Data science

Chapter 1.2 – Setting up your working environment

2025-2026

What will we be working with?

- Programming language: Python
- IDE: Visual Studio Code
  - Integrated Development Environment

- Use a virtual environment in Python
- Use Jupyter Notebooks

- Note: This is a hands-on presentation. You are supposed to follow along.

Python

- There are a few alternatives to Python for data science:
    - Julia: Python, but you can compile it, so it’s faster. Good when Python becomes too slow, but not ideal to start with.
    - R: Used mainly by statisticians, who were the first data scientists. Feels strange at first but is a very powerful language to work with data (importing/cleaning/creating graphs). Preferred language to use in Business Intelligence tools.
- We chose Python because it’s widely used

Windows, Linux or MacOS?

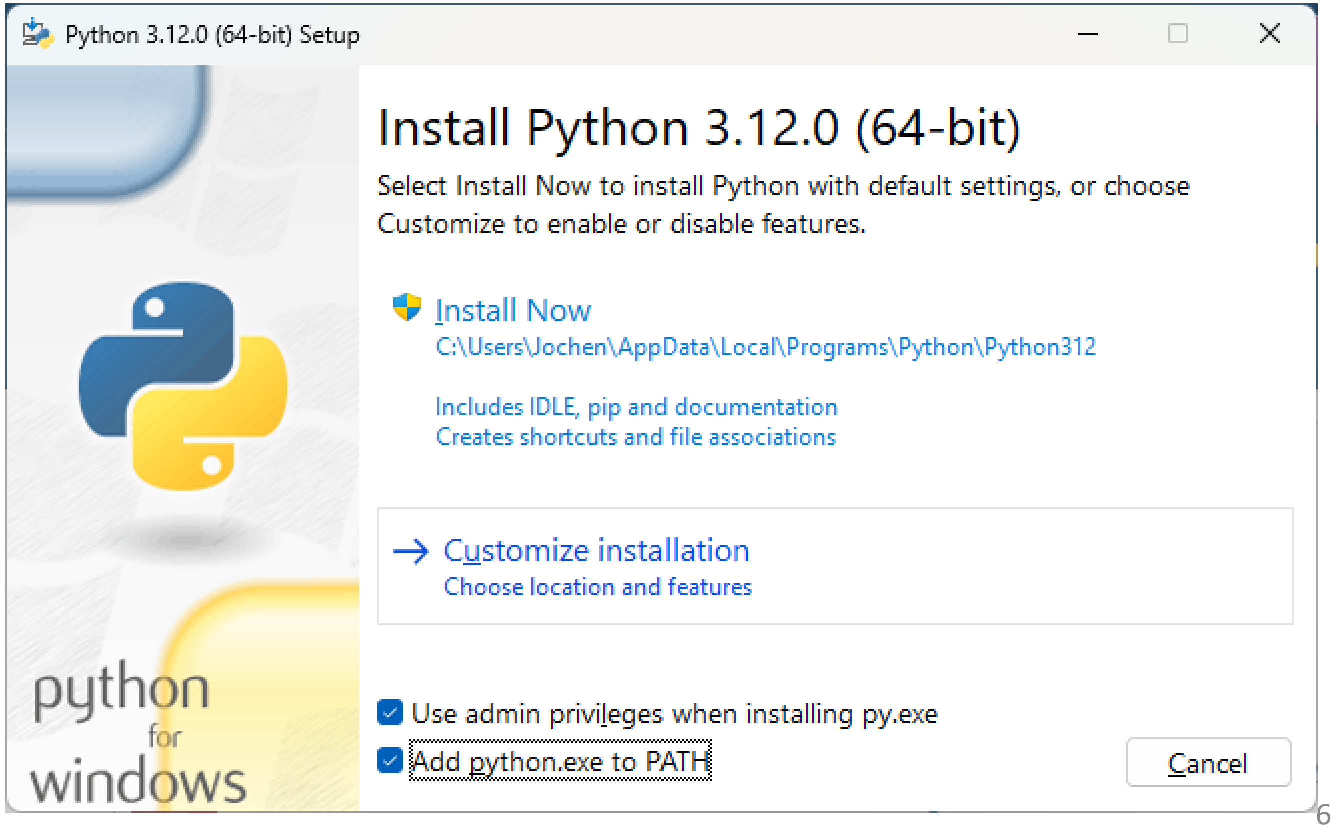
- Our examples and screenshots will always be made on Windows
  - And we know our code works on Windows
  - Because the teachers exclusively use Windows (for this course)
- But the tools we use are all cross-platform
  - You can use MacOS or Linux
- So, we decided:
  - You're free to use any OS you like
  - If you have any issues with your code, we will help you figure out what's going on
  - But we reserve the right to say "I can't find the problem so it's probably because you don't use Windows. You figure it out."

Installing Python

- Go to https://www.python.org/downloads/
- Download the latest version
- DO NOT quickly click 'next' through the installation!
  - Check the following slides

Installing Python

- First screen:
  - “Add Python to path”
  - Click “Customize installation



Python 3.12.0 (64-bit) Setup

Install Python 3.12.0 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\Jochen\AppData\Local\Programs\Python\Python312

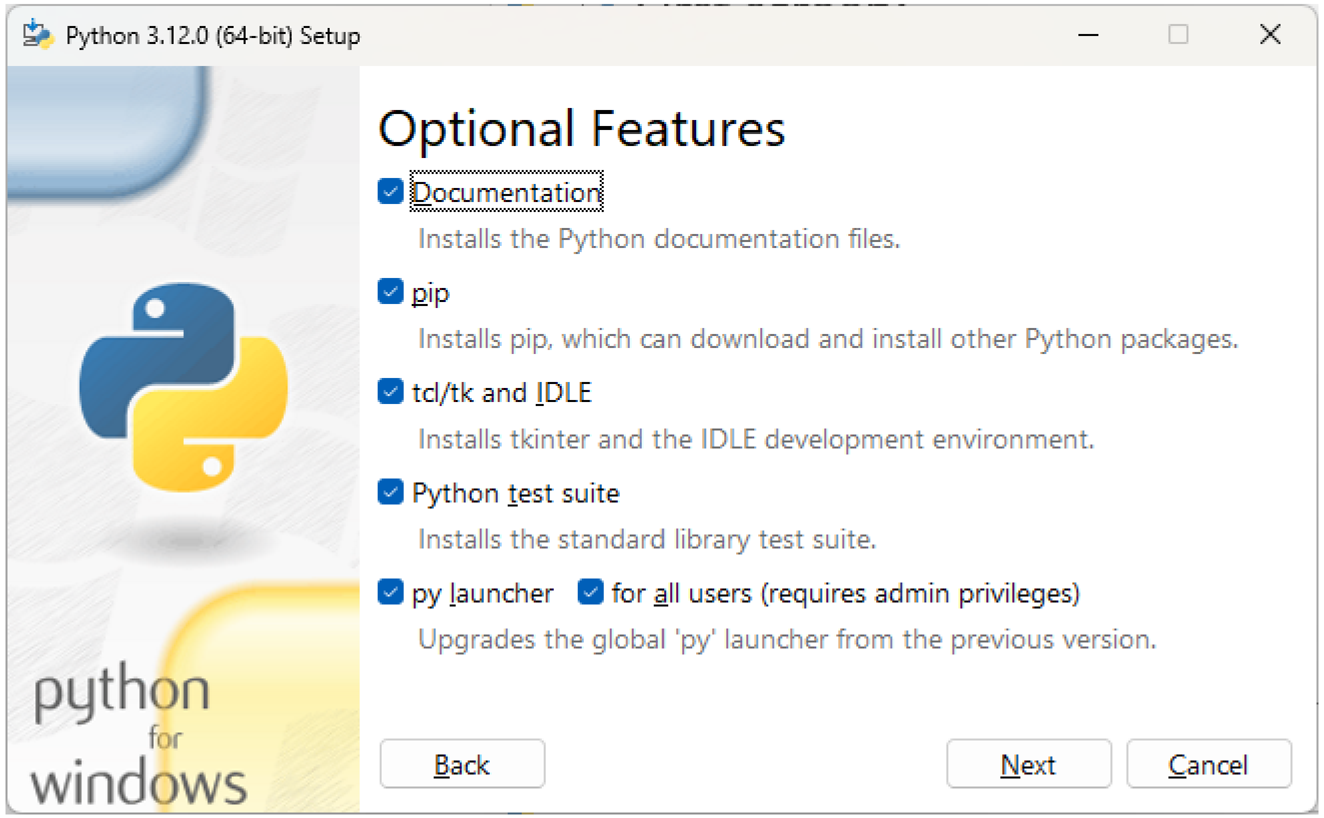
Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Use admin privileges when installing py.exe
☑ Add python.exe to PATH

Installing Python

- Leave the following screen as it is


Python 3.12.0 (64-bit) Setup

Optional Features

☑ Documentation

Installs the Python documentation files.

☑ pip

Installs pip, which can download and install other Python packages.

☑ tcl/tk and IDLE

Installs tkinter and the IDLE development environment.

☑ Python test suite

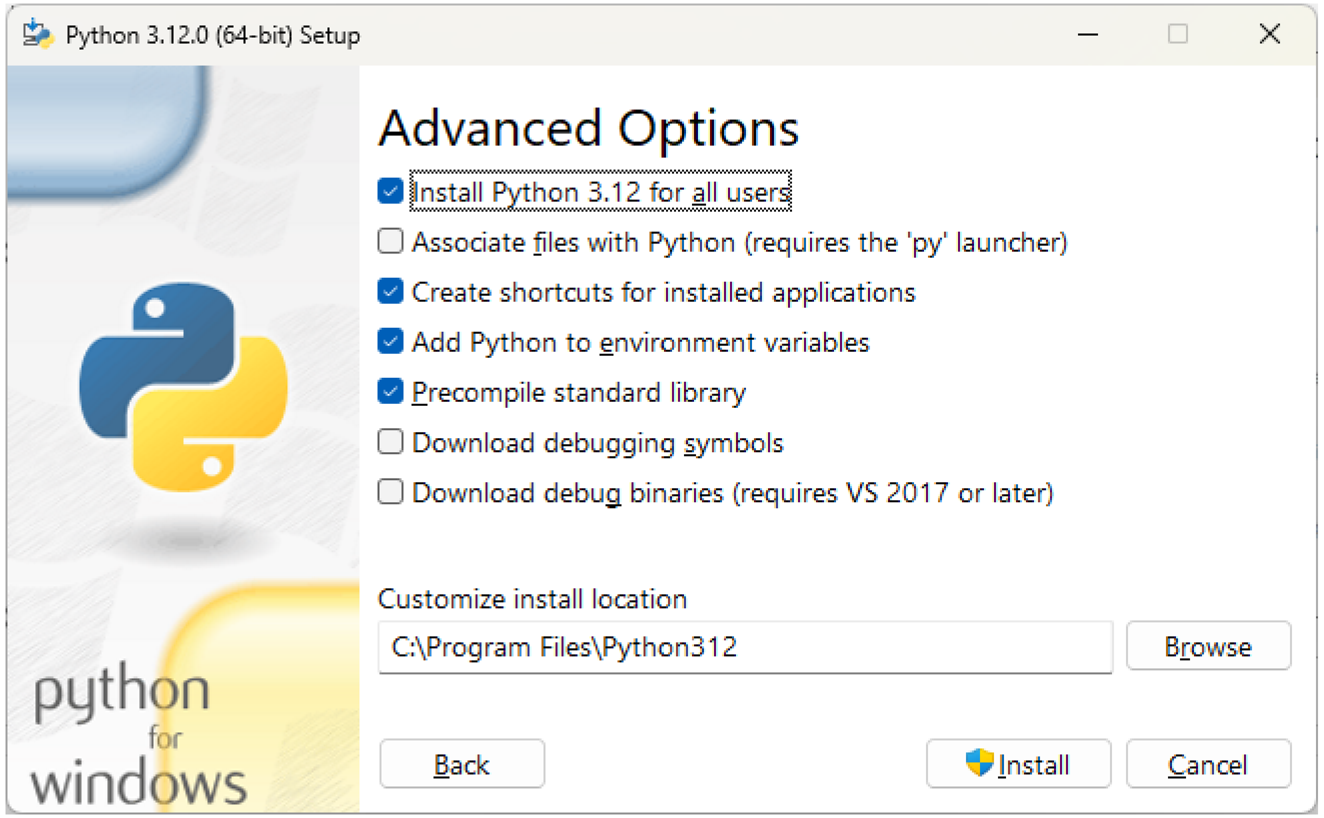
Installs the standard library test suite.

☑ py launcher ☑ for all users (requires admin privileges)

Upgrades the global 'py' launcher from the previous version.

Installing Python

- Select "Install Python for all users"



Python 3.12.0 (64-bit) Setup

Advanced Options

- [✓] Install Python 3.12 for all users
- [ ] Associate files with Python (requires the 'py' launcher)
- [✓] Create shortcuts for installed applications
- [✓] Add Python to environment variables
- [✓] Precompile standard library
- [ ] Download debugging symbols
- [ ] Download debug binaries (requires VS 2017 or later)

Customize install location

C:\Program Files\Python312

Visual Studio Code

- Microsoft has the Visual Studio IDE since the nineties
  - Big, bulky program that only runs on Windows
- But later on they released Visual Studio Code
  - Small and efficient
  - Open source
  - Works on any operating system
  - Can be used for any programming language (through plugins)
  - Integrates with GitHub
  - Can be used on a remote system
- In this course we work with **Visual Studio Code**

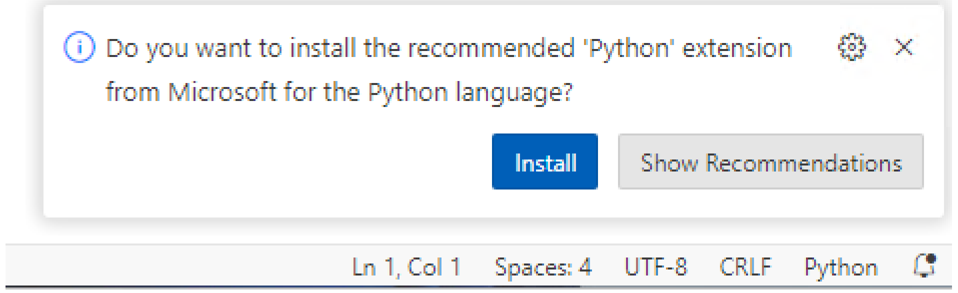
Installing Visual Studio Code

- Go to https://code.visualstudio.com/
- Download the latest version for your operating system
- Install using the default settings

- Alternatives
  - Install using https://ninite.com/
  - Install as a portable app
  - Theoretically: use https://vscode.dev/

First time usage

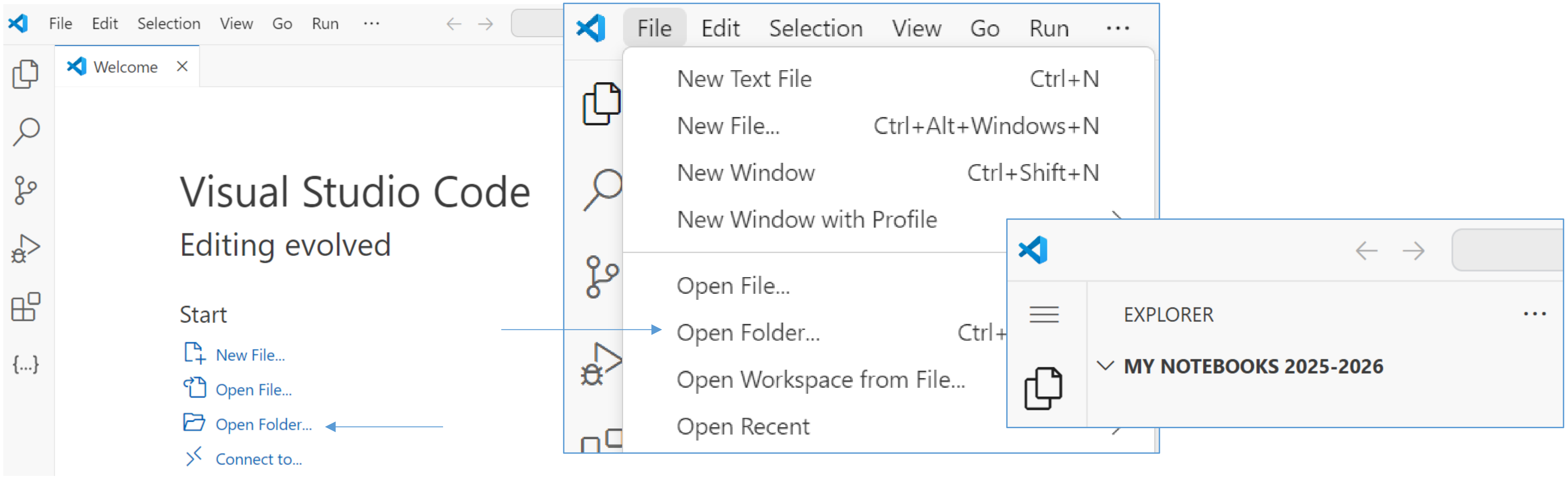
- When using VSCode for the first time, it will ask a lot
  - Default color scheme
  - Sync your settings
  - …
- All of these can be changed later, so simply choose a nice color scheme and close the window
- When starting a new type of file, there will be boxes on the lower right corner
  - Click “install”, it’ll make your life easier



Do you want to install the recommended 'Python' extension from Microsoft for the Python language?

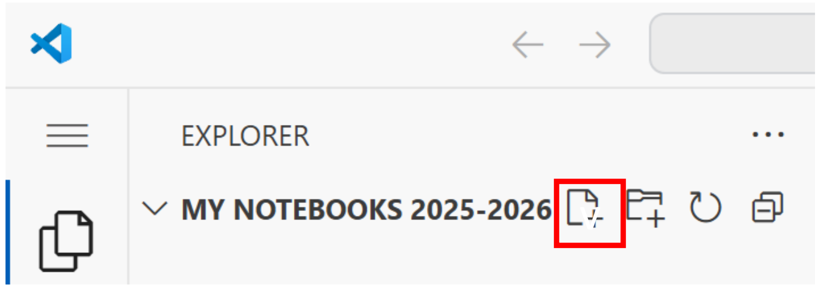
Creating your first Python script

- Create a folder for your Data Science projects
- By opening this folder in VS Code, it becomes your "workspace".

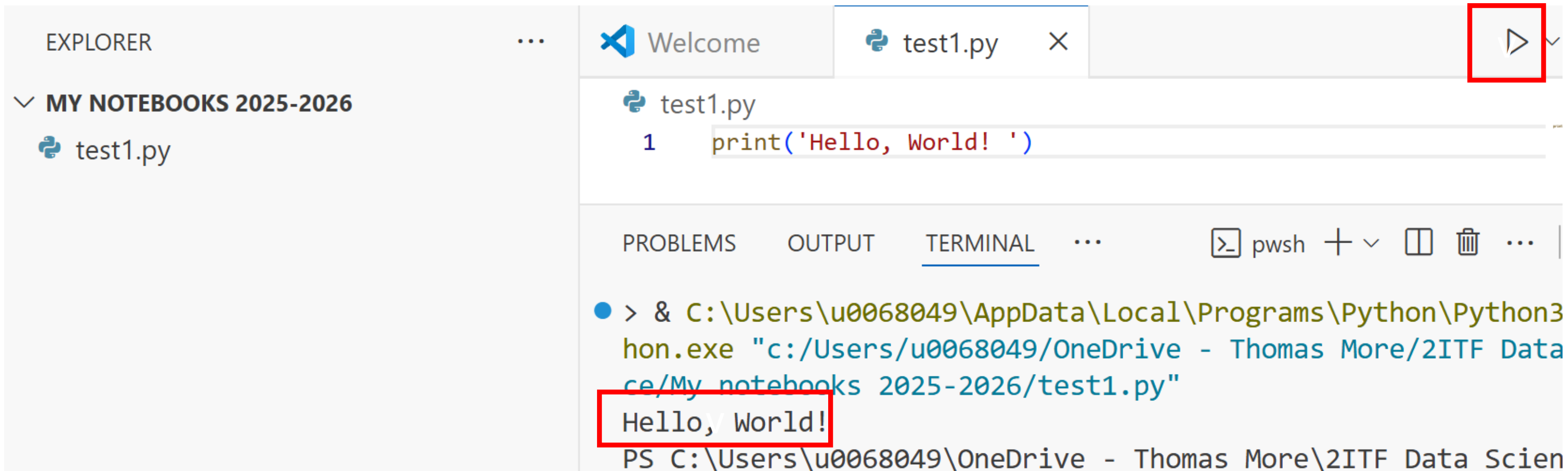


Creating your first Python script

- Click the "New file"-icon



- Call the file "test1.py"
- Add Python code that prints "Hello, world!"
- Run the code using the "play"-button in the upper right corner

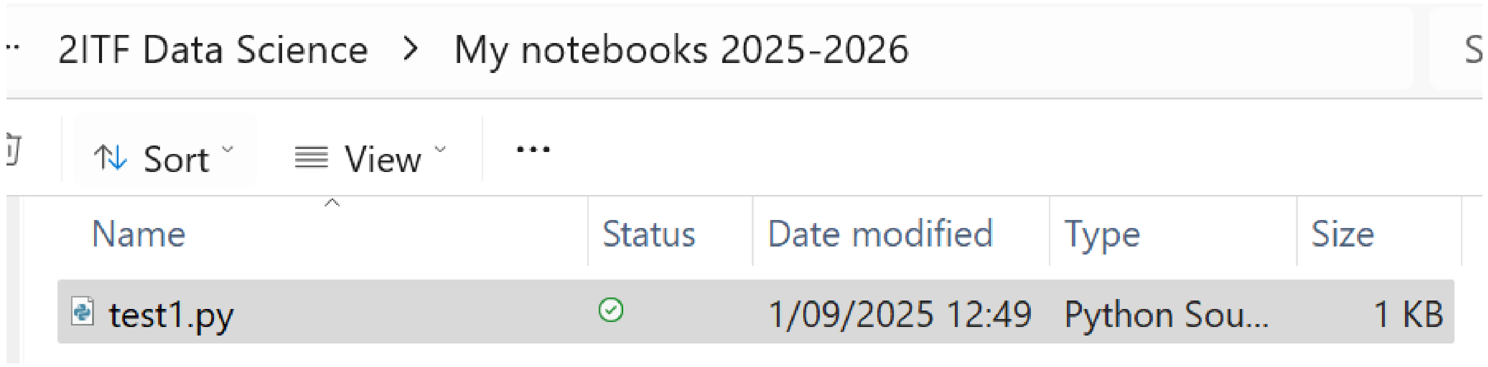


```python
print('Hello, World! ')
```

```
> & C:\Users\u0068049\AppData\Local\Programs\Python\Python3
hon.exe "c:/Users/u0068049/OneDrive - Thomas More/2ITF Data
ce/My notebooks 2025-2026/test1.py"
Hello, World!
PS C:\Users\u0068049\OneDrive - Thomas More\2ITF Data Scien
```

Creating your first Python script

- The code runs in a terminal below
- The file is also saved in your folder

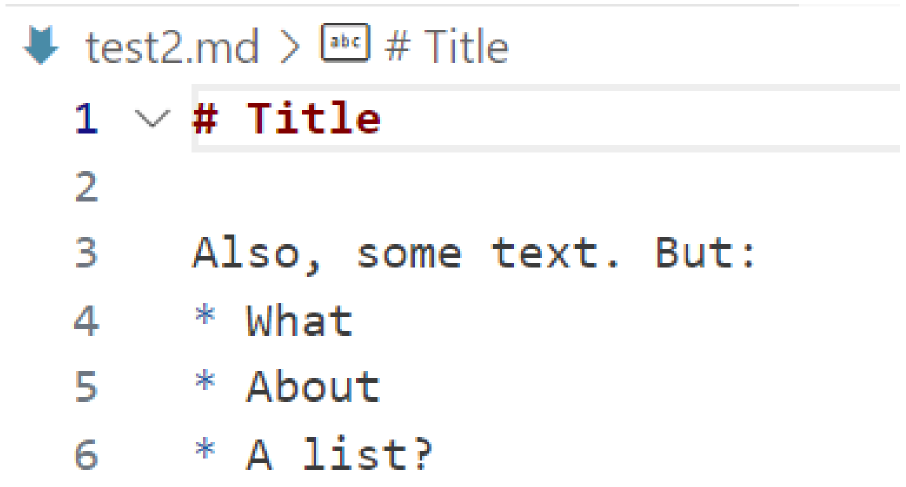


Name	Status	Date modified	Type	Size
test1.py	✓	1/09/2025 12:49	Python Sou...	1 KB

- When creating Python-scripts, this is the way to work

Creating your first Markdown file

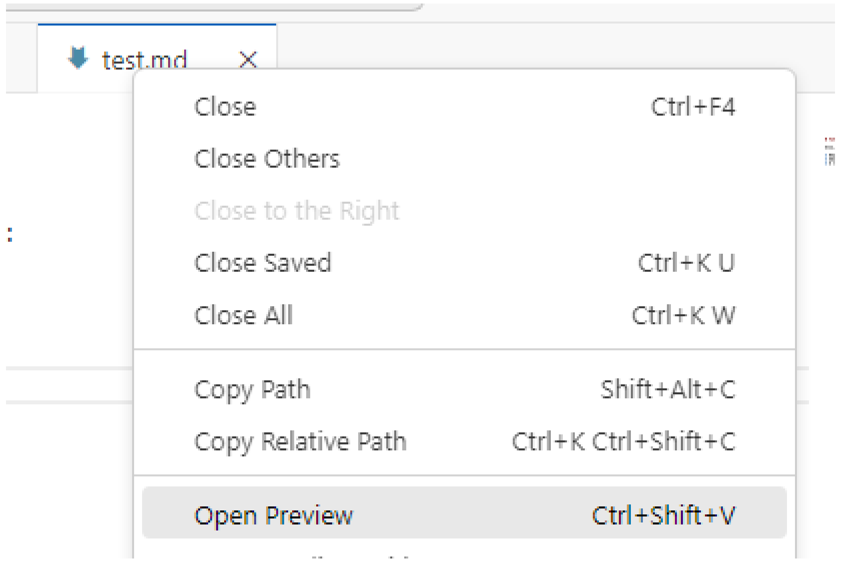
- Markdown is a way to style text using only simple markers, to help you write formatted text without using Word or HTML
- In your workspace folder, create a file and call it “test2.md”
- Add the following to the file



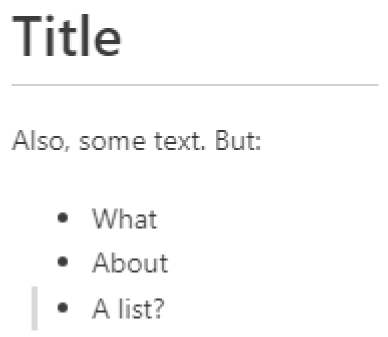
```
# Title

Also, some text. But:
* What
* About
* A list?
```

Using markdown files

- Save the file and right-click the tab, choosing “Open preview”



- The output will be the same file, but formatted
  - There are tags for adding code, images, links, …



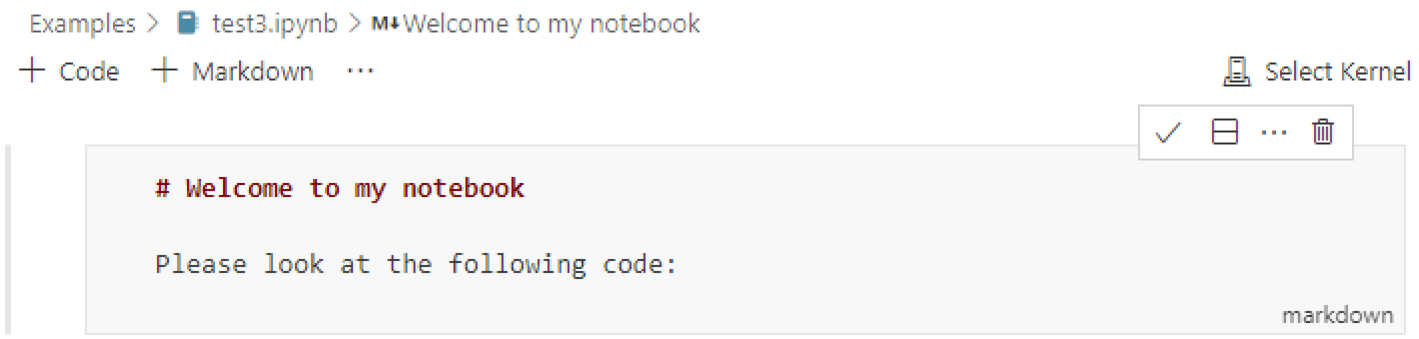
Title

Also, some text. But:

- What
- About
- A list?

Creating your first Jupyter notebook

- We did code and we did markdown, but can we combine them?
- Yes! It’s called a Jupyter notebook
- Create a new file called “test3.ipynb”
- Add a markdown-cell and add some markdown-text

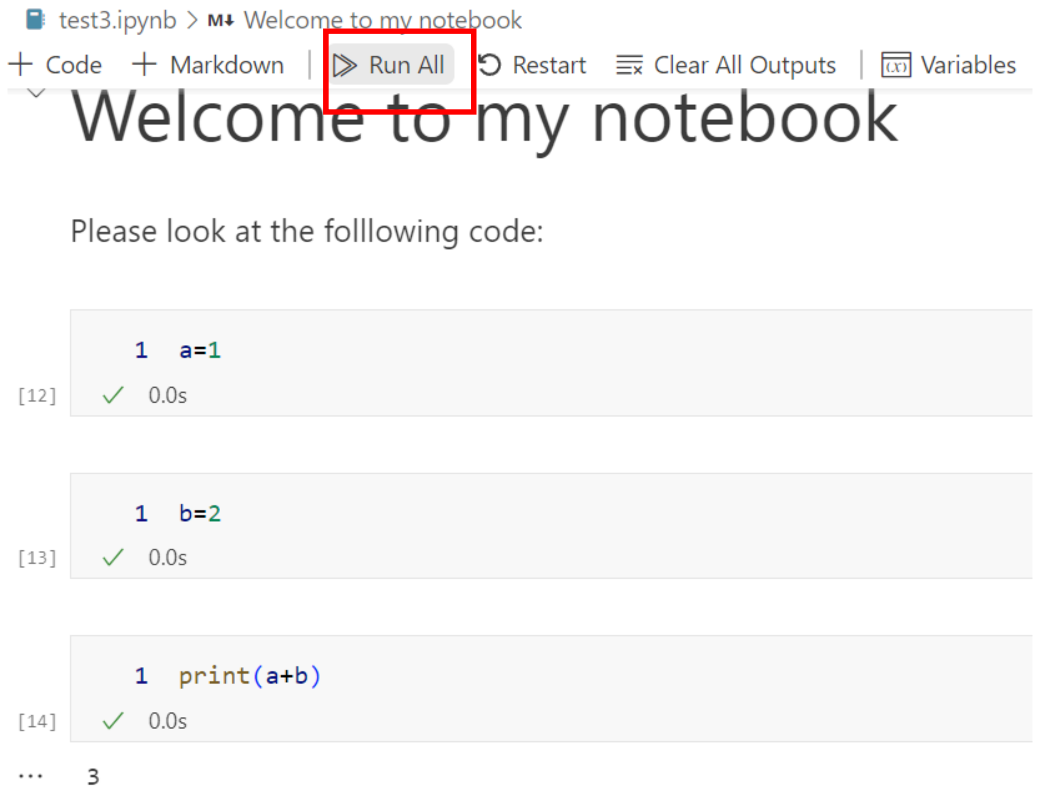


```
# Welcome to my notebook

Please look at the following code:
```

- Click the checkmark on the top right to display formatted text
- Double-click the formatted text to edit it again

Creating your first Jupyter notebook

- Add 3 code-cells:
  - In the first cell put “a = 1”
  - In the second cell put “b=2”
  - In the third cell put “print(a+b)”
- Run the cells
- Note how the variables you made in cell 1 and cell 2 still exist in cell 3
- This is the big strength of Jupyter notebooks: you can write code and define variables in one cell and reuse these variables differently in subsequent code cells



Welcome to my notebook

Please look at the folllowing code:

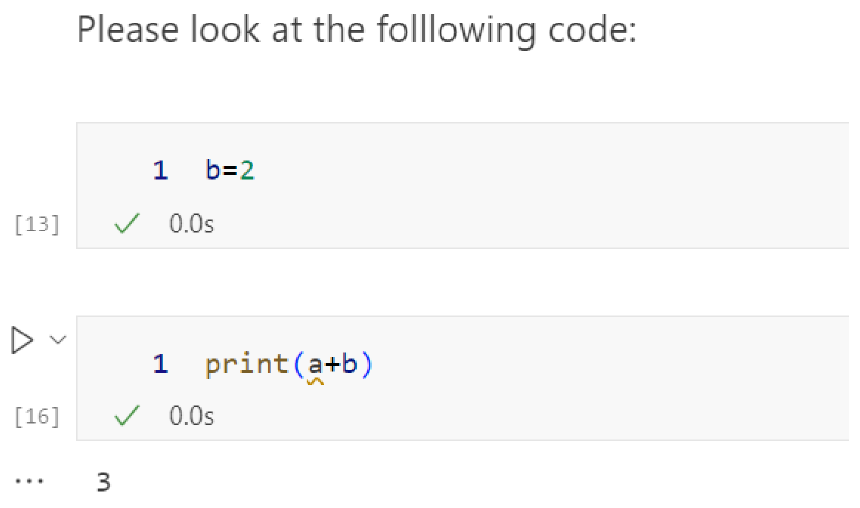
```
[12]
a=1
0.0s
```

```
[13]
b=2
0.0s
```

```
[14]
print(a+b)
0.0s
3
```

Using Jupyter notebooks

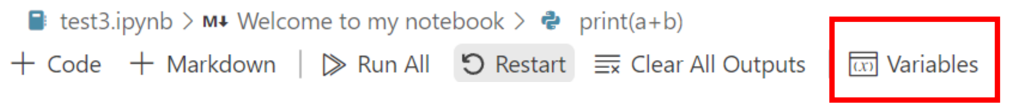
- Delete the first cell
- Now run the third cell again!
- The print still works!

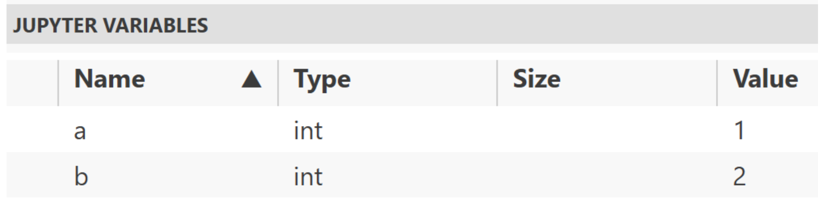


Please look at the folllowing code:

```
[13]  b=2
      ✓ 0.0s

[16]  print(a+b)
      ✓ 0.0s
...   3
```

- This is a pitfall: it's not because you deleted a cell that the result of that cell is gone. The Jupyter variables still exist and can still mess up your results.



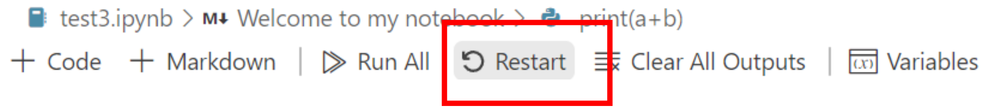


JUPYTER VARIABLES

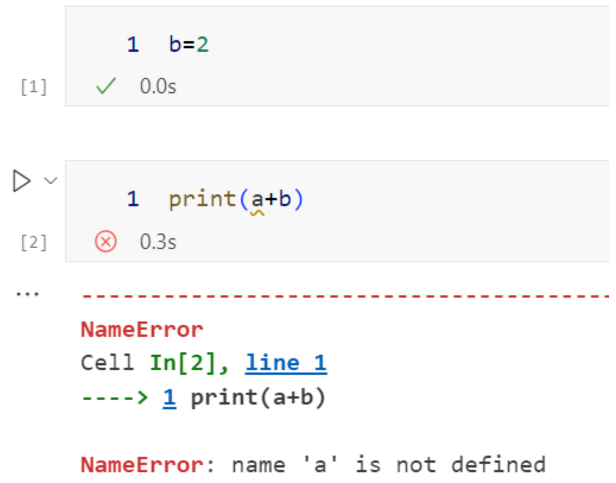
Name ▲	Type	Size	Value
a	int		1
b	int		2

Using Jupyter notebooks

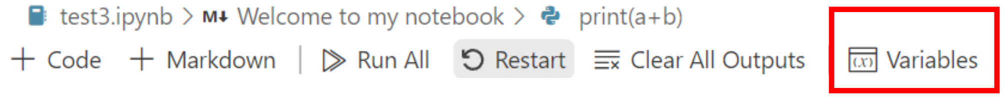
- If you want the variable ‘a’ to be cleared from memory, you must restart the kernel

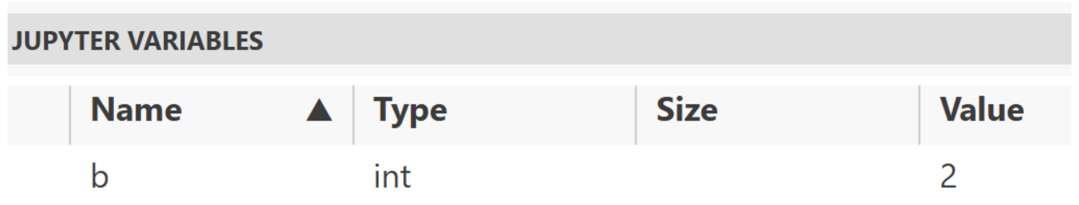

test3.ipynb > M↓ Welcome to my notebook > print(a+b)
+ Code + Markdown | Run All Restart Clear All Outputs | Variables

Please look at the folllowing code:



```
[1]  b=2
     ✓ 0.0s

[2]  print(a+b)
     ⊗ 0.3s

...  ---------------------------------------
     NameError
     Cell In[2], line 1
     ----> 1 print(a+b)

     NameError: name 'a' is not defined
```


test3.ipynb > M↓ Welcome to my notebook > print(a+b)
+ Code + Markdown | Run All Restart Clear All Outputs | Variables

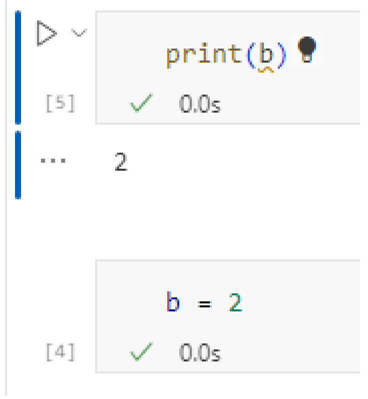


JUPYTER VARIABLES

Name ▲	Type	Size	Value
b	int		2

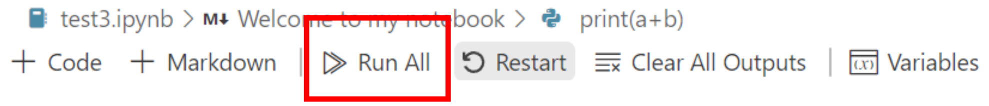
Using Jupyter notebooks

- In the first cell put "print(b)"
- In the second cell put "b = 2"
- Run the second cell first and the first cell last
- Works!



```
print(b)
[5]  0.0s
...  2

b = 2
[4]  0.0s
```

- Another pitfall: when developing you code in different cells. That is fine but remember to always end up with a notebook that you can run top to bottom.



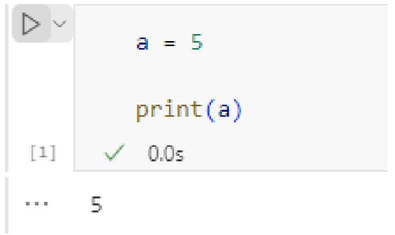
- It's impossible, or at least very frustrating, to recreate the order in which you ran cells earlier. Keep it clean and tidy.

Using Jupyter notebooks

- When restarting all variables and functions are lost
- You start over
- If you are a particularly messy developer, restart your kernel by the end of the session
    - If you can still run all cells top to bottom, it’s fine
    - If you can’t anymore, something is wrong

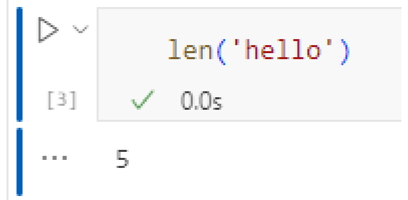
Using Jupyter notebooks

- To print the contents of a variable, you can use print()

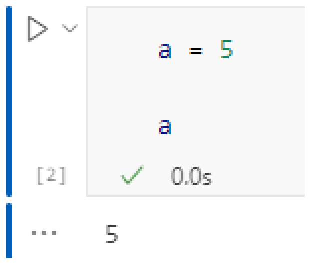


```
a = 5

print(a)
[1]  ✓ 0.0s
...  5
```

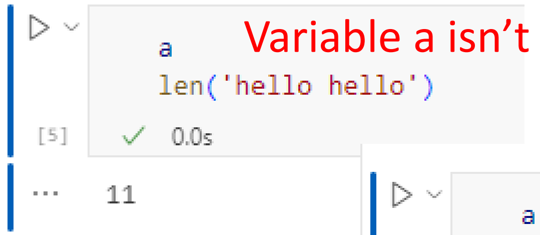
- But notebooks also print the output of whatever the last statement in the code-block is
  - A variable
  - Output of a function



```
len('hello')
[3]  ✓ 0.0s
...  5
```



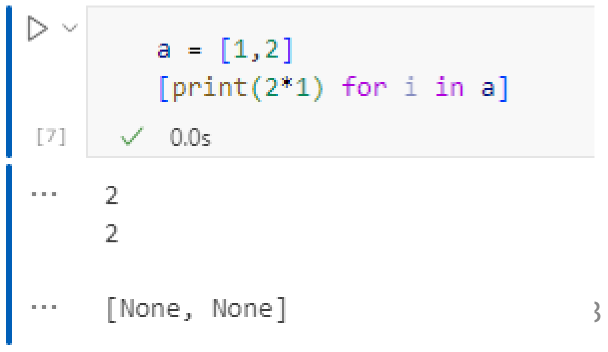
```
a = 5

a
[2]  ✓ 0.0s
...  5
```



```
a
len('hello hello')
[5]  ✓ 0.0s
...  11
```

Variable a isn't printed!

- Note however
  - This only works with the last function
  - If the last function has no return-value, None is printed

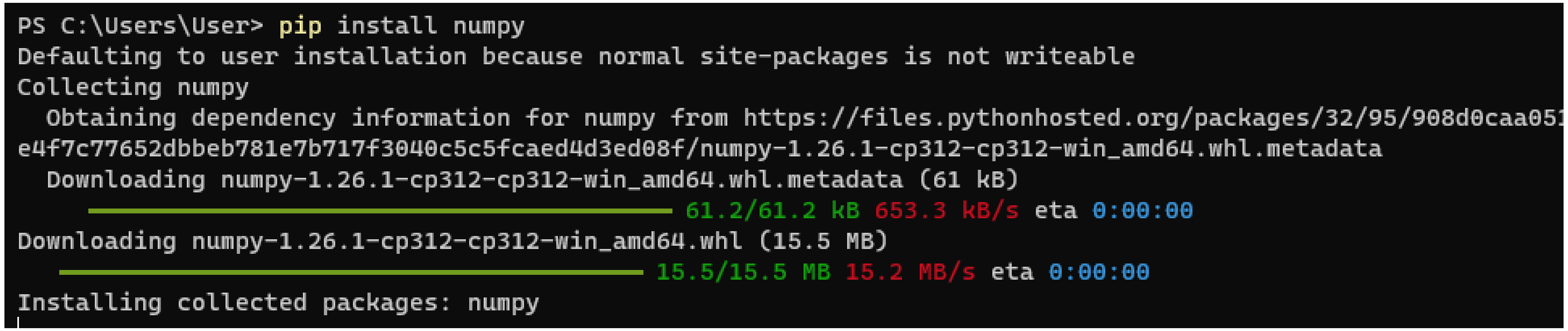


```
a = [1,2]
[print(2*1) for i in a]
[7]  ✓ 0.0s
...  2
     2

...  [None, None]
```

This last row is the output of the list comprehension ->

Libraries

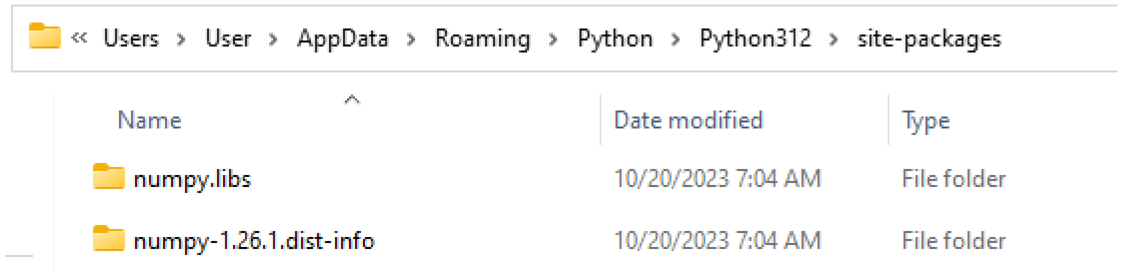
- When doing data science, you use a lot of libraries
  - Pandas, matplotlib, numpy, seaborn, …
- These libraries are open-source packages that we can use on our own system
- You can install them using pip: (Do not do this on your own system!)



```
PS C:\Users\User> pip install numpy
Defaulting to user installation because normal site-packages is not writeable
Collecting numpy
  Obtaining dependency information for numpy from https://files.pythonhosted.org/packages/32/95/908d0caa05
e4f7c77652dbbeb781e7b717f3040c5c5fcaed4d3ed08f/numpy-1.26.1-cp312-cp312-win_amd64.whl.metadata
  Downloading numpy-1.26.1-cp312-cp312-win_amd64.whl.metadata (61 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.2/61.2 kB 653.3 kB/s eta 0:00:00
Downloading numpy-1.26.1-cp312-cp312-win_amd64.whl (15.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.5/15.5 MB 15.2 MB/s eta 0:00:00
Installing collected packages: numpy
```

Libraries

- But the problem?



« Users › User › AppData › Roaming › Python › Python312 › site-packages

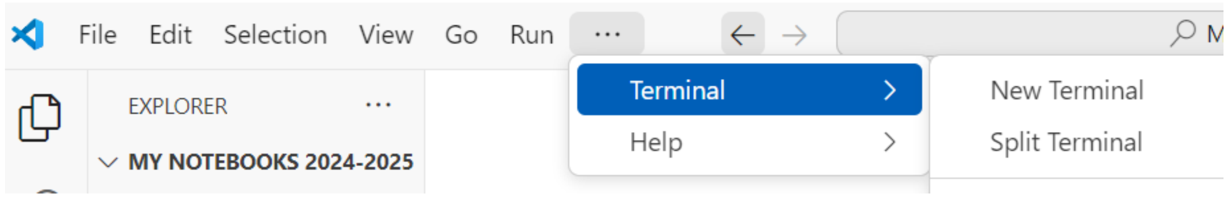
Name	Date modified	Type
numpy.libs	10/20/2023 7:04 AM	File folder
numpy-1.26.1.dist-info	10/20/2023 7:04 AM	File folder

- We just installed numpy on our system. This means that numpy will be available on every python-session we use, and that is not a best practice.
- A rule of thumb: never install packages on your system, install them in a virtual environment
- And another rule of thumb: create a new virtual environment for every new project you start

Setting up a virtual environment

This you can do on your device!

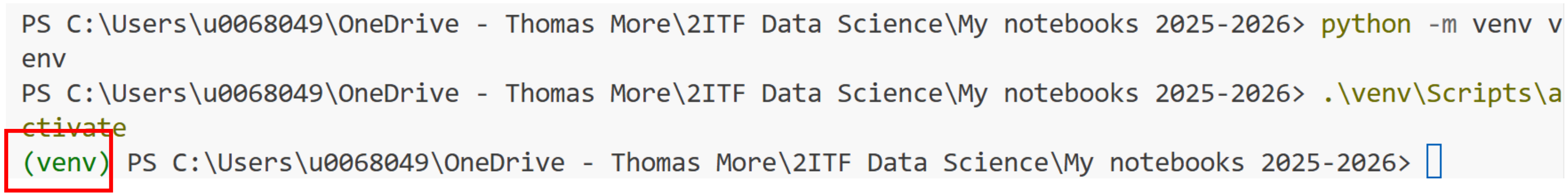
- Open VSCode and select “New terminal” on top



- In the terminal, type 2 commands:

```
python -m venv venv
.\venv\Scripts\activate
```

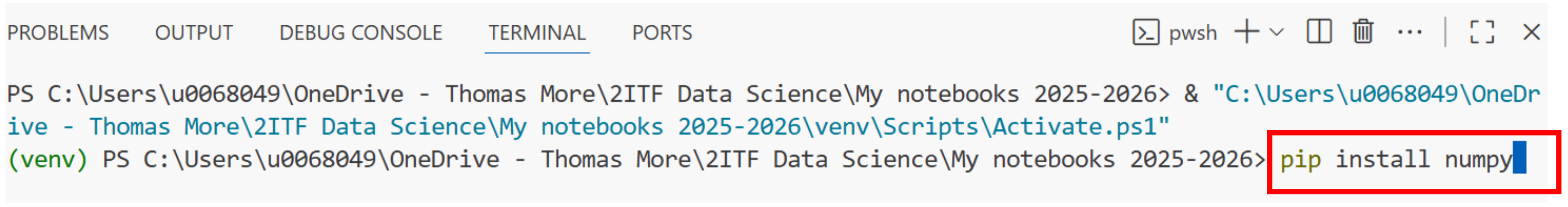
  - Note: these are two separate commands, to be run on two different lines
- The terminal changes to **(venv)**



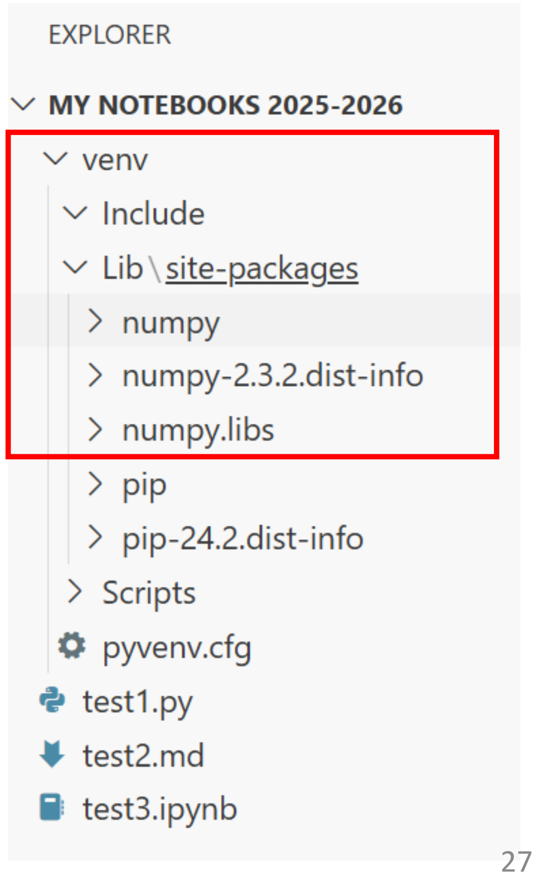
```
PS C:\Users\u0068049\OneDrive - Thomas More\2ITF Data Science\My notebooks 2025-2026> python -m venv v
env
PS C:\Users\u0068049\OneDrive - Thomas More\2ITF Data Science\My notebooks 2025-2026> .\venv\Scripts\a
ctivate
(venv) PS C:\Users\u0068049\OneDrive - Thomas More\2ITF Data Science\My notebooks 2025-2026>
```

Virtual environments

- When you now install a library with pip, this library ends up in the venv-subfolder:



```
PS C:\Users\u0068049\OneDrive - Thomas More\2ITF Data Science\My notebooks 2025-2026> & "C:\Users\u0068049\OneDr
ive - Thomas More\2ITF Data Science\My notebooks 2025-2026\venv\Scripts\Activate.ps1"
(venv) PS C:\Users\u0068049\OneDrive - Thomas More\2ITF Data Science\My notebooks 2025-2026> pip install numpy
```

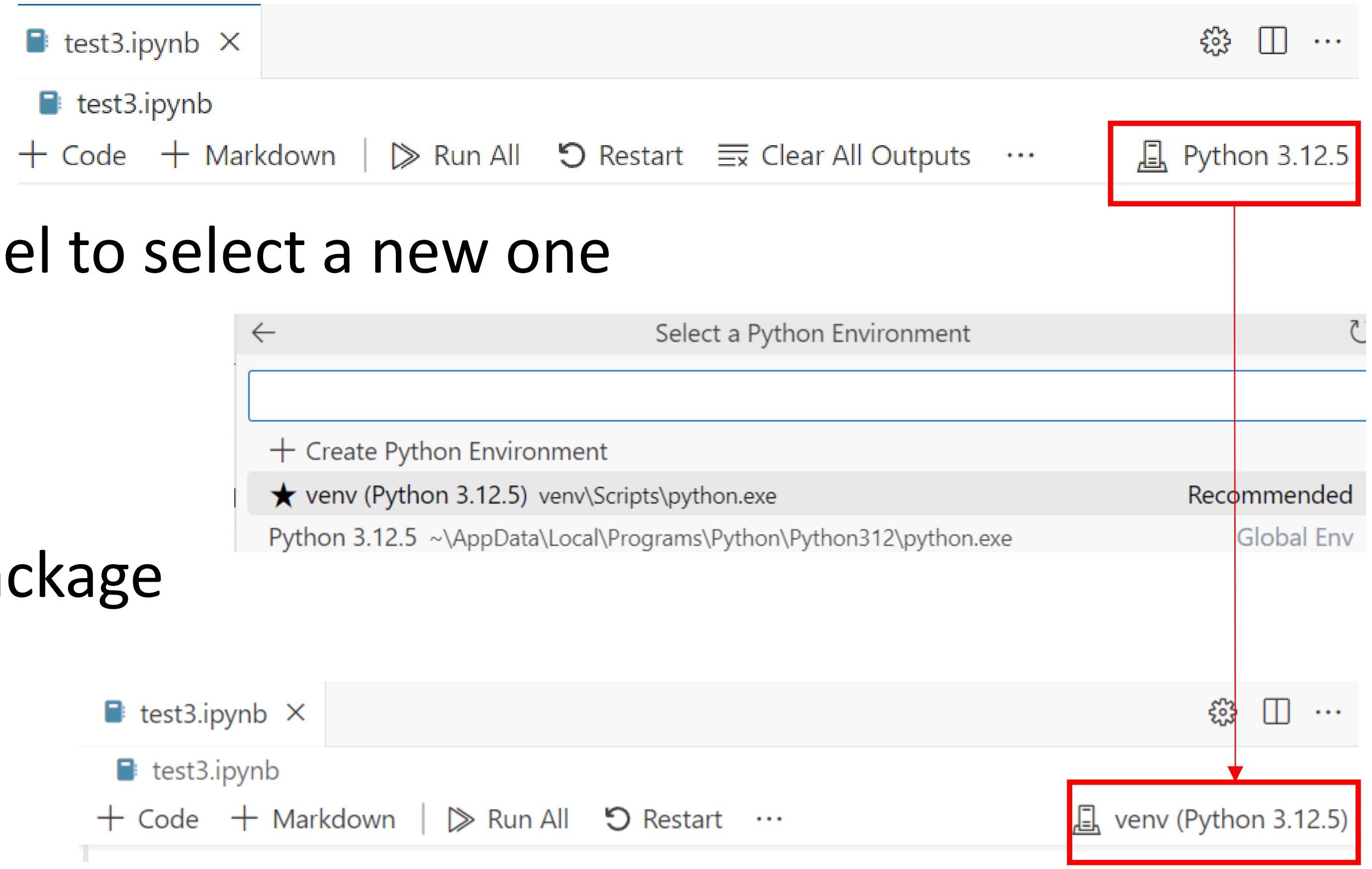


```
EXPLORER
∨ MY NOTEBOOKS 2025-2026
  ∨ venv
    ∨ Include
    ∨ Lib \ site-packages
      > numpy
      > numpy-2.3.2.dist-info
      > numpy.libs
      > pip
      > pip-24.2.dist-info
    > Scripts
    pyvenv.cfg
  test1.py
  test2.md
  test3.ipynb
```

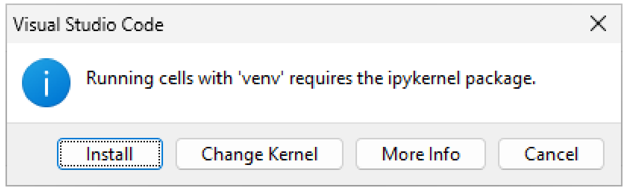
- And that is where they belong.
  Nice and close to the code that is using those libraries

Venv in Jupyter notebooks

- Once you have created a virtual environment it will be suggested as a 'kernel'
  - Open 'test3.ipynb'
  - Check the current kernel
  - Click the current global kernel to select a new one
  - Select your venv


test3.ipynb
test3.ipynb
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Python 3.12.5
Select a Python Environment
+ Create Python Environment
★ venv (Python 3.12.5) venv\Scripts\python.exe Recommended
Python 3.12.5 ~\AppData\Local\Programs\Python\Python312\python.exe Global Env
test3.ipynb
test3.ipynb
+ Code
+ Markdown
Run All
Restart
venv (Python 3.12.5)

  - (Also install the ipykernel package when asked to do so)


Visual Studio Code

Running cells with 'venv' requires the ipykernel package.

Venv in Jupyter notebooks

- Every time, you open a new notebook it will ask you which kernel to use
- Always select the venv you just made, never the default Python installation

- Nice to know: want a venv in an older version of Python?
  - Install the older version of Python, but don’t let it register in the path
  - Create the new venv using the full path to the new (old version of) Python
  - That venv will from now on always use the old Python

Filling a venv

- We'll be using quite a lot of libraries in this course
- You can install new libraries using “pip install …”
- And you can run this in a code cell by saying “!pip install …”
- And you can freeze an environment by using “pip freeze”

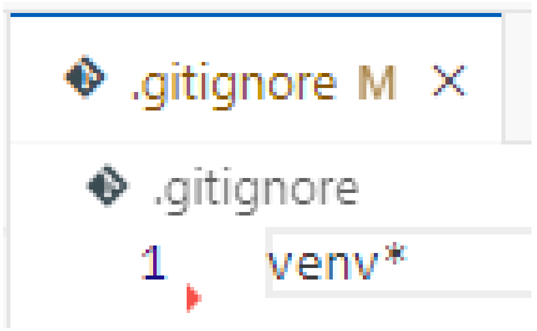
- More information in the notebook called “1 - Working with a venv.ipynb”

Venv in GitHub

- Finally, suppose you are working with a venv in a folder that is synced with GitHub, do you want to upload your venv to GitHub?

No.

- You do want to run a pip freeze every now and then and upload the requirements file
- To prevent uploading the venv to GitHub:



```
venv*
```

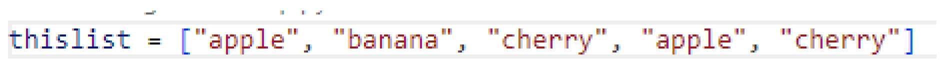
Debugging

- https://code.visualstudio.com/docs/datascience/jupyter-notebooks

- Read the above article. It will speed up your development enormously.

Multiline cursor

- Ever needed to do the same thing on multiple lines?
- Multi-Line Editing
  - Windows: Ctrl + Alt + Arrow Keys
  - Linux: Shift + Alt + Arrow Keys
  - Mac: Opt + Cmd + Arrow Keys
- Multi-cursor editing!
  - press the Alt key (or Option key on a Mac) and use the mouse to place cursors



```python
thislist = ["apple", "banana", "cherry", "apple", "cherry"]
```

Summary

- Setting up an IDE is always the first step when starting a new programming language
  - It takes some time, but is worth doing properly
- You may want to start out using a simple editor when trying a new language, but a good IDE will always help you work better in the long run
- After going through these slides, you’ll have a working setup