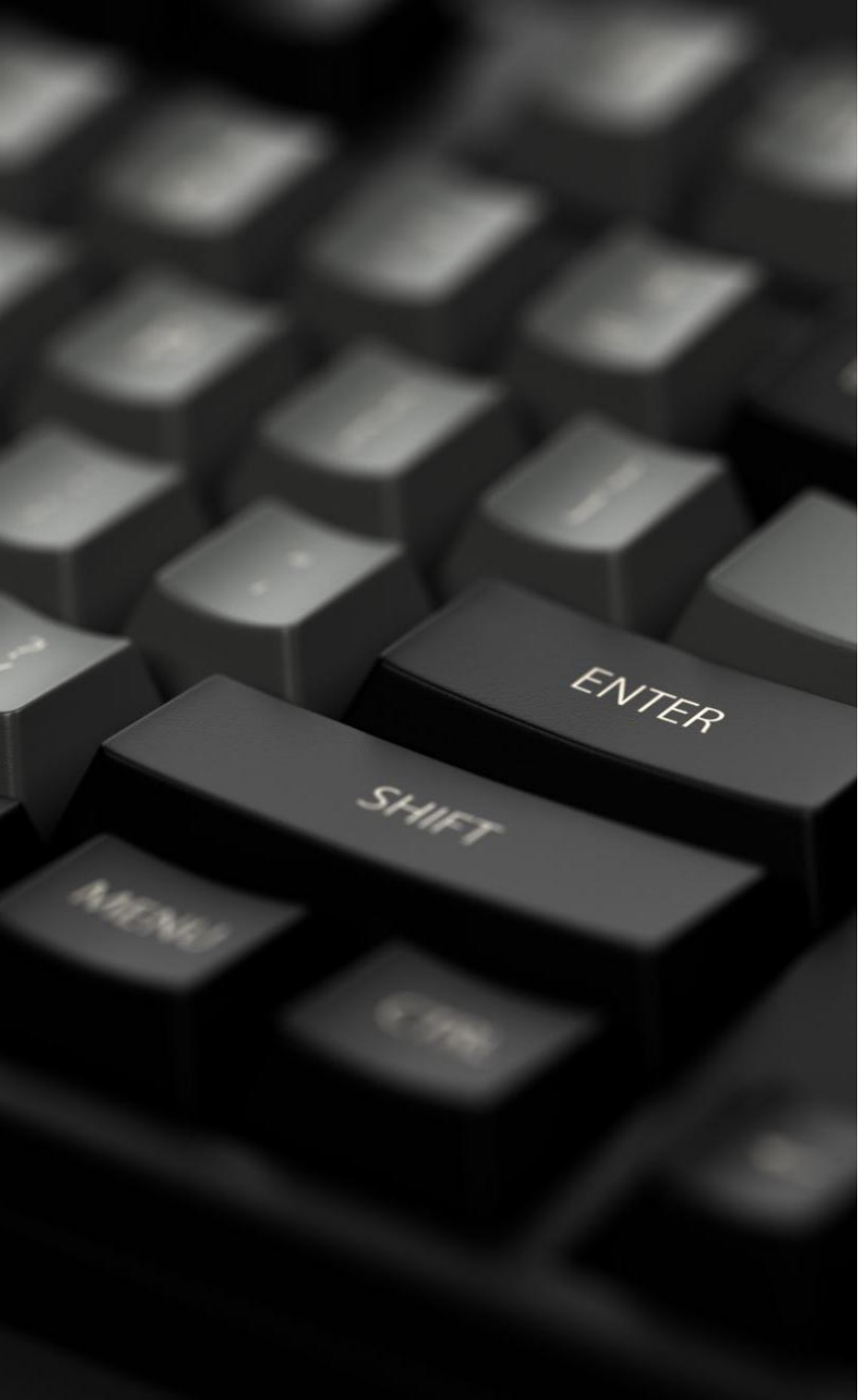
A gamified introduction to Python Programming

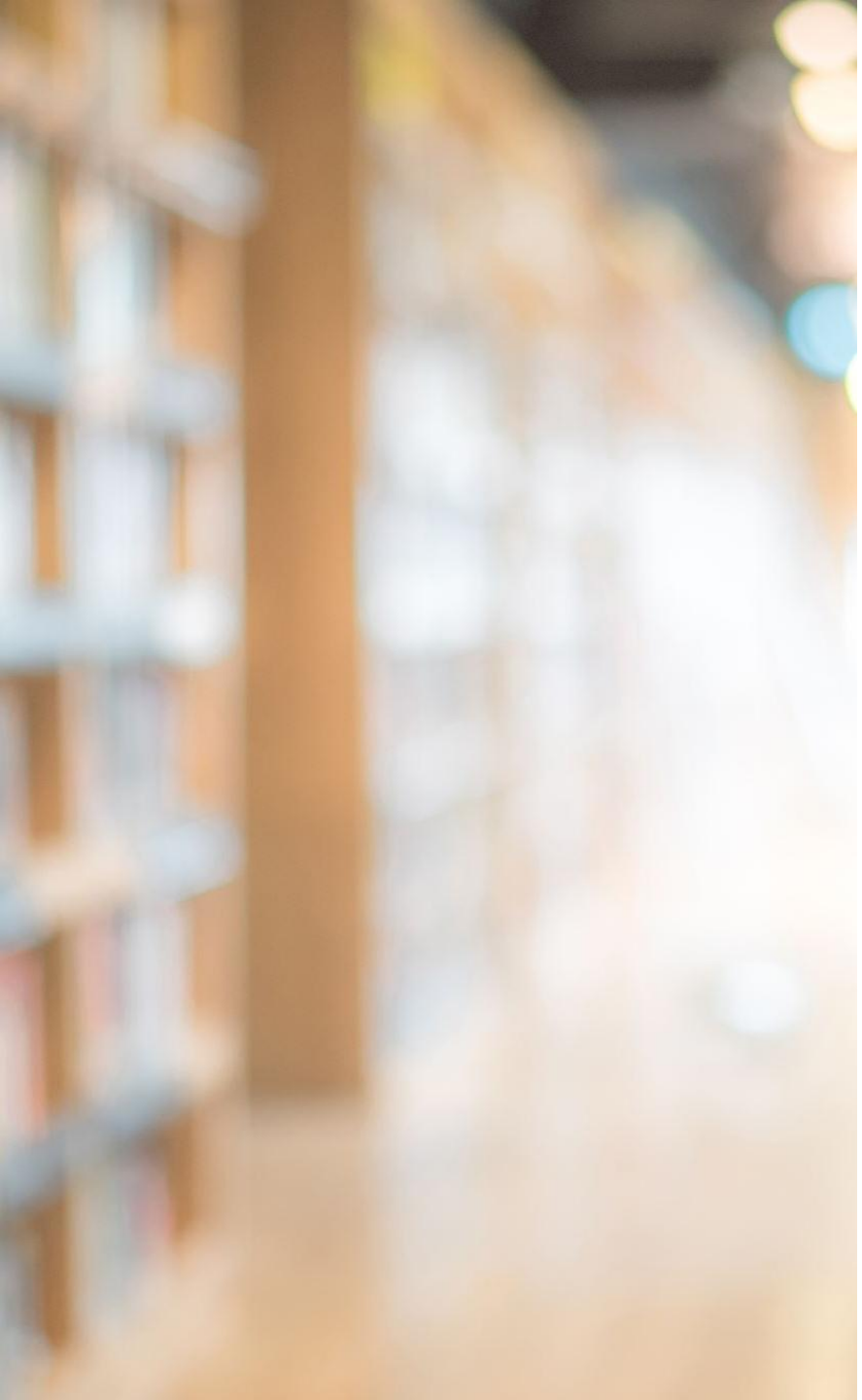
Lecture 1
Introduction

Matthieu DE MARI – Singapore University of Technology and Design



Outline (Chapter 1)

- What are the **objectives of this course**?
- What is **programming**?
- **What are the key concepts** about programming and computer science?
- What is a **programming language**?
- **How do I install and configure Python?**
- How to I write my **first programs** in Python?
- What is a **programming environment**?
- And many other things on the way!



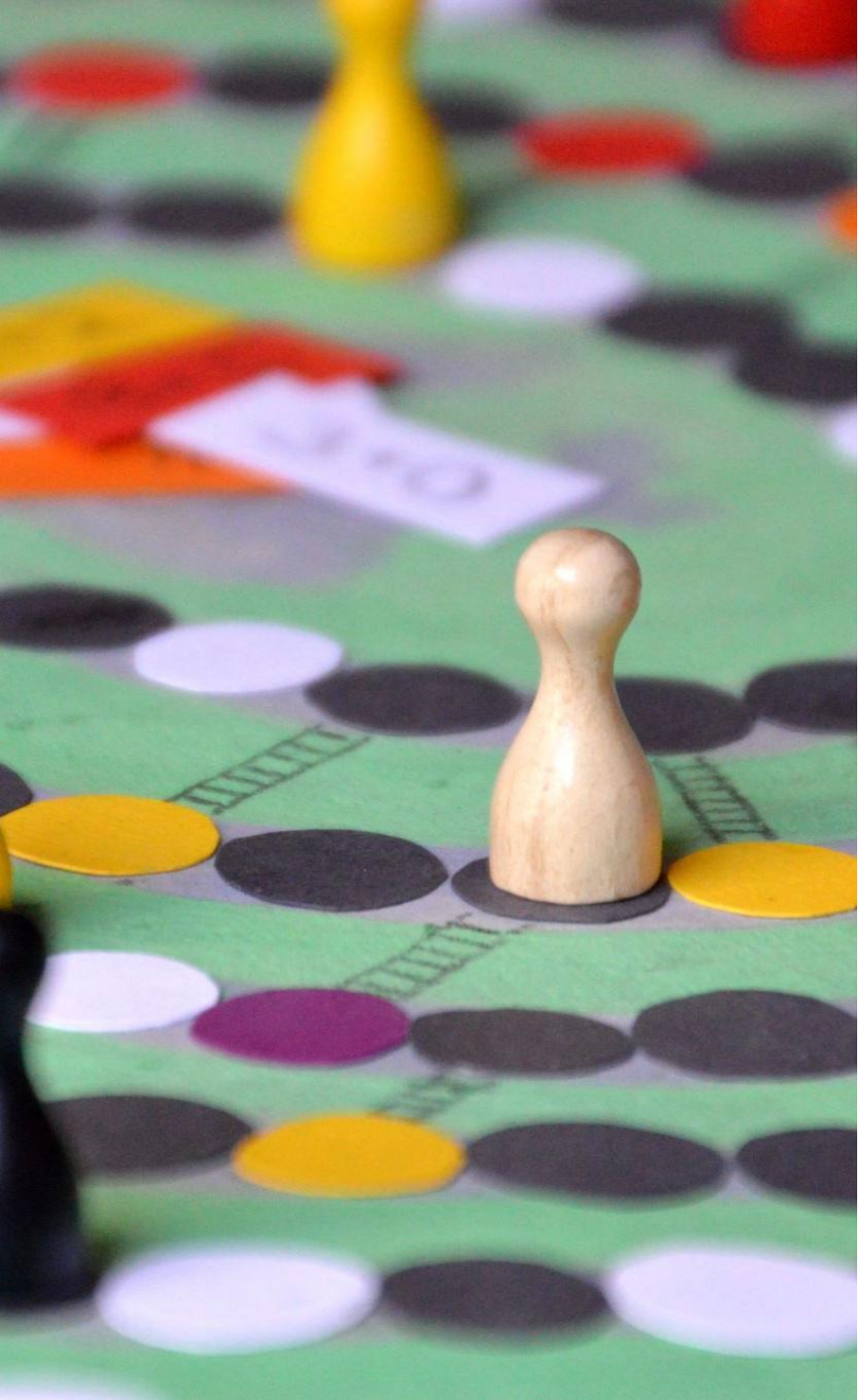
Objectives of this course

Objectives:

- Give the students an introduction to Computer Science, Programming, and Python.
- Serves as a Computer Science 101 course.

Delivery:

- Lessons include a bit of theory (PPT/PDF slides)
- Autonomous practice activities (in Jupyter Notebooks).
- Supporting videos online.

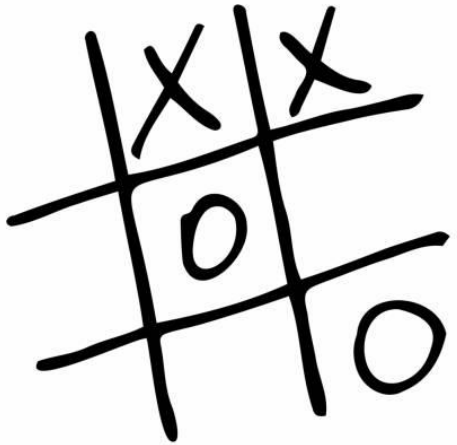


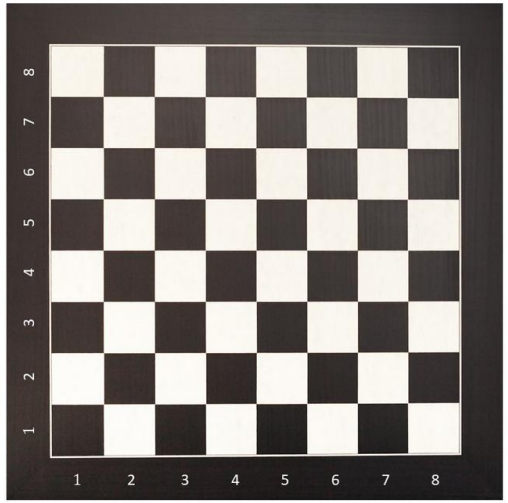
About Gamification

“Games are a good way of thinking about real problems. I use games in all my courses […] because things you have played with are things you remember.”
- *Jarrett Walker*

(Example of Gamification – Using the MiniMetro game to explain concepts of urban planning: http://humantransit.org/2014/12/learning-how-transit-works-from-mini-metro.html)

About Gamification



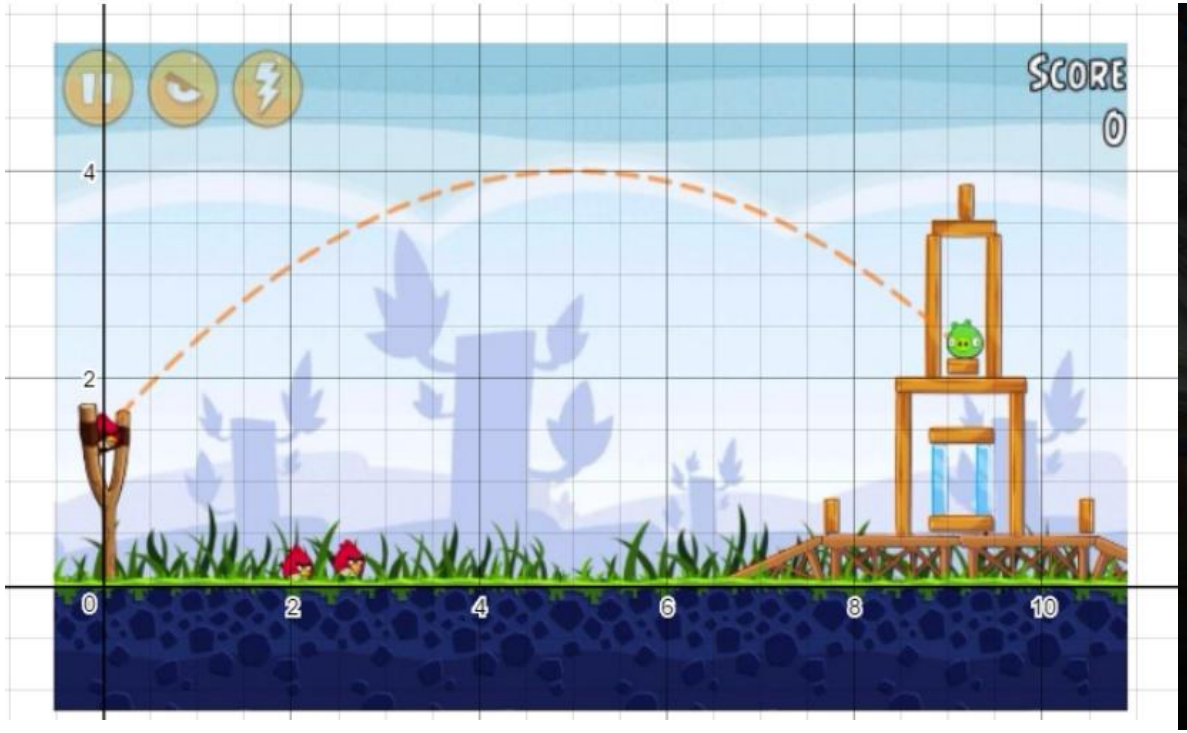


Activity 2 - Ballistics of an angry bird

Problem statement

In the angry bird game, the player has to launch birds at structures from a slingshot. The player gets to decide on an **initial angle theta (in degrees)** and **an initial speed for the bird alpha (in m/s)**.

After releasing, the angry bird goes flying into a parabolic curve, as shown in the figure below.



slido

Please download and install the Slido app on all computers you use

What is programming anyway? Use your own words!

ⓘ Start presenting to display the poll results on this slide.

Some key definitions

Definition (Programming): **Programming** refers to the process of designing and building an **executable computer program**, to accomplish specific computational tasks.

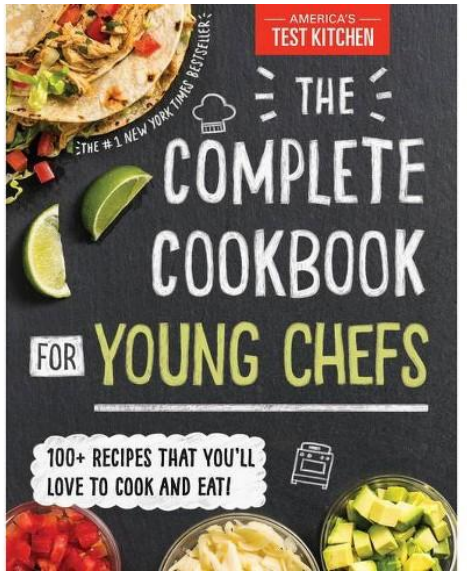
Layman definition (Programming): **Programming** consists of **defining a sequence of instructions (or algorithm)**, given by a human and that the computer must follow to accomplish a certain task.

Definition (Algorithm): An **algorithm** is a **finite sequence of instructions**, typically used to perform a computation task. When executed, on a provided set of **inputs**, it will proceed through a set of well-defined states and will eventually produce an **output**.

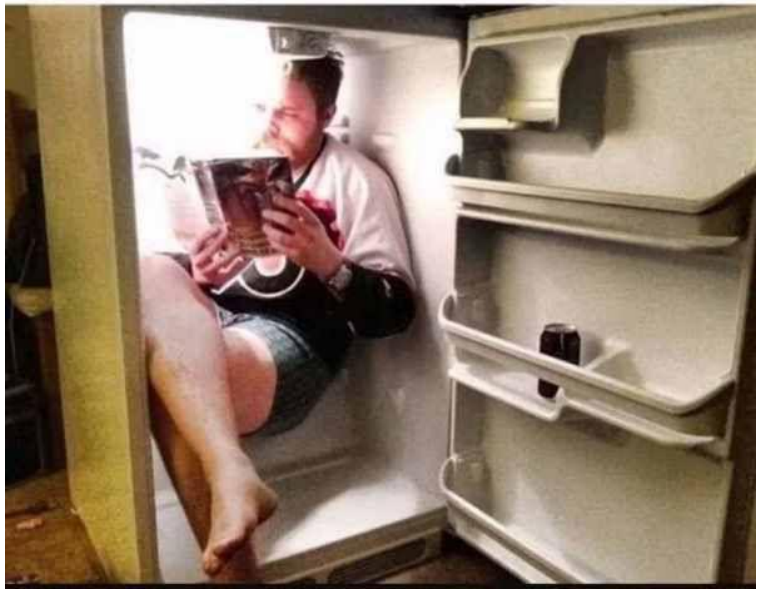
Programming and algorithms

Think of an **algorithm** as the sequence of **instructions** in a food **recipe** book!

- **Inputs** would be the **raw ingredients** you are using. **Outputs** would be the **dish produced** as a result.
- **Instructions** need to be clear and precise. *(No vague "Put in the oven" instruction without a clear temperature and exact duration being specified!)*
- **Instructions** are followed **sequentially**. *(No baking before mixing ingredients!)*

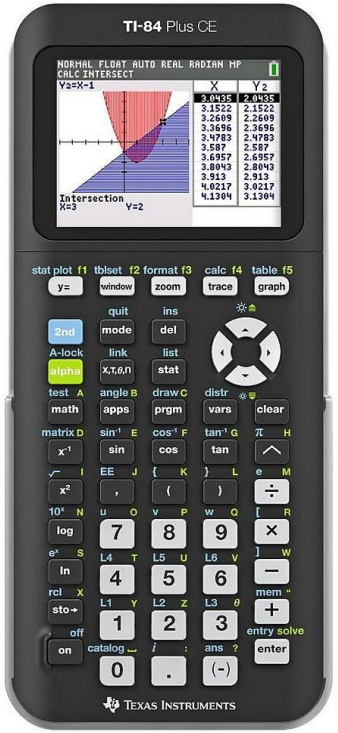


When you're cooking & the recipe says "chill in the fridge for one hour"



Programming and algorithms

Let us consider this example problem: Instructions to compute the area of a circle with radius 10.



Sequence of operations to follow

- Type 10,
- Type multiply key,
- Type 10,
- Press equal/enter key,
- Type multiply key,
- Type π,
- Press equal key again,
- What is eventually displayed is the area of the circle with radius 10.

Programming and algorithms

Let us consider this example problem: Instructions to compute the area of a circle with radius **10**.

Note: Here, the value **10** is an **input** for the algorithm.
We could replace occurrences of the value **10** with another value, e.g. **5**, and the algorithm would still produce the expected result as an **output**!

Sequence of operations to follow

- Type ~~10~~ 5,
- Type multiply key,
- Type ~~10~~ 5,
- Press equal/enter key,
- Type multiply key,
- Type π,
- Press equal key again,
- What is eventually displayed is the area of the circle with radius ~~10~~ 5.

Programming and algorithms

Note: Multiple algorithms could work for the same problem!

- Type π,
- Type multiply key,
- Type 5,
- Press equal/enter key,
- Type multiply key,
- Type 5,
- Press equal key again,
- What is eventually displayed is the area of the circle with radius 5.

- Type 5,
- Type multiply key,
- Type 5,
- Press equal/enter key,
- Type multiply key,
- Type π,
- Press equal key again,
- What is eventually displayed is the area of the circle with radius 5.

The life of a programmer

1. **Problem Analysis:** Identify a problem and understand its components.
2. **Algorithm Design:** Come up with an algorithm to solve said problem**.**
   - Step-by-step instructions,
   - To be followed sequentially,
   - With some expected inputs and outputs
3. **Coding:** Explain this algorithm to the computer, using a programming language.
4. **Testing:** Execute the program on a computer and see if it produces the expected results!

- Type 10,
- Type multiply key,
- Type 10,
- Press equal/enter key,
- Type multiply key,
- Type π,
- Press equal key again,
- You have reached your result.

ALGORITHM

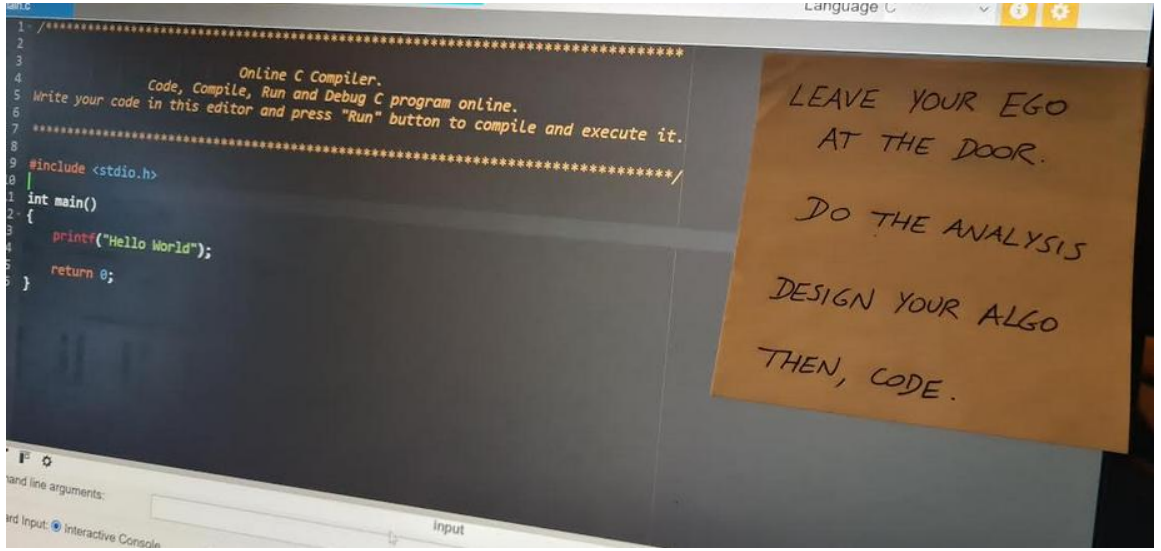
python™

A typical mistake

Typical mistake: Beginners will often skip these **first two steps** and jump straight into **coding**…
That is not the right way to go!

- Looking at your screen, trying to guess what to **code**? You probably need more time on **the first two steps.**
- After all, **coding** is simply explaining your logic/algorithm to the computer for execution later. Figuring out the right logic/algorithm is your human job!

1. **Problem Analysis:** Identify a problem and understand its components.
2. **Algorithm Design:** Come up with an algorithm to solve said problem.
   - Step-by-step instructions,
   - To be followed sequentially,
   - With some expected inputs and outputs
3. **Coding:** Explain this algorithm to the computer, using a programming language.
4. **Testing:** Execute the program on a computer and see if it produces the expected results!



A contract between CS students and teachers

Commandment #1: Learning Programming is Like Learning the Piano

- I understand that programming, like playing the piano, requires practice, patience, and perseverance.
- While theory is important, I must also actively “play the piano”, that is, practice coding. Only I can play the piano and learn by doing so.
- At first, it will be frustrating. My code may “sound bad” and fail in ways I do not yet understand, but I eventually will understand.
- And if I persist, putting in consistent effort, coding will gradually become more intuitive until it feels like second nature.

A contract between CS students and teachers

Commandment #2: I am teaching YOU, not ChatGPT!

- I commit to practicing coding myself rather than relying on tools like ChatGPT or other AIs to code in my place.
- While these tools can be great at providing guidance, I recognize that they can also solve all the beginner activities for me, without me doing any actual work.
- I understand that I cannot improve without doing the actual “piano playing”, or work, myself; and will therefore not over-rely on ChatGPT.

A contract between CS students and teachers

Commandment #3: Coding Can Be Frustrating — And That is Okay

- I accept that frustration is a natural part of the learning process.
- I will not let frustration deter me or faze me; instead, I will use it as a signal that I am challenging myself to learn something new.
- When faced with challenges, I will take a step back, analyze the problem, and persist with a positive mindset.
- I will see errors and bugs not as failures, but as opportunities to learn and indications on how to improve my code.
- I understand that debugging is an essential skill and an integral part of the programming process.

A contract between CS students and teachers

Commandment #4: Problem-Solving Starts on Paper

- I understand that before writing code, a programmer's job is to think through the problem.
- If I cannot explain, on paper, the logical steps to solve the problem, I will not be able to translate it into code.
- In any scenario where I end up looking blankly at my screen not knowing where to start my coding, I will simply close my laptop and think about the activity using pen and paper first.
- I will then embrace the use of pen and paper to organize my thoughts, draft plans, and clarify my approach.
- I will not try to randomly guess the correct code to use by trying things out, hoping they will work out great.

A contract between CS students and teachers

Commandment #5: We all have the same Learning Curve

- While I might ask friends for help, I will not blindly copy the code of my friends or compare my coding to theirs.
- Other students in the classroom might feel coding is easy, because the logic of coding feels natural to them and/or they have already done programming before.
- I might feel that coding is difficult and might find it frustrating that the other students do not struggle as much as I do.
- That is okay, we all have the same learning curve in the end, but not necessarily the same pace or the same starting point.

A contract between CS students and teachers

By signing this (clearly NOT legally binding) contract, I commit to taking an active role in my programming education, embracing challenges as opportunities to grow, and practicing regularly and fairly.

Date: _____

Signed,

_____ (Student Name)

Quick question…

Who is considered the inventor of computers and programming?

Hint: It is probably <u>NOT</u> one of these two famous persons.

(They created Windows and MacOS, which are Operating Systems a.k.a. OS, not computers)

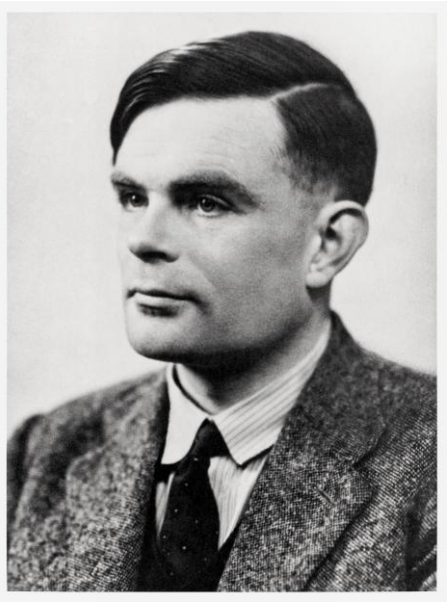
slido

Please download and install the Slido app on all computers you use

Who is considered the inventor of computers and programming?

ⓘ Start presenting to display the poll results on this slide.

The “father” of computer science



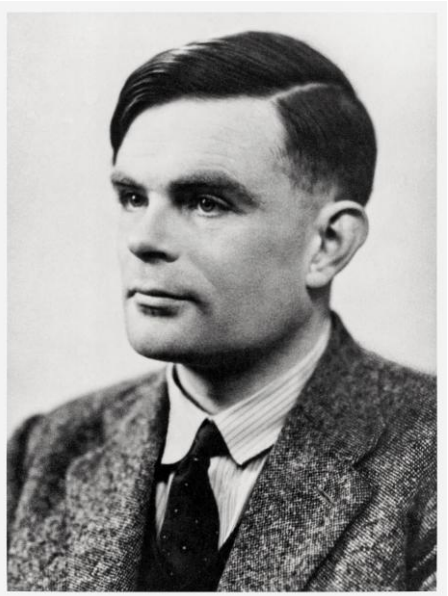
Alan Mathison Turing (1912–1954) was an English mathematician, computer scientist, logician, cryptanalyst, philosopher, and theoretical biologist.

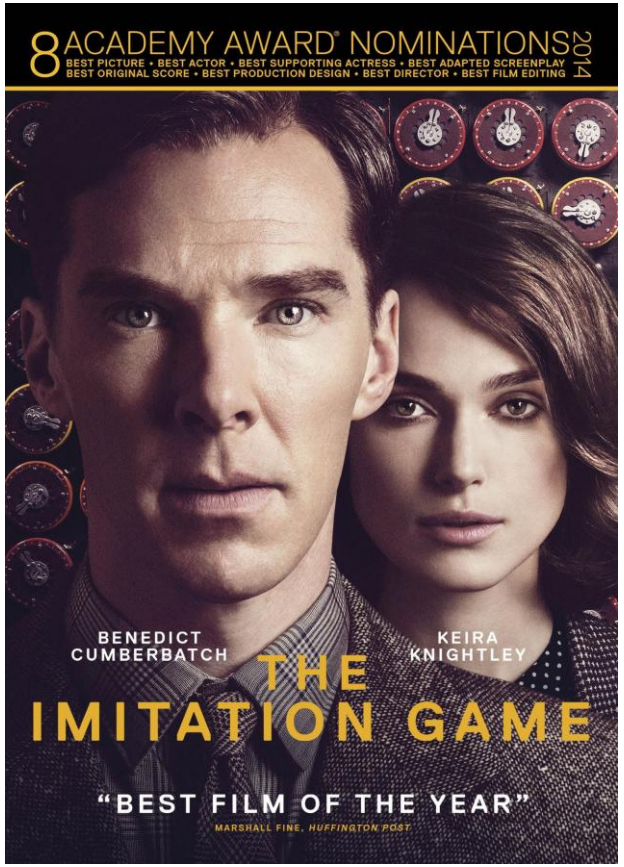
- Highly influential in the development of theoretical computer science and formalized algorithmic concepts.
- Created the first Turing machine, which can be considered the first general-purpose computer.

The “father” of computer science

During World War 2, he designed a “computer-like” machine, to decode encrypted transmissions coming from the German army (a.k.a. Enigma).

- Thanks to his decryption device, it has been estimated that the war in Europe was shortened by more than two years, saving over 14 million lives.
- Learn more about Alan Turing, by watching **The Imitation Game** movie. *(Your first homework?)*





About your computer

Your computer is good at doing two, and **only** two, things (nothing more!)

1. **Perform computational tasks (calculations),** as described by an algorithm.
2. **Remember the results of these computational tasks**, by storing them in its internal memory (and eventually retrieving these results, by accessing its memory).

And that is it. And that is all you need.

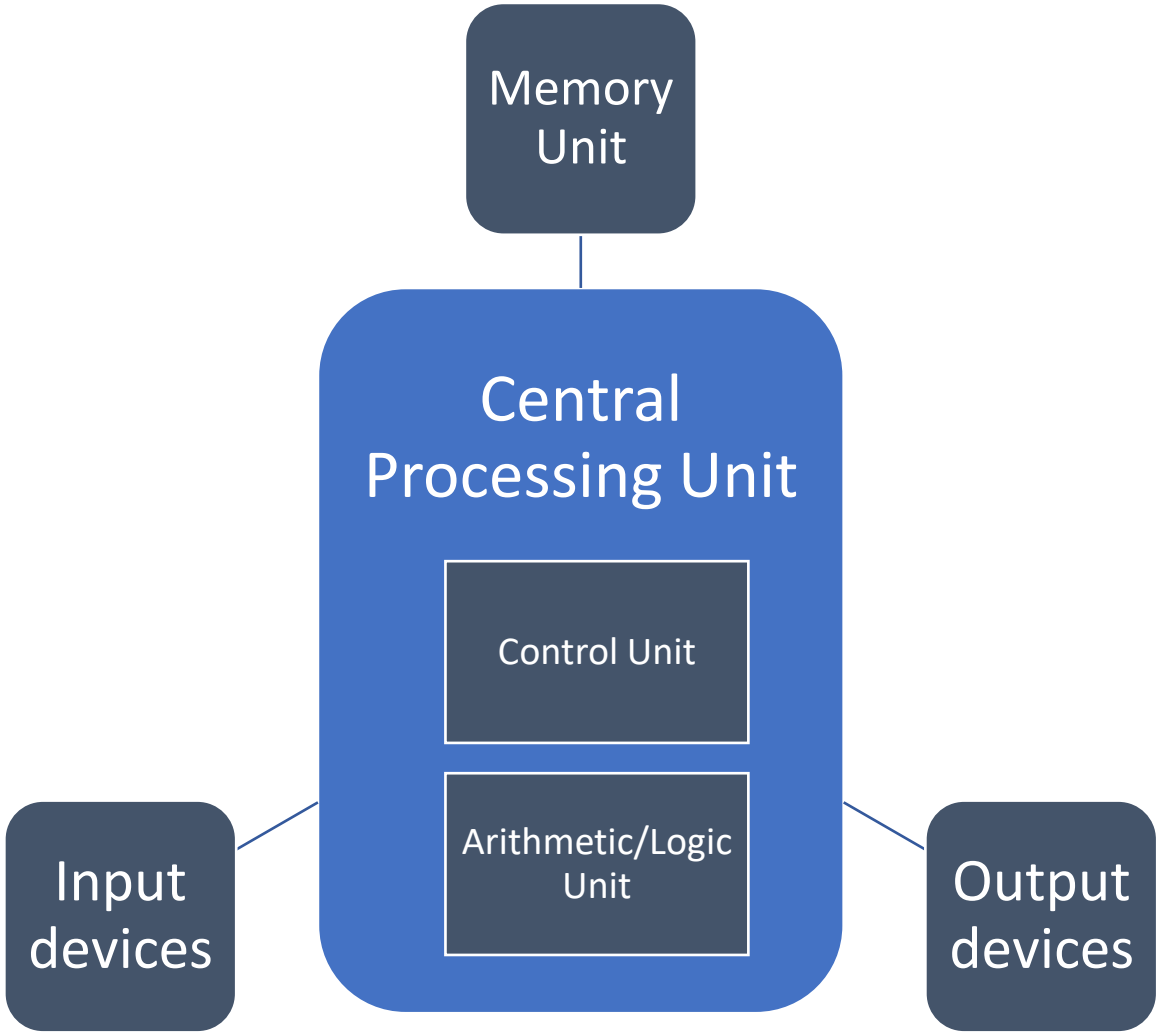
Because all operations performed by computers can be broken down into combinations of both.

The Von Neumann architecture

Definition (the Von Neumann architecture of a computer):

The **Von Neumann architecture** describes one of the first architectures for a computer.

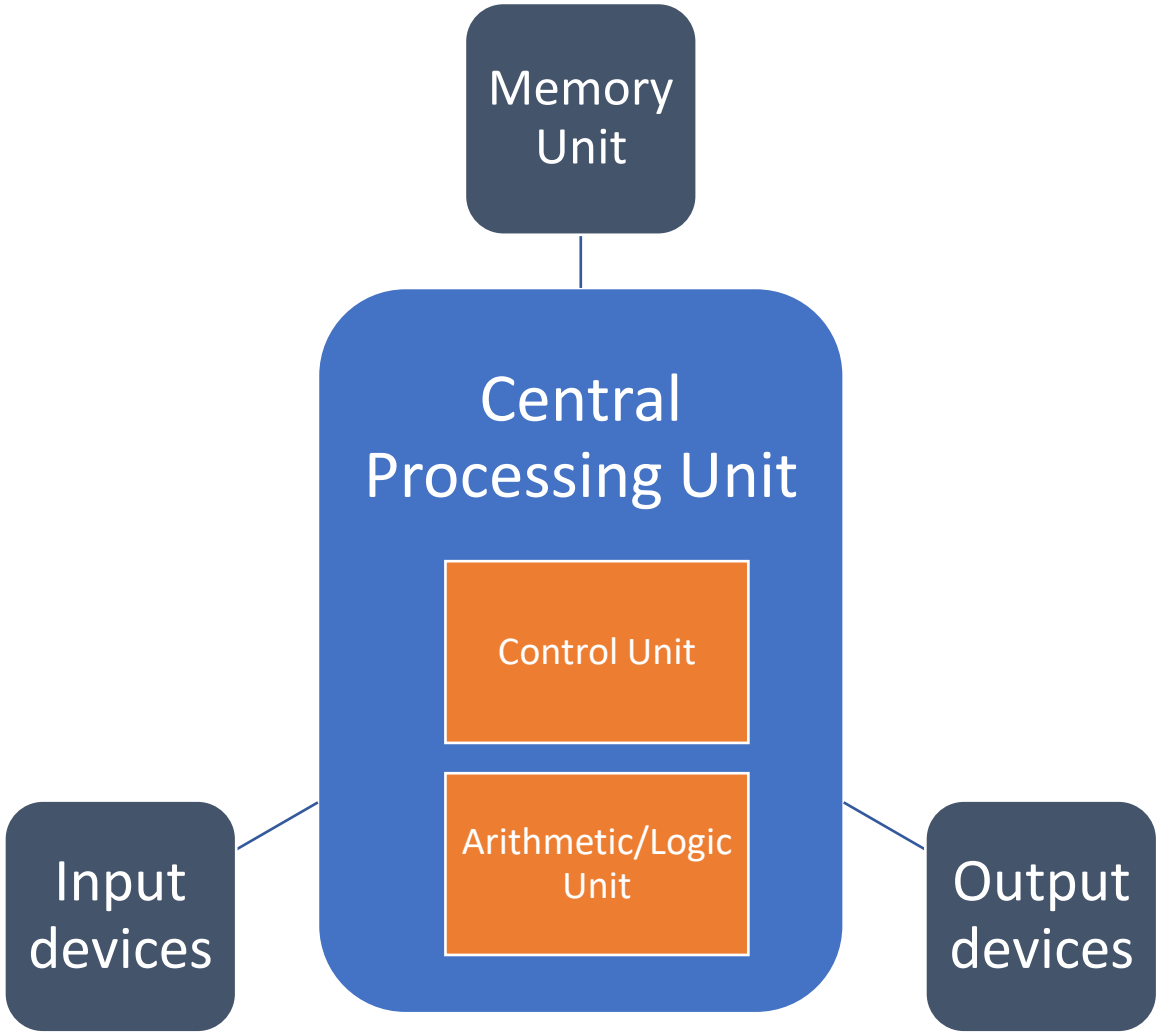
It consists of several elements that every computer possesses, and each serves a specific purpose.



The Von Neumann architecture

It first consists of a **Central Processing Unit (CPU)**, which itself consists of…

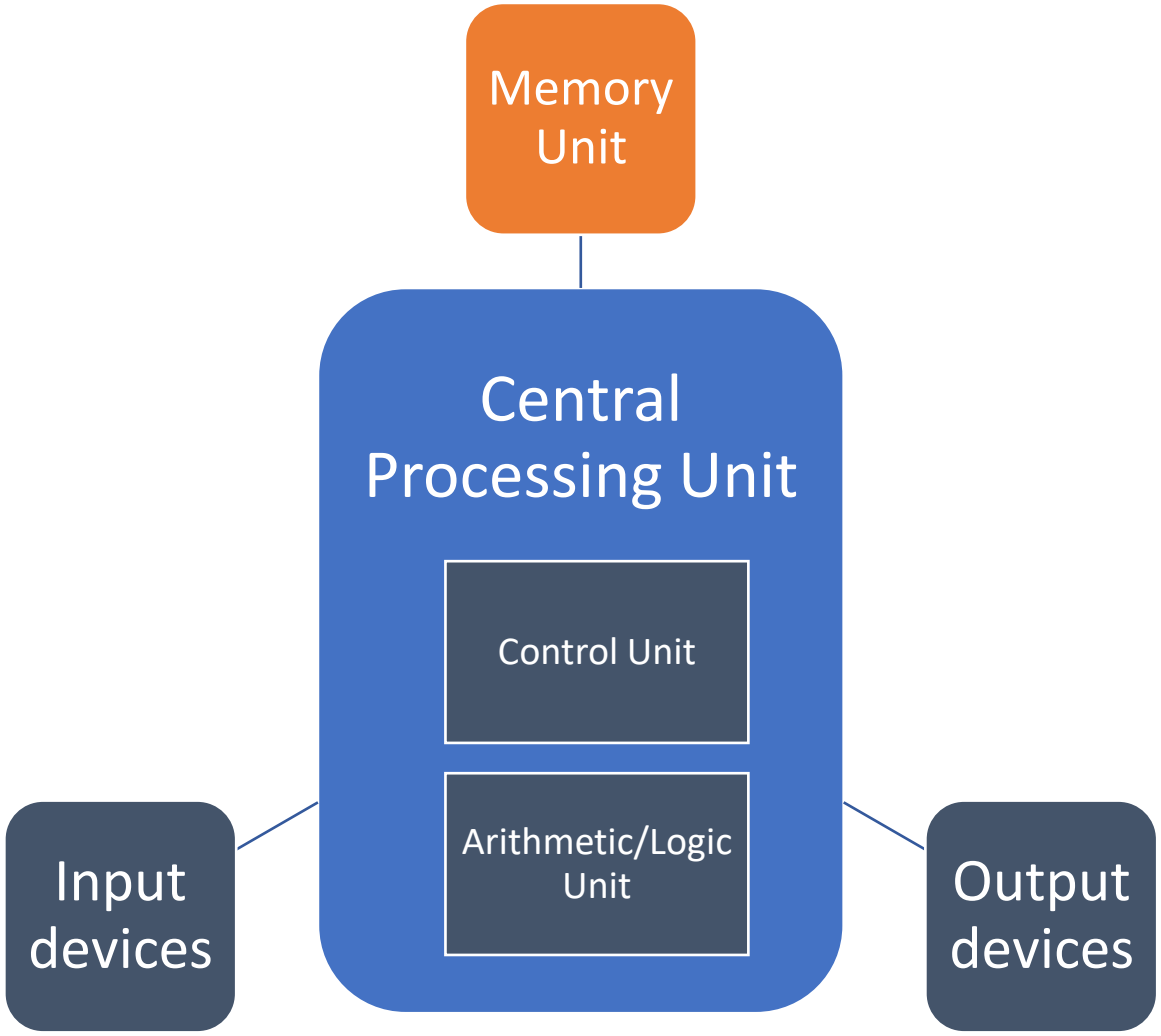
- An **Arithmetic/Logic Unit** (in charge of dealing with instructions, typically math operations, in binary 0/1),
- And **a Control Unit** (in charge of the hardware and communication between different hardware elements)



The Von Neumann architecture

It also contains…

- A **Memory Unit** (for storing and retrieving results from previous computational tasks, using binary formatting 0/1),

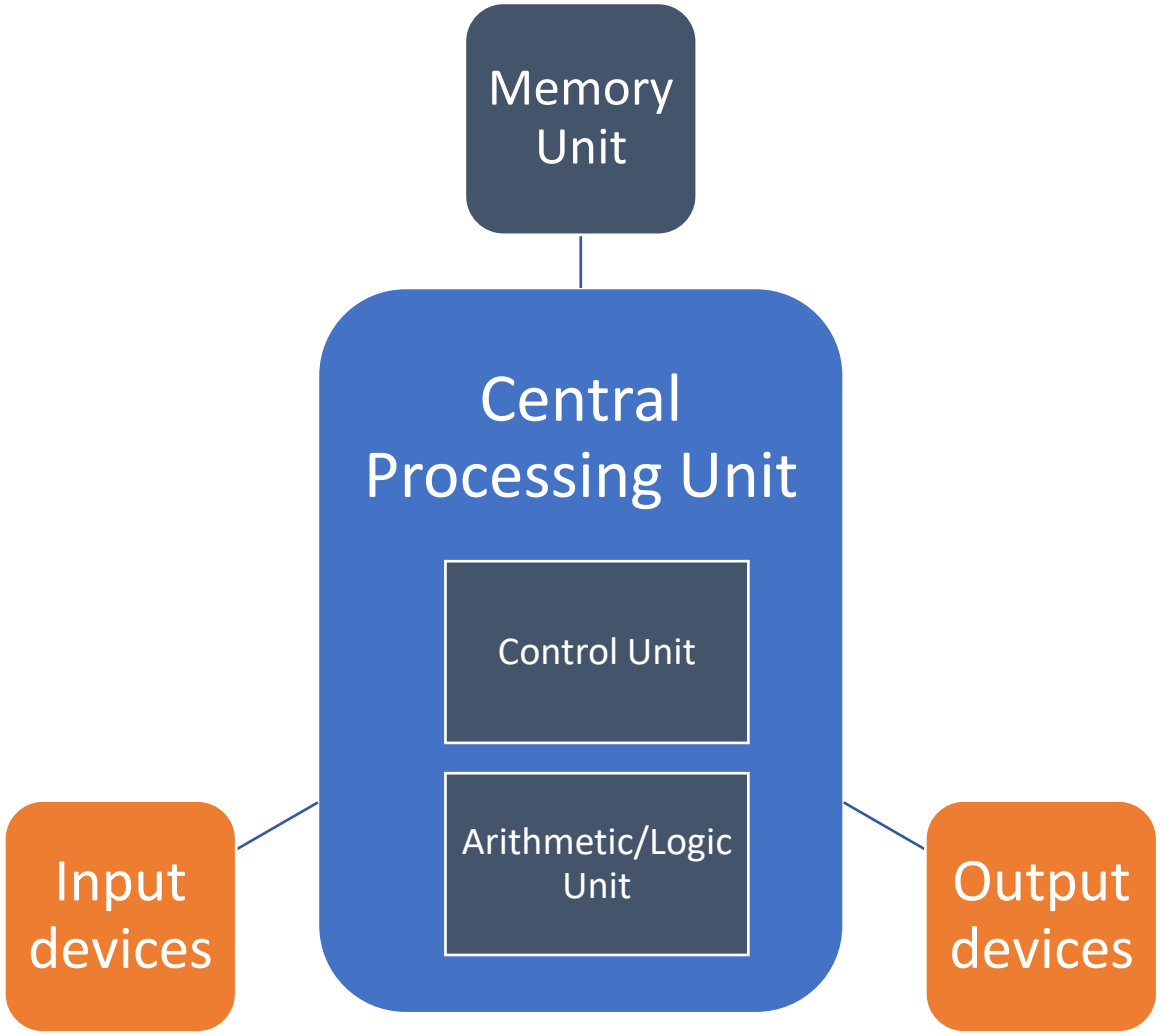


The Von Neumann architecture

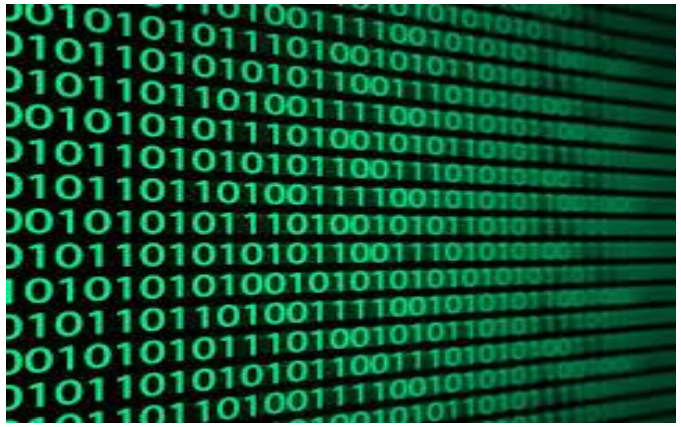
It also contains…

- A **Memory Unit** (for storing and retrieving results from previous computational tasks, using binary formatting 0/1),
- **Inputs and Outputs Devices** (e.g. mouse, keyboard, screen, microphone, webcam, etc.).

You do not need to know more about the hardware for now!

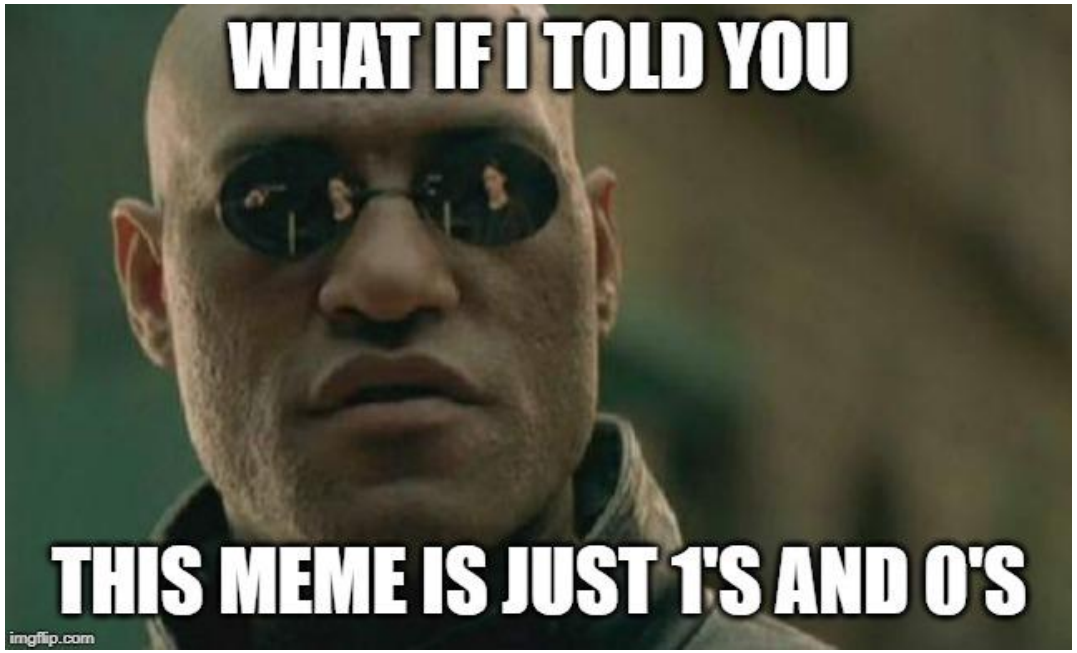


The problem of dealing with binary and the need for programming languages



Observation: In a computer, and due to hardware reasons, all operations (that is the computational and memory tasks) and are performed in **binary**.

- **Problem:** Binary is heavy and difficult to read for humans.
- **Example:** “Matthieu”, in binary, is “01001101 01100001 01110100 01110100 01101000 01101001 01100101 01110101“.



The problem of dealing with binary and the need for programming languages

Problem: How can human conveniently discuss algorithms with a computer that is designed to work only with binary?

→ **Solution:** find an intermediate language, readable by humans, to address the computer.

- **High-level language:** closer to human language (easy to learn)
- **Low-level language:** closer to binary (faster, but difficult to learn)

Generations	Languages	Characteristics
First-generation languages (1954 – 1958)	FORTRAN I, ALGOL 58, Flowmatic, IPL V	Mainly used for mathematical calculations; consists only of global data and sub-programs.
Second-generation languages (1959 – 1961)	FORTRAN II, ALGOL 60, COBOL, Lisp	Use extended to business applications; artificial intelligence; subroutines, block structure, data types introduced.
Third-generation languages (1962 – 1970)	PL/1, ALGOL 68, Pascal, Simula	Use extended to wider applications; ideas of modules and data abstraction introduced.
The generation gap (1970 – 1980)	C, FORTRAN 77	Many languages invented with few surviving; small executables, thrust towards standardization.
Enhanced popularity of object-orientated languages (1980 – 1990)	Smalltalk 80, C++, Ada83, Eiffel	Languages derived from previous ones; the idea of a class as a basic unit of abstraction.
Emergence of frameworks (1990 – present)	Visual Basic, Java, Python, J2EE, .NET, Visual C++, Visual Basic .NET	Widespread use of integrated development environments (IDE); focus on Web-based systems.

The problem of dealing with binary and the need for programming languages

Problem: How can human conveniently discuss algorithms with a computer that is designed to work only with binary?

→ **Solution:** find an intermediate language, readable by humans, to address the computer.

- **High-level language:** closer to human language (easy to learn)
- **Low-level language:** closer to binary (faster, but difficult to learn)

Generations	Languages	Characteristics
First-generation languages (1954 – 1958)	FORTRAN I, ALGOL 58, Flowmatic, IPL V	Mainly used for mathematical calculations; consists only of global data and sub-programs.
Second-generation languages (1959 – 1961)	FORTRAN II, ALGOL 60, COBOL, Lisp	Use extended to business applications; artificial intelligence; subroutines, block structure, data types introduced.
Third-generation languages (1962 – 1970)	PL/1, ALGOL 68, Pascal, Simula	Use extended to wider applications; ideas of modules and data abstraction introduced.
The generation gap (1970 – 1980)	C, FORTRAN 77	Many languages invented with few surviving; small executables, thrust towards standardization.
Enhanced popularity of object-orientated languages (1980 – 1990)	Smalltalk 80, C+ iffel	Languages derived from previous ones; the idea of a class as a basic unit of abstraction.
Emergence of frameworks (1990 – present)	Visual Basic, Java, Python, J2EE, .NET, Visual C++, Visual Basic .NET	Widespread use of integrated development environments (IDE); focus on Web-based systems.

What is the Python programming language?

Definition (Python):
Python is an **interpreted, high-level, general-purpose programming language.**

- Created by **Guido van Rossum** and first released in **1991**.
- Currently **v3**, since **2008** (v4 soon?!).
- Python's design philosophy emphasizes code **readability**. Its language constructs aim to help programmers write simple and clear, logical code for their projects.

Why learn Python?

- **High-level language:** easy to write and read, and therefore well-suited for beginners.
- **Wide variety of packages:** can be used for multiple purposes (computer software, phone apps, video games, web, etc.)
- **Dynamic typing:** in layman terms, Python is able to manage the data saved to memory, in an automated and efficient fashion, without human intervention.

- **Python is the #1 language for data science and AI at the moment:** several AI frameworks such as Tensorflow (Google AI) and Pytorch (widely used in academic research on AI), etc.

Why learn Python?

- Overall, Python is a good **entry point for beginners** in both programming and computer science, and therefore the first language we decided to teach.
- **Also, high interoperability with other languages:** Jumping to or including another programming language (Java, C, SQL, etc.) is easy, once you know Python.

Installing Python

Install it now!

- In this class, we will use **Python 3.13.7** (latest stable version available as of the 10th of Aug 2025).
- Download it here (for 32/64-bit Windows and Mac users, Linux users can get it via apt-get): https://www.python.org/downloads/

Installing Python

Important note: Check your installed programs for existing versions of Python already installed on your machines. Not a good idea to install multiple v3 versions of Python on the same machine!

Exception for (old) MacOS users: Python v2 might be installed as it is required for some components of MacOS. Do not uninstall it, and just install Python v3, keeping the v2 on the side as it is.

The Console/Terminal/Shell/Bash

Definition (Console, Terminal, Shell, or Bash):

The **console** (or terminal or shell or bash) is a text prompt expecting commands, which can be used to communicate with your computer.

- *(And in the early days of computers, it was the only way to communicate with them!)*

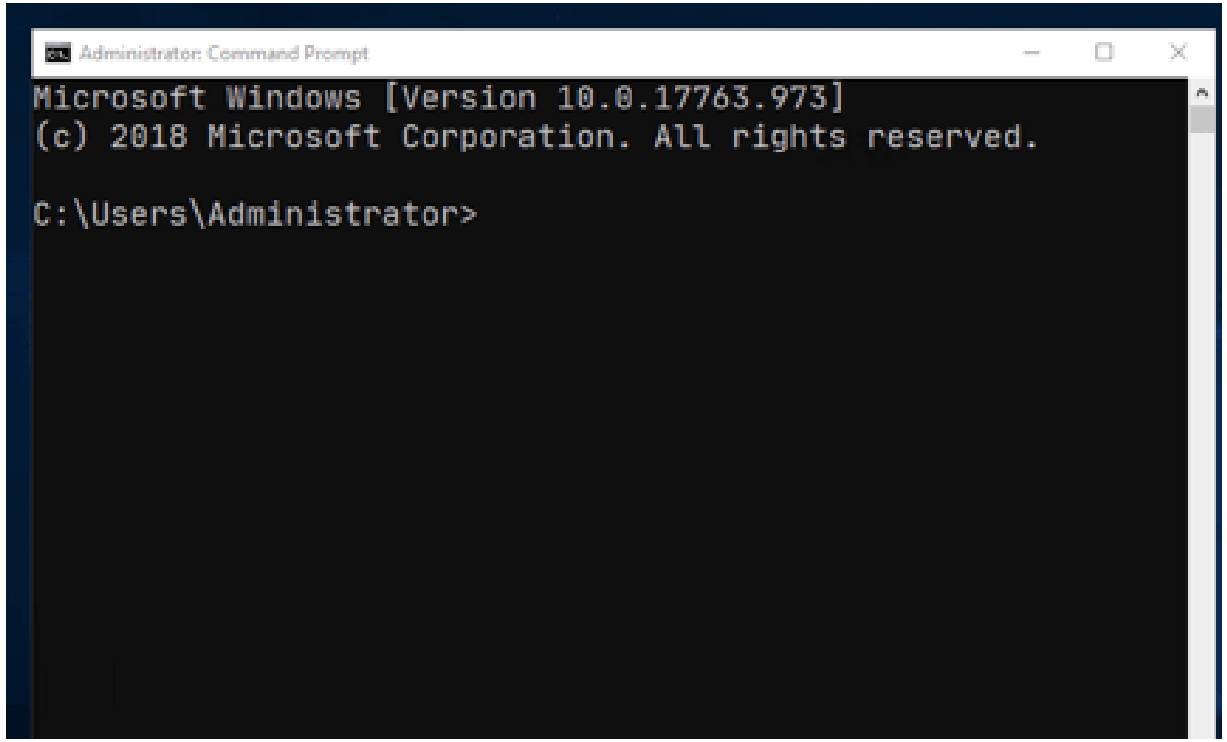
The Console/Terminal/Shell/Bash

Definition (Console, Terminal, Shell, or Bash):

The **console** (or terminal or shell or bash) is a text prompt expecting commands, which can be used to communicate with your computer.

- **Nowadays, your Operating System (OS) provides a user-friendly way to communicate with your computer and "hides" the need for a console.**

- **But consoles are still there, hidden somewhere in your computers anyway!**



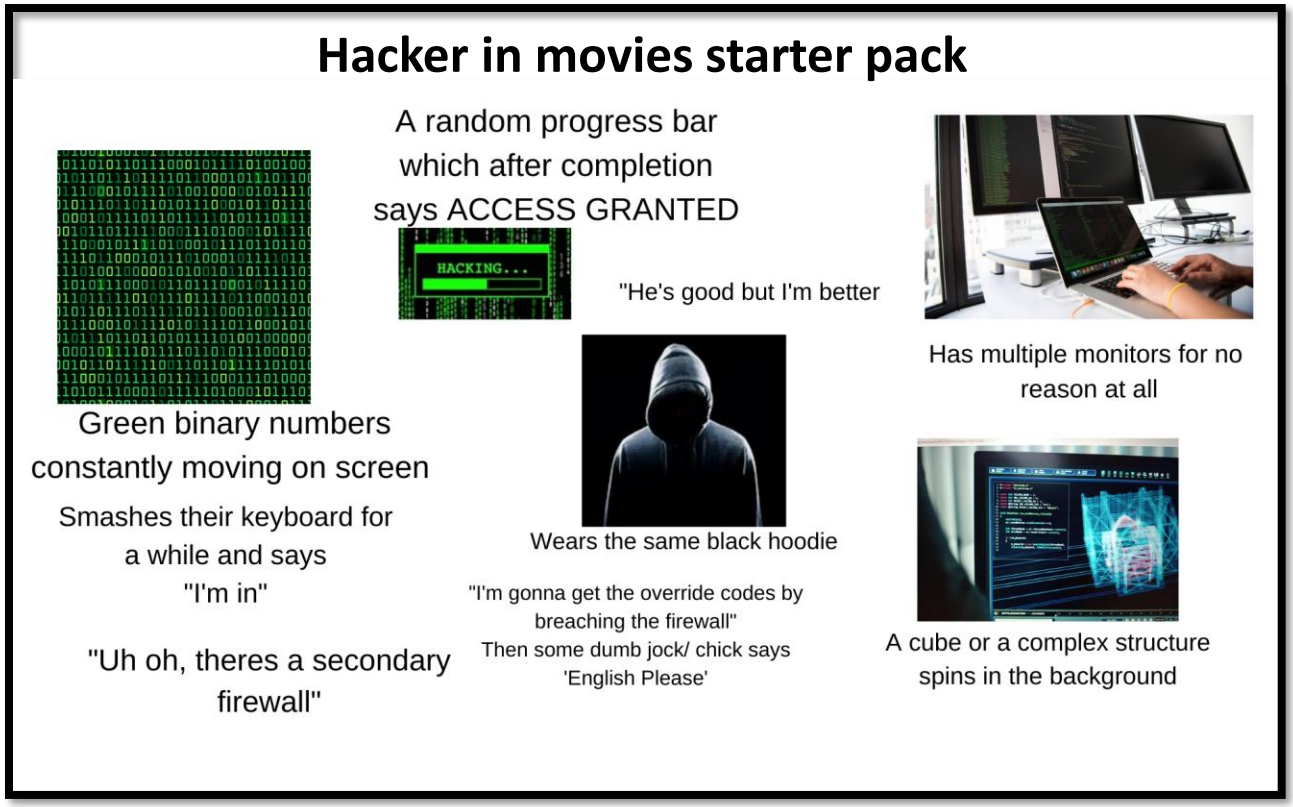
The Console/Terminal/Shell/Bash

Definition (Console, Terminal, Shell, or Bash):

The **console** (or terminal or shell or bash) is a text prompt expecting commands, which can be used to communicate with your computer.

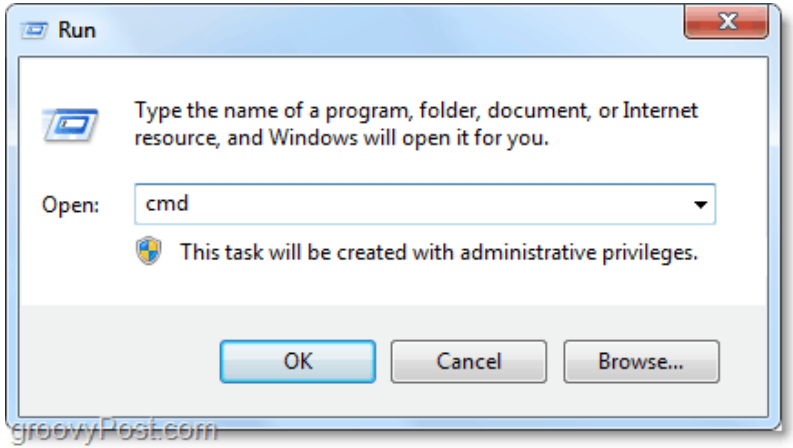
- **Nowadays, your Operating System (OS) provides a user-friendly way to communicate with your computer and "hides" the need for a console.**

(Typically, you see consoles all the time in computer/hacker movies.)



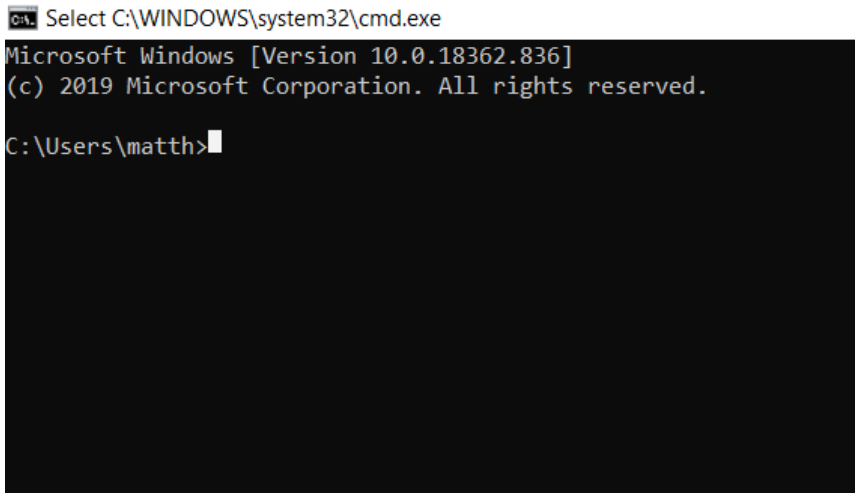
Opening a console (Windows)

Check your installation has completed appropriately, by trying to open a **console**.



Windows: the simple way.

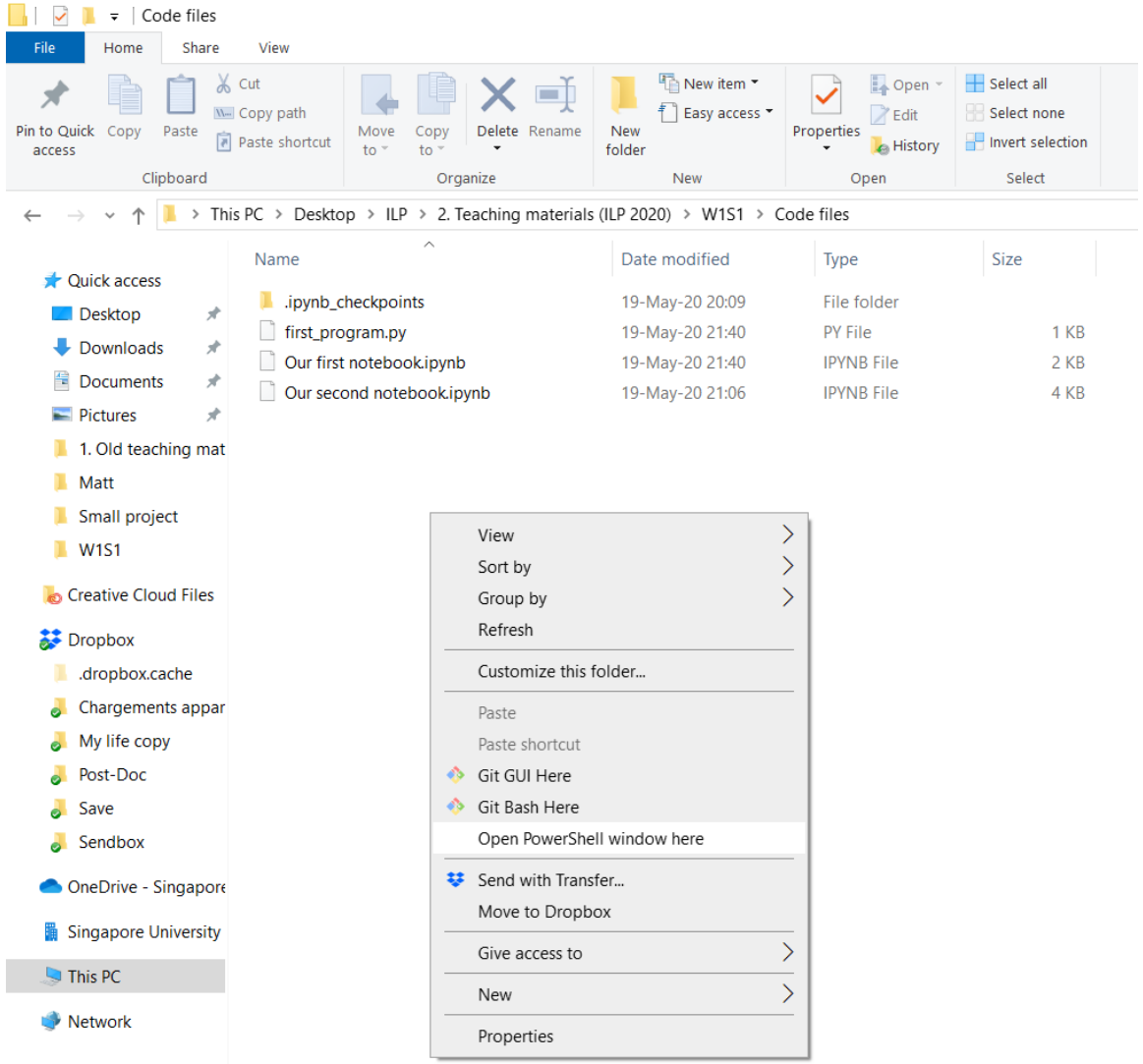
1. Press simultaneously **Windows key + R**,
2. then type **cmd (which is short for command console)**, and press **Enter.**



Opening a console (Windows – option 2)

Windows option 2: Faster and more convenient way.

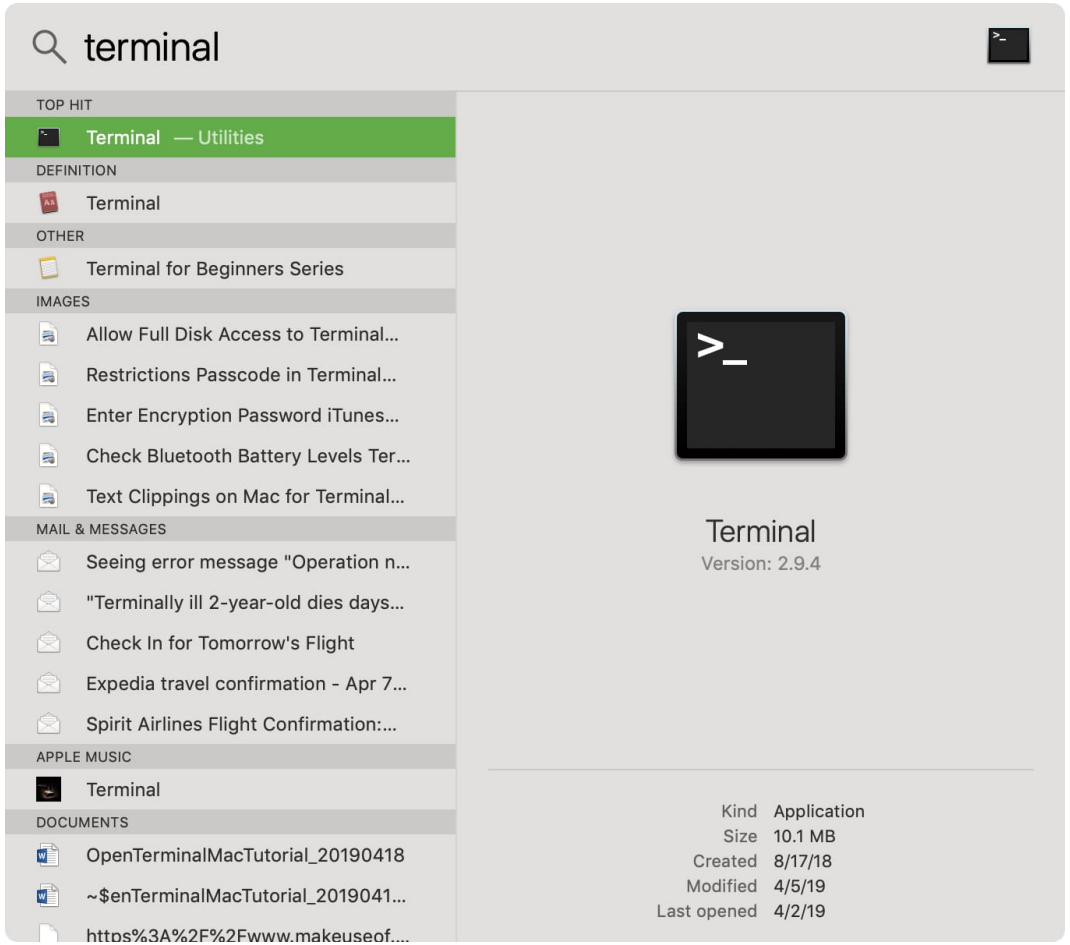
1. **Open an explorer window/file manager window** in the folder you attempt to work in.
2. **Hold shift** and **right-click** in an empty space of the explorer window.
3. **Choose "Open Powershell window here".**



Opening a console (Mac OS)

Mac: roughly the same procedure as Windows.

1. Press simultaneously **Command key + Space**,
2. Then type **Terminal**,
3. It should appear as your top result, click it.



Opening a console (Mac OS)

Mac option 2: roughly the same procedure as Windows.

1. Open a Finder window and navigate to the folder you want to work in.
2. Right-click (or Control-click) in an empty space within the folder.
3. Choose "New Terminal at Folder" from the context menu.

Debug: If you do not see the "New Terminal at Folder" option:

1. Open the “System Preferences”. Go to "Keyboard“, then the "Shortcuts" tab.
2. Select “Services” from the list on the left. Look for the “Files and Folders” section.
3. Ensure that the checkbox next to “New Terminal at Folder” is checked.

Opening a console (Mac OS)

Mac option 2: roughly the same procedure as Windows.

1. Open a Finder window and navigate to the folder you want to work in.
2. Right-click (or Control-click) in an empty space within the folder.
3. Choose "New Terminal at Folder" from the context menu.

Mac Option 3: There is also, on recent version of MacOS, the possibility of dragging and dropping a folder onto the terminal icon to open a terminal in the location of the given folder.

This might, however, only work on recent version of MacOS.

Either way, try it out, feel free to google it to figure out something that works for you!

OS mode vs. Python mode

- After installing Python, you may use the console to type Windows/Mac/OS commands (a.k.a. **OS mode**).
- Typically, commands for creating, copying, pasting, moving through folders/files, etc.

- You may also use a command to start a Python environment.
- While in the Python environment or **Python mode**, your console expects Python-type commands (not OS ones!).

→ How do you know which mode you are in?

(We will show how it is done on Windows, but roughly similar in MacOS/Linux.)

OS mode vs. Python mode

- After installing Python, you may use the console to type Windows/Mac/OS commands (a.k.a. **OS mode**).
- Typically, commands for creating, copying, pasting, moving through folders/files, etc.

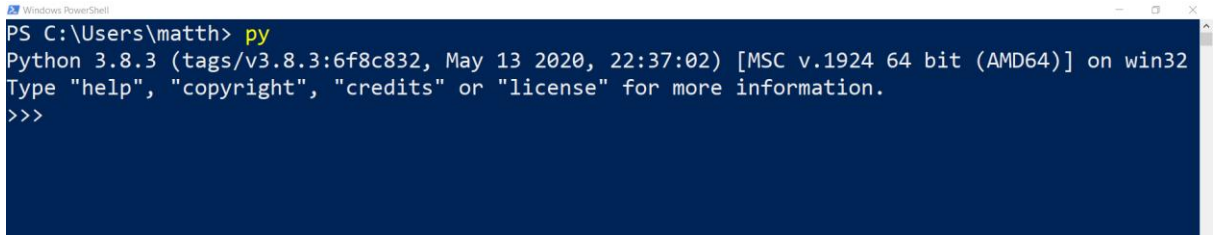
- You may also use a command to start a Python environment.
- While in the Python environment or **Python mode**, your console expects Python-type commands (not OS ones!).

Some commands to use in OS mode

Out of scope, but feel free to try some of these OS commands for fun!

- mkdir xyz: creates a folder with name “xyz”, at the current location of your terminal.
- del xyz: deletes a folder or file with name “xyz”, at the current location of your terminal.
- cd xyz: move the terminal from its current location to the subfolder with name “xyz”, assuming this subfolder is in the current location of the terminal.
- Many more OS commands for you to explore: https://devblogs.microsoft.com/scripting/table-of-basic-powershell-commands/

Your first run of Python!



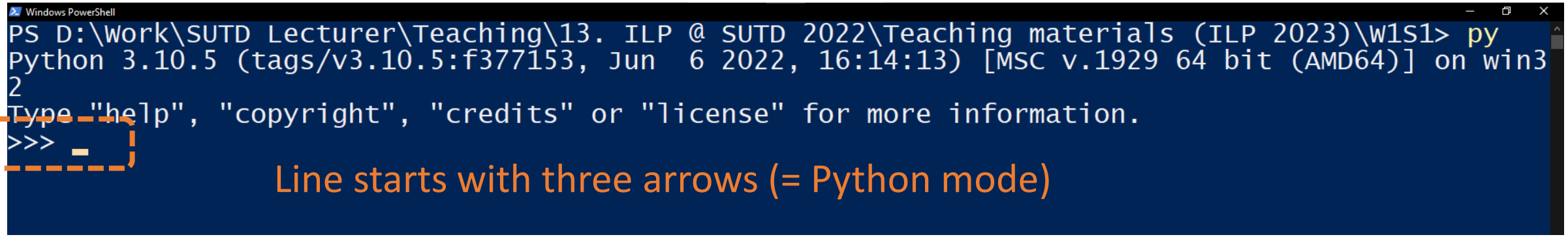
Start Python by **typing one** of the following commands in the console, while in OS mode and press **Enter!** *(Note: one of these should work, might vary depending on your machine!).*

- **py (most frequent one, works in my case)**
- py3
- python
- python3

OS mode vs. Python mode

- After installing Python, you may use the console to type Windows/Mac/OS commands (a.k.a. **OS mode**).
- Typically, commands for creating, copying, pasting, moving through folders/files, etc.

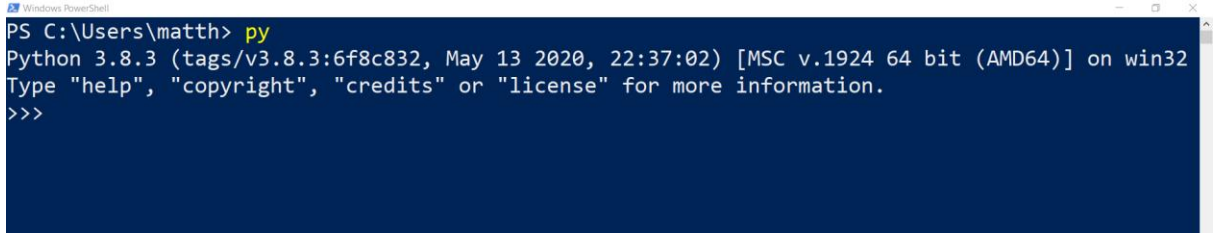
- You may also use a command to start a Python environment.
- While in the Python environment or **Python mode**, your console expects Python-type commands (not OS ones!).



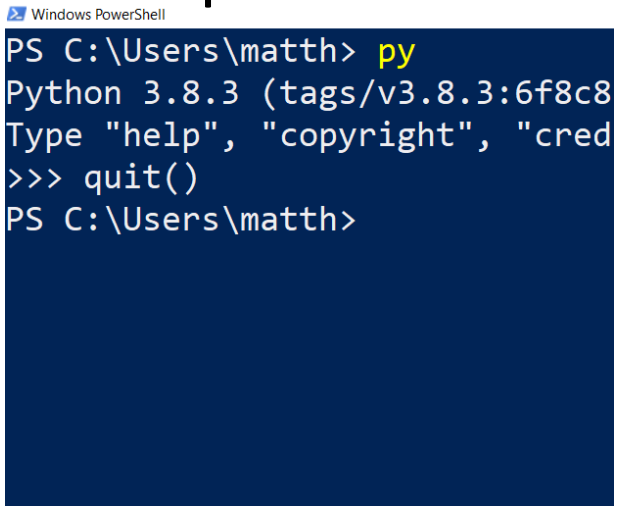
Your first run of Python!

Start Python by **typing one** of the following commands in the console, while in OS mode and press **Enter!** *(Note: one of these should work, might vary depending on your machine!).*

- **py (most frequent one, works in my case)**
- py3
- python
- python3



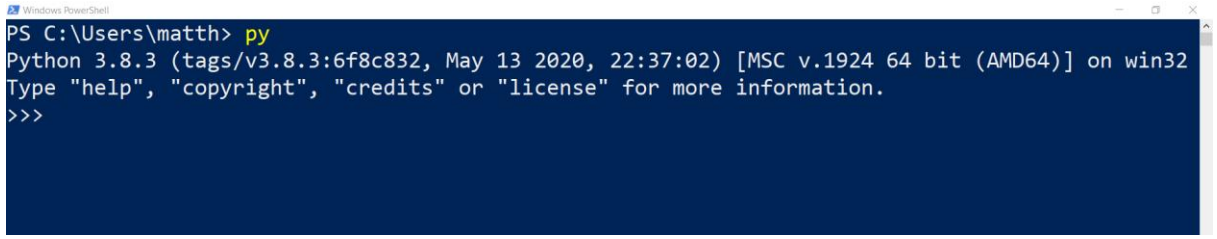
To exit the Python environment and go back to your OS console, simply type **quit()** while in Python mode and press enter.



Your first run of Python!

Start Python by **typing one** of the following commands in the console, while in OS mode and press **Enter!** *(Note: one of these should work, might vary depending on your machine!).*

- **py (most frequent one, works in my case)**
- py3
- python
- python3



Important note: Make sure this command started the correct version of Python (should be v3.X.something). Especially for MacOS users with a v2 and v3 installed!

If it starts v2, try another keyword until you find the right one to use!

Installing & updating packages in Python

- Next, let us install **packages**. (**Packages** are extra functionalities for Python.)
- To do so, run the command in **red**, below, in your console, while in the OS mode.

Note: if you are using the command **py3, python or python3** instead of **py**, adjust accordingly!

```
PS C:\Users\matth> py -m pip install numpy notebook matplotlib
```

It shall run for a while, download and install a few libraries or Python… (numpy for advanced math computation matplotlib for displays, and notebook, which we will use later on)

py –m pip install numpy notebook matplotlib

Our first program!

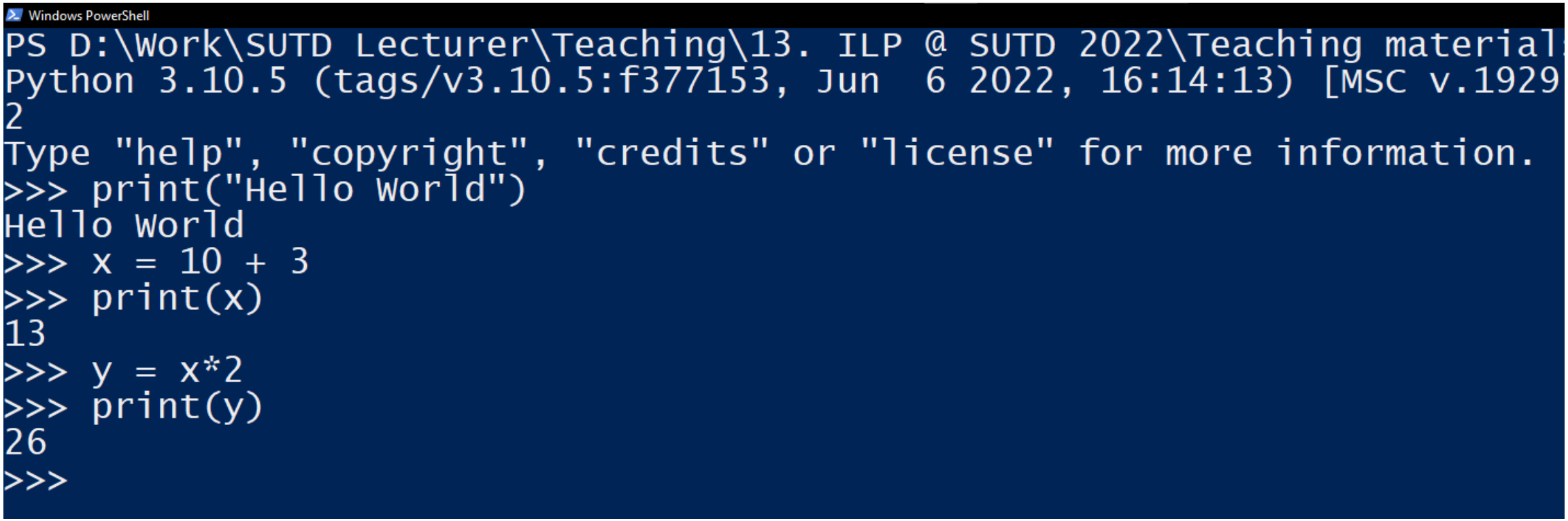
Definition (the “Hello World” program): The **“Hello World” program** is a computer program that outputs or displays the message "Hello World!". It is often used as a **sanity test** to make sure that a computer language is correctly installed.

- Start Python in a console, type **print(“Hello World!”)**, and submit by pressing Enter. It should display “Hello World!”.

When your code outputs "Hello World!"

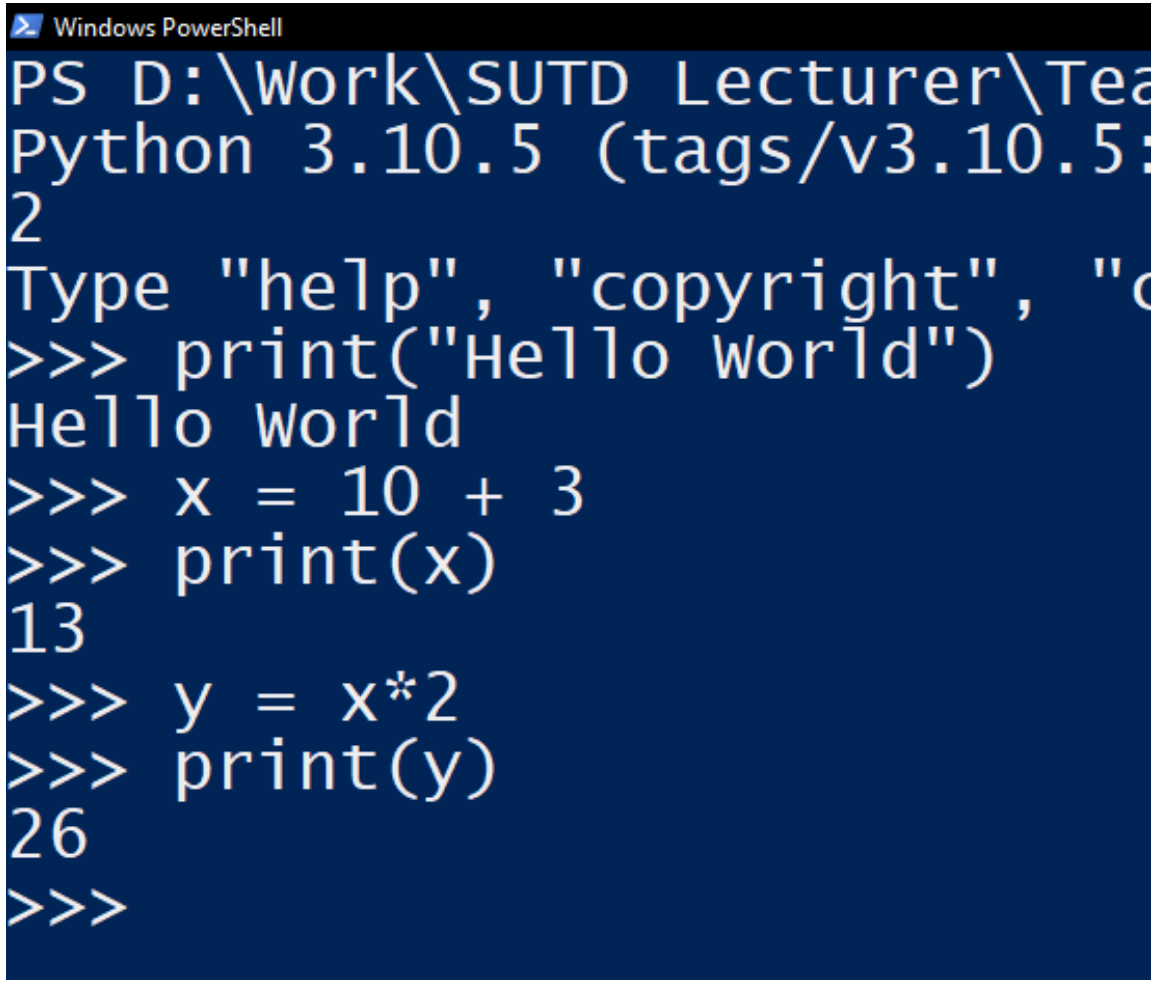
Our first program!

- Assigning something to memory is done with the = sign.



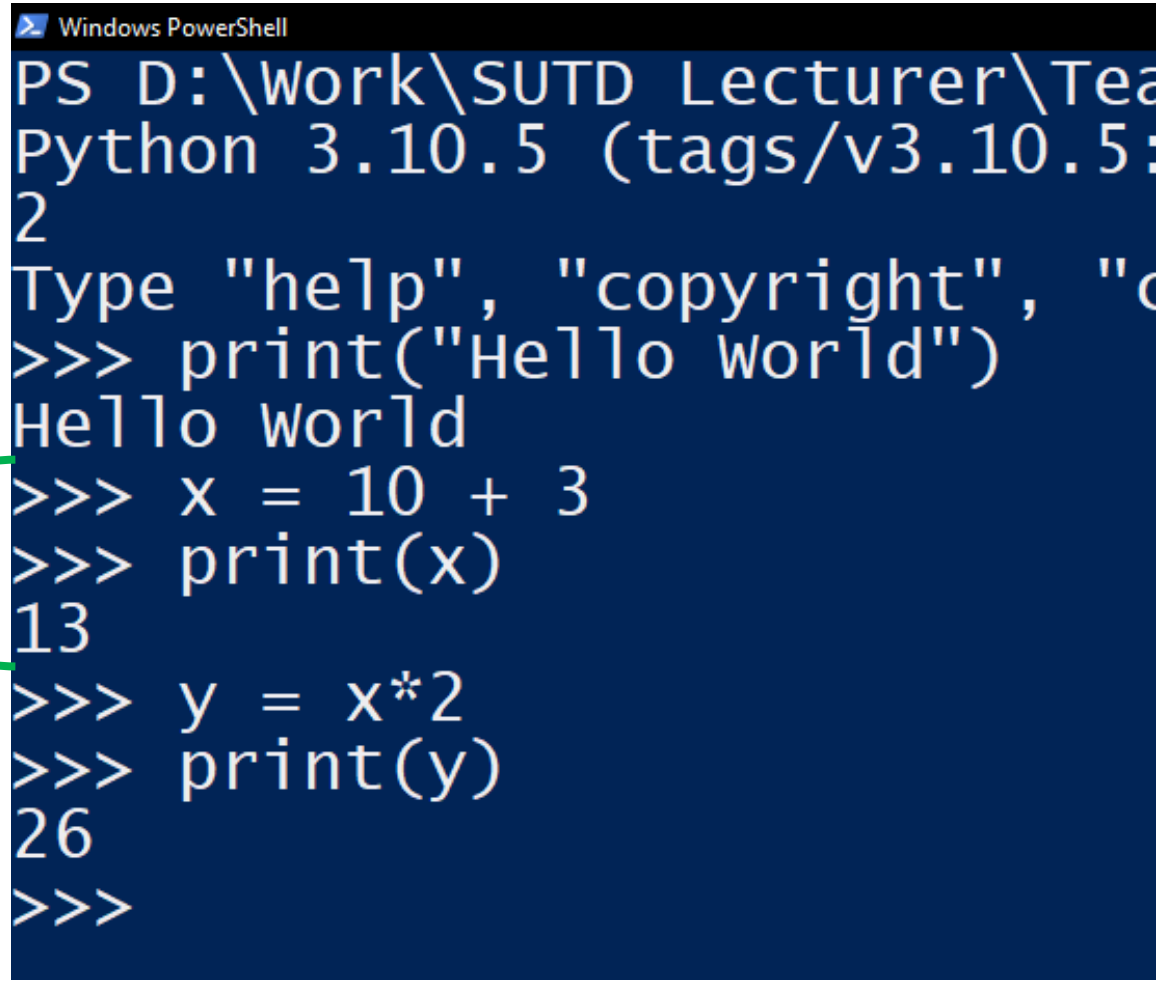
Our first program!

- Assigning something to memory is done with the = sign.
- (**Note:** The = sign does not have the same meaning as in mathematics.)
- In computer science it means:
  - Assign what is on the right-hand side of the equal sign to memory.
  - The name of this element, called a **variable**, consists of the text located on the left-hand side of the equal sign.



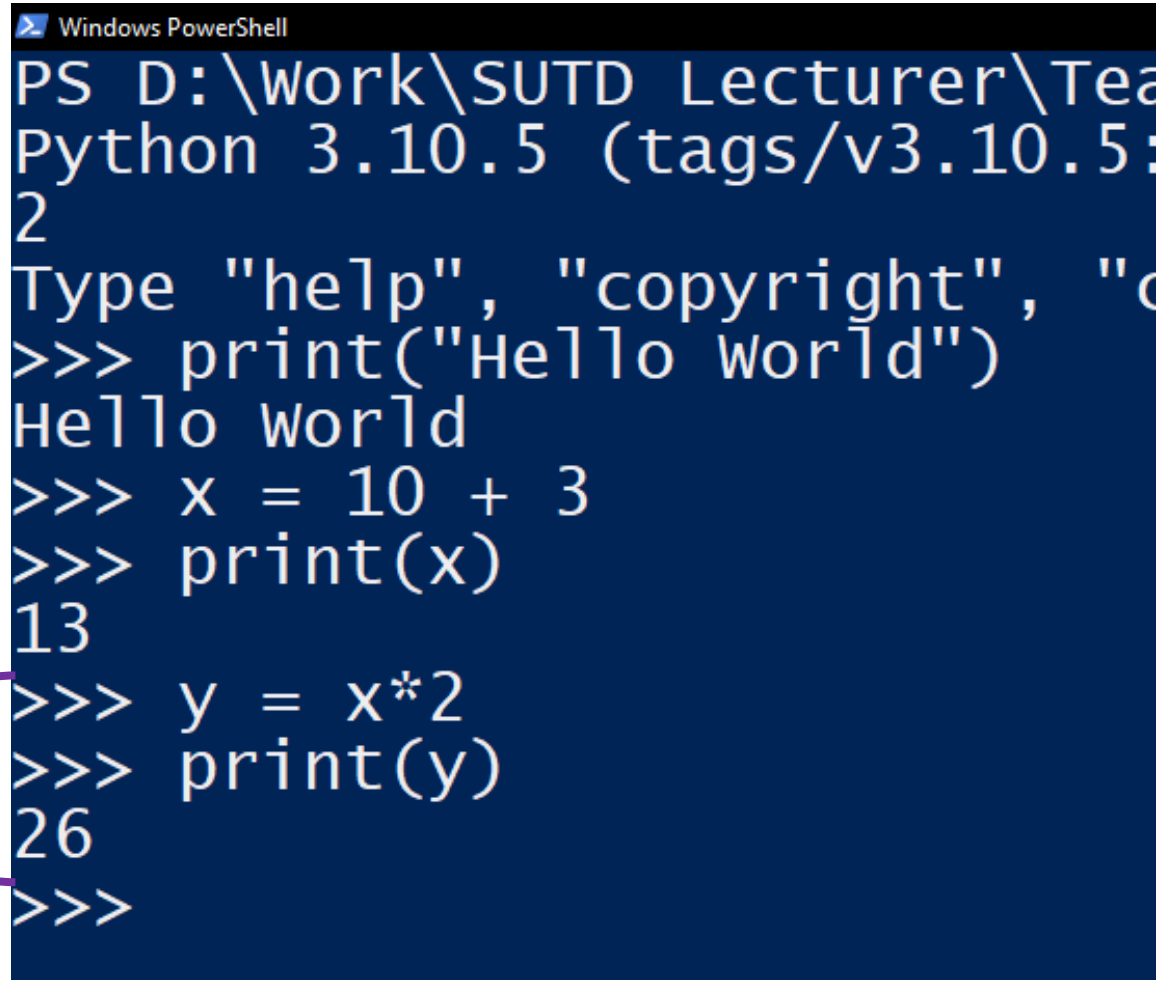
Our first program!

- Here, we have assigned the numerical value resulting from the operation $\mathbf{10 + 3}$, to a **variable** named **x**.
- Later on, we can retrieve the value stored in the variable, and – for instance – ask the computer to display it on screen for us, with the **print()** function.



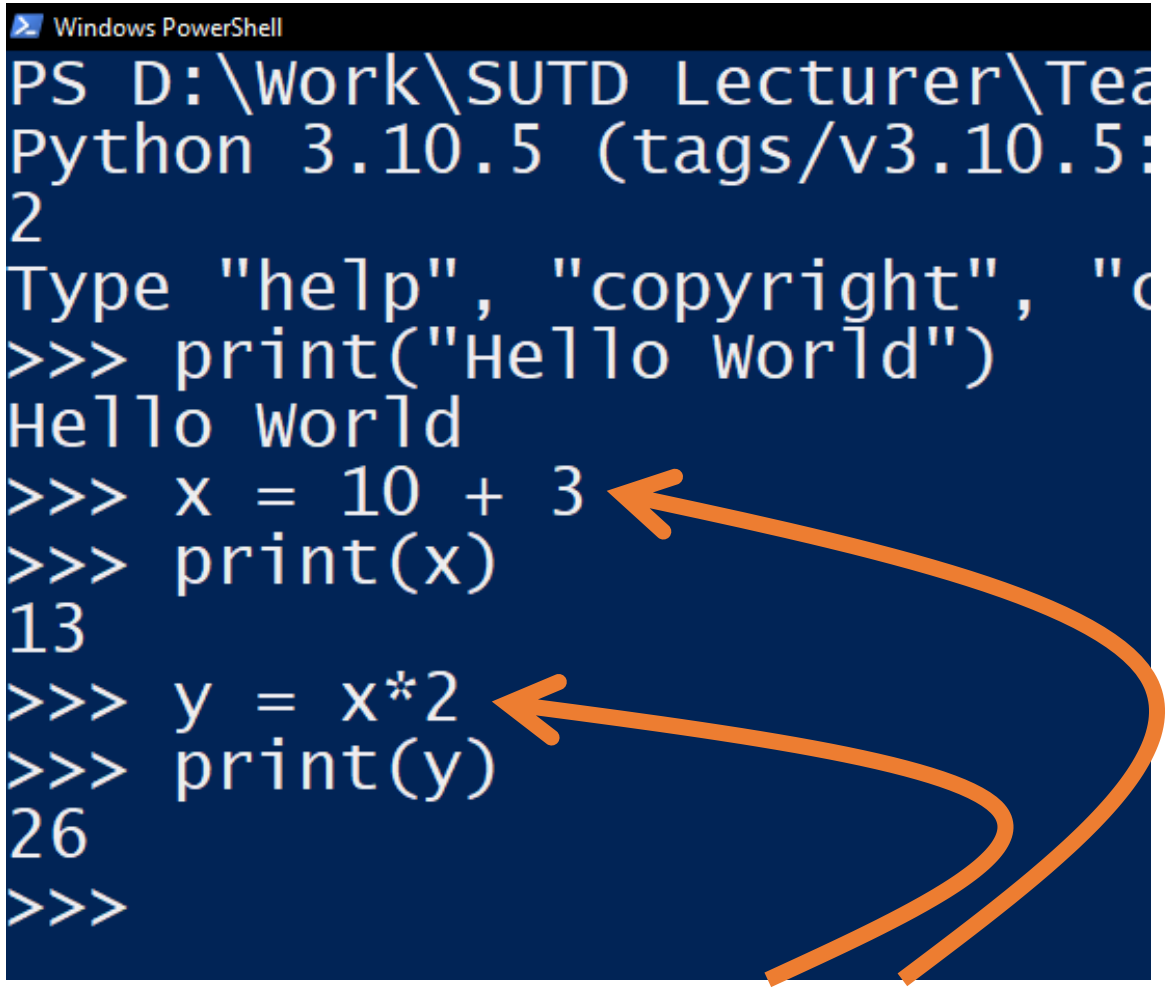
Our first program!

- **Here**, we have assigned the numerical value resulting from the operation $10 + 3$, to a **variable** named **x**.
- Later on, we can retrieve the value stored in the variable, and – for instance – ask the computer to display it on screen for us, with the **print()** function.
- Or even reuse it in **other calculations!**



Our first program!

- **Here**, we have assigned the numerical value resulting from the operation $10 + 3$, to a **variable** named **x**.
- Later on, we can retrieve the value stored in the variable, and – for instance – ask the computer to display it on screen for us, with the **print()** function.
- Or even reuse it in **other calculations!**



Important: these commands executed in the background but showed nothing in the console! Need to explicitly ask for a print()!

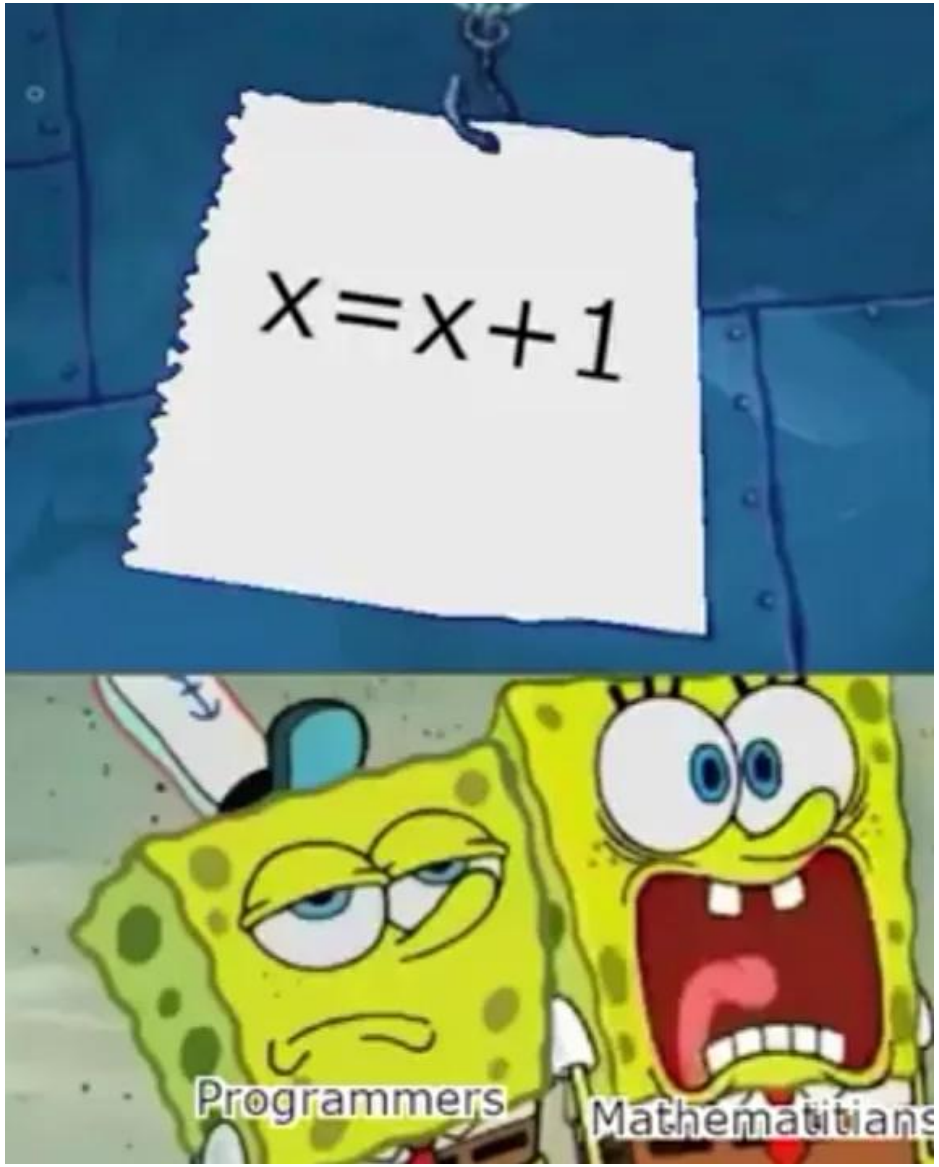
Our first program!

Assigning something to memory is done with the = sign.

(**Important Note:** The = sign does not have the same meaning as in mathematics.)

In computer science it means:

- Assign what is on the right-hand side of the equal sign to memory.
- The name of this element, called a **variable**, consists of the text on the left-hand side of the equal sign.



An important note on the use of the equal sign in computer science.

Matt’s Great advice (First of many)

Matt’s Great Advice: the print() function in Python.

The **print()** function is the most important Python function.

It is **only way** for you to check what is being computed and stored in memory at any given time.

Use it and abuse it, to check what your program is doing!

Executing a .py file in console mode

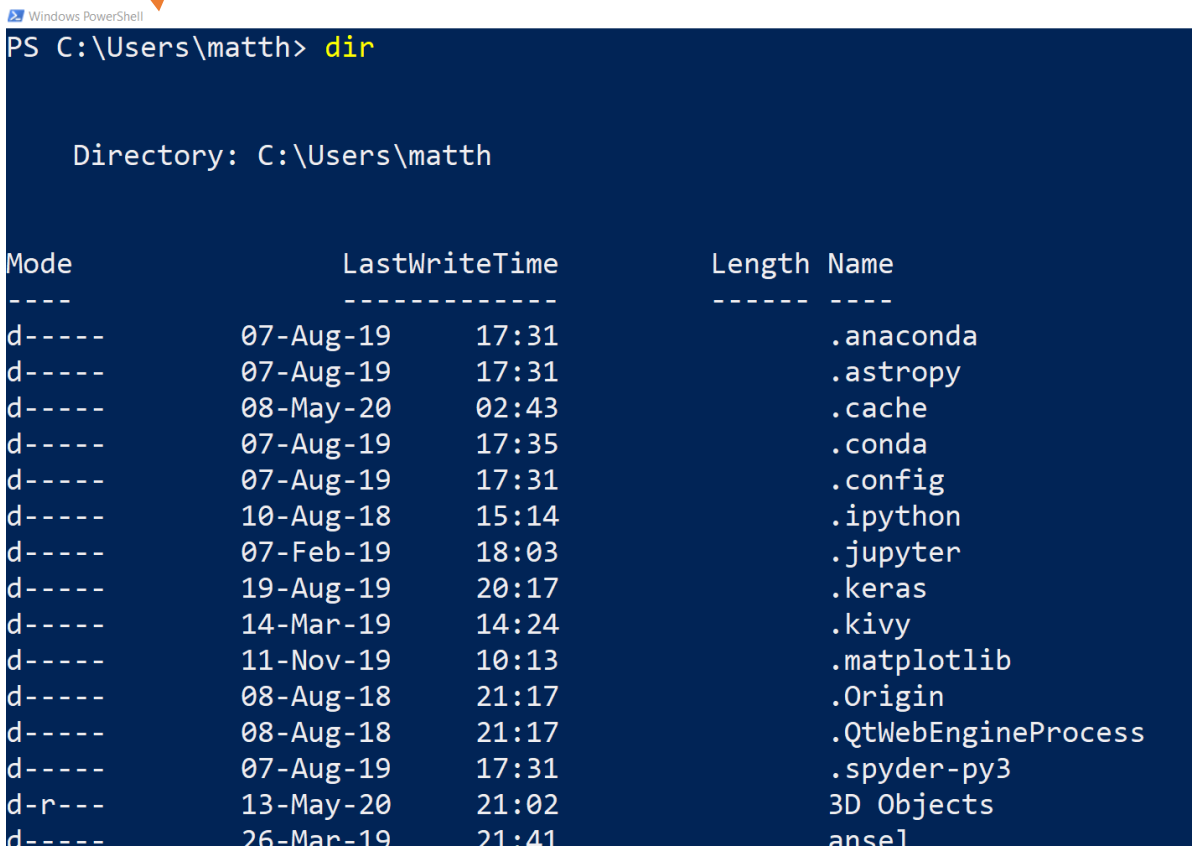
You should have downloaded a few files along with the lecture notes/slides we are currently using.

- More specifically, we will now use the **first_program.py** file, located in the “Code files” folder.
- Identify where you downloaded and stored your **first_program.py** file. We need to know where it is currently located, before moving to the next slide.

Executing a .py file in console mode

Command (dir/ls): The **dir command** (or **ls command** in Mac OS/Linux) lists the folders and files in your current location.

Current location is C:\Users\matth\



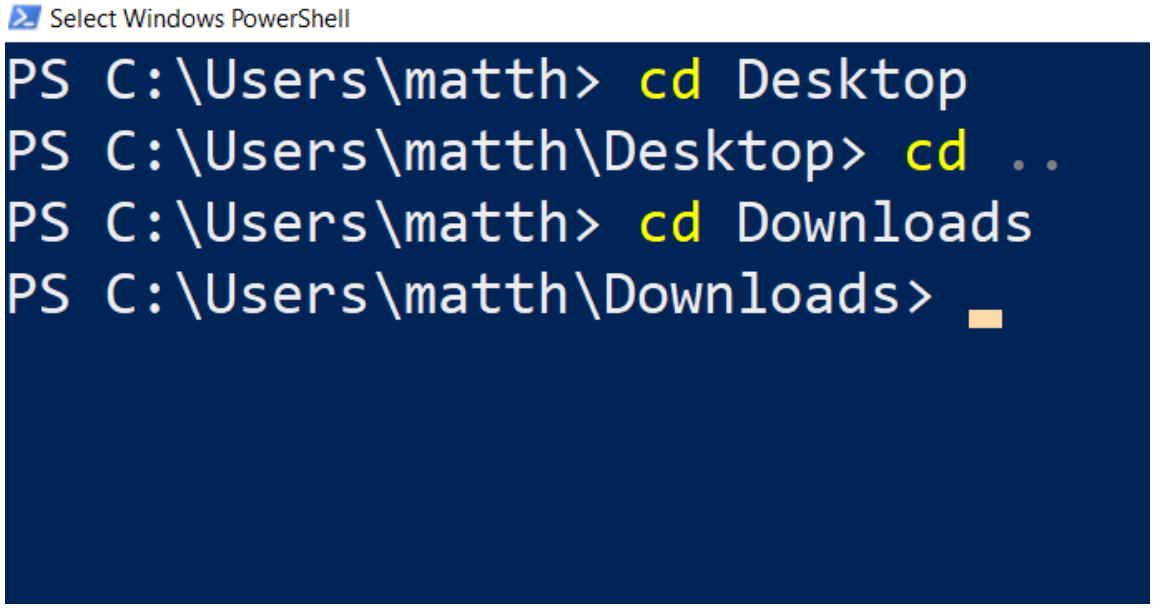
Executing a .py file in console mode

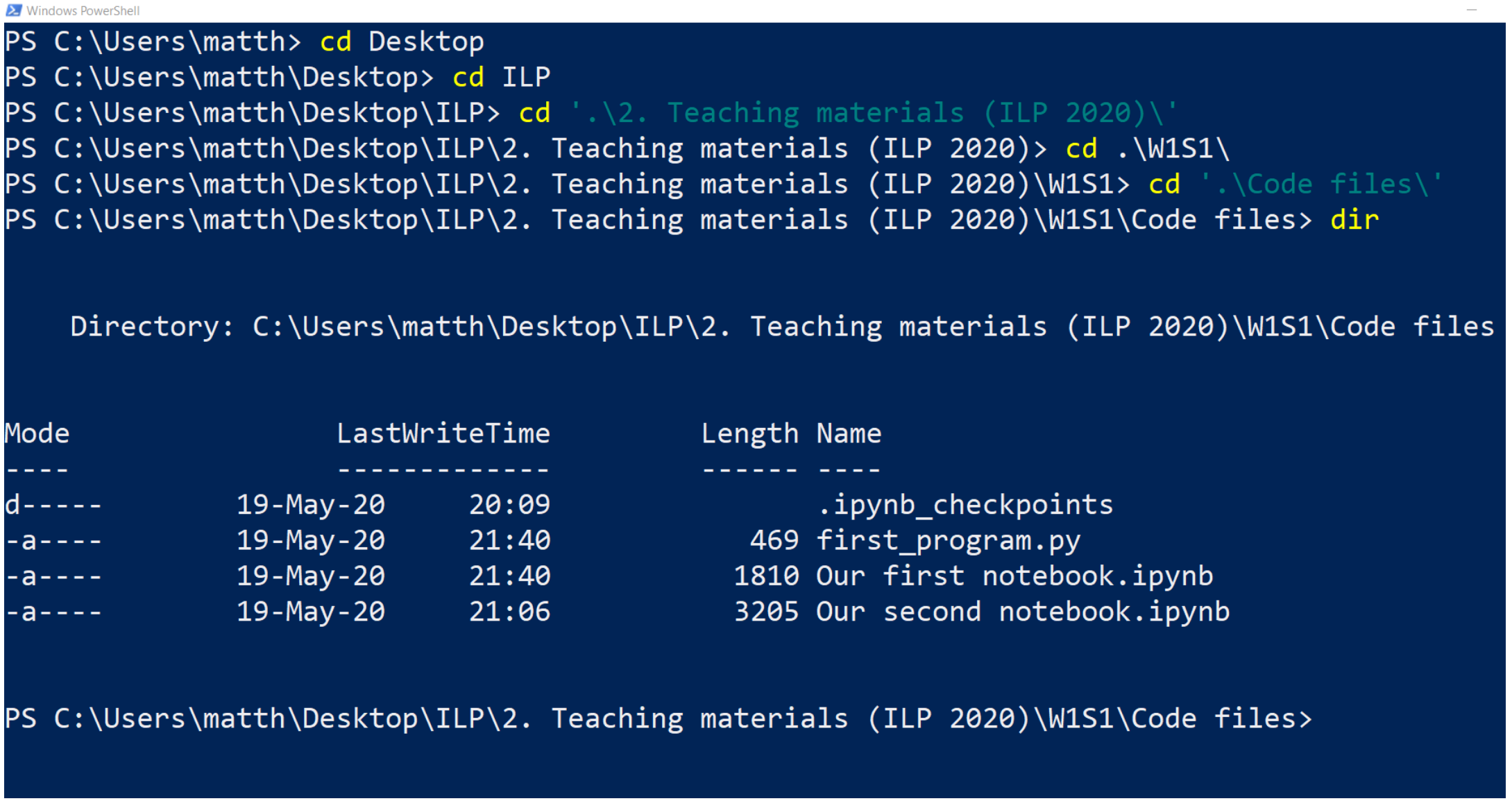
Command (dir/ls): The **dir command** (or **ls command** in Mac OS/Linux) lists the folders and files in your current location.

Command (cd): The **cd command** changes your current location to another folder, reachable from your current location in **dir/ls**.

- **Note:** the command "**cd ..**" moves you back one level.

Observe how the current location is changing every time.



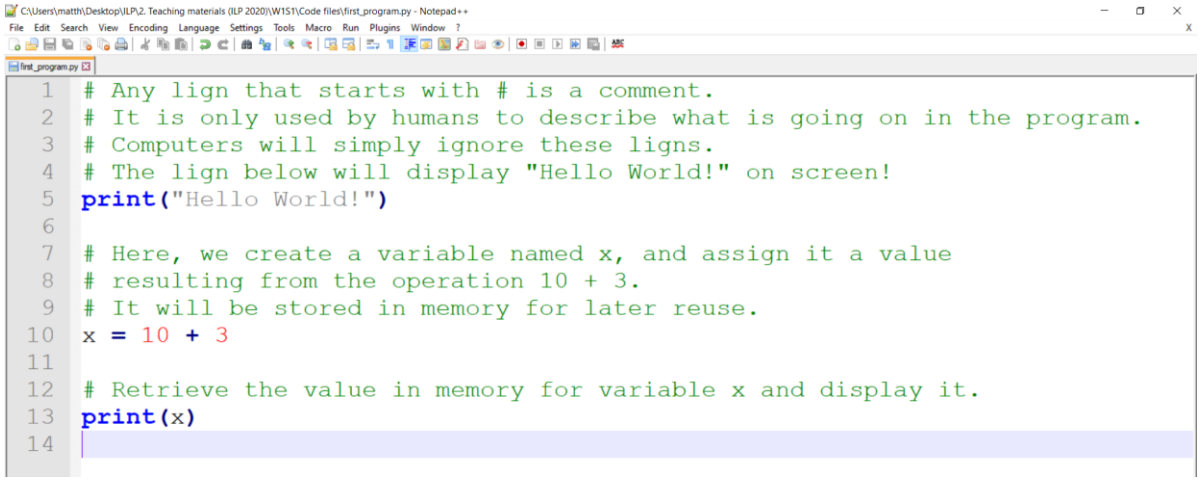


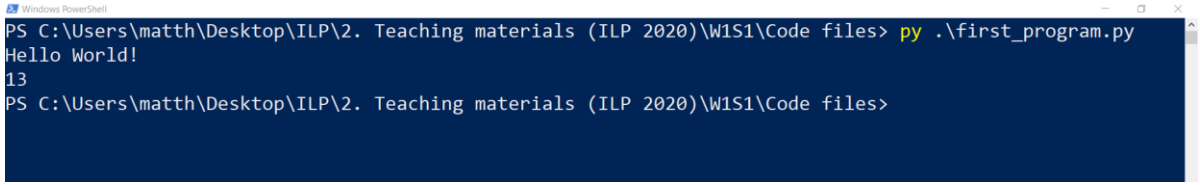
→ Now, use cd/dir/ls commands to move to the location of your **first_program.py** file !

Checking your .py file

- Open your **first_program.py** file with any text editor (specifically do it by right clicking and asking to open with a text editor such as Notepad or Notepad++).
- You should recognize the code we used earlier.
- Later on, you can run the code in the **first_program.py** file, all at once, by typing the following command in your console

py first_program.py





Running Python from an IDE

Problem: Typing code in a text editor and running it from console is not exactly convenient… In fact, it requires to alt-tab a lot!

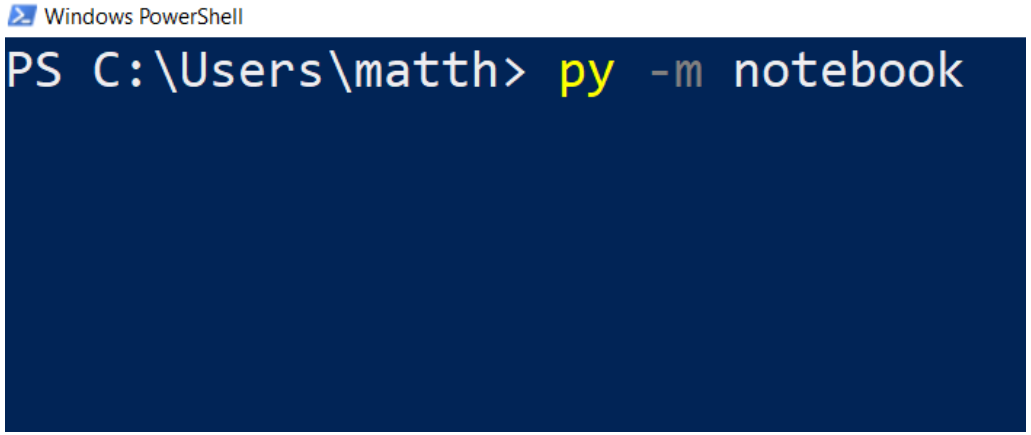
→ **Suggestion:** we should use an **Interactive Development Environment (IDE)**, which makes the coding easier for us.

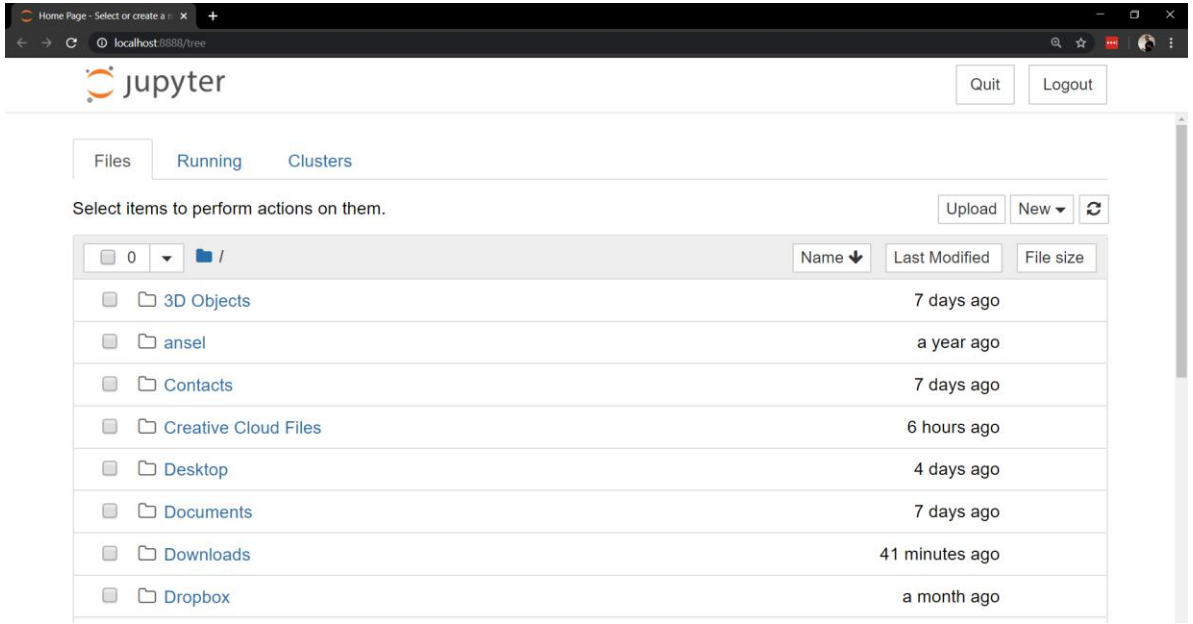
→ In this course, I suggest to use **Jupyter Notebook and/or Jupyter Lab environments**, but you might look online for other IDEs and learn about them if you want!

Running Python from an IDE, such as a Jupyter Notebook

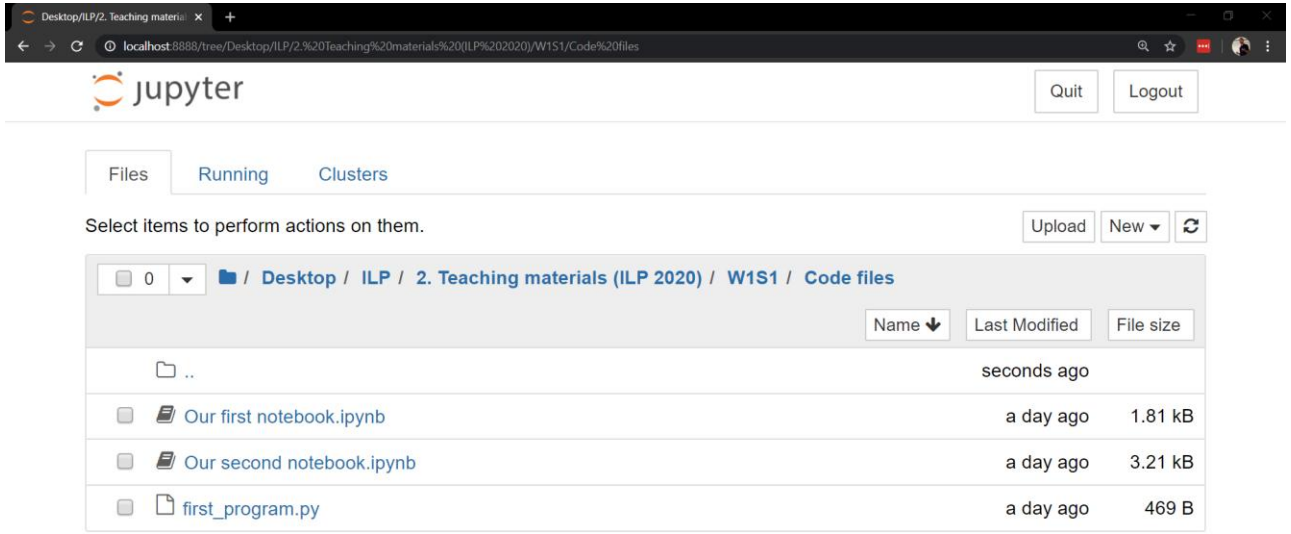
Return to your console, **in OS mode** (use **quit()** while in Python mode to return to OS mode if needed).

- Type **py –m notebook**, and press enter to submit and call the **Python notebook module.**
- **It should open a notebook window/tab in your web browser.**





Running Python from an IDE, such as a Jupyter Notebook



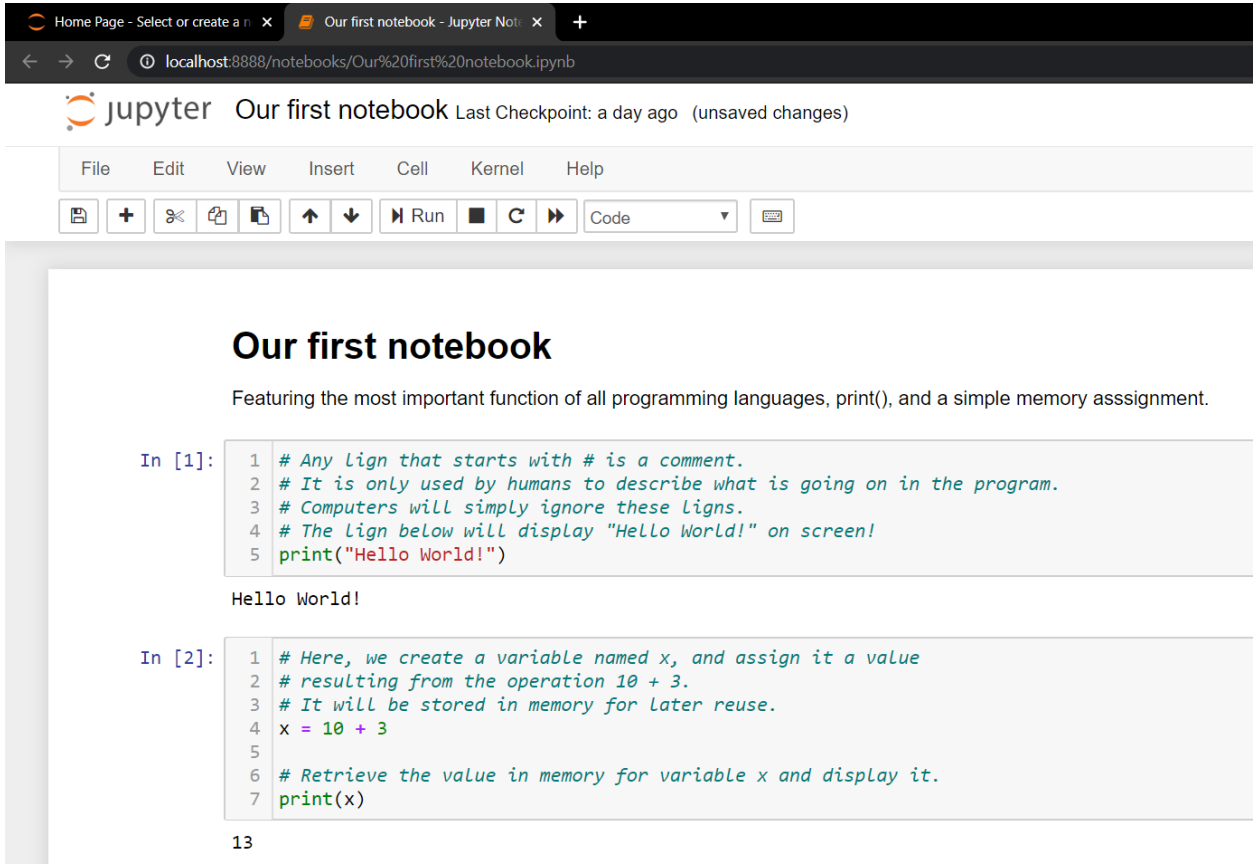
Notebooks are a more convenient way to navigate your files (rather than using cd repeatedly) and to program using Python.

- Move to the folder where the code you downloaded for today's lecture is.
- Let us try opening "Our first notebook.ipynb" by clicking it.
- *(Files ending in .ipynb are Jupyter Notebooks)*

Running Python from an IDE, such as a Jupyter Notebook

Notebooks provide a mixed combination of

- **text blocks** (in Markdown text language, ignore that)
- and **code blocks** (in Python, these blocks have a “In […]” on their left side).
- Try executing a cell of code by selecting it and pressing **Shift+Enter**!
- A lot more convenient, isn’t it?

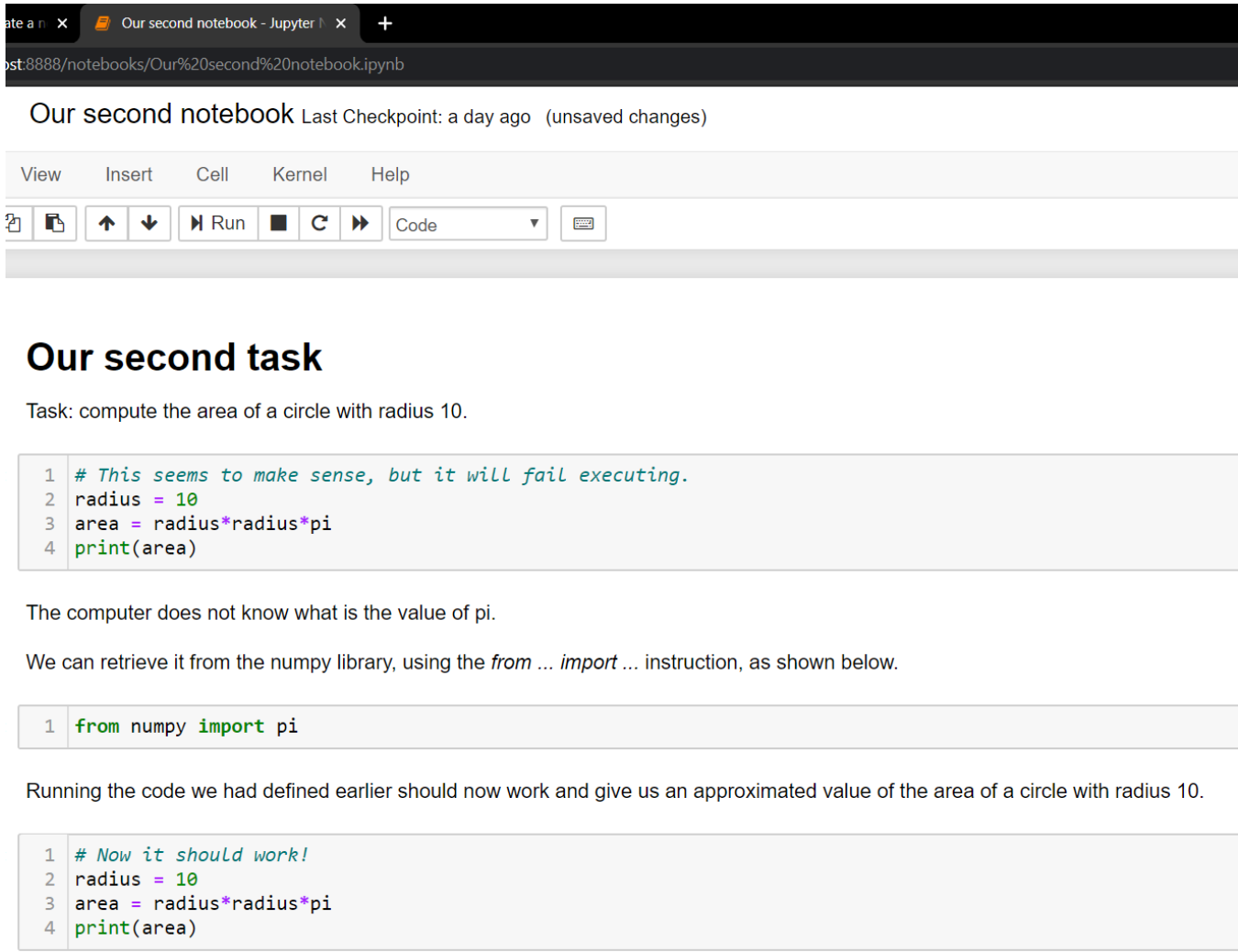


Our second task

Now that we know how to run Python in a Notebook environment, let us consider a **second task…**

We will ask the computer to compute the area of a circle with radius 10.

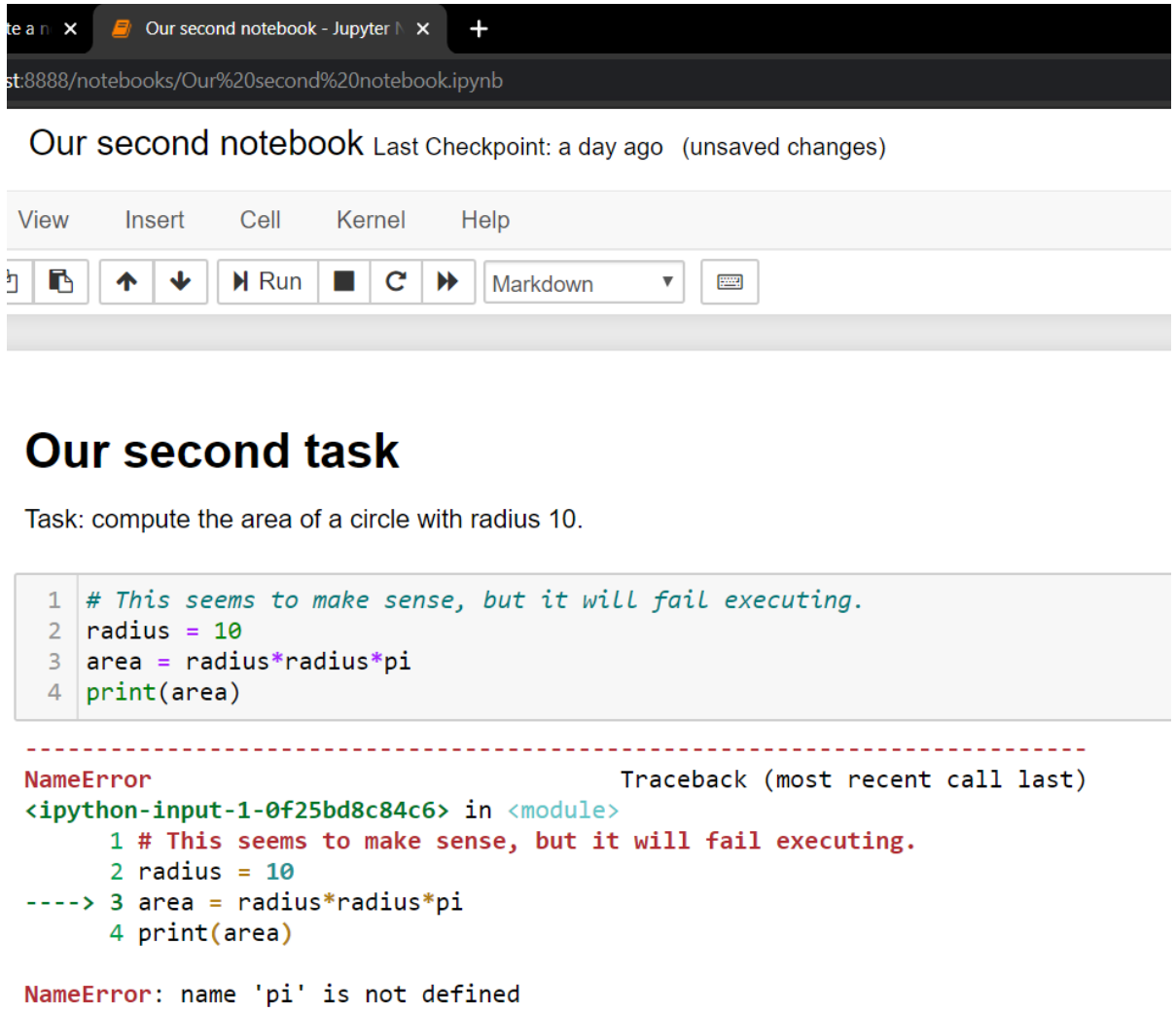
- Open the second notebook.
- **Note:** in computer science, the multiplication operation is denoted $*$, not $\times$.



A problem with our second task…

Problem: While the task seems easy mathematically speaking, we have a problem… **We need the value of the constant pi.**

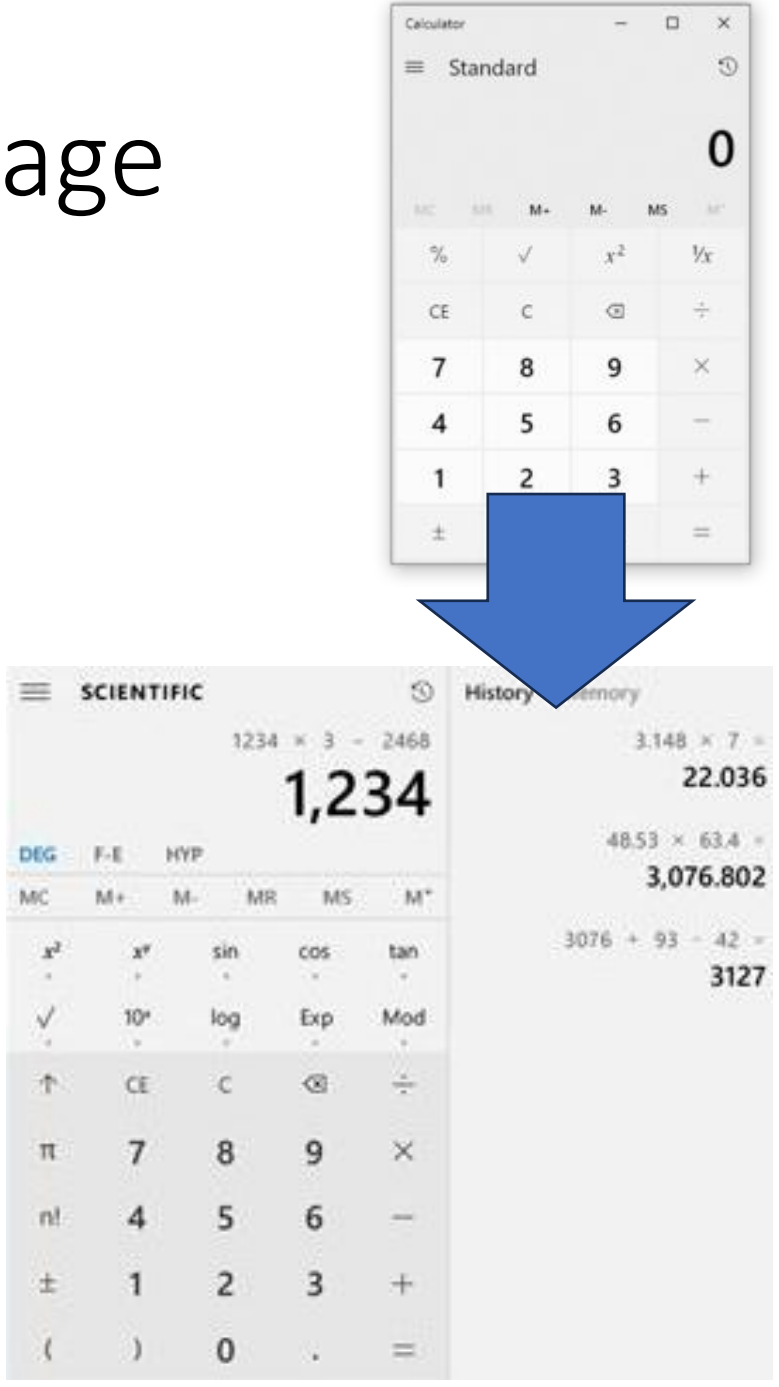
- Python does not know it by default like your calculator…
- We could create a variable named pi and assign it the value 3.14, but it is better (and more accurate) to retrieve it from a **package**.



Importing from a Python package

By default, Python is a simple yet powerful calculator, which can only perform basic calculations (additions, multiplications, etc.)

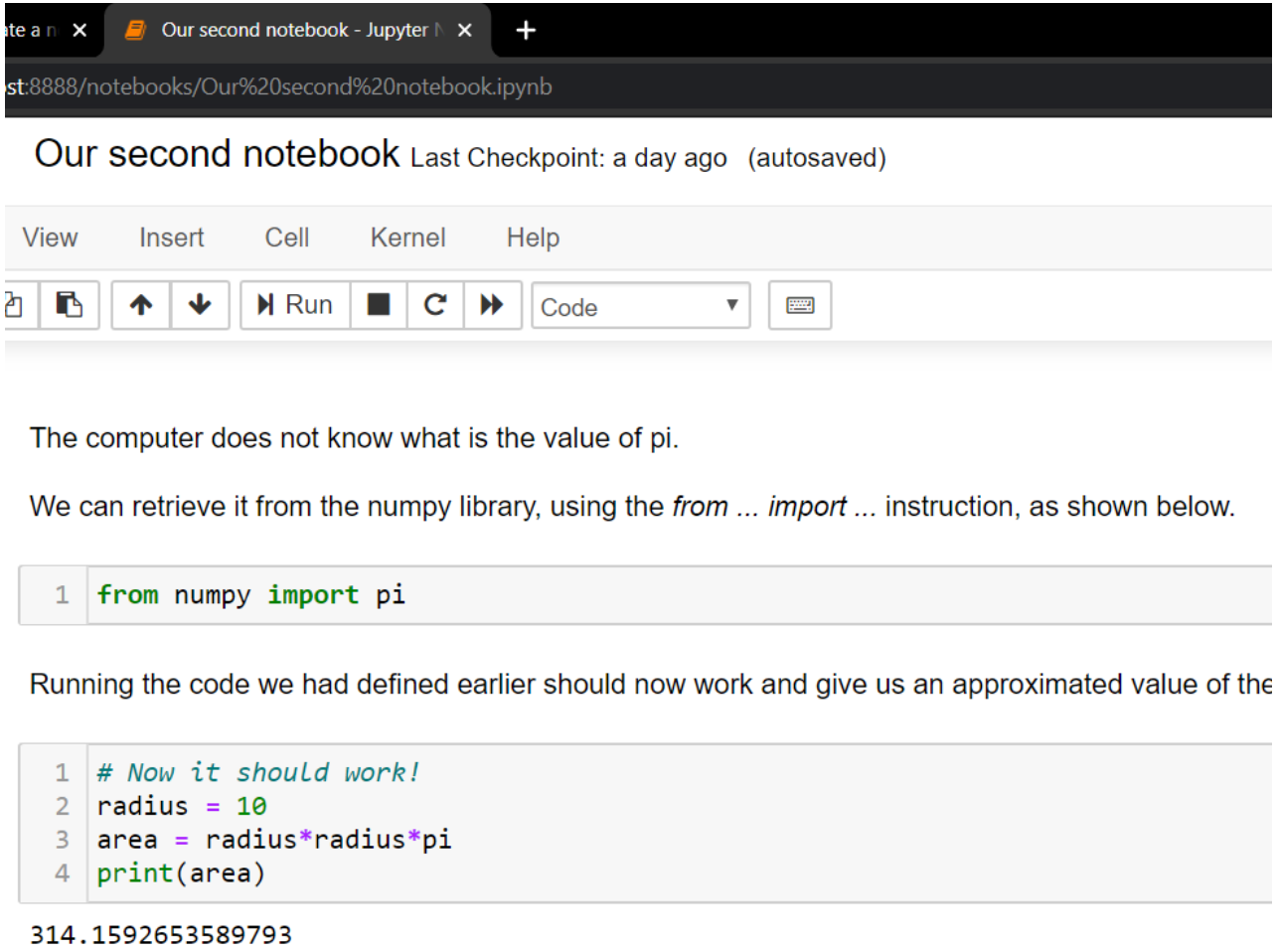
- If we need more advanced concepts, we need to **import** them from a **package**.
- **Importing** ≈ adding a specific button to your calculator so it can run more advanced operations!



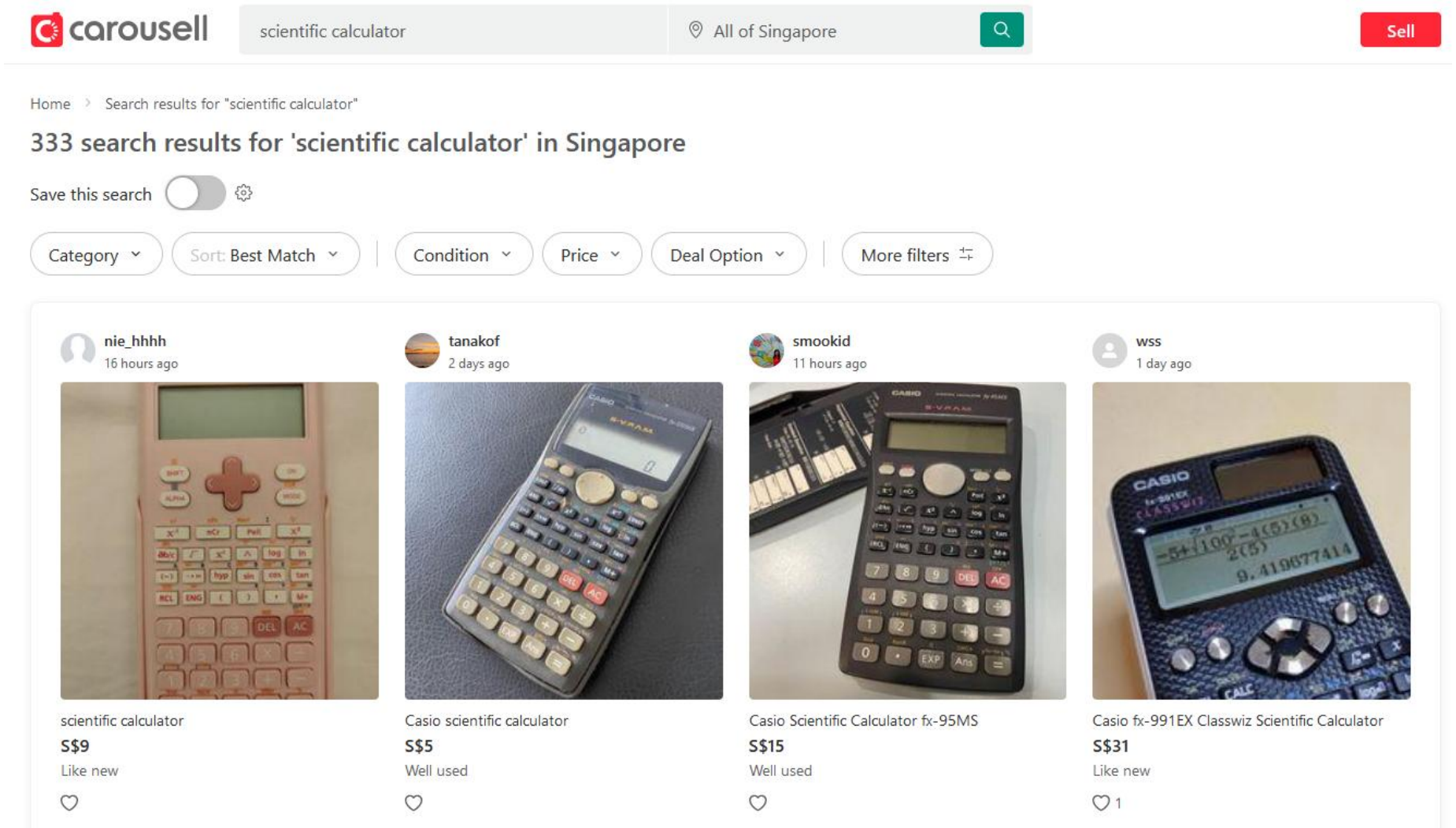
Importing from a Python package

By default, Python is a simple yet powerful calculator, which can only perform basic calculations (additions, multiplications, etc.)

- If we need more advanced concepts, we need to **import** them from a **package**.
- **Importing** ≈ adding a specific button to your calculator so it can run more advanced operations!



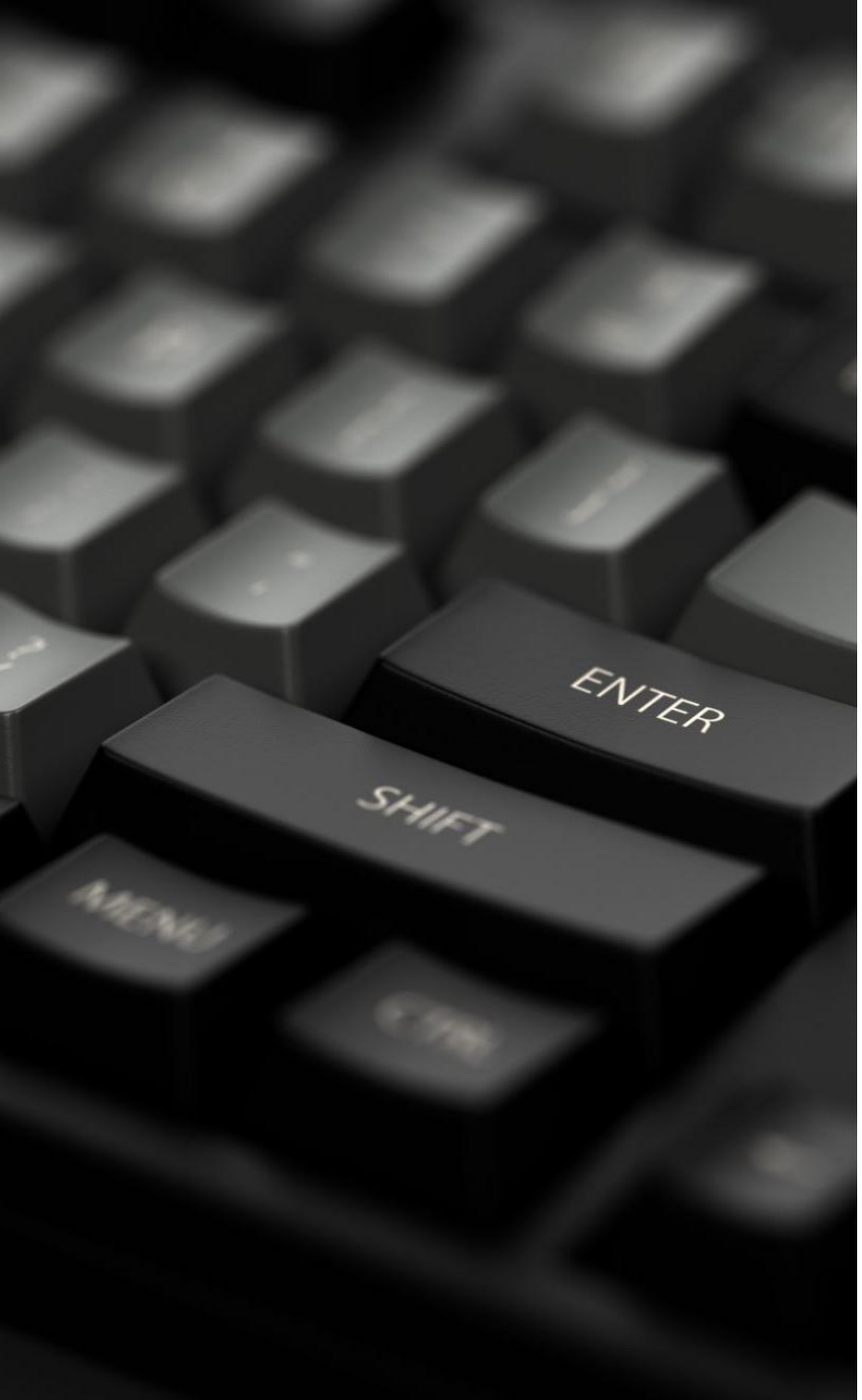
By the way, this means that you do not need your calculator anymore, maybe time to give it away.



Congratulations, you now have a Python-compatible machine, ready to run!

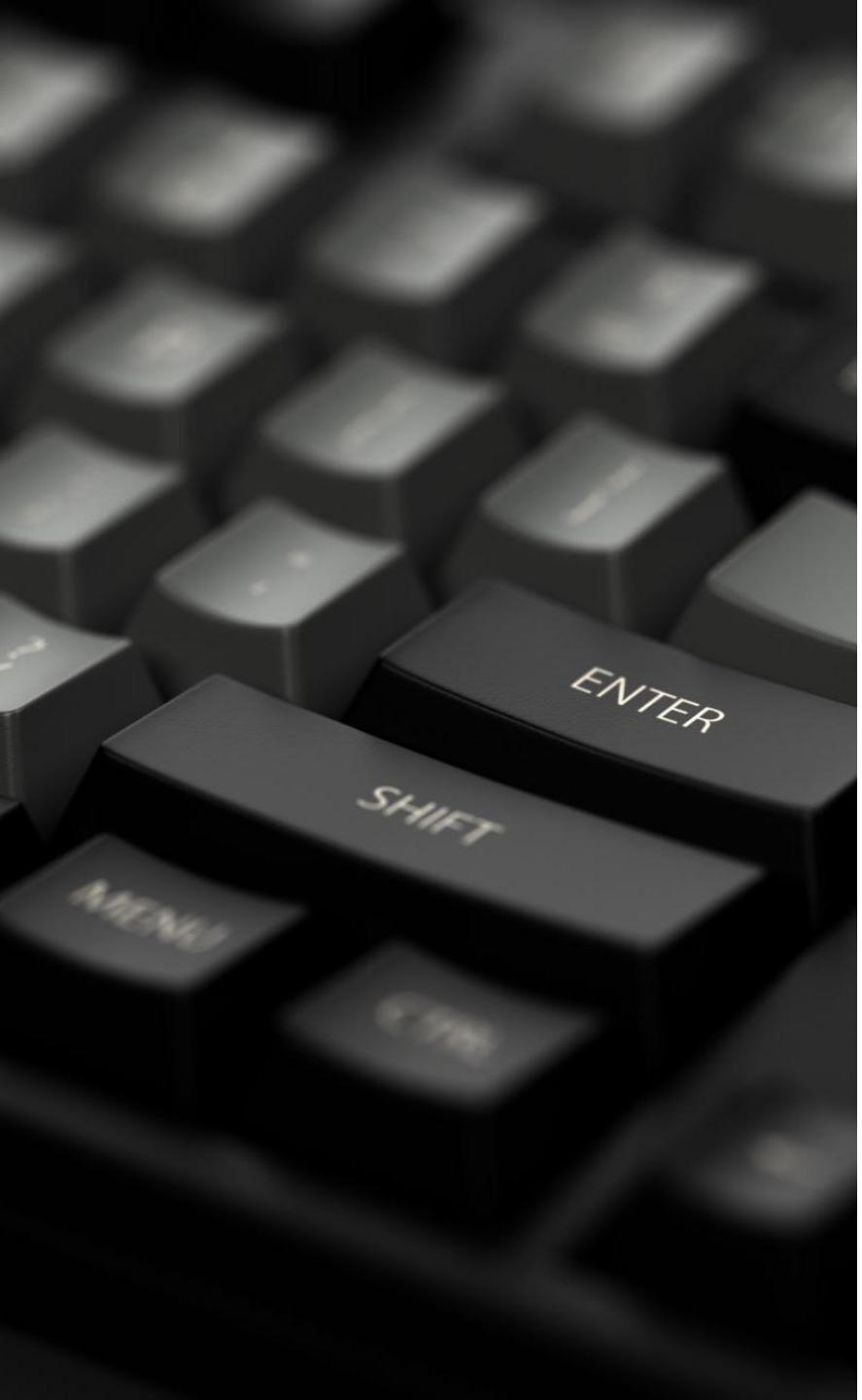
Feel free to play around a bit more if you want!

python™



Conclusion (Chapter 1)

- What is **programming**?
- **What are the key concepts** about programming and computer science?
- What is a **programming language**?
- **How do I install and configure Python?**
- How to I write my **first programs** in Python?
- What is a **programming environment**?



Debugging time

→If you were able to execute the two codes (in the Jupyter Notebooks): you are officially done for today.

→ Let me know ASAP if you have encountered technical issues and we will try to fix them together.