove indent from selected lines

Comment out lines

Select one or more lines and press Ctrl + / to turn them into comments.
This is useful for testing — you can hide lines without deleting them.
Press Ctrl + / again to uncomment them.

13 Common Errors and Fixes

Error message	Likely cause and fix
`SyntaxError: invalid syntax`	A typing mistake in your code — often a missing colon (:), bracket, or quote. Check the line shown in the error.
`NameError: name 'x' is not defined`	'x' (or whatever the name is) has not been created yet. Check your spelling or make sure you assigned the variable before using it.
`IndentationError`	The indentation (spaces at the start of a line) is wrong. Python uses indentation to group code — make sure it is consistent (4 spaces per level).

Error message	Likely cause and fix
`TypeError`	You are using the wrong type of data — for example, trying to add a number to a string. Use int() or str() to convert values.
`FileNotFoundError: No such file or directory`	Python cannot find the file. Make sure you uploaded it using the + tab button and spelled the filename exactly right.
`ModuleNotFoundError`	The library has not been installed yet. Go to Packages and install it first.
`ZeroDivisionError`	Your program tried to divide a number by zero. Add a check before the division.
`Nothing happens when I click Run`	Check that the Run button is not grey — Python may still be loading. Wait a few seconds and try again.

14 Glossary

Term	Meaning
Algorithm	A step-by-step set of instructions that solves a problem
Array	An ordered collection of values (called a list in standard Python; arrays are used in NumPy)
Boolean	A value that is either True or False
Comment	A line starting with # that Python ignores — used to leave notes in code
CSV	Comma-Separated Values — a simple file format for storing data in a table
DataFrame	A Pandas table of data with rows and columns, similar to a spreadsheet
Dictionary	A Python collection that stores data as key: value pairs
Function	A named block of code that does a specific task, called with function_name()
Import	Bringing a library into your program so you can use its features
Indentation	Spaces at the start of a line that show which block of code a line belongs to
Library	A collection of extra Python code that adds new features
Loop	A block of code that repeats — for loops repeat a set number of times, while loops repeat until a condition is False
Matplotlib	A Python library for drawing charts and graphs
Module	A single Python file that can be imported into other programs
NumPy	A Python library for fast maths on arrays of numbers

Term	Meaning
Output	What your program displays on screen using print()
Pandas	A Python library for working with tables of data
Parameter	A value you pass into a function — e.g. the 100 in forward(100)
Pyodide	The technology that makes Python run inside a web browser
String	Text data surrounded by quotes — e.g. "Hello"
Syntax	The rules about how Python code must be written
Variable	A named storage location for a value — e.g. score = 85

Python Playground

Student User Manual — Level 2 Edition