1. Introduction

Course: Python

Who are we?

Dr. Bardh Prenkaj

Ph.D. Computer Science

Prof. Ass. Marco Cascio

Ph.D Computer Science

Flavio Giorgi

M.Sc. Computer Science

Leonardo Picchiami

M.Sc. Computer Science

What is this course about? (Scope & Goal)

- The basic **concepts** of programming
- Programming in **Python**
- The **tools** you need to programm (as far as necessary)
- Gaining **practical experience** with all of the above
- Disclaimer: the actual **work** starts **after** this **course**!

How do we do it? (Organization)

- We provide you with:
  - **Concepts** and **tools** for Python program development (the theory)
  - **Exercises** and programming tasks (the practice)
- You need to:
  - **Ask** questions (at any time) and **engage** with your classmates! (we are a **small group**)
  - **Work** through the tasks (on your device)
- Course Material:
  - **We are using Moodle!**
  - **Slides**
  - **Lecture Notes**

What do we cover? (Content)

- Introduction
  - Foundation of Programming
  - Practical / Setup
- Primitives (Data Types)
- Operations
- Collections
- Control Statements
  - If-statements
  - Loops
- Functions
- Classes

1.1 Getting started

- What is a python program?
- Environment

What is a Python program?

Problem

Program

Python

Algorithm

Programming Language

Problem

Description: A **task** or question that requires a solution or answer

Example: Calculate the mean of some given numbers

Algorithm

Description: An abstract **step-by-step procedure** to solve a problem

Example:

1. Sum up all numbers
2. Count all numbers
3. Divide the sum by the count

Program

Description: A set of **instructions** telling your computer how to execute an Algorithm

Example:

```
sum = 0
count = 0
for number in numbers:
    sum = sum + number
    count = count + 1
mean = sum / count
```

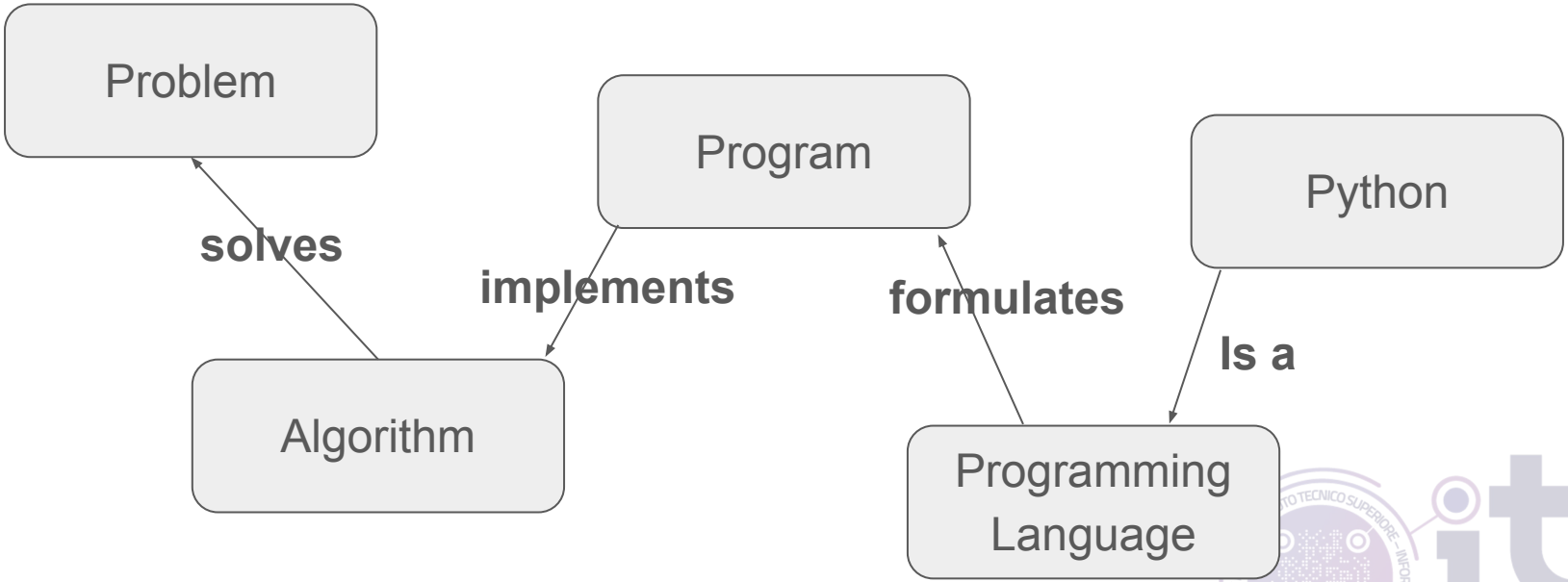
The Programm **text** is the program **code**

Programming Language

Description: A **specific language** for formulating programs

Example: **Python**

What is a Python program?



What environment does a program need?

Permanent Storage

- Your computer's “hard drive”
- “Slow” to access
- E.g. HDD, SSD, NAS, cloud storage, etc.

Your Data

Python program code

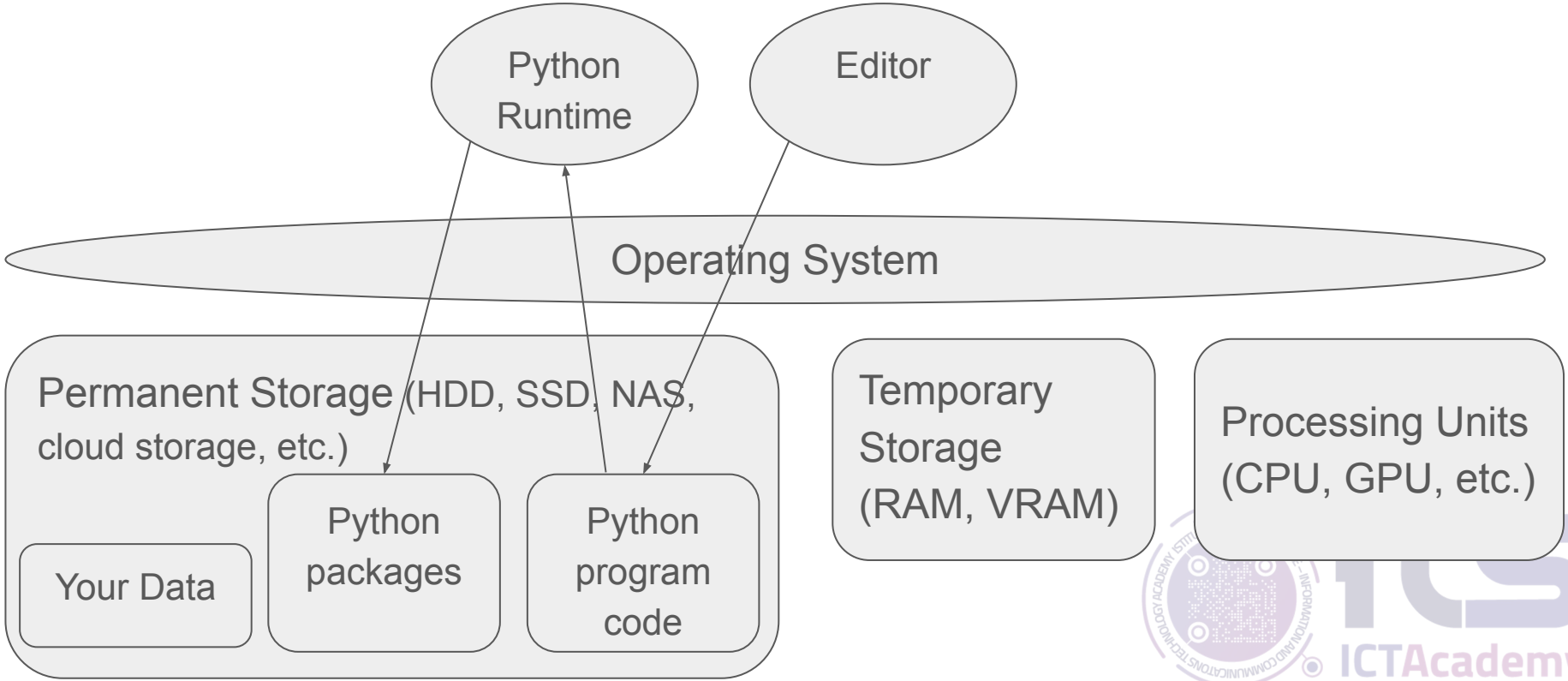
Processing Units

- Do the actual computation
- Central Processing unit (CPU)
- Graphics Processing Unit (GPU)

Temporary Storage

- Your computer's “memory”
- Fast to access
- Only temporary
- E.g. RAM, VRAM

How does the program interact with the environment?

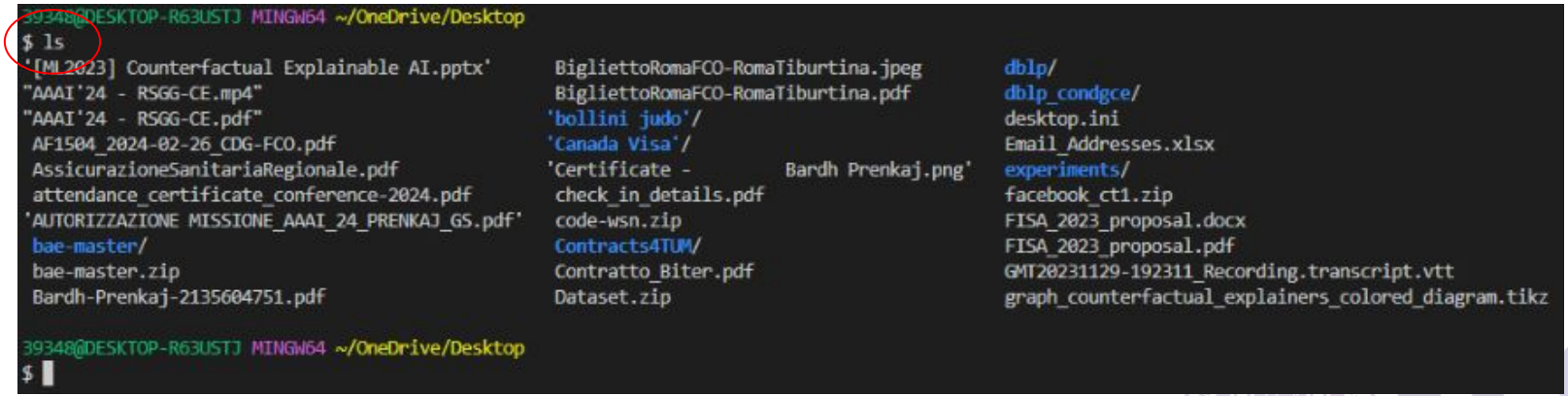


1.2 Setting up our tools: Command Line

- Commands:
  - ls
  - cd
  - mkdir
  - touch
  - rm

Command-Line

- With the **command-line** (cmd) you can communicate directly with your computer (i.e. operating system)
- Example: For listing the content inside a folder (`ls`)



Command-Line - Try it yourself

1. Open command-line: (Use the shortcut **ctrl+alt+t**)
2. Go to your desktop: **`cd`** `Desktop`
3. Create now folder: **`mkdir`** `<name of your folder>`
4. Show your Desktop content:
   - Linux: `ls`
5. Remove folder: **`rm -r`** `<name of your folder>`

Command-line - Navigating directories

- Change your current directory (**working directory**) with `cd <target>`

```
39348@DESKTOP-R63USTJ MINGW64 ~/OneDrive/Desktop
$ cd dblp

39348@DESKTOP-R63USTJ MINGW64 ~/OneDrive/Desktop/dblp
$ mkdir cartella-di-prova
```

- For going up the folder structure use “..” as target

```
39348@DESKTOP-R63USTJ MINGW64 ~/OneDrive/Desktop/dblp
$ cd cartella-di-prova/

39348@DESKTOP-R63USTJ MINGW64 ~/OneDrive/Desktop/dblp/cartella-di-prova
$
```

```
39348@DESKTOP-R63USTJ MINGW64 ~/OneDrive/Desktop/dblp/cartella-di-prova
$ cd ..

39348@DESKTOP-R63USTJ MINGW64 ~/OneDrive/Desktop/dblp
$
```

Create your first “empty” Python file

You can do that using the command **touch** *<file_name>*

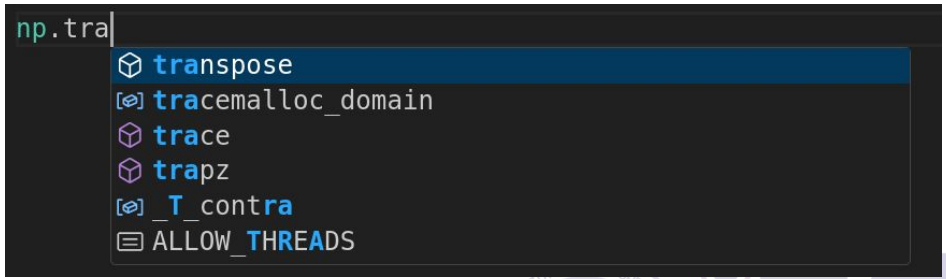
```
39348@DESKTOP-R63USTJ MINGW64 ~/OneDrive/Desktop/dblp/cartella-di-prova
$ touch first_python_program.py

39348@DESKTOP-R63USTJ MINGW64 ~/OneDrive/Desktop/dblp/cartella-di-prova
$ ls
first_python_program.py

39348@DESKTOP-R63USTJ MINGW64 ~/OneDrive/Desktop/dblp/cartella-di-prova
$
```

1.3 Setting up our tools: Integrated Development Environment (IDE)

- IDE
- VSCode

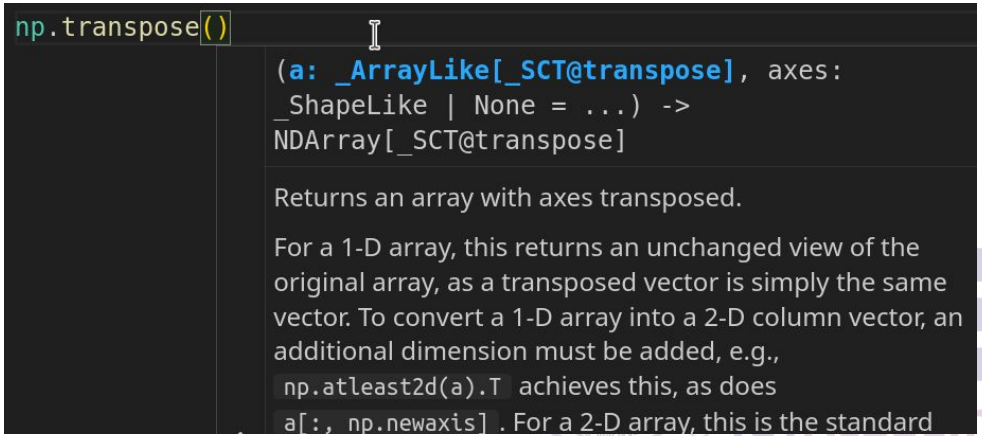
Visual Studio Code - Programming more comfortably!

- **Code editors** are great for developing programs!
  - Visual Studio code is actually more like a *Integrated Development Environment* (IDE)
- Write code and **save** it as a file
- **Highlight errors** in code
- **Autocomplete** and recommendations



Visual Studio Code - Programming more comfortably!

- **Code editors** are great for developing programs!
  - Visual Studio code is actually more like a *Integrated Development Environment* (IDE)
- Write code and **save** it as a file
- **Highlight errors** in code
- **Autocomplete** and recommendations
- **Documentation** and references

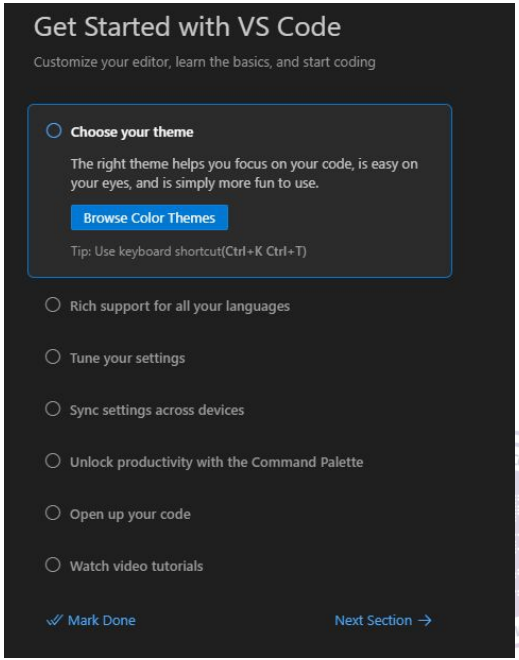


Visual Studio Code - Programming more comfortably!

- **Code editors** are great for developing programs!
  - Visual Studio code is actually more like a *Integrated Development Environment* (IDE)
- Write code and **save** it as a file
- **Highlight errors** in code
- **Autocomplete** and recommendations
- **Documentation** and references
- Code **versioning** control with Git → more on Git later!

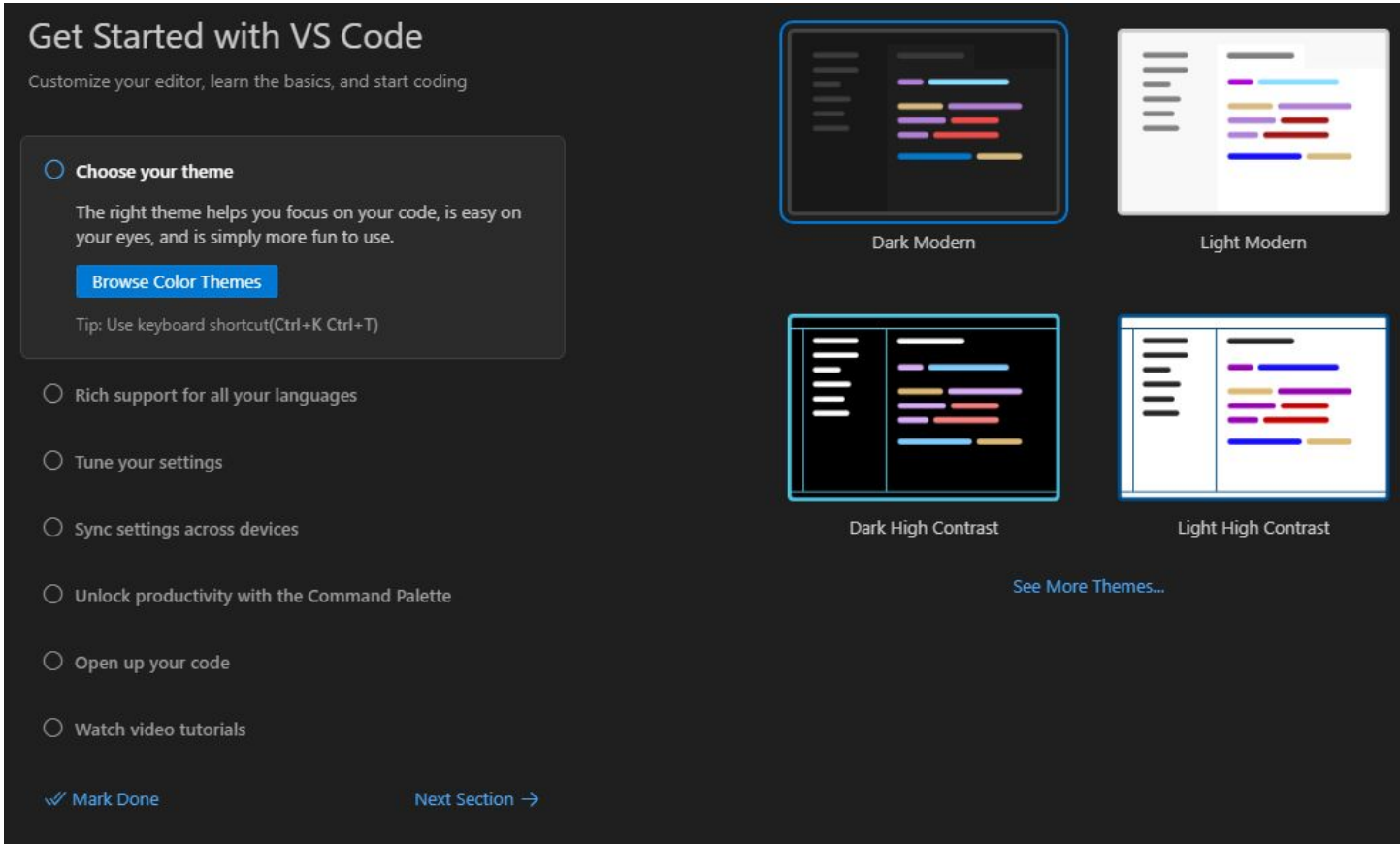
Visual Studio Code - Installation

- Go to code.visualstudio.com
- **Download** installer & **Install** (should be similar for Windows & Mac)
  - Select **all** additional tasks
- **Wait** for the others if you are at "Get Started with VS Code" (We will do the setup together)



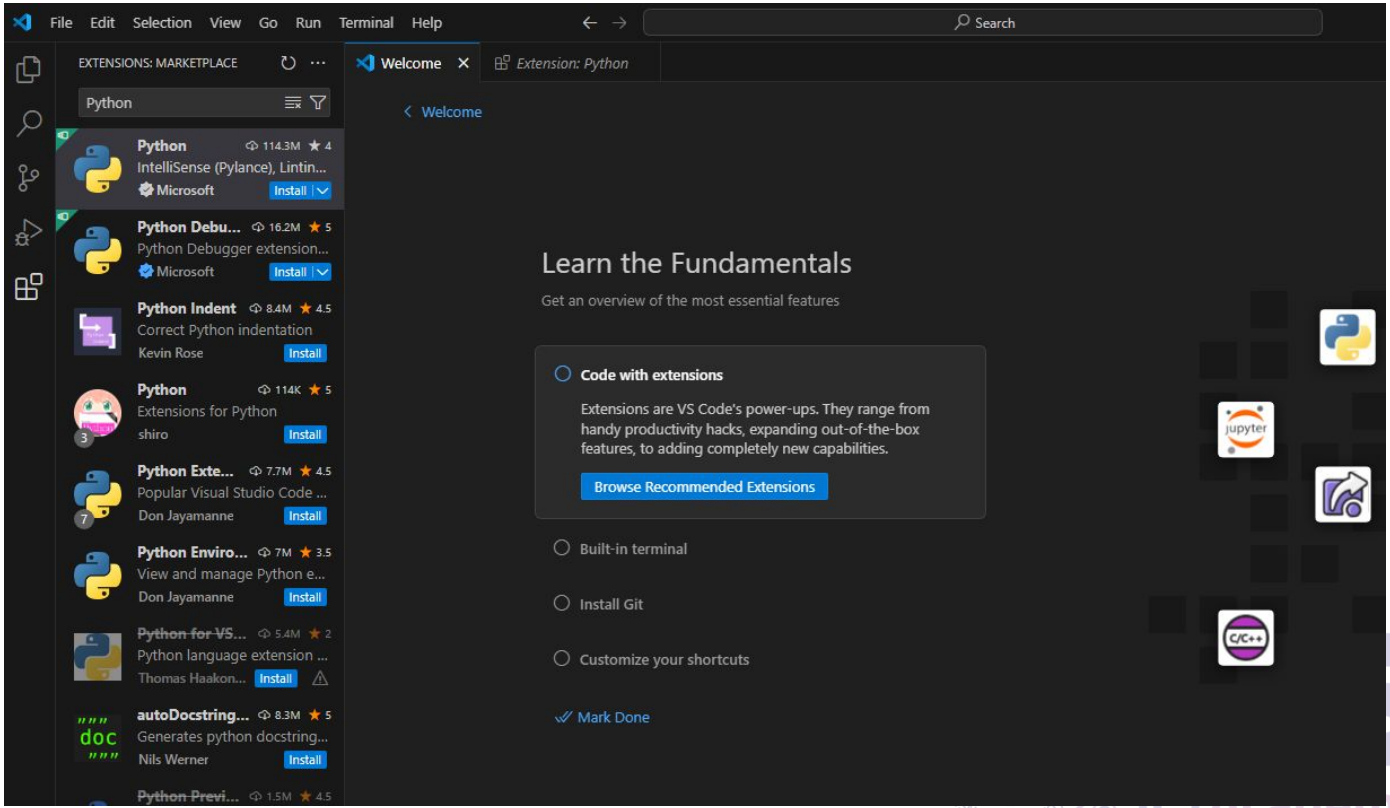
Visual Studio Code - Setup - Theme

1. **Choose** Theme
2. Click “**Next Section**”

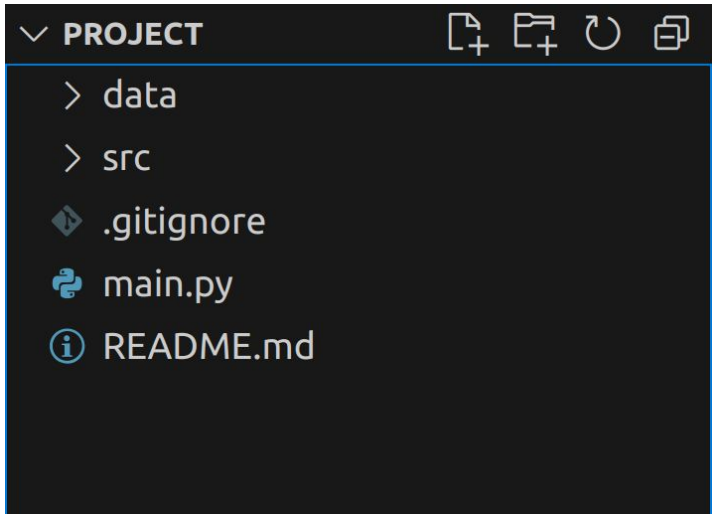


Visual Studio Code - Setup - Python Extension

1. Click “**Browse Recommended Extensions**”
2. Type “**Python**” in the extension **search** bar
3. **Install** the “**Python**” extension
4. If you get the “Get Started with Python Development” Tab opens; **wait** for the others

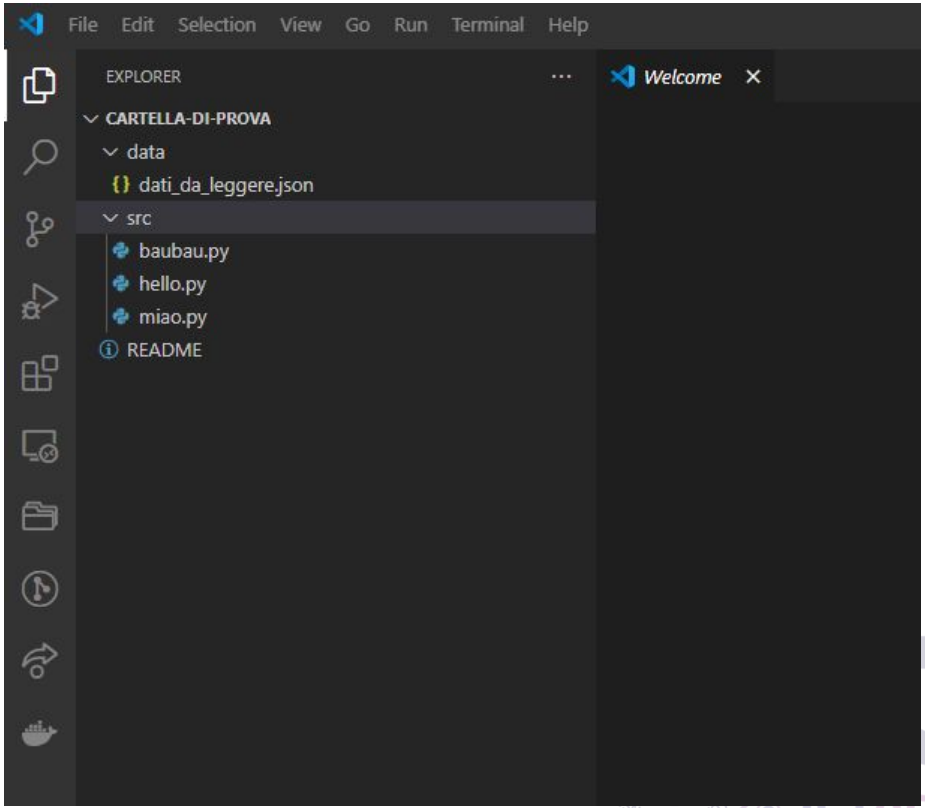


Exercise: Let’s create a “standardized” Python project structure using: mkdir, cd, touch, ls



Visual Studio Code - Setup - Open the project

1. Click “**File > Open Folder**”
2. Choose the parent folder you just created
3. Open the file `hello.py` and write `print(“Hello World”)`



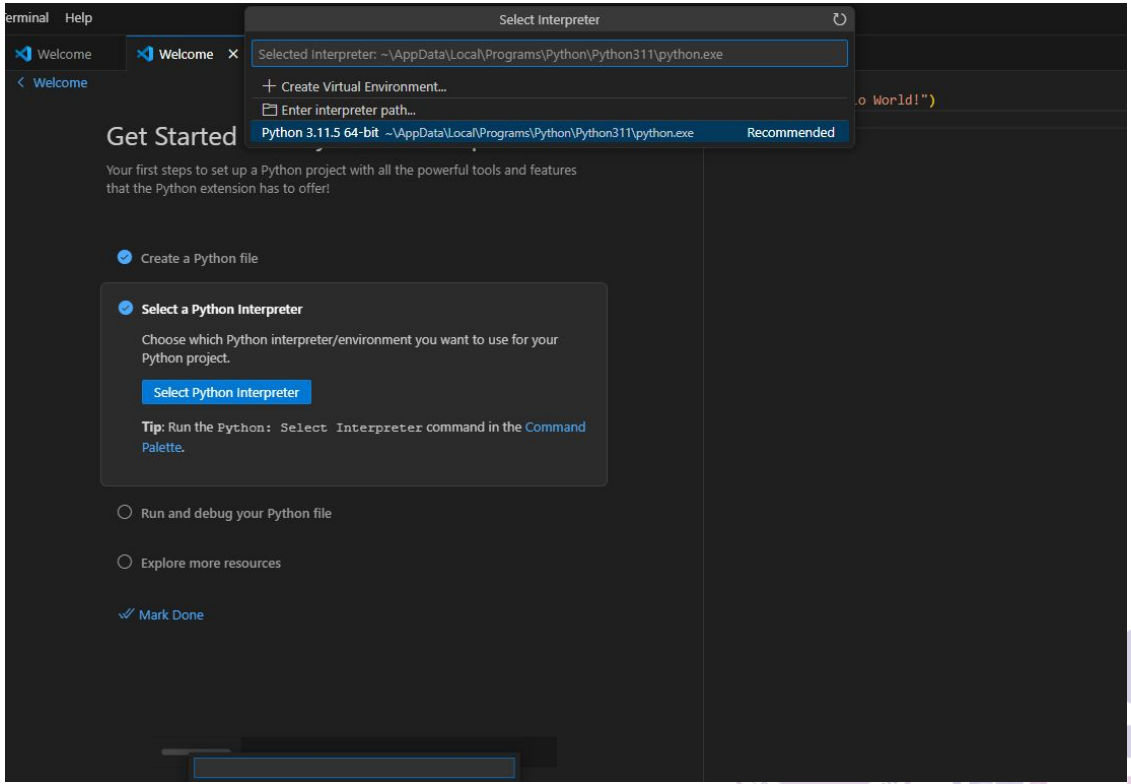
Visual Studio Code - Setup - Running hello.py

1. Click “**Terminal > New terminal**”
2. Navigate to the parent folder of `hello.py`
3. Once there, write `python hello.py`

```
39348@DESKTOP-R63USTJ MINGW64 ~/OneDrive/Desktop/dblp/cartella-di-prova/src
$ python hello.py
Hello World
```

Visual Studio Code - Setup - Python environment

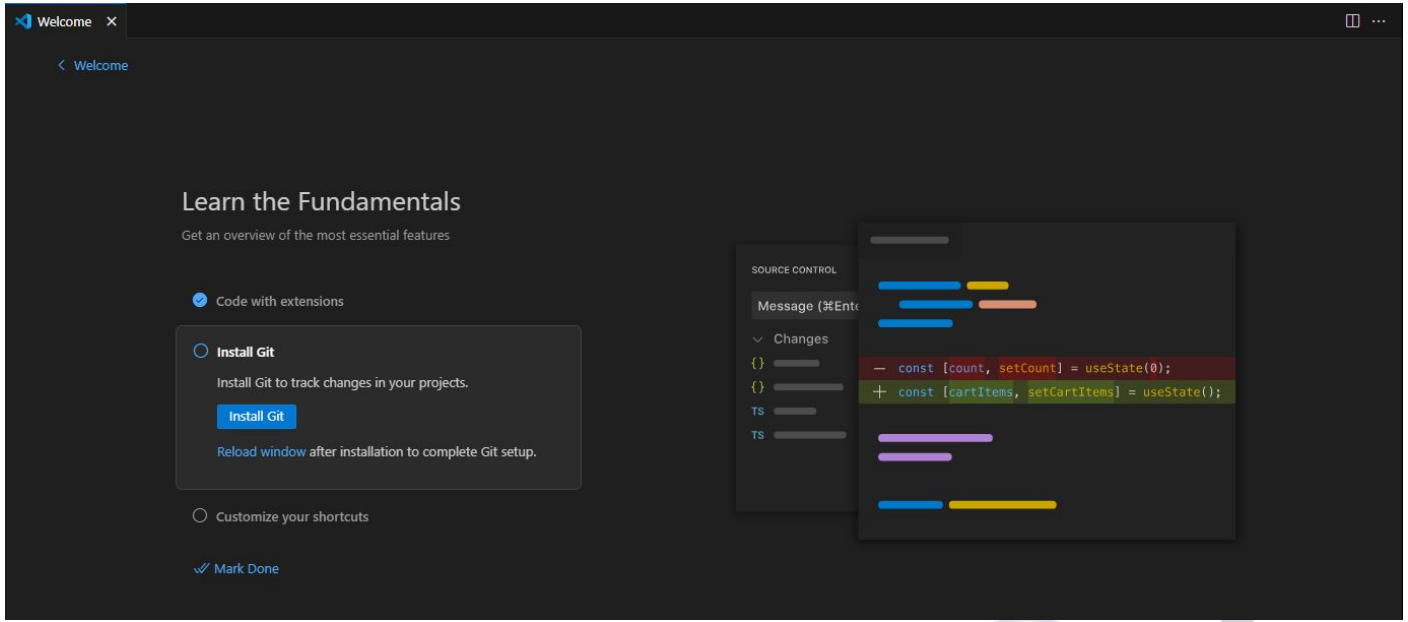
1. Click “**Create Python File**”
2. **Write** in file:
   `print(“Hello World!”)`
3. Go to “**Select** a Python **Interpreter**”
4. **Select** your python **version**.
5. **Go back** into the python **file**



Visual Studio Code - Setup - Git Installation

1. Click “**Install Git**”
2. **Follow** the **instructions** on the pop-up Website.

A. **Download** & **execute** Installer
B. Use **default** settings.
C. **Use VScode** as “**default editor**”
D. Otherwise default options again (there are many)



3. Click “**Reload window**” in VScode