Step-by-Step Environment Set-Up

Install Visual Studio Code

What it is? A code editor.

Why do I need it? You don't *need* it, but it makes coding easier by colour-coding code, auto-completions, and highlighting errors.

Where do I get it? Here

Install Python

What it is? The programming language.

Why do I need it? To tell your computer how to execute programs written in Python syntax.

Where to get it? Here

Install Python Extension

What it is? Additional helpful add-ons for the coding editor.

Why do I need it? To get Python-specific help while you code.

Where to get it? VSCode → ctrl + shift + x → search 'Python Extension Pack' → install

Get a GitHub Account

What it is? An open-source online platform for storing and sharing code.

Why do I need it? To back up your code, collaborate with others, and showcase your projects.

Where to get it? Here

Install Git

What it is? A local version control system.

Why do I need it? To track changes to your code and to upload your changes to GitHub.

Where to get it? Here

Install Git Extension & Sign In

What it is? Code editor add-ons for working with Git.

Why do I need it? To use Git through VSCode interface rather than the terminal.

Where to get it? VSCode → ctrl + shift + x → search 'GitHub Pull Requests' → install; sign into your GitHub account in VSCode → Accounts (bottom left corner)

Configure Git in VSCode

What it is? Telling your VSCode to communicate with *your* GitHub.

Why do I need it? To identify whose changes are being saved on GitHub.

How do I do this? VSCode → ctrl + ‘ → run `git config --global user.name "[your username]"` → run `git config --global user.email "[your email]"`

Now you're ready to start!

Starting a Project

🚀 Create a Project

1. In VSCode, go to Explorer (paper icon or ctrl + shift + E)
2. Hit ‘Open Folder’ and navigate to your project’s folder (or create a new one)
3. Select the desired folder and hit ‘Open’
4. In the Explorer, navigate to the folder title on the left-hand side and hit ‘New File’
5. Name your file and add `.py` at the end to let VSCode know this is a Python file

🚀 Upload Your Project Online

1. In VSCode, go to Source Control (branch icon or ctrl + shift + G)
2. Hit ‘Initialize Repository’
3. In the text field on the left-hand side, type a quick message to describe your file and hit ‘Commit’
4. Once committed, hit ‘Publish Branch’
5. In the dropdown menu, select ‘Publish to GitHub public repository’
6. Double-check on GitHub that there is a repository with your project’s name containing your file

🚀 Propose Changes to Others’ Projects

1. On GitHub, go to a repository you want to modify, hit the green ‘Code’ button, and copy the URL
2. In VSCode, go to Source Control (ctrl + shift + G) and hit ‘Clone Repository’
3. Paste the copied URL into the revealed text field and hit enter
4. Navigate to where you want to save the repository and hit ‘Select as Repository Destination’
5. Source Control → ‘More Actions’ (3 dots next to ‘CHANGES’) → ‘Branch’ → ‘Create Branch’
6. Name your branch and hit enter
7. Make changes to the code
8. In the VSCode GitHub tab, hit ‘Create Pull Request’ (next to ‘PULL REQUESTS’) and write a message

🚀 Review Others’ Change Proposals

❶ In your VSCode, hit the ‘sync changes’ icon (bottom left bar)

↓

❷ In the VS Code GitHub Pull Requests tab, find your unresolved pull requests

↓

❸ Select a pull request and hit ‘Open Changes’ (icon next to the request title)

↓

❹ Review changes and add comments by hitting ‘+’ on the changed lines

↓

❺ Hit the request’s title to open a tab with details of the request

↓

❻ Select ‘Approve’ for incorporating the change, ‘Comment’ for resolving the change by commenting on it, or ‘Request Changes’ to request changes to the proposed change

Primitive Data Types

Name	Stores	Examples
`int`	integer value	`3, 7, 42`
`float`	float value	`3.14, 2.0`
`str`	text	`‘Sherlock’, "Holmes"`
`bool`	true or false values	`True, False`

You can print the data type of `x` by running `print(type(x))`. Learn more about other data types here 🔗.

Operators

Assignment Operators

Operator	Function	Example	Equivalent
`=`	assigns a value to name	`x = 2`	
`+=`	increases assigned value by new value	`x += 3`	`x = x + 3`
`-=`	decreases assigned value by new value	`x -= 3`	`x = x - 3`
`**=`	raises assigned value to the power of new value	`x **= 3`	`x = x**3`

Arithmetic Operators

Operator	Returns	Example
`+`	addition of two values	`x + y`
`-`	subtraction of two values	`x - y - z`
`*`	multiplication of two values	`x * y`
`/`	division of two values	`x / y`
`**`	exponentiation (power) of two values	`x ** y`
`%`	remainder after division (modulo)	`x % y`

Logical Operators

Operator	Returns	Example
`and`	True if both conditions are true	`x > 5 and y < 10`
`or`	True if at least one condition is true	`x > 5 or y < 10`
`not`	True if condition is false (negation)	`not x > 5`

Comparison Operators

Operator	Returns	Example
`==`	True if values are equal	`x == 5`
`!=`	True if values are not equal	`x != 5`
`>`	True if left value is greater	`x > 5`
`<`	True if left value is smaller	`x < 5`
`>=`	True if left value is greater or equal	`x >= 5`
`<=`	True if left value is smaller or equal	`x >= 5`

Learn more about other operators here 🔗.

Couple General Tips

1. **Many issues can be resolved by reading notifications and manuals closely.** When the computer is trying to communicate, take a moment to read what it's saying—it's usually trying to help.
2. **Most programmers Google (and LLM!) stuff all the time.** Whatever you're struggling with, someone has probably struggled with before; go out there and see what's helpful.
3. Thanks to the Jupyter extension you installed as a part of the Python Extension Pack, **typing `#%%` will split your code into 'cells' that you can run independently** (no need to make a new file for each task).

In-Class Problems

01 Create a new repository with at least one file through VSCode and upload it to GitHub.

02 Propose a change to someone else's repository.

03 Define variables `x`, `y`, and `z` so that they are of different types each. Print their types.

04 Write a program that prints the product of two sums.

05 Assign value 3 to `x` and value 10 to `y`. Print 28 using only these two variables and arithmetic operators.

06 Create separate variables for various types of personal data (name, DOB, email etc.). Then print this data.

07 Create variables with hourly wage and hours worked. Print gross pay and pay after 23% tax.

08 Assume a savings account with an initial $1000. The owner adds $100 every month at 2.1% interest rate (calculated monthly). Print the amount saved after 2 years.

09 Assign `3749` to a variable named `seconds`, then print the number of hours and minutes without the multiplication or division operator.

10 01/01/2025 fell on Wednesday. Use operators to compute what day of the week the 175th day of 2025 fell on.

11 The boolean variable `quartered` stores whether or not the number 4792 is divisible by 4. Print the value of `quartered`.

12 Recipe calls for 4 cups of flour and serves 5 people. Scale it to serve 7 people. Compare the results of when you store the number of cups as an integer versus versus float in your calculation.

13 Create a variable storing a (realistic) age. Create a variable that stores True if the age is above 18, False otherwise. Print the variable.

14 Create two variables, one for the number of cookies and one for the number of people attending. Create a boolean checking whether each person gets at least one cookie, then print the result.

15 Weather is considered 'unpleasant' if the temperature falls below 17 or rises above 27. Store the temperature as `x` and create a boolean variable called `unpleasant` that evaluates whether the weather is unpleasant. Test with temperatures 15, 22, and 30.