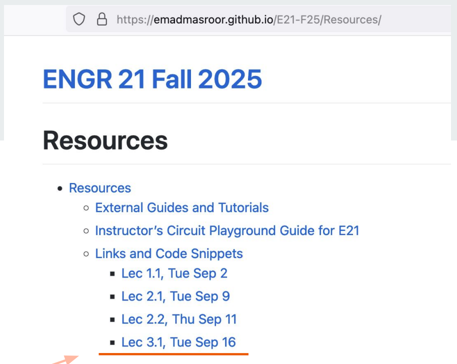
ENGR 21: Computer Engineering Fundamentals

Lecture 6
Thursday, September 18, 2025

Test #1
8:30 to 8:55

Saving Data in Files On the Circuit Playground Bluefruit


https://emadmasroor.github.io/E21-F25/Resources/

ENGR 21 Fall 2025

Resources

- Resources
  - External Guides and Tutorials
  - Instructor's Circuit Playground Guide for E21
  - Links and Code Snippets
    - Lec 1.1, Tue Sep 2
    - Lec 2.1, Tue Sep 9
    - Lec 2.2, Thu Sep 11
    - Lec 3.1, Tue Sep 16

Storing Data on the Circuit Playground Bluefruit

It is possible to store data on the board, even when disconnected from your PC and powered with battery

Task: Download `boot.py` and save it to `CIRCUITPY`.

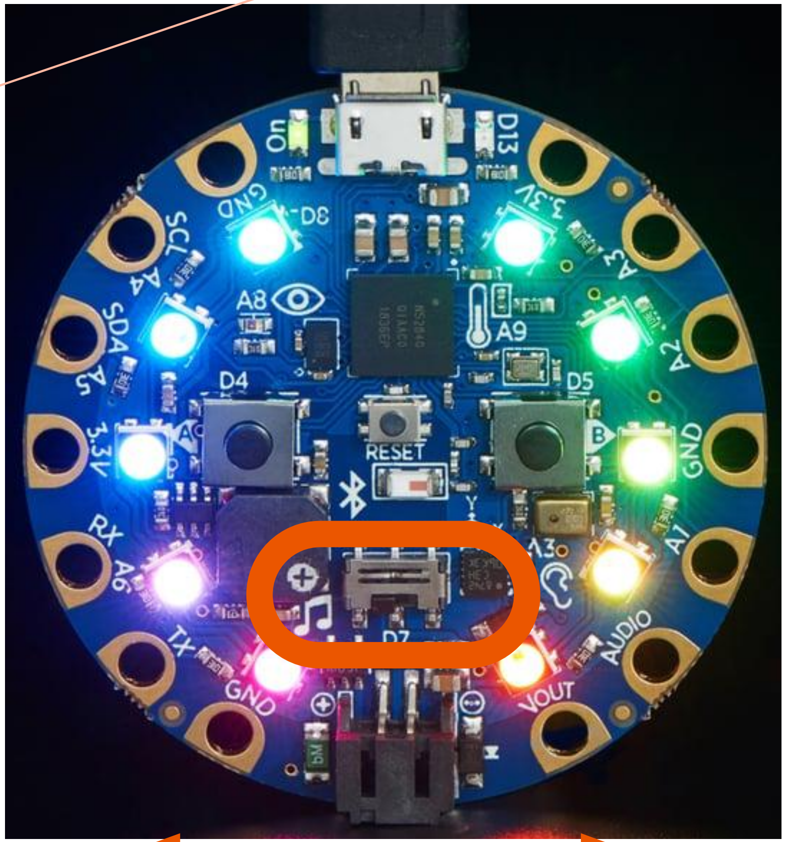
After boot.py is on your board, you will be able to switch between

1. Read-only mode
2. Writable mode

Either your computer can save files to your board, or CircuitPython can save files to your board. not both!!

To switch between modes:

- Slide switch
- Eject `CIRCUITPY` from OS
- Press reset button

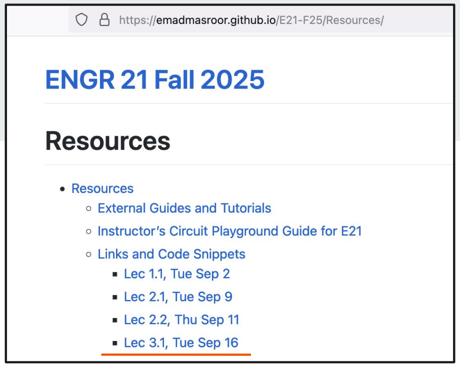


Read-only (by CircuitPython)
Computer can write

Writable (by CircuitPython)
Computer can't write

Storing Data on the Circuit Playground Bluefruit

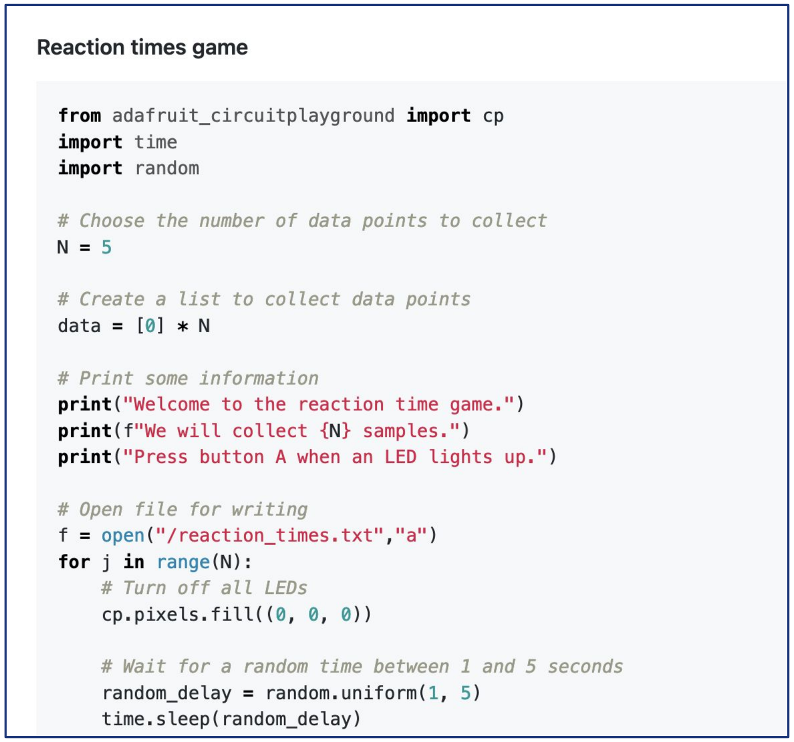
Download Reaction times game from Resources page


https://emadmasroor.github.io/E21-F25/Resources/

ENGR 21 Fall 2025

Resources

- Resources
  - External Guides and Tutorials
  - Instructor's Circuit Playground Guide for E21
  - Links and Code Snippets
    - Lec 1.1, Tue Sep 2
    - Lec 2.1, Tue Sep 9
    - Lec 2.2, Thu Sep 11
    - Lec 3.1, Tue Sep 16


Reaction times game

```
from adafruit_circuitplayground import cp
import time
import random

# Choose the number of data points to collect
N = 5

# Create a list to collect data points
data = [0] * N

# Print some information
print("Welcome to the reaction time game.")
print(f"We will collect {N} samples.")
print("Press button A when an LED lights up.")

# Open file for writing
f = open("/reaction_times.txt","a")
for j in range(N):
    # Turn off all LEDs
    cp.pixels.fill((0, 0, 0))

    # Wait for a random time between 1 and 5 seconds
    random_delay = random.uniform(1, 5)
    time.sleep(random_delay)
```

```
# Open the file
f = open("/reaction_times.txt","a")

# Write to file
f.write(f"{reaction_time:.4f}\n")
f.flush()

# Close the f
f.close()
```

Activity: Modify Accelerometer to collect data

Goal: Save 30 seconds of accelerometer data when switched on with battery power, into a text file on board the CIRCUITPY.

Start from the code from Reactim Time Game and Accelerometer

(both from Resources Page)

Resources :
- ↳ Accelerometer code
- ↳ Code that writes reaction_times.txt

Combine into one file

Functions in (Circuit) Python

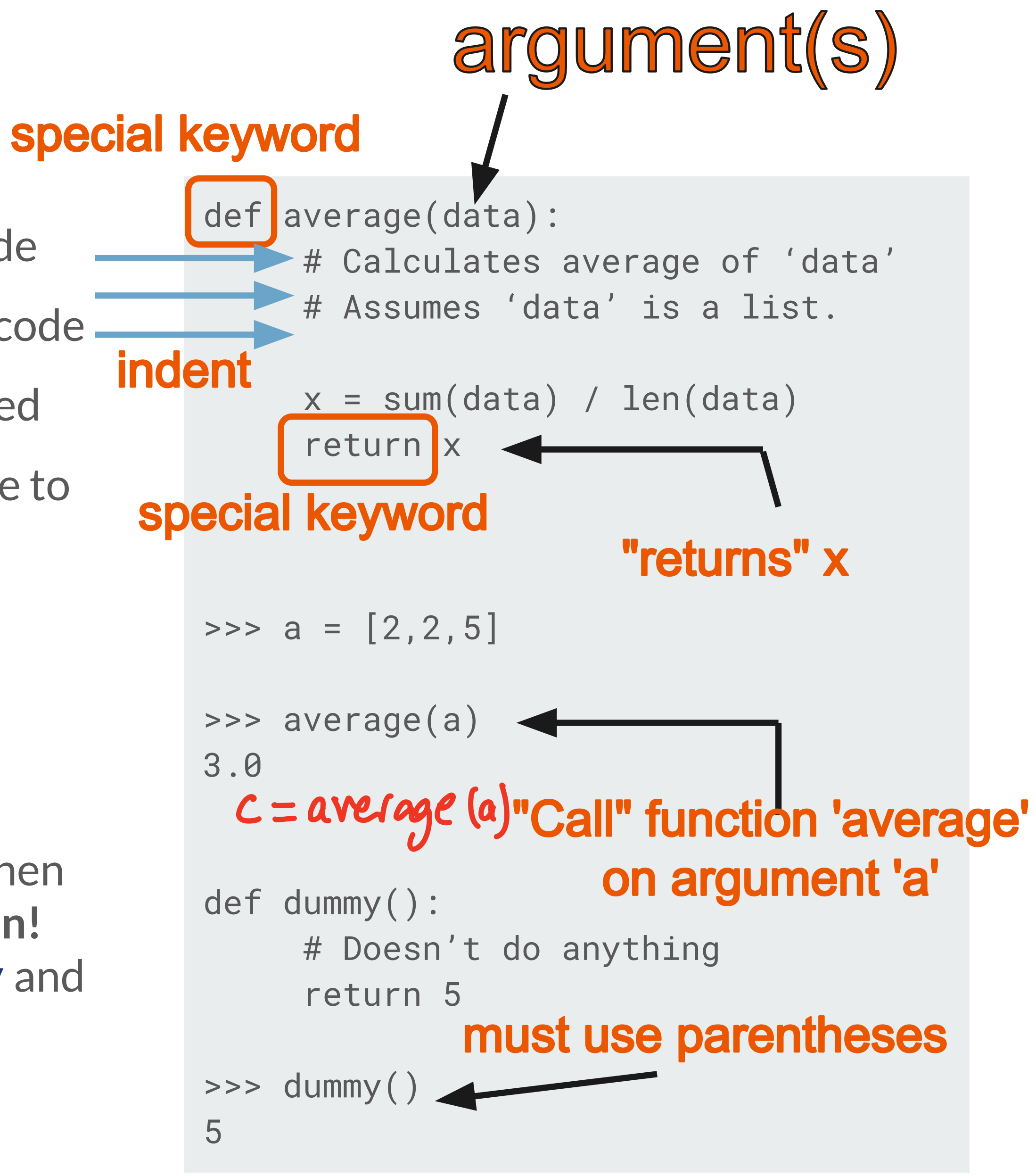
Remove `boot.py` from `CIRCUITPY` now if you'd like!

Anatomy of a function

- A function is a reusable piece of code that you can 'call' elsewhere in the code
- Keeps your code clean and organized
- Good software engineering practice to write (almost all) your code using functions

For CircuitPython, you can:

- Write a function at the REPL. It is then **only available for that REPL session!**
- Or you can write it inside `code.py` and use it inside `code.py`.



```
def average(data):
    # Calculates average of 'data'
    # Assumes 'data' is a list.

    x = sum(data) / len(data)
    return x


>>> a = [2,2,5]

>>> average(a)
3.0

def dummy():
    # Doesn't do anything
    return 5

>>> dummy()
5
```

What's possible with functions in Python

- Multiple arguments
  - Arguments can be <u>any</u> type
- No arguments
  - Remember to call it using parentheses
- Call one function inside another
  - As long as both functions have been defined
  - The order in which you define functions does not matter.
- A function that doesn't return anything

```
def func1(a,b):
    # Adds a and 2b
    return a + 2b
```

order of args matters!

```
>>> func1(3,4)
11

def func2():
    # Estimate pi
    return 22/7
>>> func2()
3.142...

def func3(x):
    # Call func1
    return func1(x,x)
```

Scoping Rules

- Functions have their own “scope”
- The same symbol can represent different things inside and outside a function.

```
>>> a = 5

>>> def func3(a,b):
...     print("The value of a is ",a)
...     return a + 4*b
...

>>> func3(6,4)
The value of a is  6
22

>>> a
5
```

Keyword Arguments

It is possible to enter the arguments explicitly by name instead of using the order.

If you use keyword arguments, order of arguments does not matter

```
def subtract(big,small):
    return big - small

>>> subtract(5,3)
2

>>> subtract(3,5)
-2

>>> subtract(big=5,small=3)
2

>>> subtract(small=3,big=5)
2
```

Best practices

- If using the REPL, can define functions 'on the fly'
- If collecting your code in a *.py file,
  - Plan out what functions you will need
  - Define all your functions first
  - Then write the body of your code
- Advanced:
  - Once your code gets long enough, you'll want to wrap some functions inside **modules**

Write a function to control NeoPixels

Write a function that accepts 3 arguments:

1. a **`string`** denoting the color
2. an **`int`** denoting the pixel number
3. A number denoting the intensity

Task: Complete the starter code and save it as code.py

```
import adafruit_circuitplayground as cp
import time
def lightUp(color,n,p):
    # Lights up pixel number n using color "color"
    # 'Color' should be a string, either 'red',
'blue', or 'green'
    # n should be an int between 0 and 9
    # p should be any number between 1 and 255
    return 42

# Now call it inside a while loop
while True:
    lightUp(‘red’,8,45)
    time.sleep(3)
    lightUp('green',5,200)
    time.sleep(3)
    cp.play_tone(440,3)
```

Installing Python on the computer

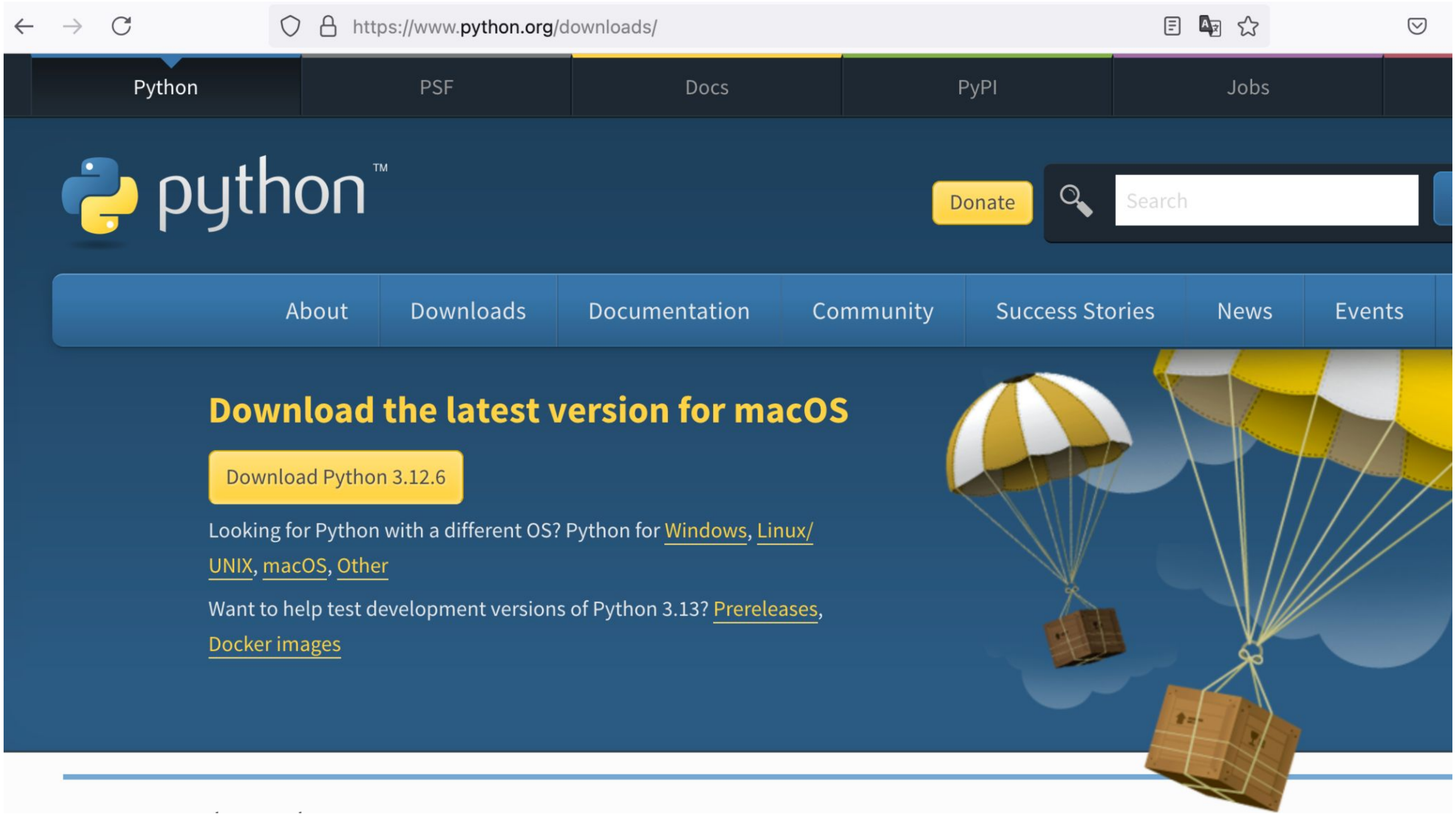
Starting with this week’

Python Installation Office Hours

with Prof. Masroor
Friday 1 - 2:30 PM
Singer 112 & TBD

The goal is to have (Desktop) Python up and running on your computer by this weekend https://www.python.org/downloads/

Installing Python on your computer



Download the latest version for macOS

Looking for Python with a different OS? Python for Windows, Linux/UNIX, macOS, Other

Want to help test development versions of Python 3.13? Prereleases, Docker images

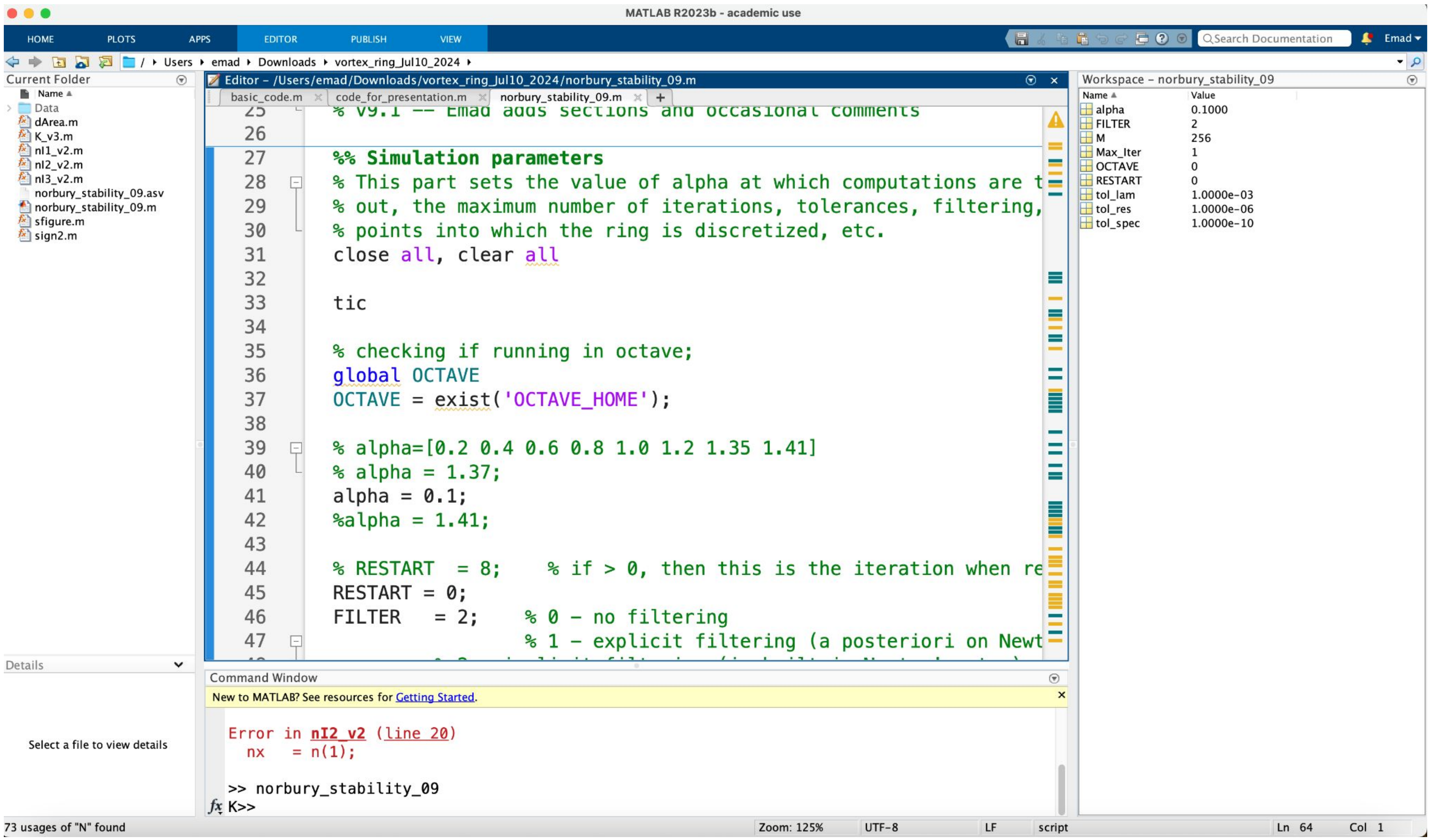
Choice of IDE (Integrated Development Environment)

What’s an IDE?

- A program that lets you interface with a programming language
- Usually a “visual” interface
- Multiple IDEs can be installed; they will use the same underlying programming language

IDEs you (may) have seen before

MATLAB has a built-in IDE



```
% v9.1 -- Emad adds sections and occasional comments

%% Simulation parameters
% This part sets the value of alpha at which computations are t
% out, the maximum number of iterations, tolerances, filtering,
% points into which the ring is discretized, etc.
close all, clear all

tic

% checking if running in octave;
global OCTAVE
OCTAVE = exist('OCTAVE_HOME');

% alpha=[0.2 0.4 0.6 0.8 1.0 1.2 1.35 1.41]
% alpha = 1.37;
alpha = 0.1;
%alpha = 1.41;

% RESTART  = 8;     % if > 0, then this is the iteration when re
RESTART = 0;
FILTER   = 2;    % 0 - no filtering
                 % 1 - explicit filtering (a posteriori on Newt
```

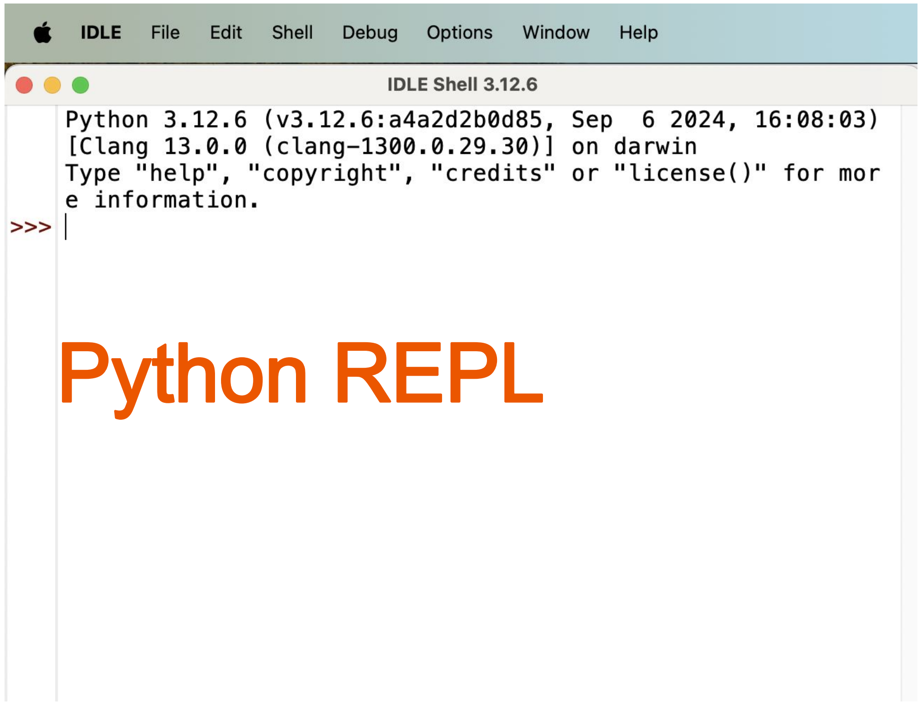
Name	Value
alpha	0.1000
FILTER	2
M	256
Max_Iter	1
OCTAVE	0
RESTART	0
tol_lam	1.0000e-03
tol_res	1.0000e-06
tol_spec	1.0000e-10

```
Error in nI2_v2 (line 20)
  nx   = n(1);

>> norbury_stability_09
K>>
```

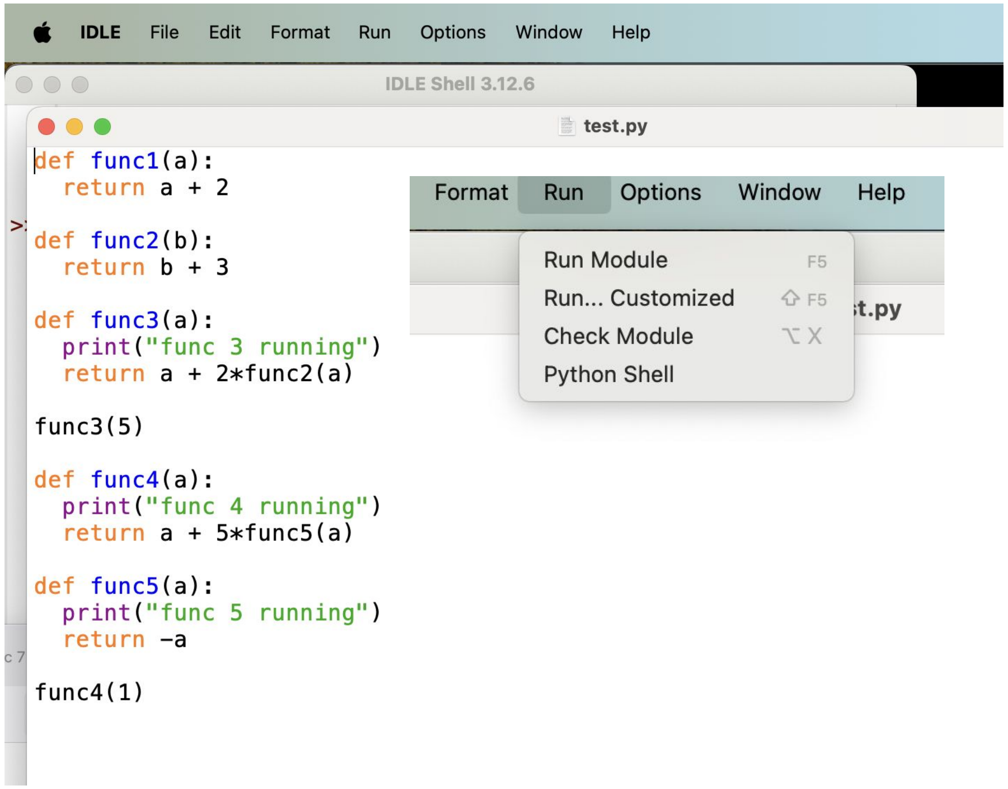
The simplest IDE for Python: IDLE (Integrated Development and Learning Environment)

Comes pre-installed with Python if you get it from www.python.org/downloads

Search for 'IDLE' in start menu or Launchpad



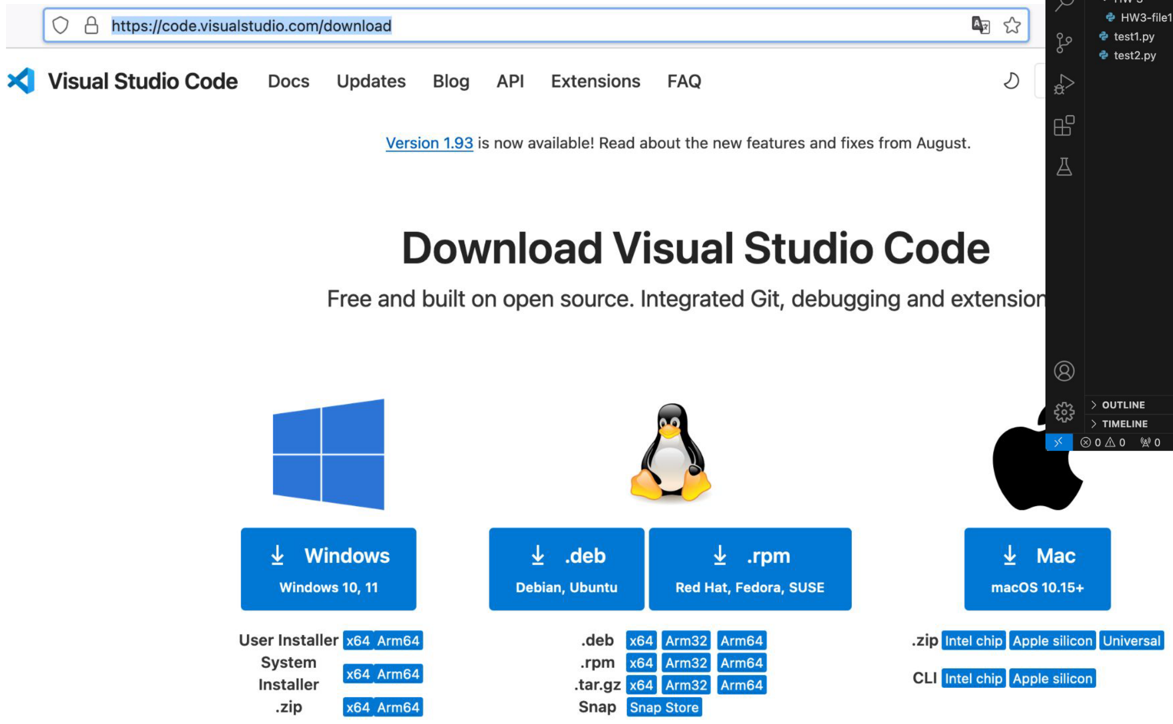
```
Python 3.12.6 (v3.12.6:a4a2d2b0d85, Sep  6 2024, 16:08:03)
[Clang 13.0.0 (clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license()" for mor
e information.
>>> 
```



```
def func1(a):
  return a + 2

def func2(b):
  return b + 3

def func3(a):
  print("func 3 running")
  return a + 2*func2(a)

func3(5)

def func4(a):
  print("func 4 running")
  return a + 5*func5(a)

def func5(a):
  print("func 5 running")
  return -a

func4(1)
```

Visual Studio Code (VS Code)

- Download & Install VS Code
- Install Python Extension for VS Code



https://code.visualstudio.com/download

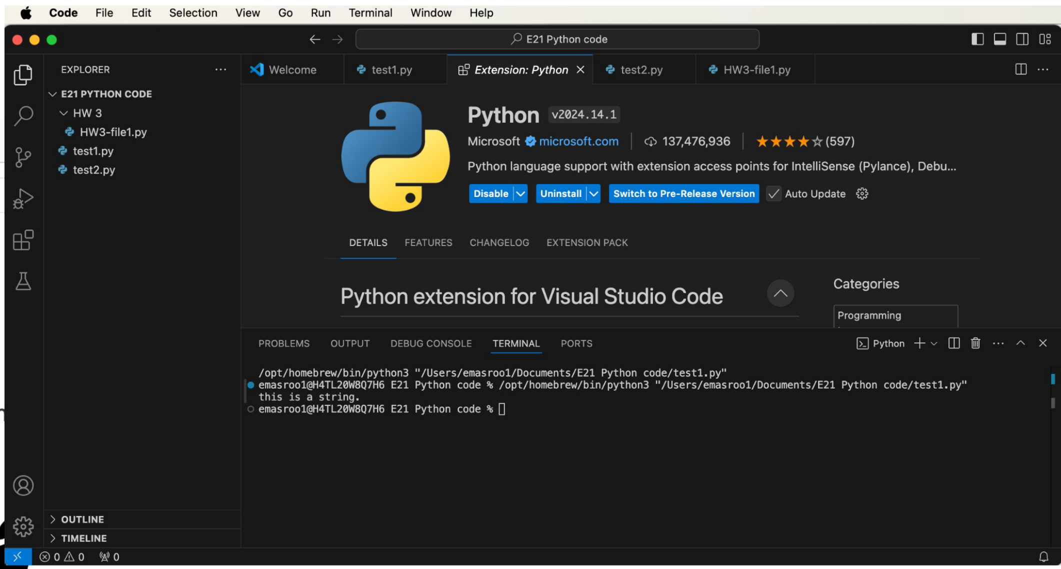
Visual Studio Code · Docs · Updates · Blog · API · Extensions · FAQ

Version 1.93 is now available! Read about the new features and fixes from August.

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extension

Windows — Windows 10, 11	User Installer x64 Arm64; System Installer x64 Arm64; .zip x64 Arm64
.deb — Debian, Ubuntu; .rpm — Red Hat, Fedora, SUSE	.deb x64 Arm32 Arm64; .rpm x64 Arm32 Arm64; .tar.gz x64 Arm32 Arm64; Snap Snap Store
Mac — macOS 10.15+	.zip Intel chip Apple silicon Universal; CLI Intel chip Apple silicon



Python v2024.14.1
Microsoft microsoft.com 137,476,936 ★★★★☆ (597)
Python language support with extension access points for IntelliSense (Pylance), Debu...

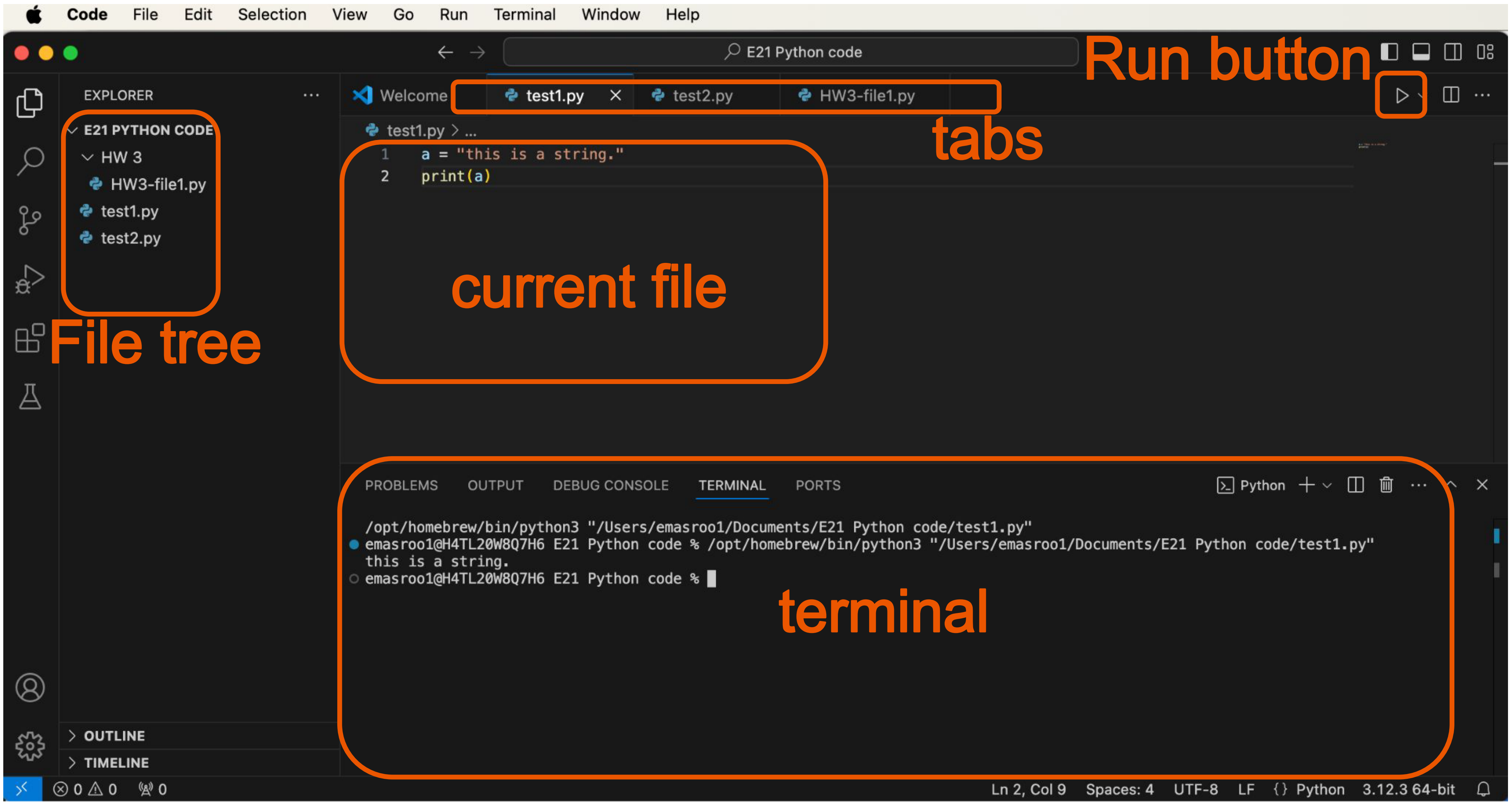
Python extension for Visual Studio Code

```
/opt/homebrew/bin/python3 "/Users/emasroo1/Documents/E21 Python code/test1.py"
emasroo1@H4TL20W8Q7H6 E21 Python code % /opt/homebrew/bin/python3 "/Users/emasroo1/Documents/E21 Python code/test1.py"
this is a string.
emasroo1@H4TL20W8Q7H6 E21 Python code % 
```

https://code.visualstudio.com/download

https://marketplace.visualstudio.com/

Anatomy of VS Code

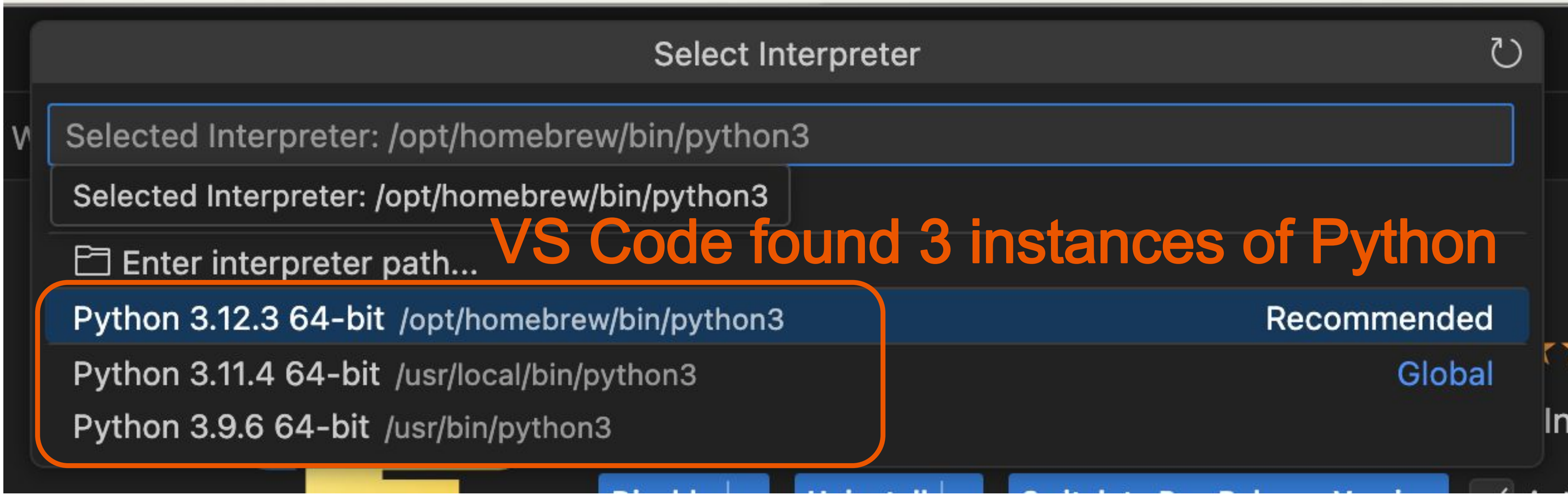


```python
a = "this is a string."
print(a)
```

```
/opt/homebrew/bin/python3 "/Users/emasroo1/Documents/E21 Python code/test1.py"
● emasroo1@H4TL20W8Q7H6 E21 Python code % /opt/homebrew/bin/python3 "/Users/emasroo1/Documents/E21 Python code/test1.py"
this is a string.
○ emasroo1@H4TL20W8Q7H6 E21 Python code %
```

Python Versions inside VS code

- You may have more than one installation of Python on your computer
- How to "tell VS Code which one to use"
    - View → Command Palette → "Python: select Interpreter"



Select Interpreter

Selected Interpreter: /opt/homebrew/bin/python3

Selected Interpreter: /opt/homebrew/bin/python3

Enter interpreter path...

VS Code found 3 instances of Python

```
Python 3.12.3 64-bit  /opt/homebrew/bin/python3    Recommended
Python 3.11.4 64-bit  /usr/local/bin/python3       Global
Python 3.9.6 64-bit   /usr/bin/python3
```