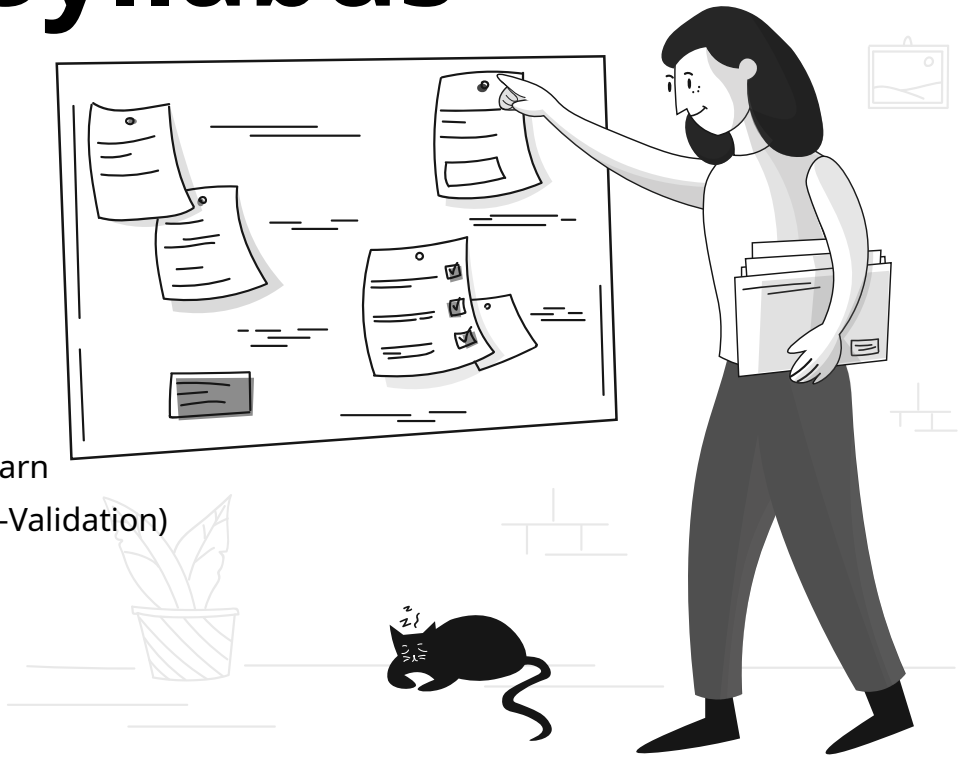
Environment Setup

Eng | Mahmoud Tariq

08/26/2025

Course Syllabus



- **Intro to Machine Learning & Environment Setup**
  - Task: Setup the envrionment
- Foundations: Key Linear Algebra & Stats + Hands-on NumPy
- Data Handling & Visualization + Hands-on Pandas
- The ML Workflow + Linear Regression with Scikit-learn
- Classification Models (Logistic Regression, KNN) with Scikit-learn
- Model Evaluation & Improvement (Metrics, Overfitting, Cross-Validation)
- Mini-Project: Putting it all together with Scikit-learn
- Introduction to Neural Networks
- Building a Neural Network with a Framework
- Course Project Description + Where to Go From Here

Tools used in this course:

- Vscode (optional, alternatives: jupyter, kaggle, colab)
- Python
- Git & Github
- That’s it!

Setup: Python

- Install if you haven’t already:
  - https://www.python.org/downloads/

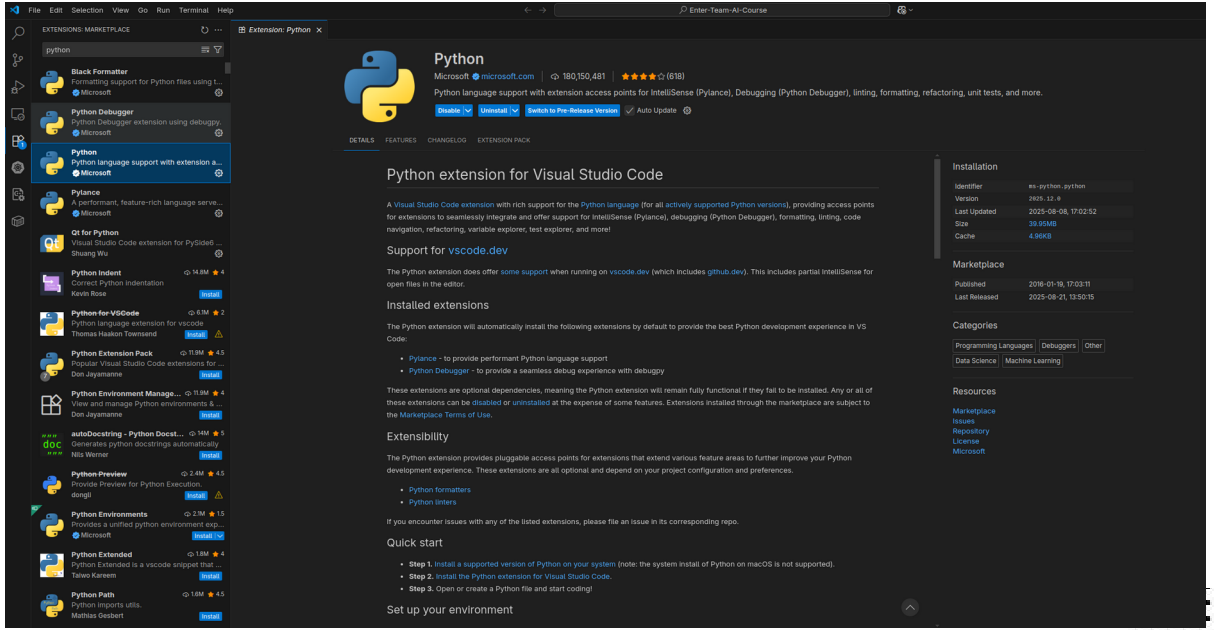
Setup: Vscode

- Install if you haven’t already:
  - https://code.visualstudio.com/

Setup: Vscode

- Now we need to install some extensions:

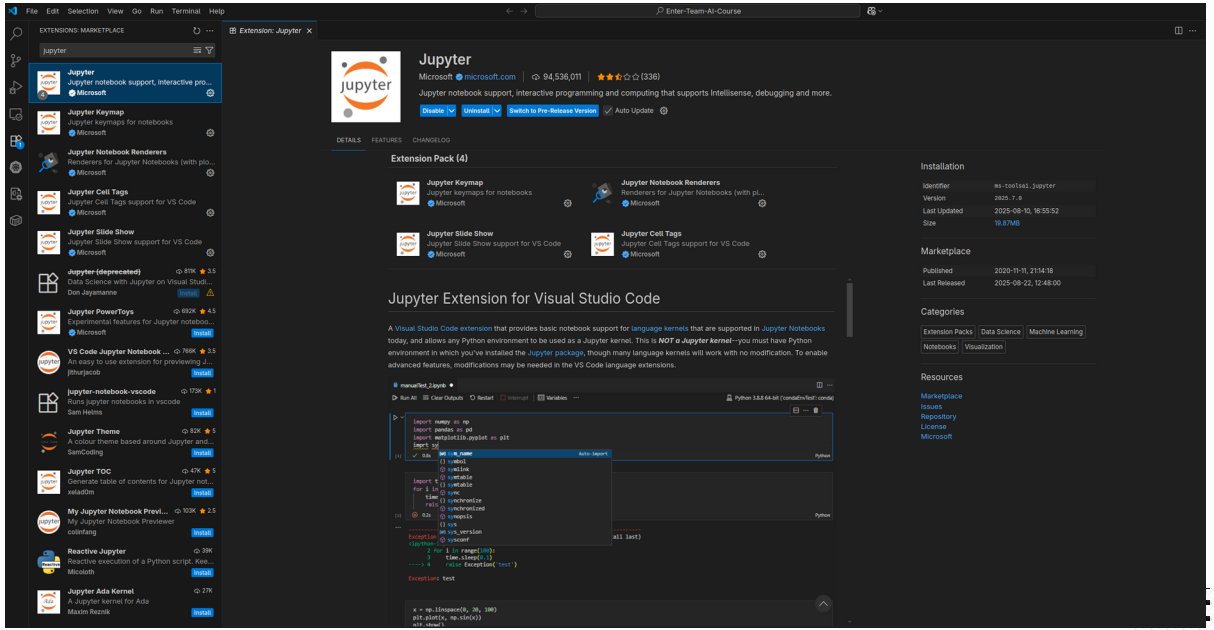
1) Python for autocomplete and syntax highlight:



Setup: Vscode

- Now we need to install some extensions:

2) Jupyter for reading and running Python notebooks (.ipynb):



Setup: Git

- Install if you haven’t already:
  - https://gitforwindows.org/

Setup: Github

- Sign up if you haven’t already:
  - https://github.com/signup

Course Repository

"

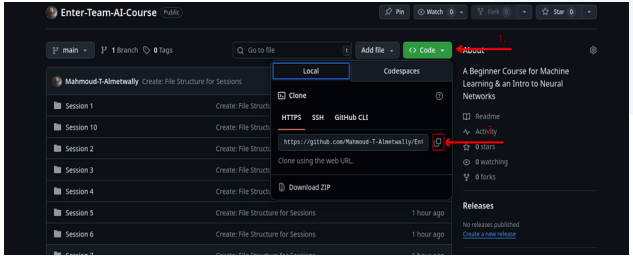
Intro to Machine learning & Neural Networks

"

How to: Get Code & Materials

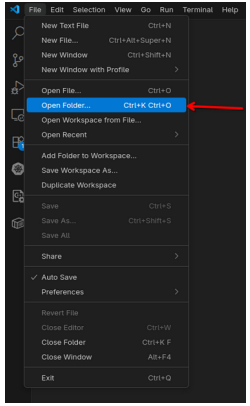
01

- Copy link:



02

- Choose a Folder

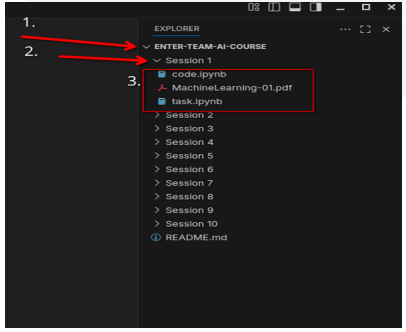


03

- Clone repo

04

- Open & Navigate



Python Environments

- What is a Python environment?
- Why should you use it?
- How to use it?

Python Environments: What is it?

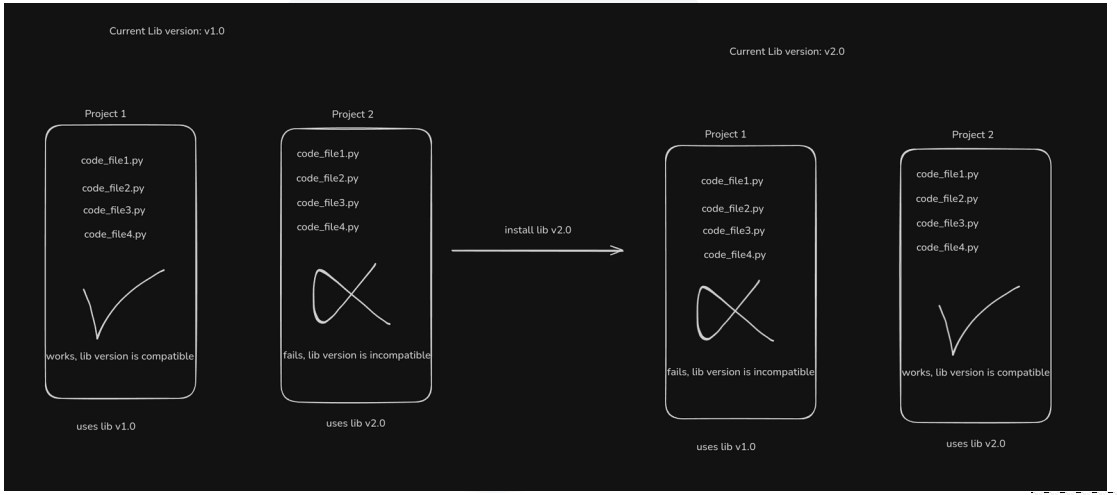
- A Python Environment is a tool for seperating projects' tools/modules and preventing conflicts
  - Module/library: a module is code written by someone else and reused across projects
  - Conflict: a conflict occurs when using two versions of the same module/library globally

    Which causes an unintentional fail in projects that use a different version

Python Environments: Why?

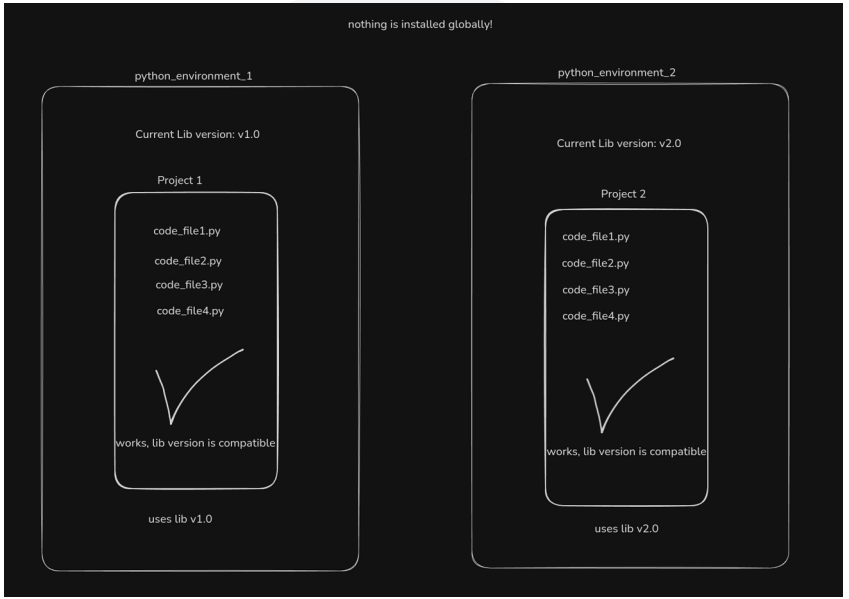
- Imagine this: you have a project that uses library called: “lib” with version v1.0

  then you decide to create another project with the same library but this time you decide to use the newer version of “lib”, v2.0, and recklessly decide to install it globally, but this version of “lib” causes the older version v1.0 code to error out and fail, So, how do we fix this?



Python Environments: Why?

- Fix: introducing, Environments. With Environments we can install libraries/modules in projects seperately, allowing each project to use it’s respective version of the library.



Python Environments: How?

- To use Python environments first we ensure that python is properly installed:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

[Mahmoud@MahmoudArch Enter-Team-AI-Course]$ python --version
Python 3.13.7
[Mahmoud@MahmoudArch Enter-Team-AI-Course]$
```

Python Environments: How?

- Then, we use the built-in package “venv” from python to create a new environment:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
                                       1.                  3.
[Mahmoud@MahmoudArch Enter-Team-AI-Course]$ python -m venv python_environment_name
                                                  2.
```

- The command is comprimised of 3 parts:
  1. “python -m”: python is used when trying to run .py code, paired with “-m” for “module” to run the code as a module
  2. “venv”: which stands for ‘virtual environment’ is the package used for managing/creating python environments
  3. “python_environment_name”: is replaced in the actual command with the environment name

Python Environments: How?

- Lastly, we activate the environment:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

[Mahmoud@MahmoudArch Enter-Team-AI-Course]$ python_environment_name/Scripts/activate
```

- And to exit the Python environment:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

[Mahmoud@MahmoudArch Enter-Team-AI-Course]$ deactivate
```

Python Environments: How?

- Now, using the environment is simple, first we activate it like previously:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

[Mahmoud@MahmoudArch Enter-Team-AI-Course]$ python_environment_name/Scripts/activate
```

- Then, we use python or pip normally and your environment will be used:

1. env is the environment name

2. a normal python command

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

(env) [Mahmoud@MahmoudArch Enter-Team-AI-Course]$ pip install numpy
Collecting numpy
  Using cached numpy-2.3.2-cp313-cp313-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata (62 kB)
Using cached numpy-2.3.2-cp313-cp313-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (16.6 MB)
Installing collected packages: numpy
Successfully installed numpy-2.3.2
(env) [Mahmoud@MahmoudArch Enter-Team-AI-Course]$
```

Python Notebooks

- What is a Python Notebooks?
- Why should you use it?

Python Notebooks: What is it?

- A Python Notebook (a file with the .ipynb extension): is a python file that allows the user to the following:
  - Run the code in the order they desire.
  - Run a piece of code (a cell) multiple times.
  - Write markup around code for easy explainations and writing equations.

Python Notebooks: What is it?

- Another benefit of Notebooks is that output is stored for each cell (will be demonstrated shortly), which means that you don’t have to run the code each time you want to view the output.

Python Notebooks: Demonstartion

Session 1 > code.ipynb > This is a Markdown cell

Generate | Code | Markdown | Run All | Restart | Clear All Outputs | Jupyter Variables | Outline ··· env (Python 3.13.7)

Generate | Code | Markdown

[1] 0.0s Python

```python
print("hello world") # Running one cell doesn't run all the code. (Notice that the output is directly below not in console)
```

```
hello world
```

[2] 0.0s Python

```python
1 + 1 # Raw values are printed directly.
```

```
2
```

[6] 0.0s Python

```python
arr # This will cause an error at first. (Hint: try running the below cells first)
```

```
[1, 2, 3]
```

[5] 0.0s Python

```python
arr = [1, 2, 3] # However assignments don't print
```

[4] 0.0s Python

```python
arr # Variables are treated the same as values if no operation is done on them. (Notice variables are stored over all cells once the cell that defines the variable is ran)
```

```
[1, 2, 3]
```

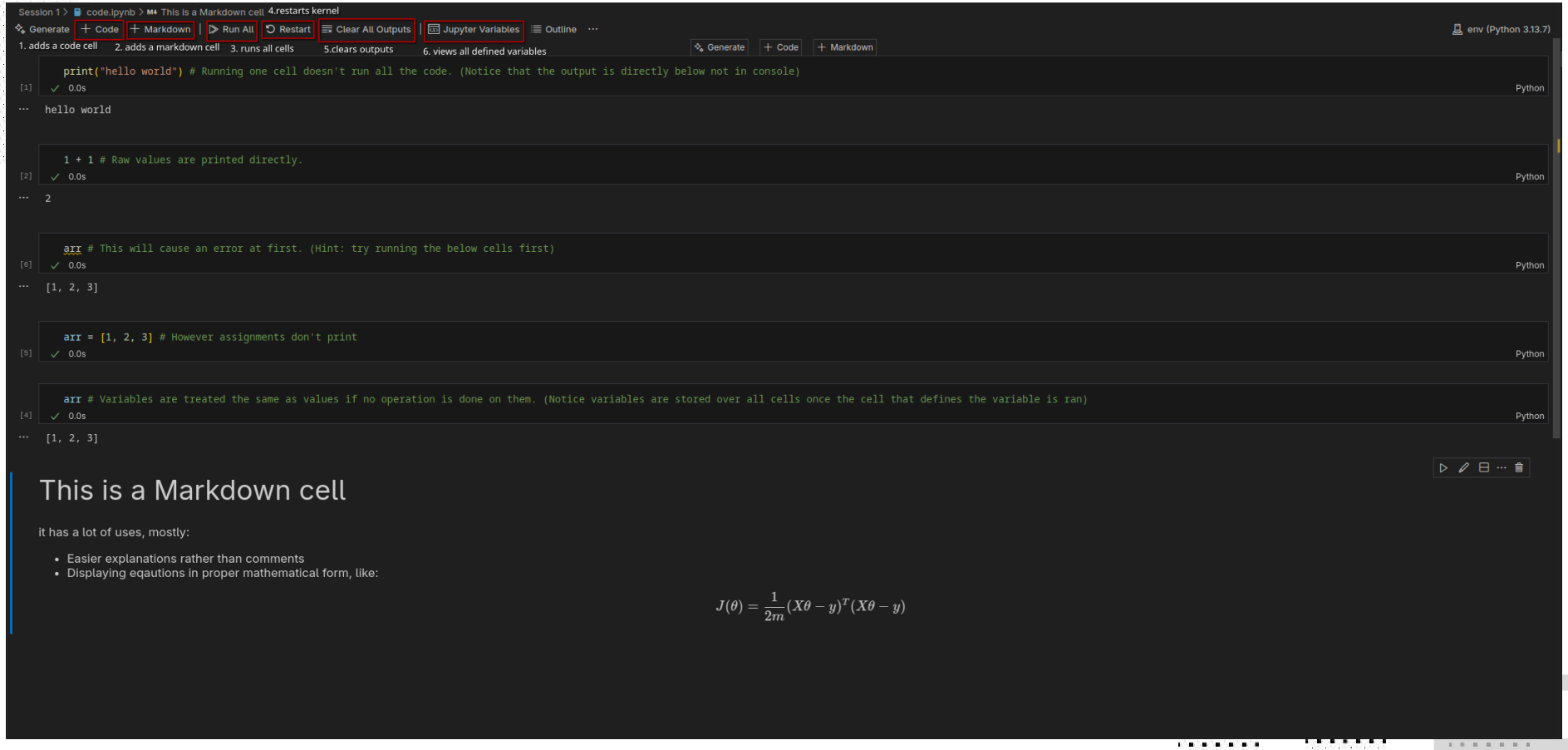
This is a Markdown cell

it has a lot of uses, mostly:

- Easier explanations rather than comments
- Displaying eqautions in proper mathematical form, like:

$$J(\theta) = \frac{1}{2m}(X\theta - y)^T(X\theta - y)$$

Python Notebooks: Demonstartion

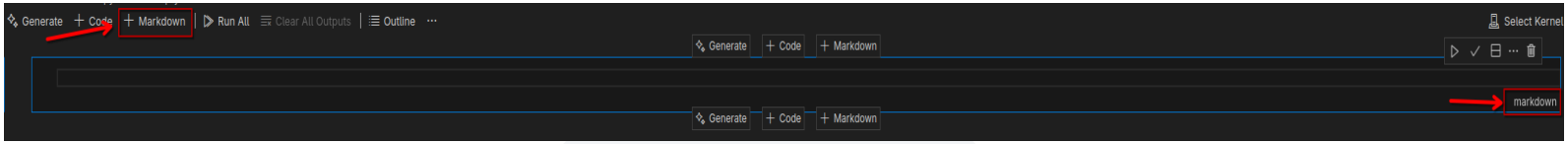


Python Notebooks: Demonstration

1. Adds a new empty Code cell at the bottom.
2. Adds a new empty Markup cell at the bottom.
3. Runs all Code cells from the beginning to the end sequentially.
4. Restarts the kernel (will be explained shortly).
5. Clears all outputs, but doesn’t remove already defined variables.
6. Allows you to view all the defined variables in your code and the values they hold.

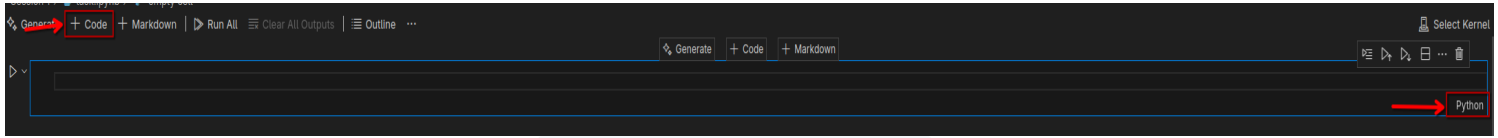
Python Notebooks: Demonstration

- Adds a new empty Code cell at the bottom.



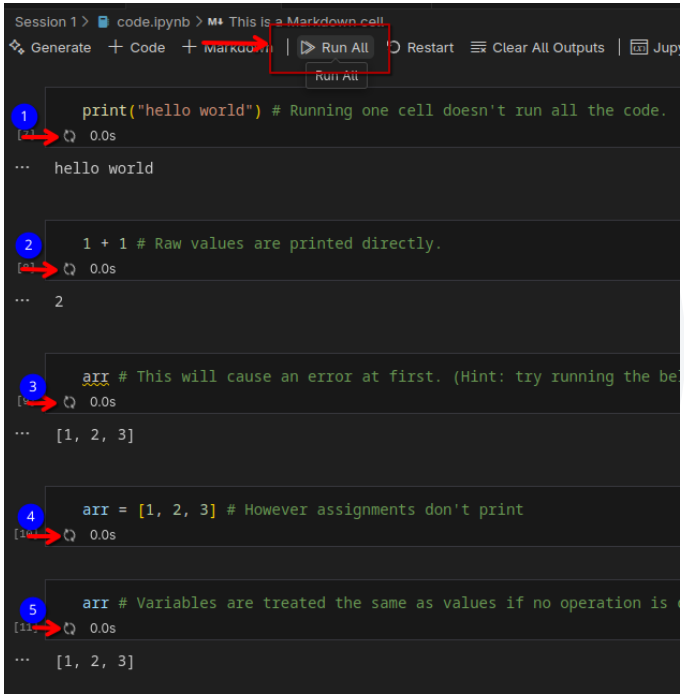
Python Notebooks: Demonstration

- Adds a new empty Markup cell at the bottom.



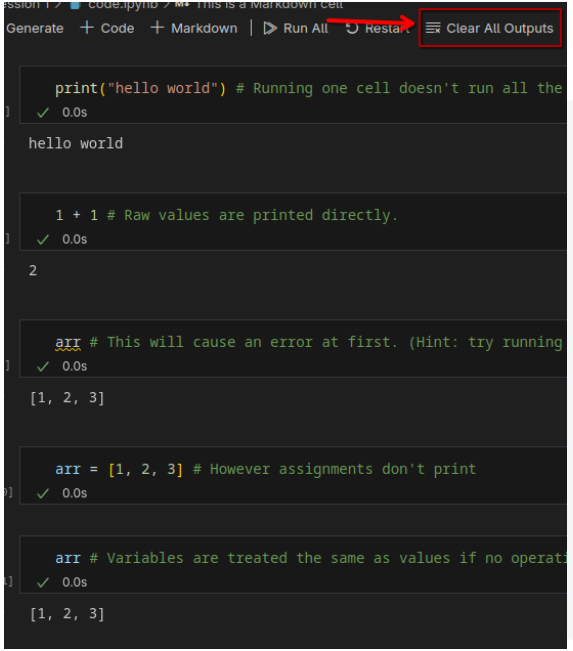
Python Notebooks: Demonstration

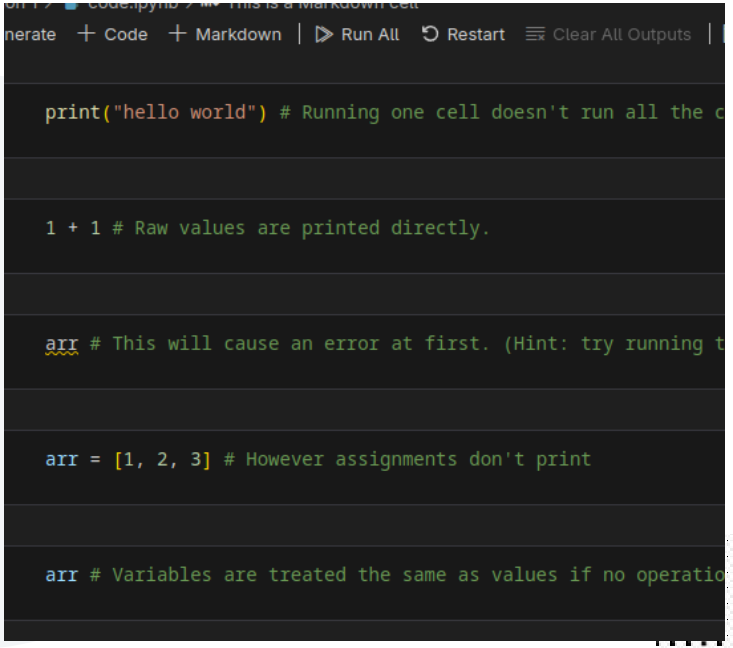
- Runs all Code cells from the beginning to the end sequentially.



Python Notebooks: Demonstration

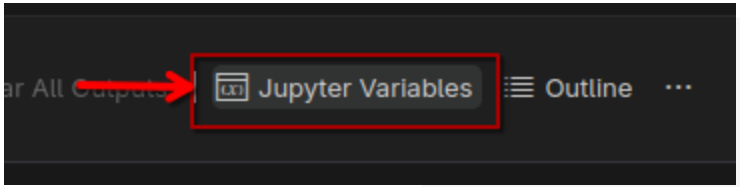
- Clears all outputs, but doesn’t remove already defined variables.

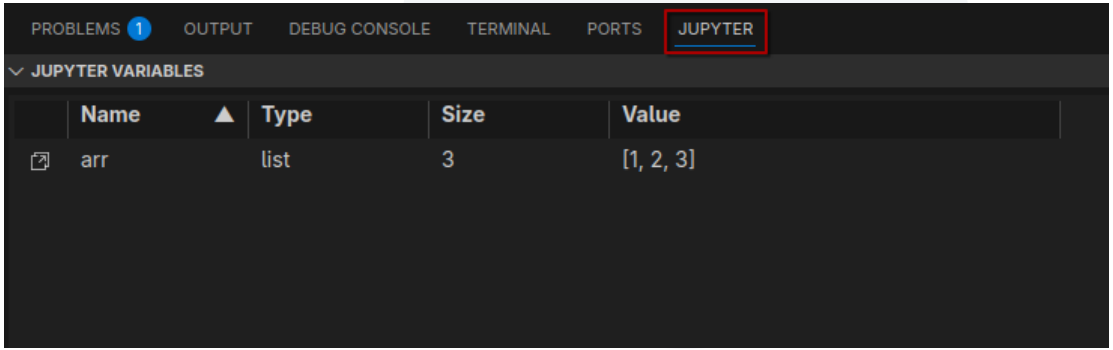




Python Notebooks: Demonstration

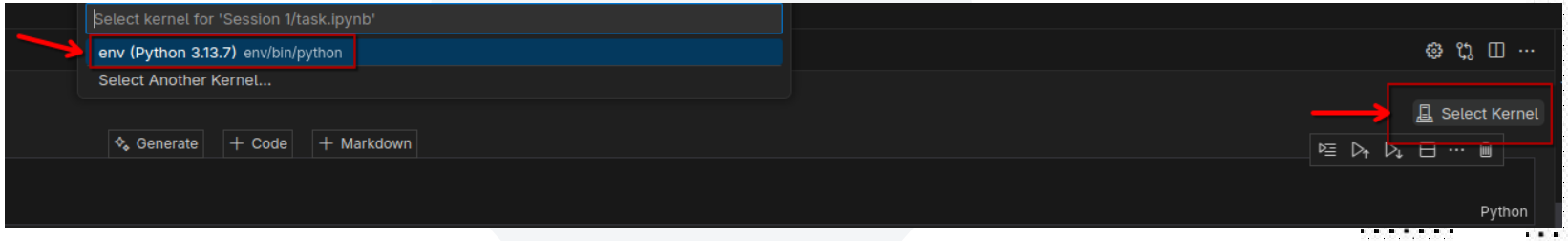
- Allows you to view all the variables defined in your code and the values they hold.





Python Notebooks: Kernels

- Notebook kernels: kernels are what run our code in the notebook, they also determine what language we are using (yes, you can use other languages in notebooks as well), they also store our variables.
- Why should you care? Because you can choose the kernel you use in your notebooks!

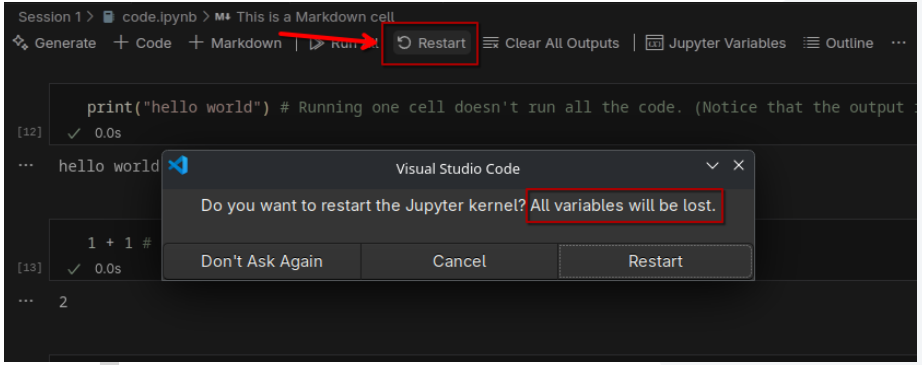


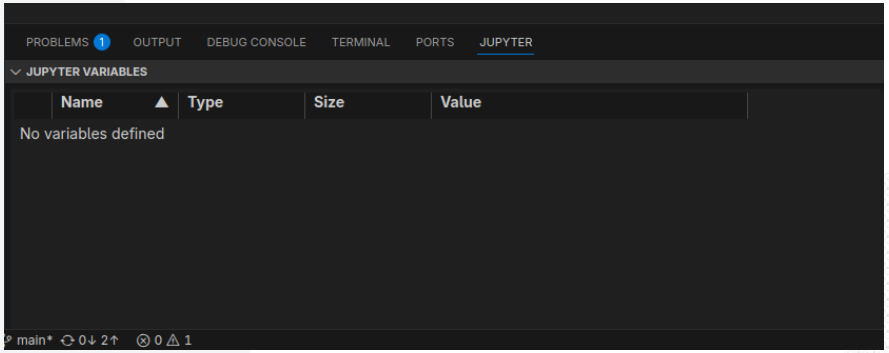
Python Notebooks: Kernels

- Choosing your kernels may sound confusing, but it’s best practice to just choose the one from your environment (like in previous slide), however that is not the only choice, you can also choose the global python version that is installed in your machine under “Select Another kernel…” however that is not recommended and can cause mismatches between installed packages in the python environment and the global version.

Python Notebooks: Kernel

- Restarting the kernel removes all variables from memory, but doesn't clear outputs. Allows you to recover from having too much memory used or errors.





Task 01:

- Clone Course Repo into an appropriate folder.
- Create a python environmnet called “env” using the “venv” package.
- Using pip, install the following packages:
  - numpy
  - pandas
  - scikit-learn
  - matplotlib
- Run the “task.ipynb”, after confirming success, upload a screenshot of the output.
- Watch the following playlist for the next session: Linear Algebra by 3Blue1Brown

Thank you