

# ILP 2022 – W1S1

## Getting started

Matthieu DE MARI – Singapore University of Technology and Design



SINGAPORE UNIVERSITY OF  
TECHNOLOGY AND DESIGN

# A quick word about me

- Matthieu (**Matt**) DE MARI
- Lecturer at SUTD (Python, Deep Learning, AI, and more)
- Information Systems Technology and Design (ISTD) pillar/faculty
- PhD from CentraleSupelec (France)
- Email: [matthieu\\_demari@sutd.edu.sg](mailto:matthieu_demari@sutd.edu.sg)
- Office @ SUTD: 1.702.27



# Outline (Week1, Session1 – W1S1)

- About this course: syllabus, objective, eDimension, practices, etc.
- What is programming?
- Key concepts about programming and computer science.
- Programming languages.
- **Installing and configuring Python, and extra packages.**
- Our first programs!

# Objectives of this Summer School

## **Objectives:**

- Give the students an introduction to Computer Science,
- Programming,
- and Python.

## **Delivery:**

- 3x 2h-lessons per week
- Lessons include a bit of theory (PPT slides) and practice activities (in Jupyter Notebooks)

# Topics

- **Week1 Session 1 (W1S1):** Getting started, key concepts, configuring and installing Python
- **W1S2:** Variables, math operators, comments, printing and getting
- **W1S3:** None type, Boolean types and functions
- **W2S1:** More practice on functions
- **W2S2:** If, elif, else, while, break statements
- **W2S3:** More practice on if, elif, else, while
- **W3S1:** For loops, generators and recursion
- **W3S2:** The list type
- **W3S3:** Advanced concepts on for loops and list type

# Topics

- **W4S1:** Debugging, errors, asserts and time
- **W4S2:** Numpy library (part 1) and imports
- **W5S1:** More on Numpy (part2), randomness
- **W5S2:** Everything about strings
- **W5S3:** Dictionaries and object-oriented thinking
- **W6S1:** Object-oriented programming (part 1)
- **W6S2:** Object-oriented programming (part 2)
- **W6S3:** Recap and mini-project!

# Edimension

## **Teaching materials**

- PPT/PDF contain the lecture materials (PPT preferred)
- Activities notebooks and their answers
- The teaching materials will be uploaded on eDimension and made available on the same day.
- <https://edimension.sutd.edu.sg/>

# Homeworks, extras and exams

- **In-class activities:** activities, done together during online lessons. Solutions are provided on other notebooks.

- **Homeworks?** Nope.

But

- **Extra practice:** basic exercises and notions, to practice the concepts seen in class a bit more.
- **Extra challenges:** advanced versions of the activities discussed in class.
- None of them are mandatory.  
Solutions are provided (except for challenges!)



Plot twist: 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, international consultant in public transit network design.  
(He also has a fantastic blog about using strategy and planning games, like MiniMetro and SimCity to explain and understand notions of transportation network and city design)

<http://humantransit.org/2014/12/learning-how-transit-works-from-mini-metro.html>

# Plot twist: 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.



# Plot twist: GAMIFICATION

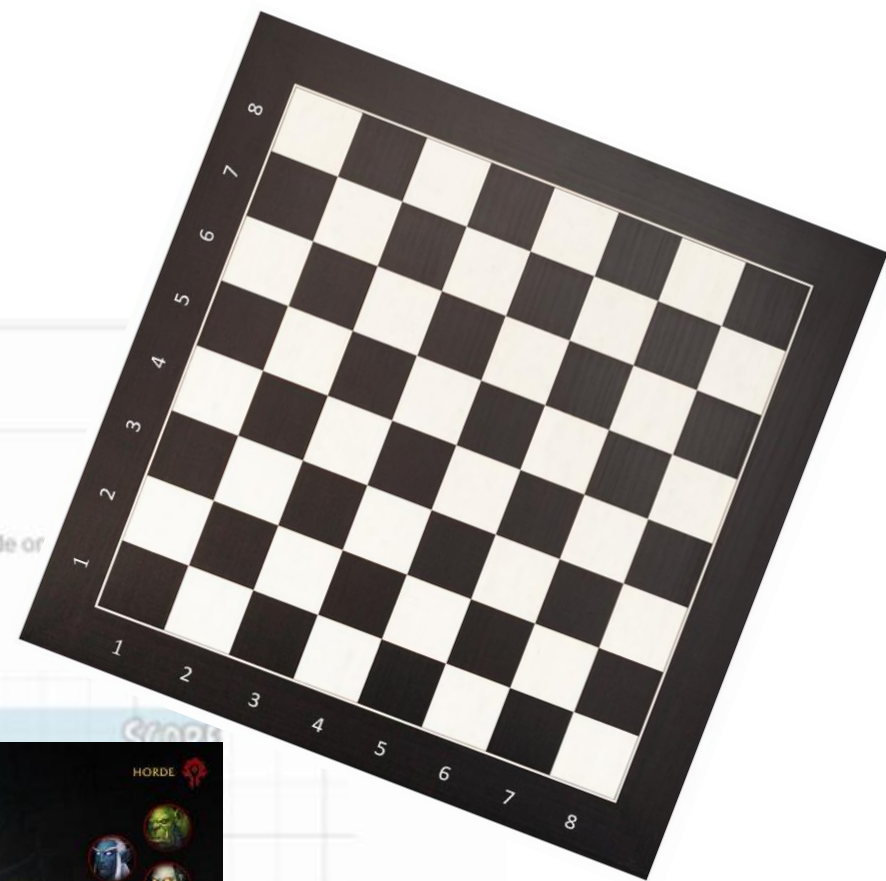


## Activity 2 - Ballistics of an angry bird

### Problem statement

In the angry bird game, the player has to launch birds at structures from a slingshot. The player gets to decide on the launch speed for the bird alpha (in  $m/s$ ).

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



# Survey

- During this class, I might often use online “surveys”.
- These help me check your understanding of this class and adjust accordingly... Please fill them!
- Speaking of... **Here is your first survey: what is “programming”?**

<https://forms.gle/stRPXnu5ZZrjMkCw5>

# Programming: definition

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

It involves tasks such as

- **analysis,**
- designing **algorithms,**
- and implementing said **algorithms** in a chosen programming language (a.k.a. **coding**).

# Programming: definition

- **Definition (Programming):**  
**Programming** refers to the process of designing and building an executable computer program, to accomplish a specific computational task.
- **Layman definition (Programming):**  
**Programming** consists of defining a sequence of instructions that the computer must follow to accomplish a task.

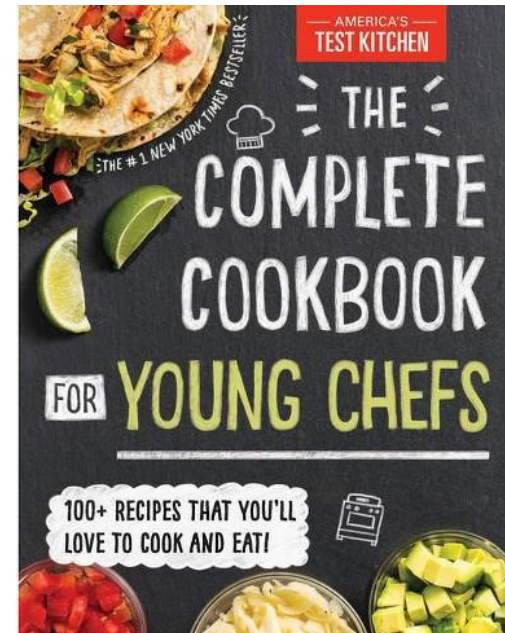
It involves tasks such as

- **analysis,**
- designing **algorithms,**
- and implementing said **algorithms** in a chosen programming language (a.k.a. **coding**).



# Programming: some analogies

- Think of it as a food **recipe** book!  
A recipe requires you to follow and execute a set of instructions to cook a recipe from scratch!

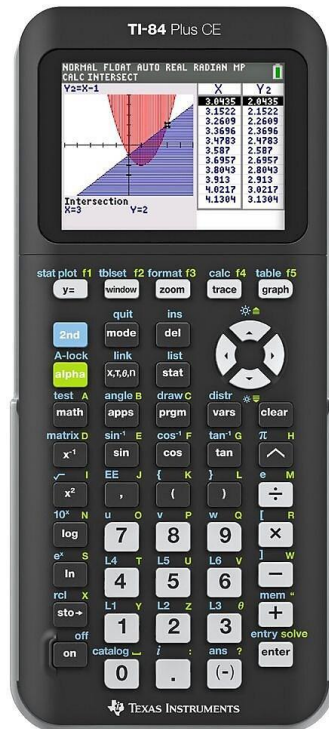


When you're cooking & the recipe says "chill in the fridge for one hour"



# Programming: some analogies

- Or a **sequence of operations** you would normally do on a calculator to find a result.
- **Example problem:** compute the area of a circle with radius 10.



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



# Algorithm: definition

- **Definition (Algorithm):** An **algorithm** is a finite sequence of instructions, typically used to solve a class of problems or perform a computation task.

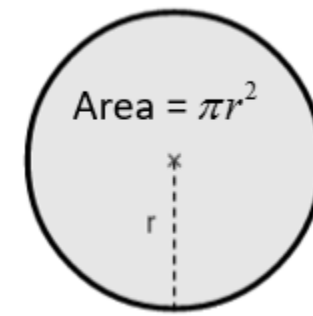
When executed, on a provided set of **inputs**, it will proceed through a set of well-defined states and will eventually produce an output.

- **Example problem:** compute the area of a circle with radius 10.
  - Type **10**,
  - Type multiply key,
  - Type **10**,
  - Press equal/enter key,
  - Type multiply key,
  - Type  $\pi$ ,
  - Press equal key again,
  - You have reached your result.

ALGORITHM

# The life of a programmer

1. **Identify a problem.**
2. **Come up with a general algorithm to solve it.**
  - Step-by-step instructions.
  - Think of it as a 'recipe' or a 'flowchart'.
3. **Represent this algorithm as a program, using a chosen programming language.**
4. **Execute the program on a computer!**



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

ALGORITHM



Quick question

**→ Who is considered the inventor of computers and “programming”?**

# Quick question

→ **Who is considered the inventor of computers and “programming”?**

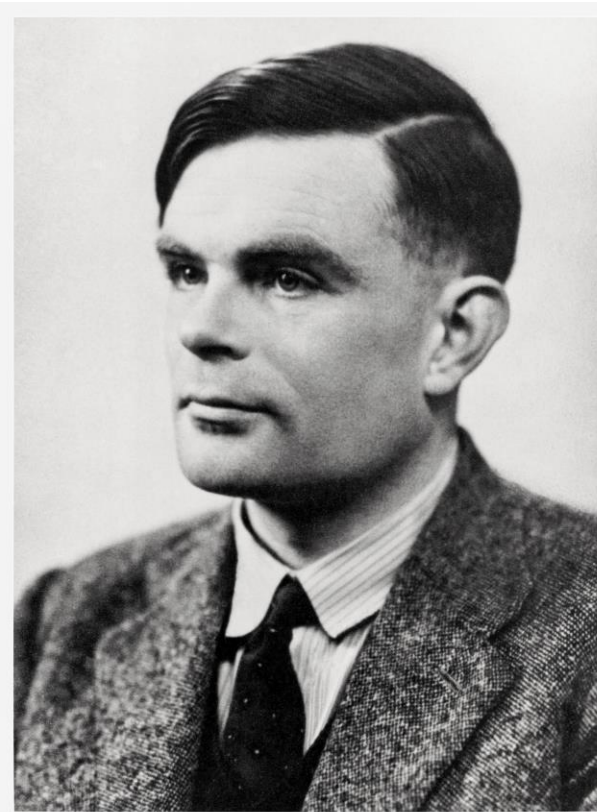
*Hint: it is NOT one of the persons below.*



# The “father” of computer science

**Alan Mathison Turing** (1912–1954)

was an English mathematician, computer scientist, logician, cryptanalyst, philosopher, and theoretical biologist.

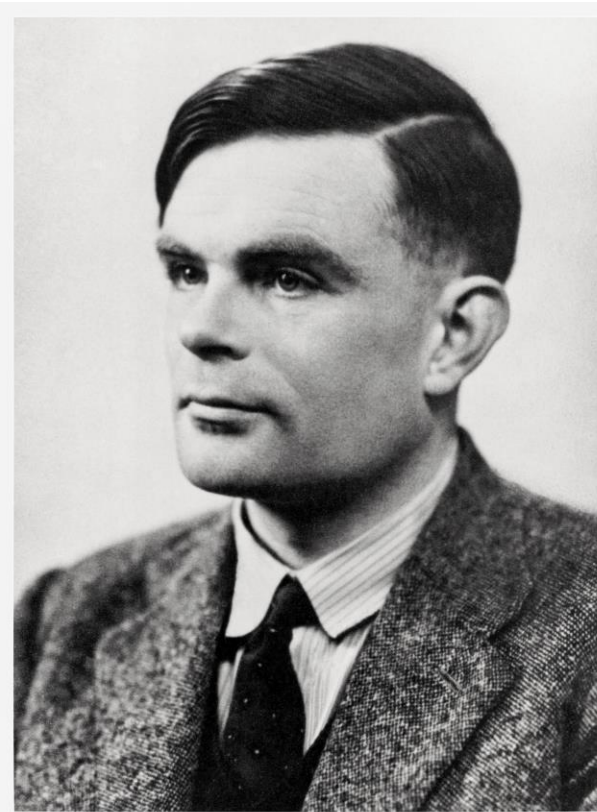


# The “father” of computer science

**Alan Mathison Turing** (1912–1954)

was an English mathematician, computer scientist, logician, cryptanalyst, philosopher, and theoretical biologist.

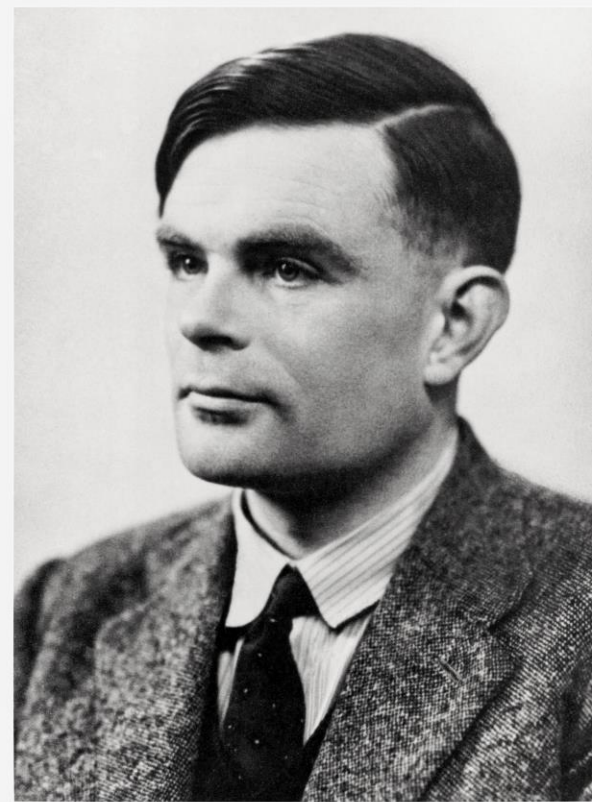
- Turing was highly influential in the development of theoretical computer science, formalized algorithmic concepts, and created the first Turing machine, which can be considered the first model of a general-purpose computer.





# The “father” of computer science

- During World War 2, he came up with an idea of a “computer-like” machine, to decode encrypted transmissions from the German army.
- Thanks to his decryption device, it has been estimated the war in Europe was shortened by more than two years and saved over 14 million lives.
- Learn more about Alan Turing, by watching **The Imitation Game** movie.



# About your computer

Your computer is good at doing two, and **only** two, things

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



# About your computer

Your computer is good at doing two, and **only** two, things

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

**And that is it.**

# About your computer

Your computer is good at doing two, and **only** two, things

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

**And that is it.**

**And that is all you need.**

# About your computer

Your computer is good at doing two, and **only** two, things

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

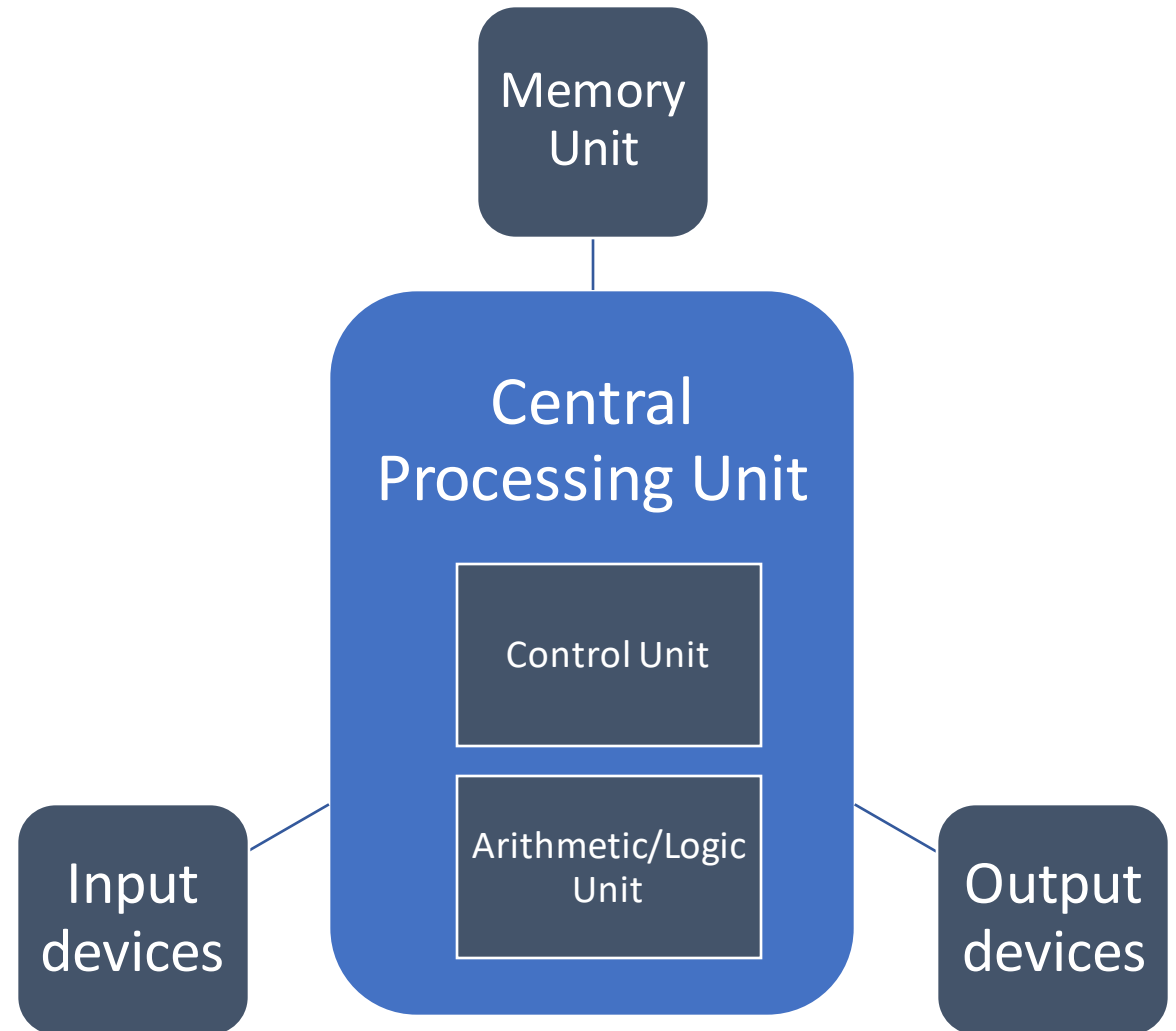
**And that is it.**

**And that is all you need.**

**Because all operations performed by computers nowadays can be broken down to combinations of both aforementioned operations.**

# The Von Neumann architecture

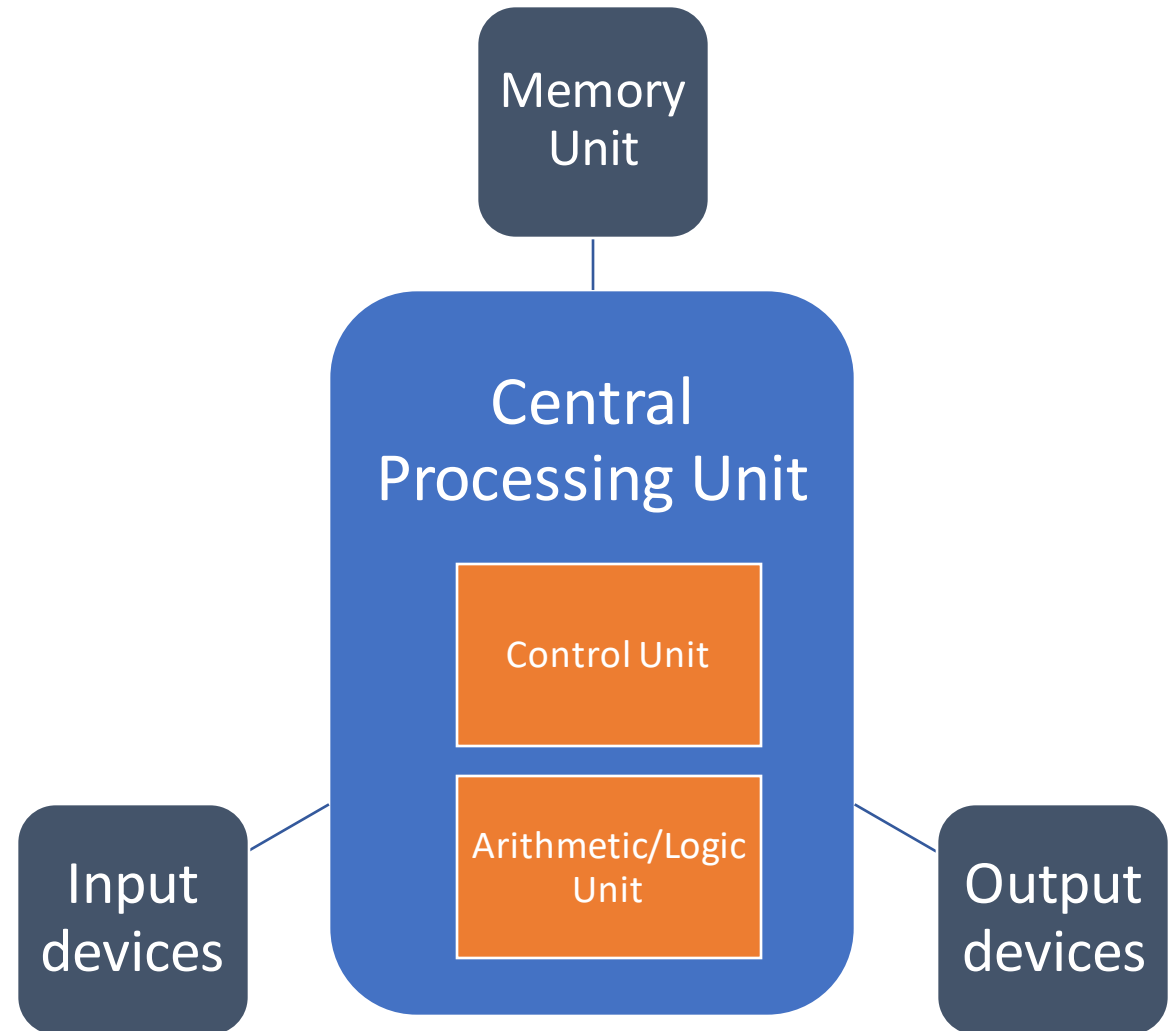
- **Definition (Von Neumann architecture):** The Von Neumann architecture describes one of the first architectures for a computer.



# The Von Neumann architecture

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

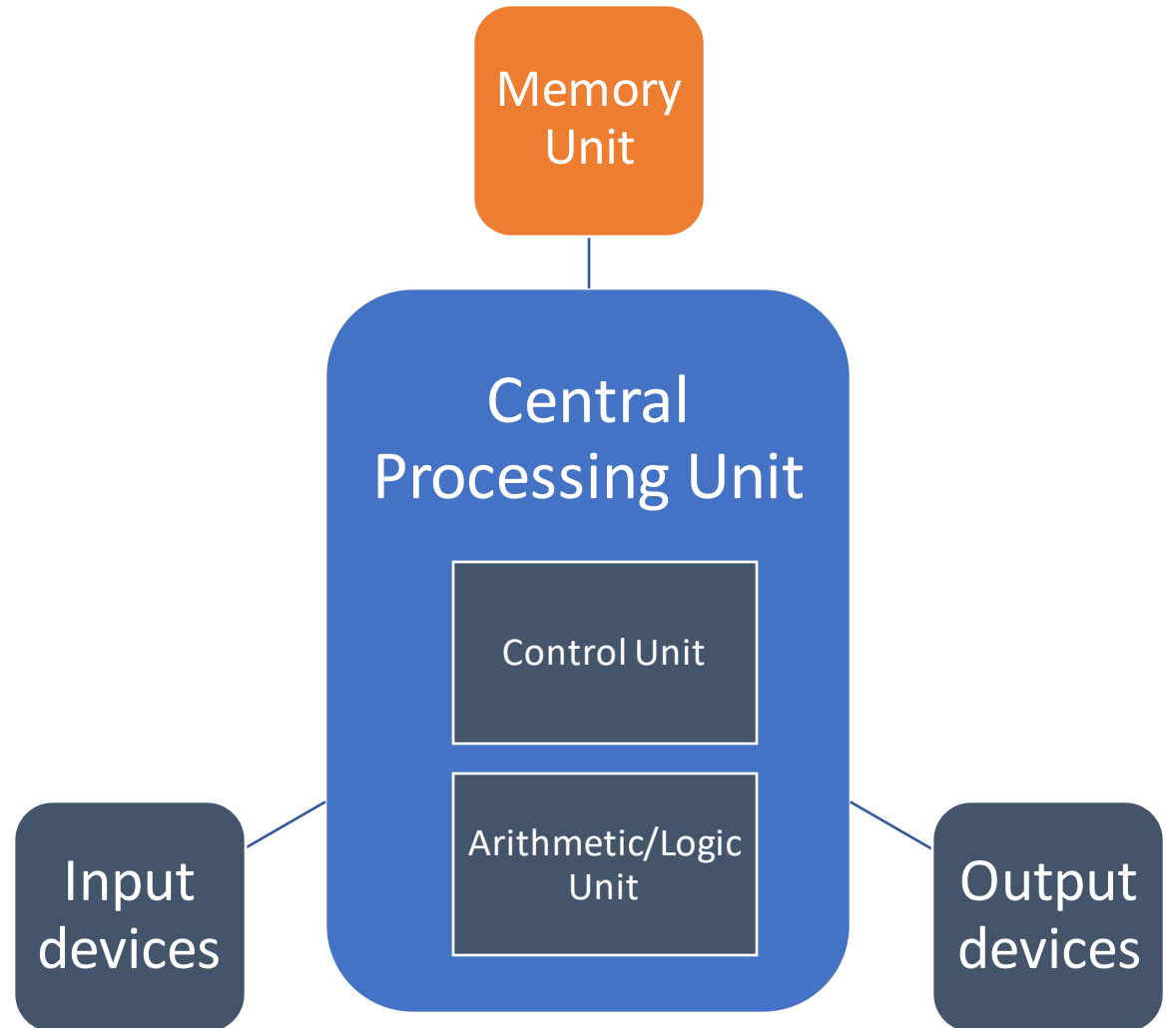
- An **Arithmetic/Logic Unit** (in charge of dealing with instructions, typically math operations, in binary 0/1),
- And a **Control Unit** (in charge of the hardware and communication between different hardware elements)



# The Von Neumann architecture

It also contains...

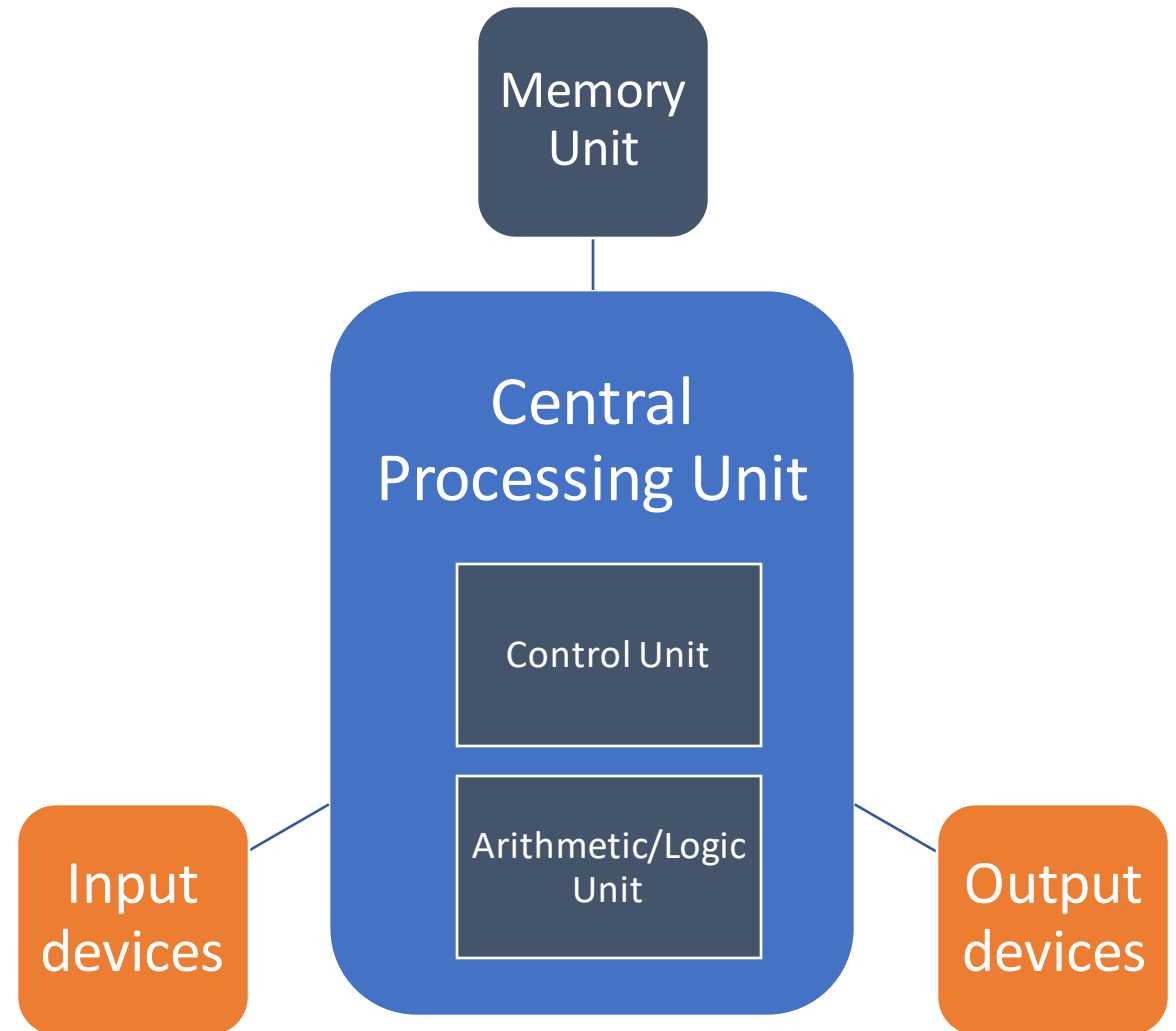
- A **Memory Unit** (for storing and retrieving results from previous computational tasks, using binary formatting 0/1),



# The Von Neumann architecture

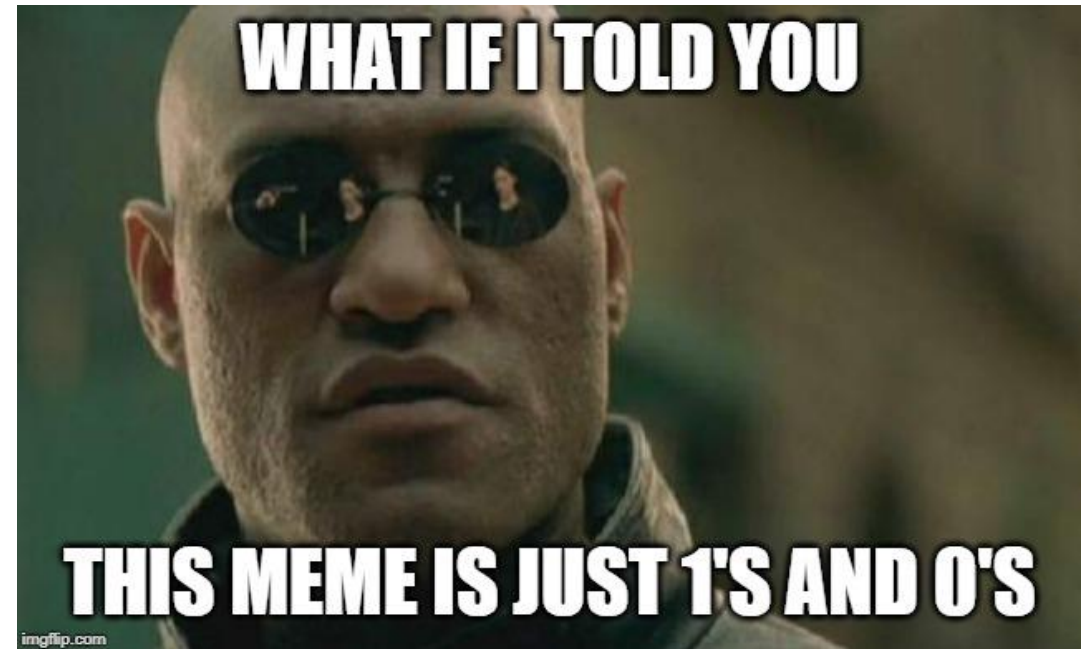
It also contains...

- 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.).



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

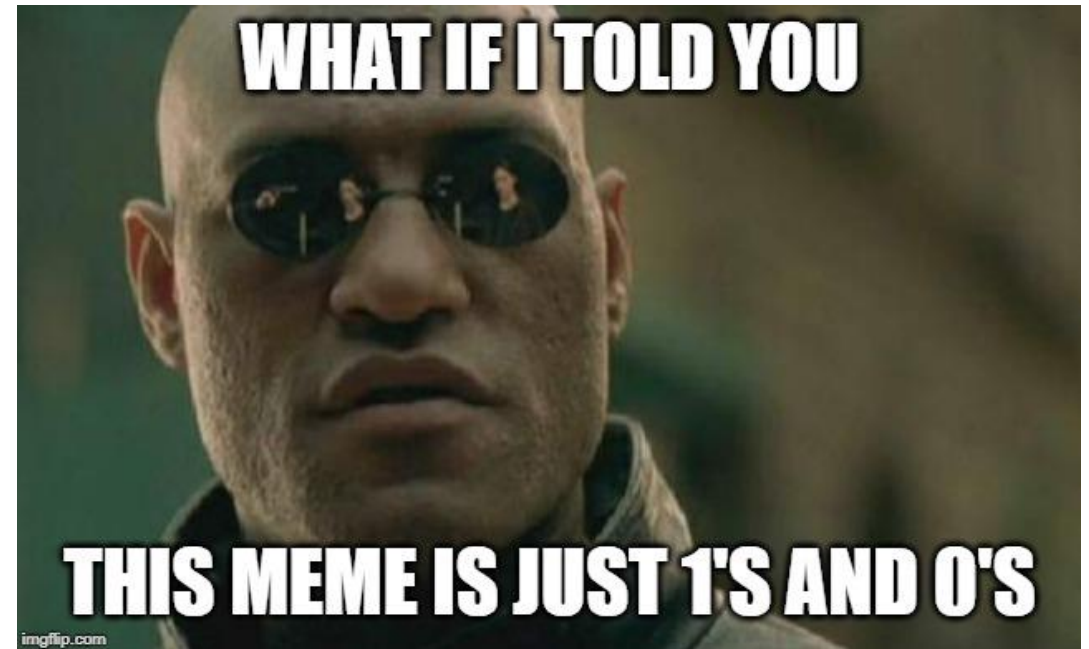
- **Observation:** in a computer, all operations (computational and memory tasks) and are performed in **binary**.





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

- **Observation:** in a computer, all operations (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

- **Observation:** in a computer, all operations (computational and memory tasks) and are performed in **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

- **Observation:** in a computer, all operations (computational and memory tasks) and are performed in **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++ 	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.

# Python: what is it?

**About Python:** Python is an interpreted, high-level, general-purpose programming language.

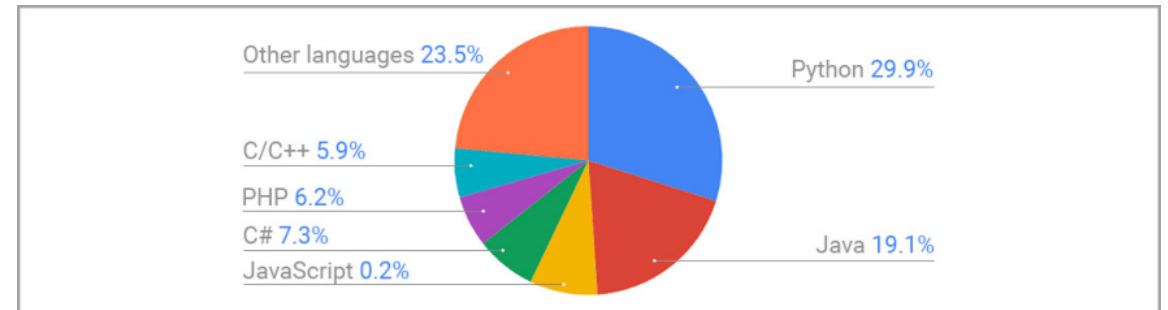
- Created by Guido van Rossum and first released in 1991.
- Currently on its **v3**, since **2008**.
- Python's design philosophy emphasizes code **readability**.
- Its language constructs aim to help programmers write clear, logical code for small and large-scale projects.



# 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 frameworks such as Tensorflow (Google AI), Pytorch (widely used in academic research on AI), etc.



# Why learn Python?

- Python is widely used in IT companies.
- Also, high interoperability with other languages: Jumping to or including another language (Java, C, SQL, etc.) is easy, once you know Python.



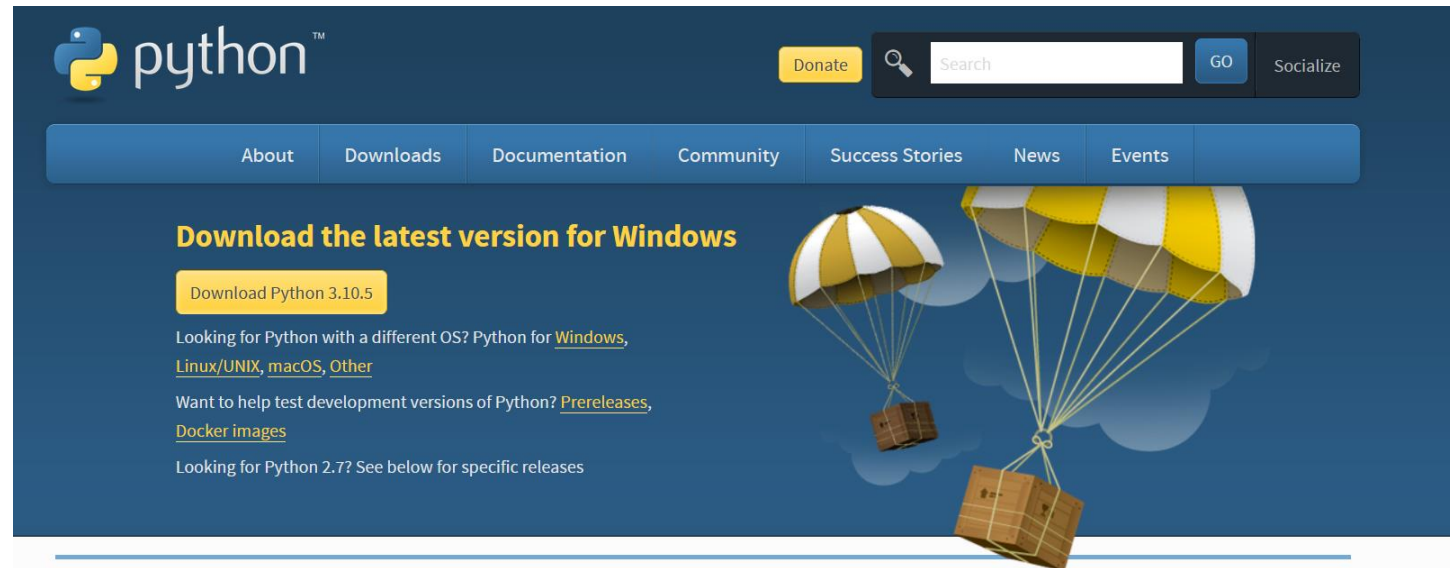
→ Overall, Python is a good **entry point** for beginners in both programming and computer science, and therefore the first language we teach in SUTD.

# Installing Python

*Install it now!*

- In this class, we will use **Python 3.10(.5)** (latest stable version as on the 12<sup>th</sup> of July 2022).
- Download it here (for 32/64-bit Windows and Mac users, Linux users can get it via apt-get or SoftwareCenter)

<https://www.python.org/downloads/>





# The Console/Terminal/Shell/Bash

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

- The console 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!)

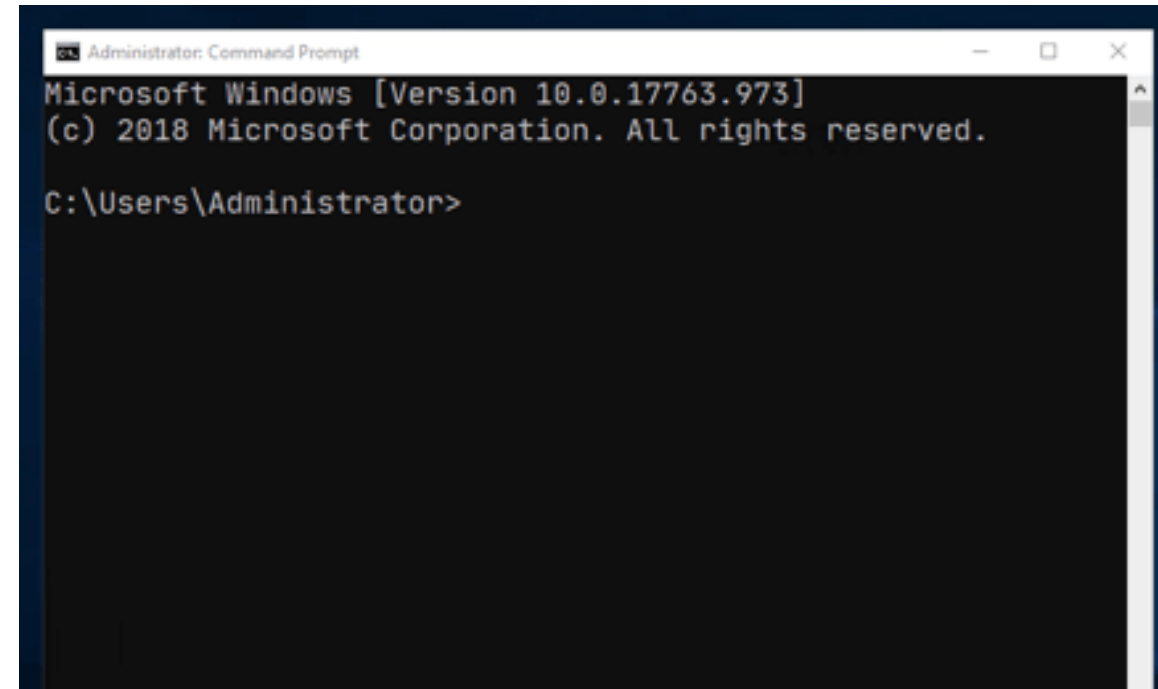




# The Console/Terminal/Shell/Bash

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

- The console is a text prompt expecting commands, which can be used to communicate with your computer.
- Nowadays, your Operating System (OS) provides an easier and more user-friendly way to communicate with your computer and “hides” the need for 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 is a text prompt expecting commands, which can be used to communicate with your computer.
- Nowadays, your Operating System (OS) provides an easier and more user-friendly way to communicate with your computer and “hides” the need for console.

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

## Hacker in movies starter pack



Green binary numbers constantly moving on screen

Smashes their keyboard for a while and says "I'm in"

"Uh oh, theres a secondary firewall"

A random progress bar which after completion says ACCESS GRANTED



"He's good but I'm better"



Wears the same black hoodie

"I'm gonna get the override codes by breaching the firewall"

Then some dumb jock/ chick says 'English Please'



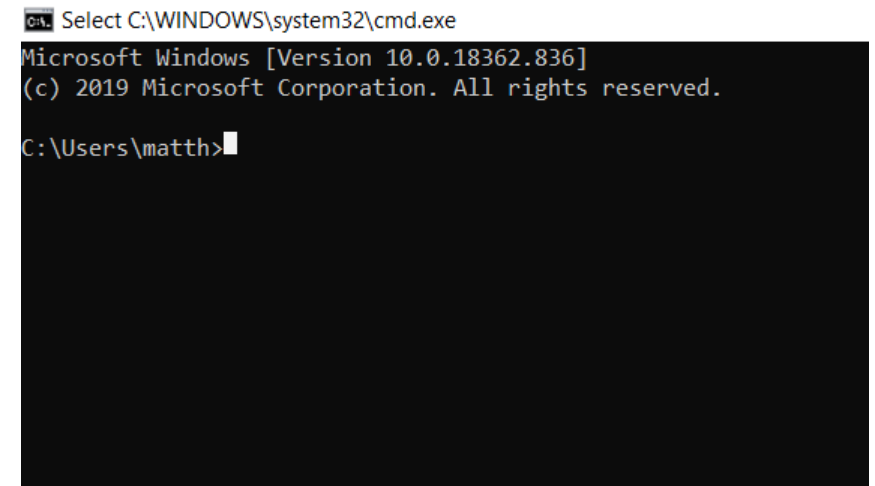
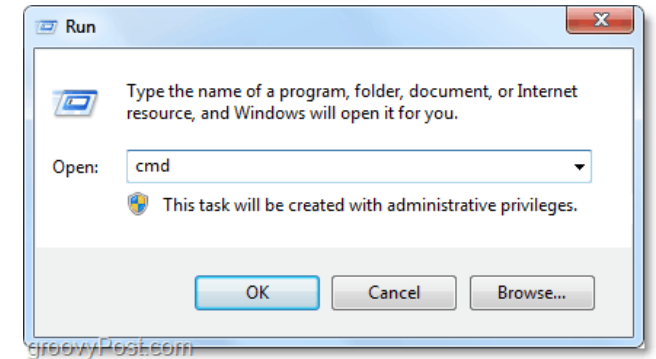
Has multiple monitors for no reason at all



A cube or a complex structure spins in the background

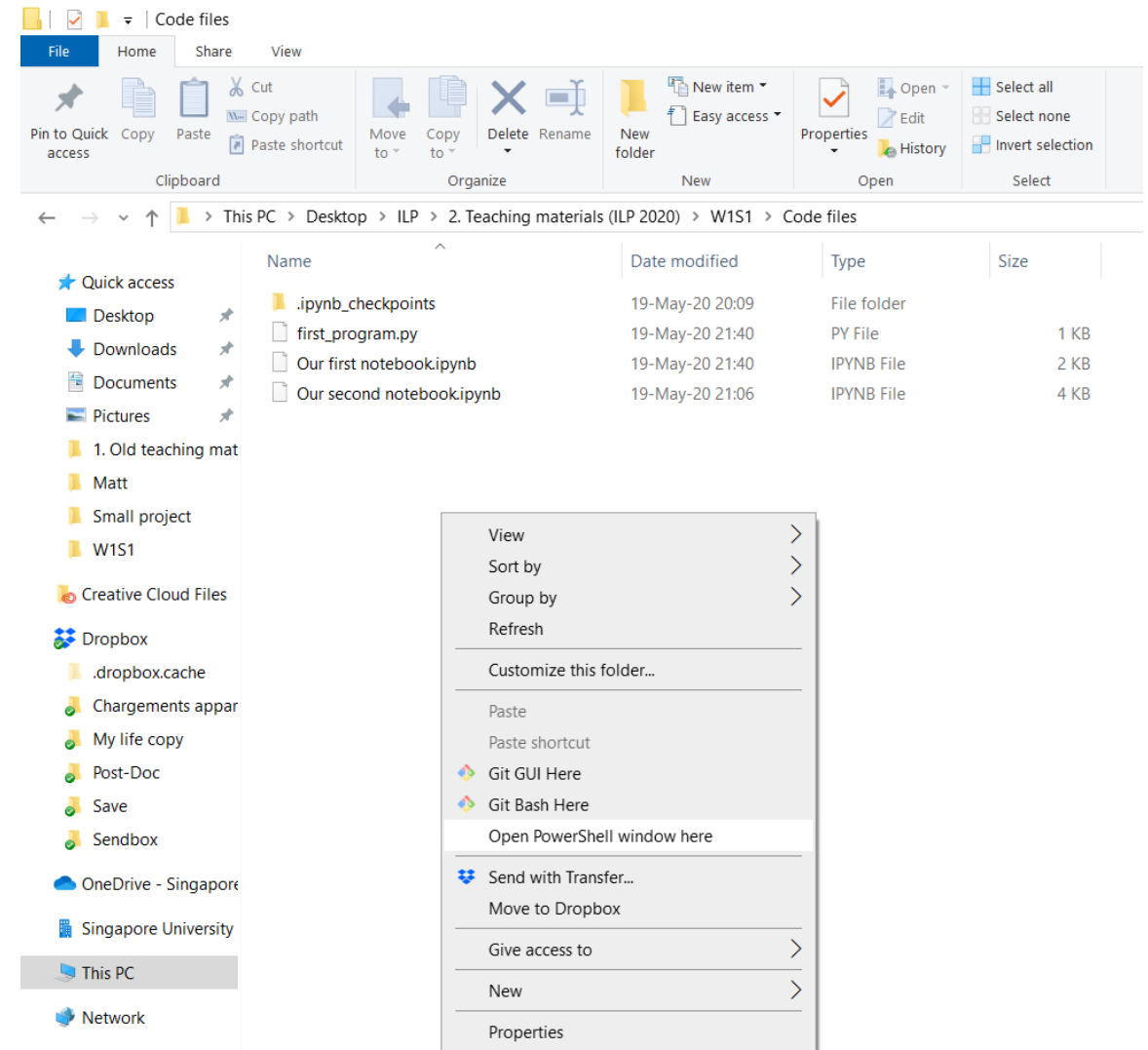
# Opening a console (Windows)

- Check your installation has completed appropriately, by trying to open a **console**.
- **Windows:** the cool way.
  1. Press simultaneously **Windows key + R**,
  2. then type **cmd** (which is short for **command console**), and press **Enter**.



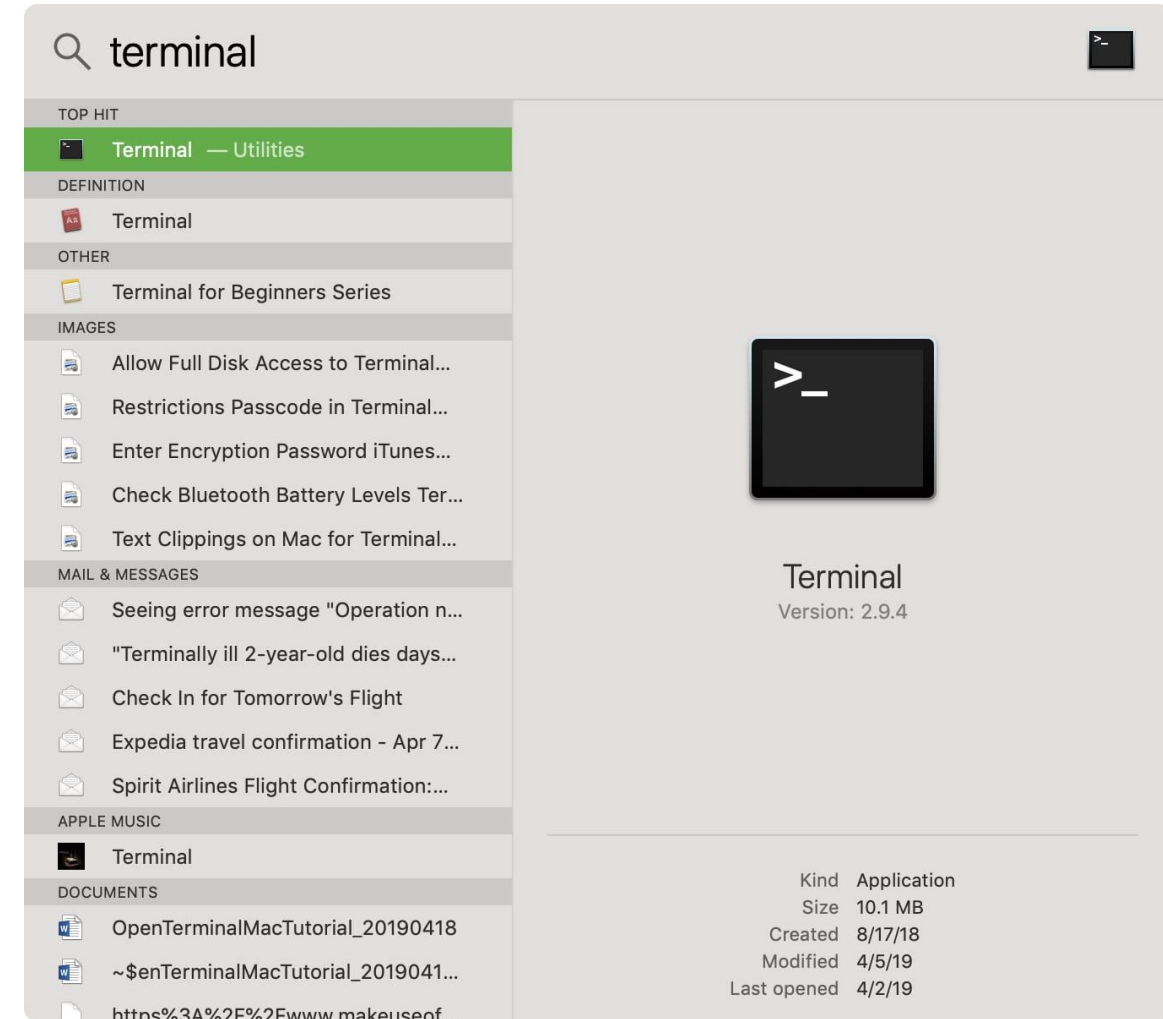
# Opening a console (Windows – option 2)

- **Windows option 2:** faster and more convenient way, in my opinion
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
  1. Press simultaneously **Command key + Space**,
  2. Then type **Terminal**,
  3. It should appear as your top result, click it



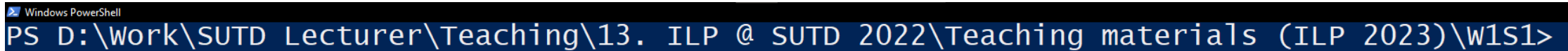
# OS mode vs. Python mode

- 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 or 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!).

→ But how do you know which mode you are in?

# 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 or 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!).

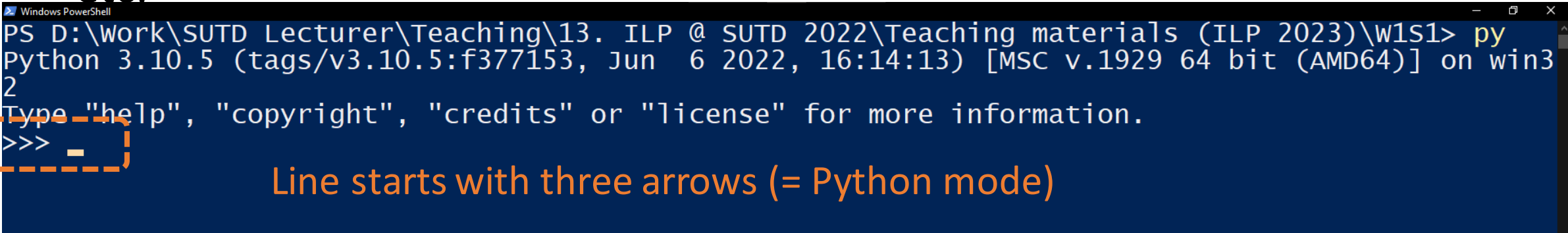


```
PS D:\Work\SUTD Lecturer\Teaching\13. ILP @ SUTD 2022\Teaching materials (ILP 2023)\w1s1>
```

Line starts with current file folder description (= OS mode)

# 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 or 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!).



```
Windows PowerShell
PS D:\Work\SUTD Lecturer\Teaching\13. ILP @ SUTD 2022\Teaching materials (ILP 2023)\w1s1> py
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> _
```

Line starts with three arrows (= Python mode)

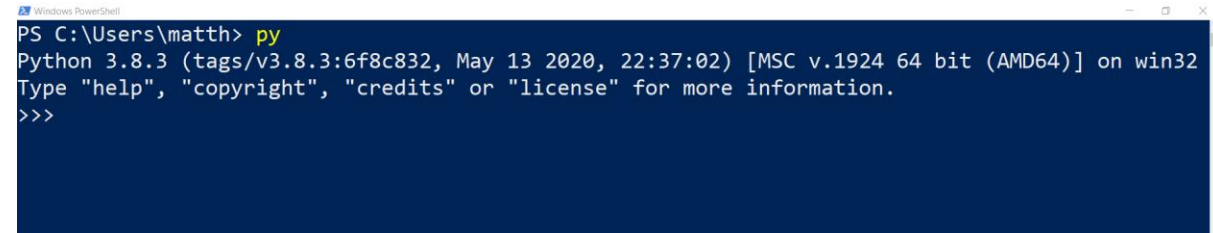


# Your first run of Python!

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

A screenshot of a Windows PowerShell terminal window. The title bar at the top reads "Windows PowerShell". The command prompt shows the user's location as "C:\Users\matth" and the command "py" has been entered. The output displays the Python version "3.8.3" and build information, followed by a prompt for help. The terminal background is dark blue.

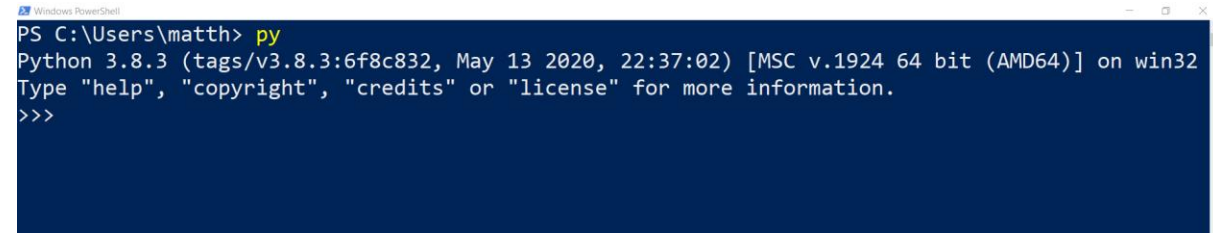
```
PS C:\Users\matth> py
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

# Your first run of Python!

Start Python by **typing one** of the following commands in the console, while in OS mode and press **Enter**!

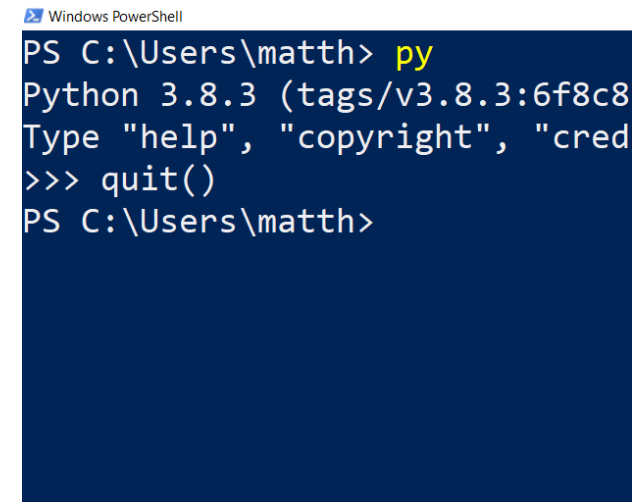
(Note: one of these should work, might vary depending on your machine!)

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



```
Windows PowerShell
PS C:\Users\matth> py
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

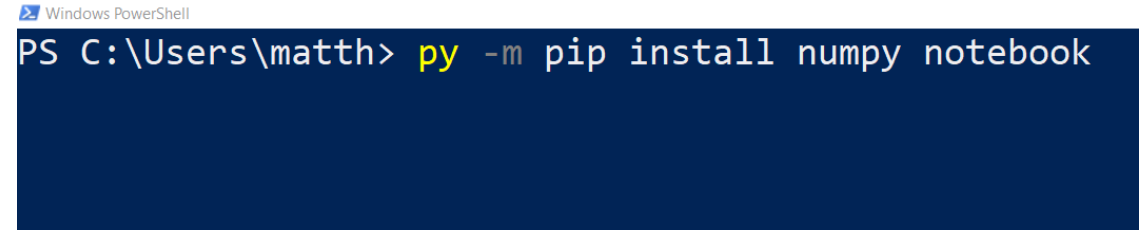
To exit the Python environment and go back to your OS console, simply type `quit()` and press enter.



```
Windows PowerShell
PS C:\Users\matth> py
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()
PS C:\Users\matth>
```

# Installing & updating packages in Python

- Next, let us install **packages**.  
(**Packages** are extra functionalities for Python.)

A screenshot of a Windows PowerShell terminal window. The title bar at the top reads "Windows PowerShell". The command prompt shows the path "C:\Users\matth>" followed by the command "py -m pip install numpy notebook". The text is white on a dark blue background.

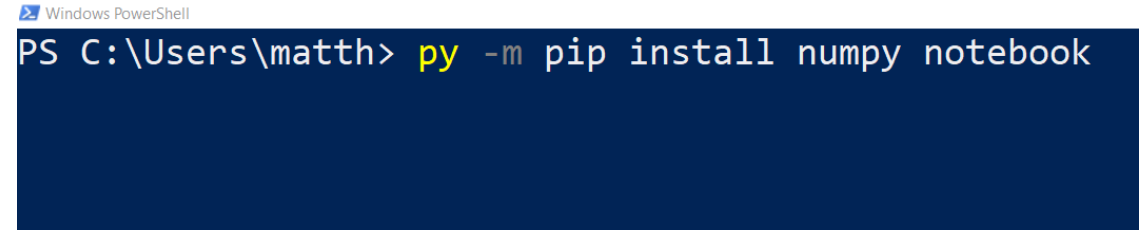
```
Windows PowerShell
PS C:\Users\matth> py -m pip install numpy notebook
```

# Installing & updating packages in Python

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

**py -m pip install numpy notebook**

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



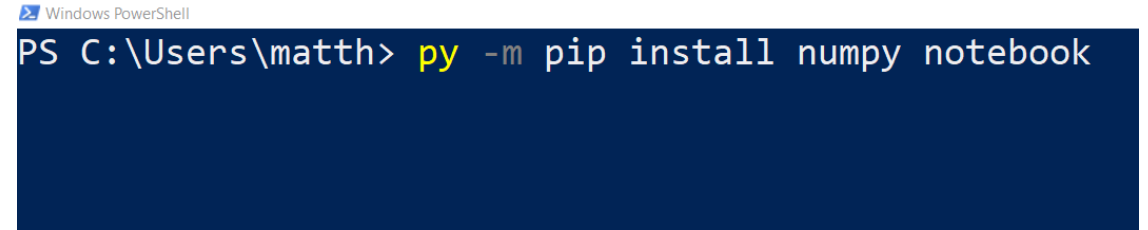
```
Windows PowerShell
PS C:\Users\matth> py -m pip install numpy notebook
```

# Installing & updating packages in Python

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

**py -m pip install numpy notebook**

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

A screenshot of a Windows PowerShell terminal window. The title bar at the top says "Windows PowerShell". The command prompt shows "PS C:\Users\matth> py -m pip install numpy notebook". The text is white on a dark blue background.

It shall run for a while, download and install a few things...  
(numpy for advanced math computation and notebook, which we will use later on)

# Our first program!

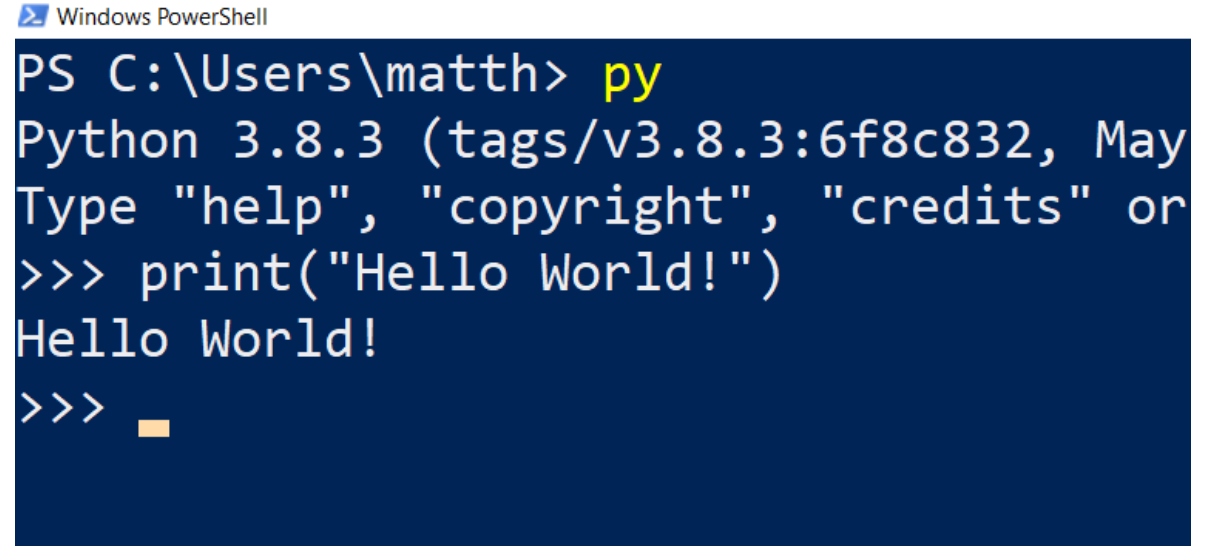
- Start python in a console, type **print("Hello World!")**, and submit by pressing Enter. It should display "Hello World!".

Windows PowerShell

```
PS C:\Users\matth> py
Python 3.8.3 (tags/v3.8.3:6f8c832, May
Type "help", "copyright", "credits" or
>>> print("Hello World!")
Hello World!
>>> █
```

# Our first program!

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

A screenshot of a Windows PowerShell terminal window. The title bar at the top reads "Windows PowerShell". The command prompt shows the user at the C:\Users\matth directory, having typed 'py' to launch Python. The Python version 3.8.3 is displayed, along with some version information. The user has entered the command 'print("Hello World!")' and the output 'Hello World!' is shown. The prompt is now ready for the next command.

```
PS C:\Users\matth> py
Python 3.8.3 (tags/v3.8.3:6f8c832, May
Type "help", "copyright", "credits" or
>>> print("Hello World!")
Hello World!
>>> 
```

# Our first program!

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

```
Windows PowerShell
PS C:\Users\matth> py
Python 3.8.3 (tags/v3.8.3:6f8c832, May
Type "help", "copyright", "credits" or
>>> print("Hello World!")
Hello World!
>>> █
```

When your code outputs "Hello World!"





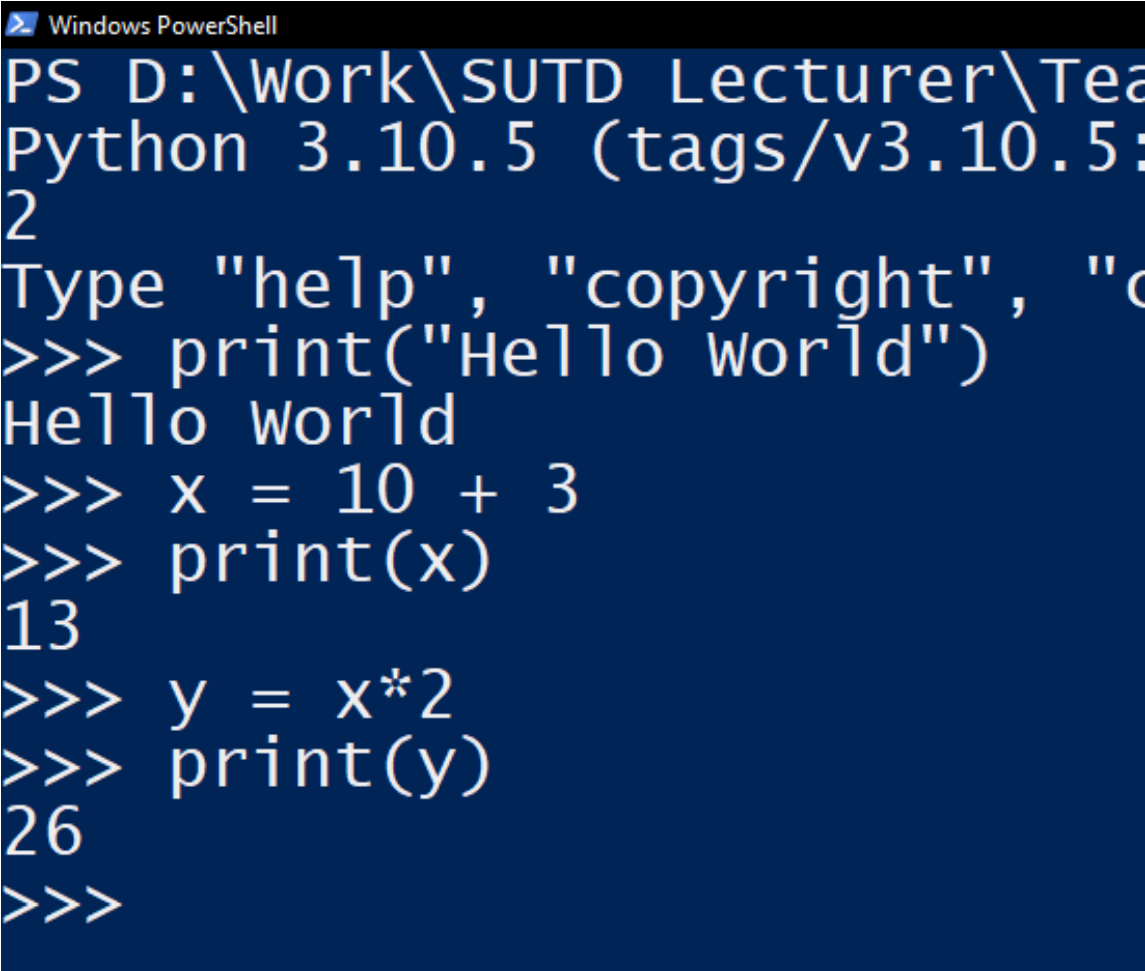
# Our first program!

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

```
Windows PowerShell
PS D:\Work\SUTD Lecturer\Teaching\13. ILP @ SUTD 2022\Teaching material\
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929
2
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world")
Hello world
>>> x = 10 + 3
>>> print(x)
13
>>> y = x*2
>>> print(y)
26
>>>
```

# Our first program!

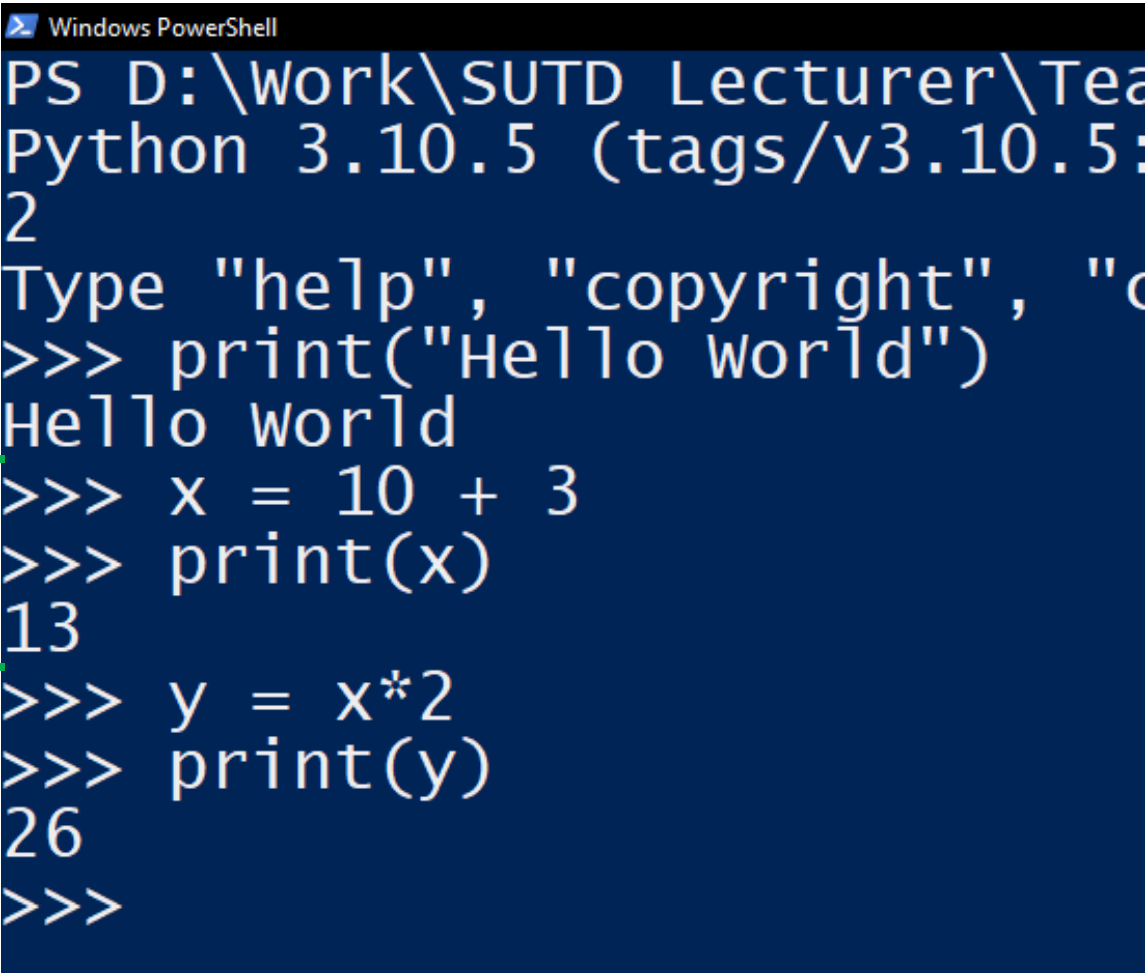
- Assigning something to memory is done with the **=** sign.
- **(Note:** The **=** sign does not have the same meaning as in mathematics.)
- In computer science it means:
  - Assign what is on the right-hand side of the equal sign to memory.
  - The name of this element, called a **variable**, consists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



```
Windows PowerShell
PS D:\Work\SUTD Lecturer\Tea
Python 3.10.5 (tags/v3.10.5
2
Type "help", "copyright", "c
>>> print("Hello World")
Hello World
>>> x = 10 + 3
>>> print(x)
13
>>> y = x*2
>>> print(y)
26
>>>
```

# Our first program!

- **Here**, we have assigned the numerical value resulting from the operation **10 + 3**, to a **variable** named **x**.

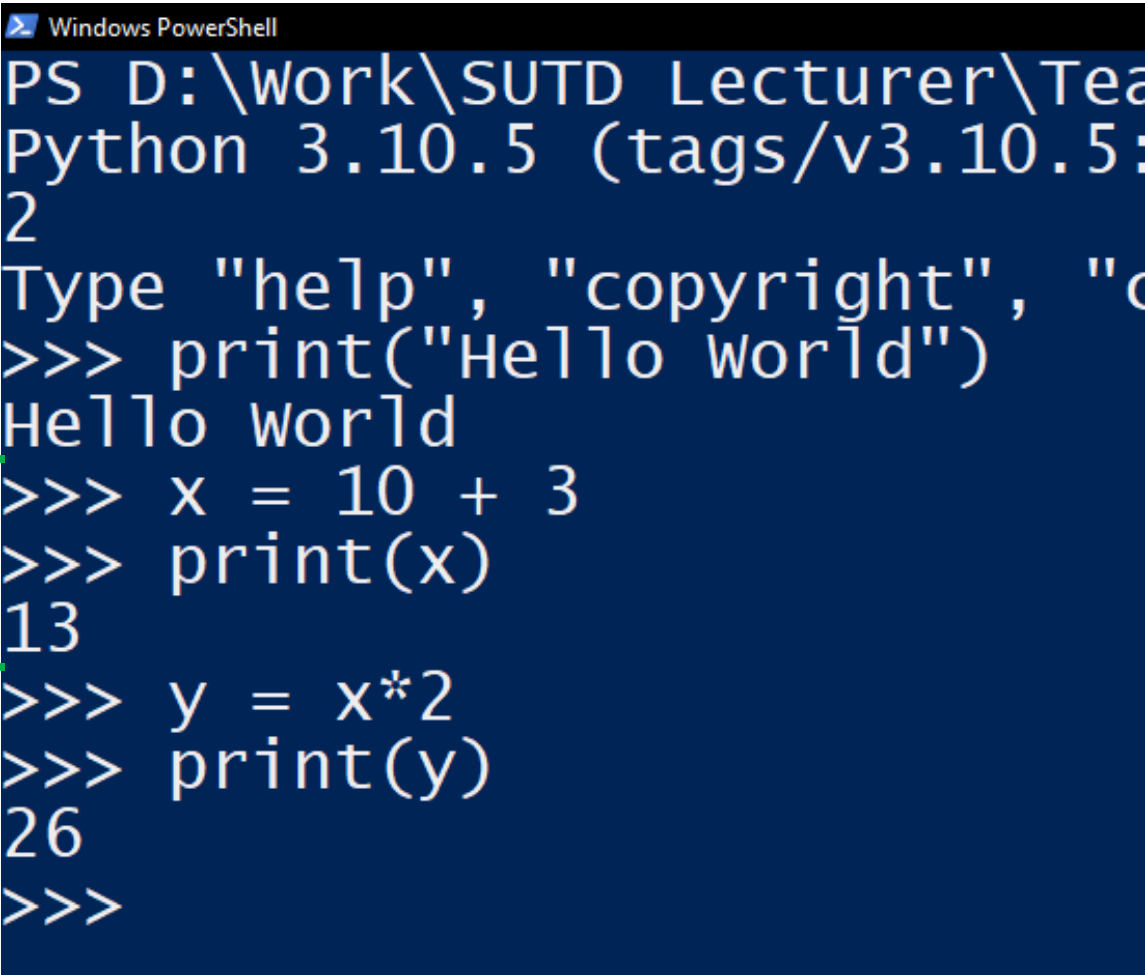


A screenshot of a Windows PowerShell terminal window with a dark blue background and white text. The window title is 'Windows PowerShell'. The prompt is 'PS D:\Work\SUTD Lecturer\Tea'. The user has entered 'Python 3.10.5 (tags/v3.10.5:2' and the prompt has moved to the next line. The user then enters 'Type "help", "copyright", "c' and the prompt moves to the next line. The user enters '>>> print("Hello World")' and the output 'Hello World' is displayed. The user enters '>>> x = 10 + 3' and the prompt moves to the next line. The user enters '>>> print(x)' and the output '13' is displayed. The user enters '>>> y = x\*2' and the prompt moves to the next line. The user enters '>>> print(y)' and the output '26' is displayed. The prompt '>>>' is shown at the bottom. A green bracket is drawn on the left side of the terminal, grouping the lines from '>>> x = 10 + 3' to '>>> print(y)'. The bracket is open on the left and closed on the right, pointing to the assignment and subsequent operations.

```
Windows PowerShell
PS D:\Work\SUTD Lecturer\Tea
Python 3.10.5 (tags/v3.10.5:
2
Type "help", "copyright", "c
>>> print("Hello World")
Hello World
>>> x = 10 + 3
>>> print(x)
13
>>> y = x*2
>>> print(y)
26
>>>
```

# 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 print it on screen for us, with the **print()** function.

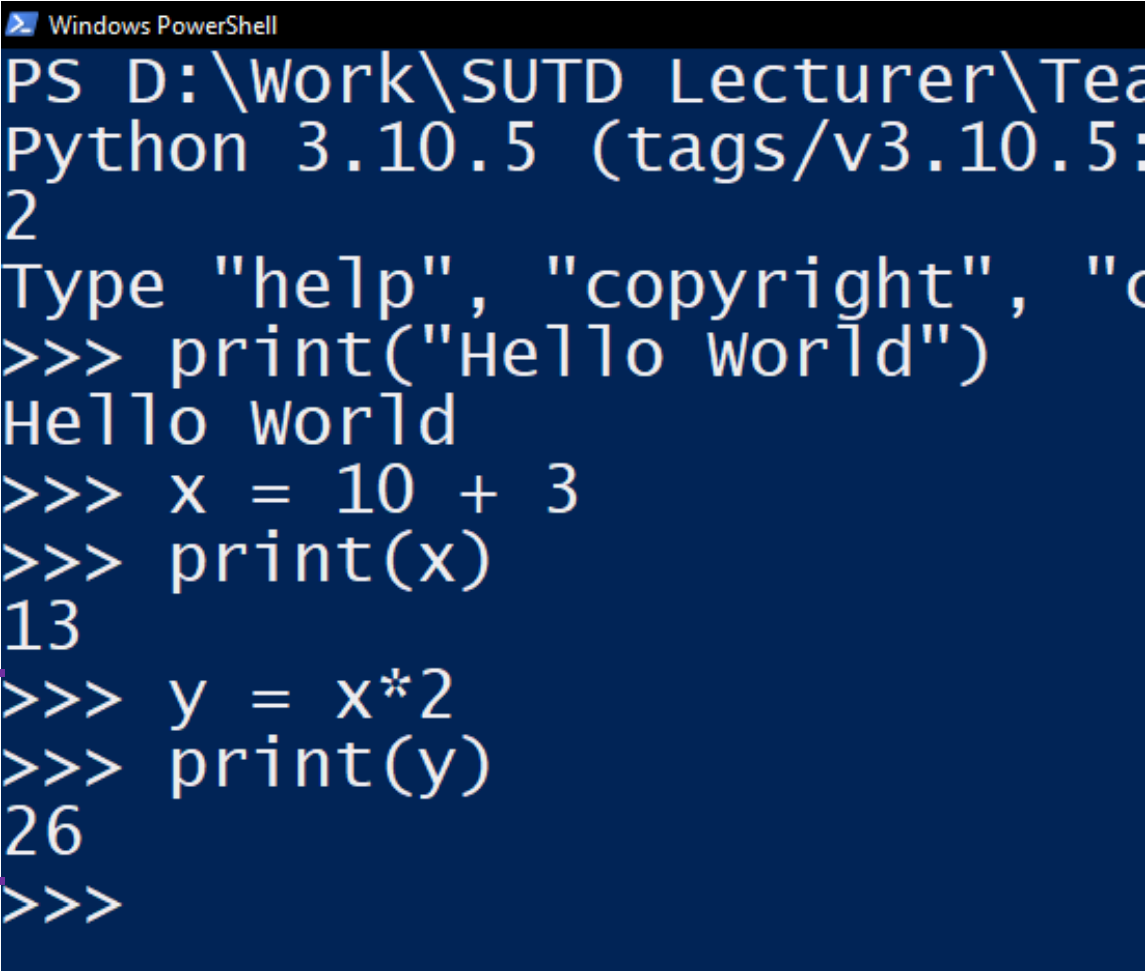


A screenshot of a Windows PowerShell terminal window with a dark blue background and white text. The window title is 'Windows PowerShell'. The prompt is 'PS D:\Work\SUTD Lecturer\Tea'. The user has entered 'Python 3.10.5 (tags/v3.10.5:2' and the prompt has moved to the next line. The user then enters 'Type "help", "copyright", "c' and the prompt moves to the next line. The user enters '>>> print("Hello World")' and the output 'Hello World' is displayed. The user enters '>>> x = 10 + 3' and the prompt moves to the next line. The user enters '>>> print(x)' and the output '13' is displayed. The user enters '>>> y = x\*2' and the prompt moves to the next line. The user enters '>>> print(y)' and the output '26' is displayed. The prompt '>>>' is shown on the final line. A green bracket is drawn on the left side of the terminal, grouping the lines from '>>> x = 10 + 3' to '>>> print(y)'. The text 'Hello World' is also highlighted in green.

```
PS D:\Work\SUTD Lecturer\Tea
Python 3.10.5 (tags/v3.10.5:
2
Type "help", "copyright", "c
>>> print("Hello World")
Hello World
>>> x = 10 + 3
>>> print(x)
13
>>> y = x*2
>>> print(y)
26
>>>
```

# 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 print it on screen for us, with the **print()** function.
- Or even reuse it in **other calculations!**

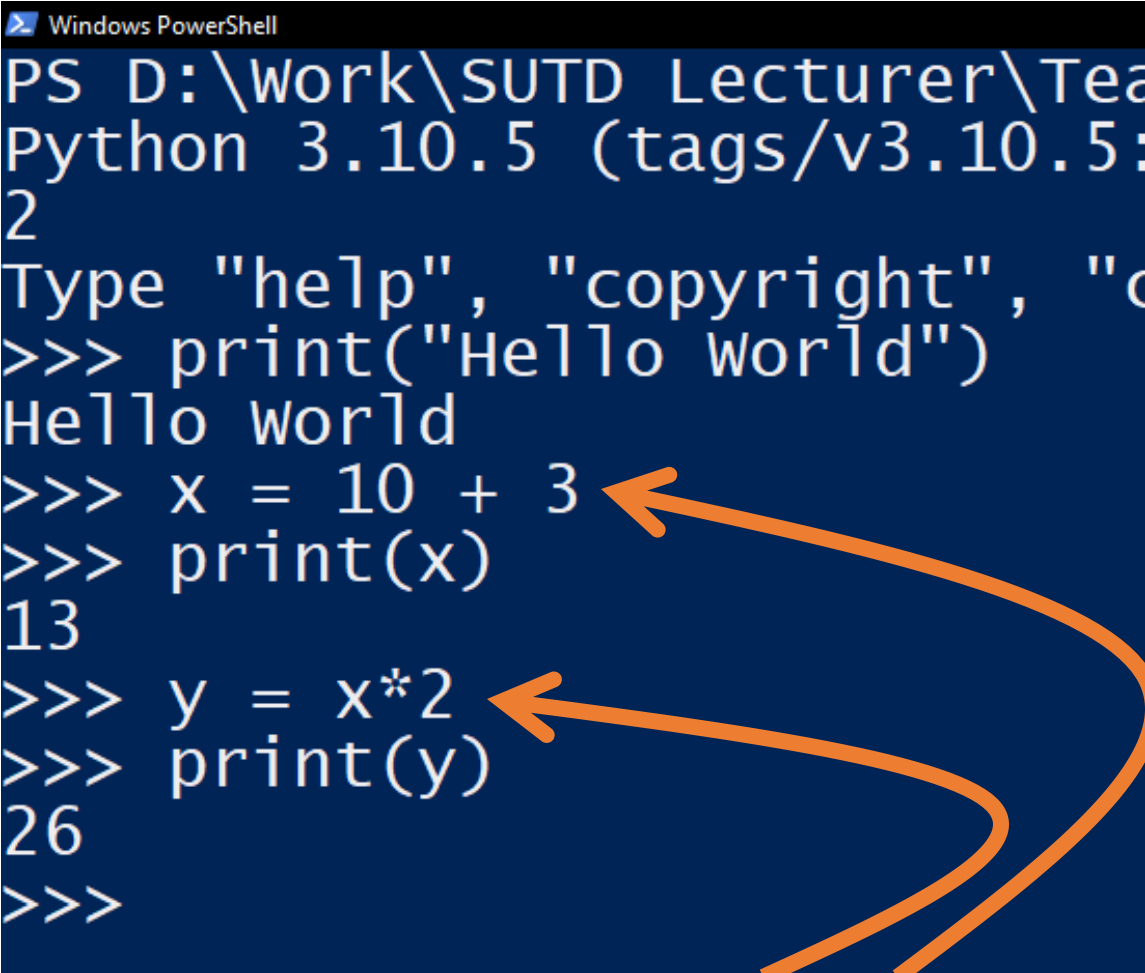


```
Windows PowerShell
PS D:\Work\SUTD Lecturer\Tea
Python 3.10.5 (tags/v3.10.5
2
Type "help", "copyright", "c
>>> print("Hello World")
Hello World
>>> x = 10 + 3
>>> print(x)
13
>>> y = x*2
>>> print(y)
26
>>>
```

A screenshot of a Windows PowerShell terminal window. The title bar reads 'Windows PowerShell'. The command prompt shows the directory 'D:\Work\SUTD Lecturer\Tea' and the Python version 'Python 3.10.5 (tags/v3.10.5... 2'. The user has entered several Python commands: 'print("Hello World")' which outputs 'Hello World', 'x = 10 + 3' followed by 'print(x)' which outputs '13', and 'y = x\*2' followed by 'print(y)' which outputs '26'. The prompt '>>>' is visible at the end of the last line. A purple bracket is drawn on the left side of the terminal, grouping the last three lines of code.

# 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 print it on screen for us, with the **print()** function.
- Or even reuse it in **other calculations!**

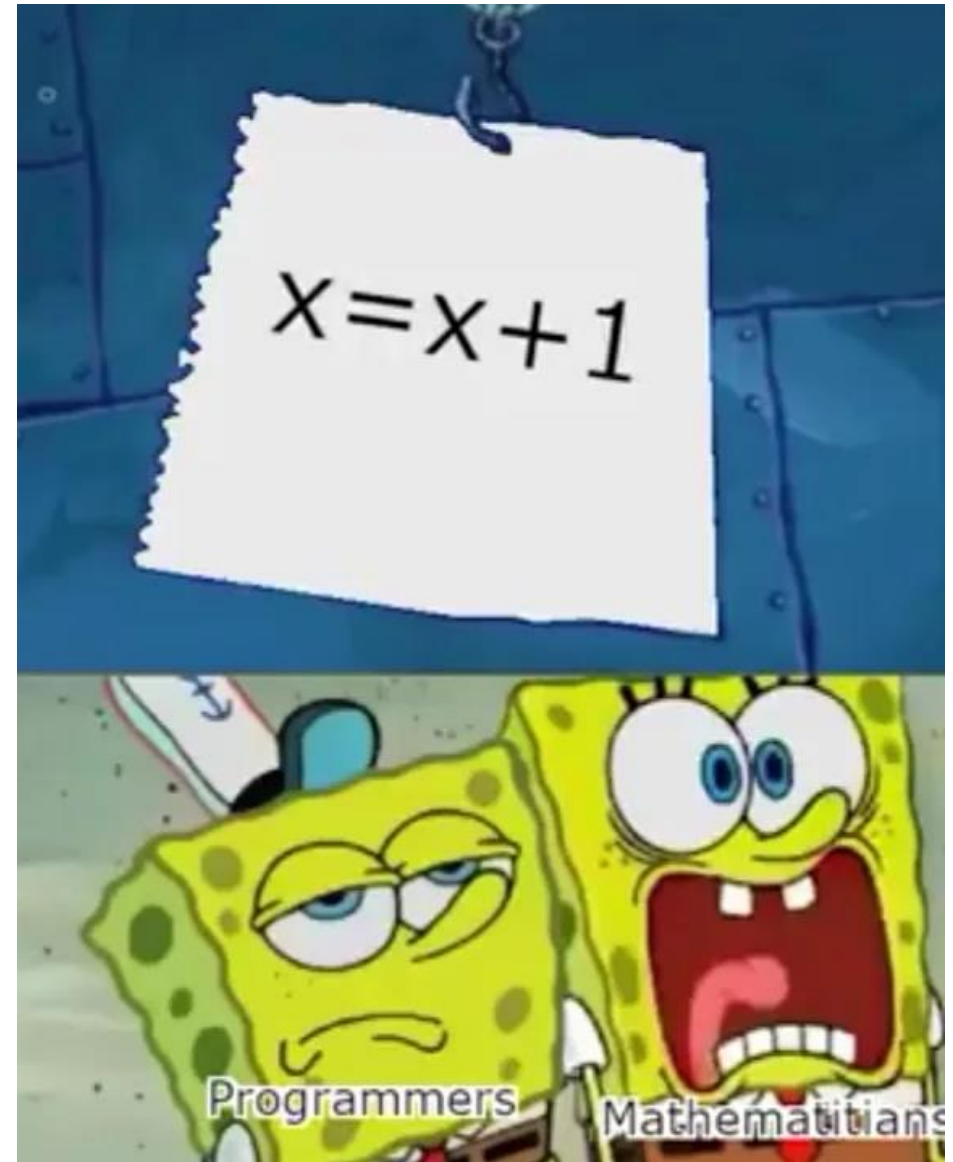


```
Windows PowerShell
PS D:\Work\SUTD Lecturer\Tea
Python 3.10.5 (tags/v3.10.5:
2
Type "help", "copyright", "c
>>> print("Hello World")
Hello World
>>> x = 10 + 3
>>> print(x)
13
>>> y = x*2
>>> print(y)
26
>>>
```

**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.
- **(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 #1

**Matt's Great Advice #1: 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 on eDimension.
- More specifically, we will now use the **first\_program.py** file.

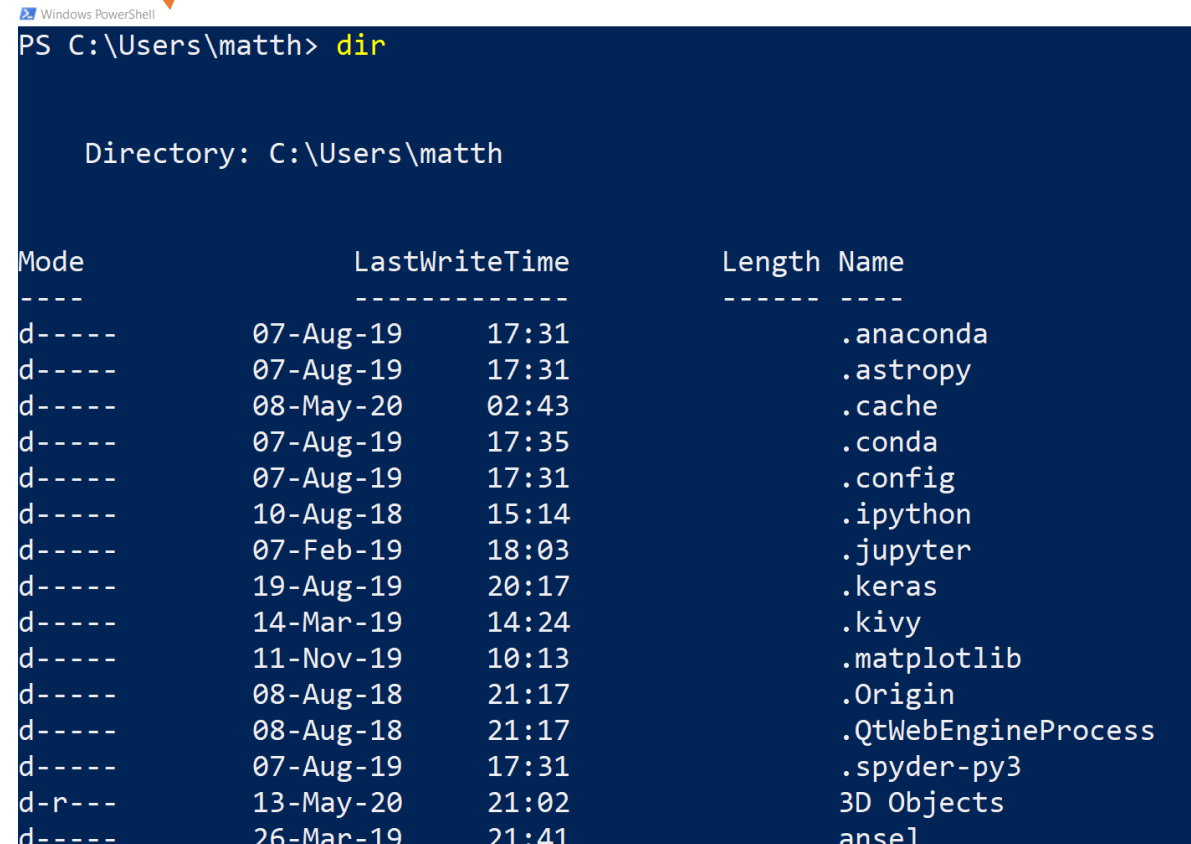
# Executing a .py file in console mode

- You should have downloaded a few files along with the lecture notes on eDimension.
- More specifically, we will now use the **first\_program.py** file.
- Identify where your **first\_program.py** file 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\



The screenshot shows a Windows PowerShell window with the title bar 'Windows PowerShell'. The prompt is 'PS C:\Users\matth>' and the command 'dir' has been entered. The output shows the directory 'C:\Users\matth' and a list of files and folders. An orange arrow points from the text 'Current location is C:\Users\matth\' to the prompt in the PowerShell window.

```
Directory: C:\Users\matth
```

Mode	LastWriteTime	Length	Name
d----	07-Aug-19 17:31		.anaconda
d----	07-Aug-19 17:31		.astropy
d----	08-May-20 02:43		.cache
d----	07-Aug-19 17:35		.conda
d----	07-Aug-19 17:31		.config
d----	10-Aug-18 15:14		.ipython
d----	07-Feb-19 18:03		.jupyter
d----	19-Aug-19 20:17		.keras
d----	14-Mar-19 14:24		.kivy
d----	11-Nov-19 10:13		.matplotlib
d----	08-Aug-18 21:17		.Origin
d----	08-Aug-18 21:17		.QtWebEngineProcess
d----	07-Aug-19 17:31		.spyder-py3
d-r--	13-May-20 21:02		3D Objects
d----	26-Mar-19 21:41		ansel

# 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**.

Observe how the current location is changing every time.

Select Windows PowerShell

```
PS C:\Users\matth> cd Desktop
PS C:\Users\matth\Desktop> cd ..
PS C:\Users\matth> cd Downloads
PS C:\Users\matth\Downloads> _
```

# 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.

Select Windows PowerShell

```
PS C:\Users\matth> cd Desktop
PS C:\Users\matth\Desktop> cd ..
PS C:\Users\matth> cd Downloads
PS C:\Users\matth\Downloads> _
```

```
PS C:\Users\matth> cd Desktop
PS C:\Users\matth\Desktop> cd ILP
PS C:\Users\matth\Desktop\ILP> cd '..\2. Teaching materials (ILP 2020)\'
PS C:\Users\matth\Desktop\ILP\2. Teaching materials (ILP 2020)> cd '..\W1S1\'
PS C:\Users\matth\Desktop\ILP\2. Teaching materials (ILP 2020)\W1S1> cd '..\Code files\'
PS C:\Users\matth\Desktop\ILP\2. Teaching materials (ILP 2020)\W1S1\Code files> dir

Directory: C:\Users\matth\Desktop\ILP\2. Teaching materials (ILP 2020)\W1S1\Code files

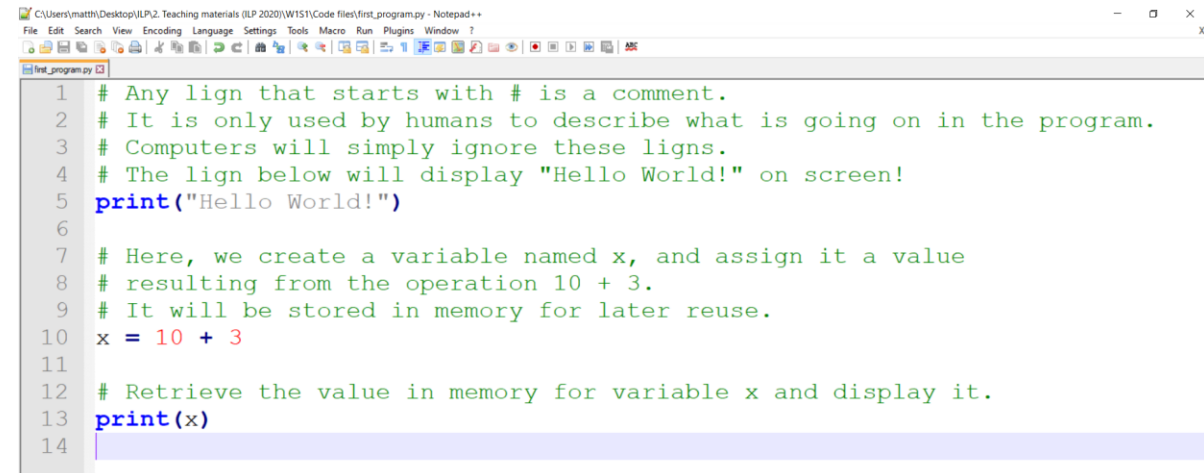
Mode                LastWriteTime         Length Name
----                -
d-----          19-May-20    20:09             .ipynb_checkpoints
-a-----          19-May-20    21:40           469 first_program.py
-a-----          19-May-20    21:40        1810 Our first notebook.ipynb
-a-----          19-May-20    21:06        3205 Our second notebook.ipynb

PS C:\Users\matth\Desktop\ILP\2. Teaching materials (ILP 2020)\W1S1\Code files>
```

→ 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).
- Recognize the code we used earlier.

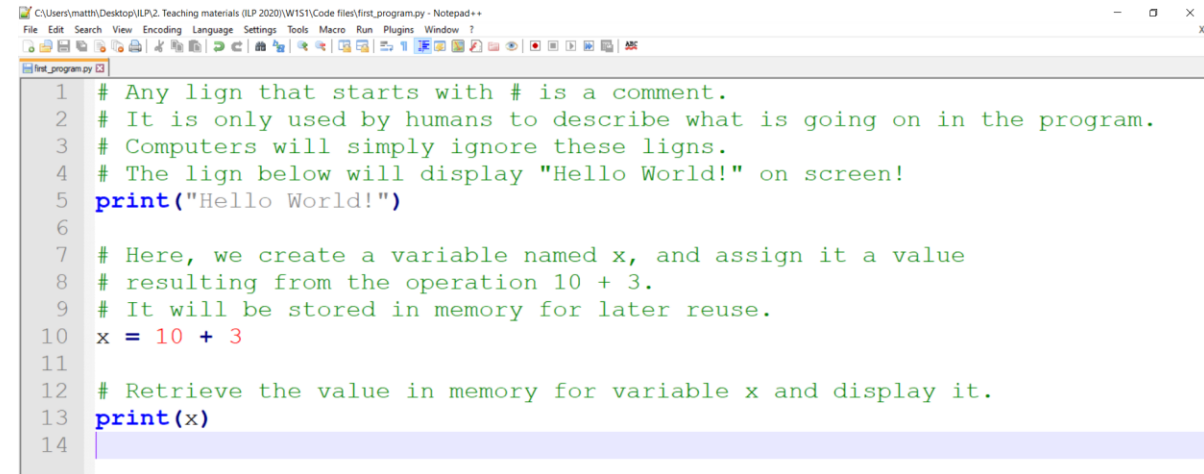


```
1 # Any line that starts with # is a comment.
2 # It is only used by humans to describe what is going on in the program.
3 # Computers will simply ignore these lines.
4 # The line below will display "Hello World!" on screen!
5 print("Hello World!")
6
7 # Here, we create a variable named x, and assign it a value
8 # resulting from the operation 10 + 3.
9 # It will be stored in memory for later reuse.
10 x = 10 + 3
11
12 # Retrieve the value in memory for variable x and display it.
13 print(x)
14
```

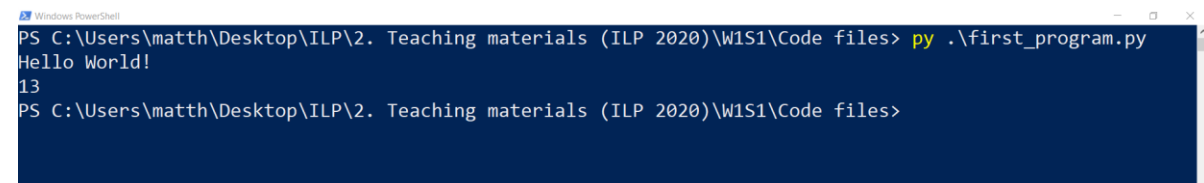
# Checking your .py file

- Open your **first\_program.py** file with any text editor (specifically do it by right clicking and asking to open with a text editor).
- 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**



```
1 # Any line that starts with # is a comment.
2 # It is only used by humans to describe what is going on in the program.
3 # Computers will simply ignore these lines.
4 # The line below will display "Hello World!" on screen!
5 print("Hello World!")
6
7 # Here, we create a variable named x, and assign it a value
8 # resulting from the operation 10 + 3.
9 # It will be stored in memory for later reuse.
10 x = 10 + 3
11
12 # Retrieve the value in memory for variable x and display it.
13 print(x)
14
```



```
PS C:\Users\matth\Desktop\ILP\2. Teaching materials (ILP 2020)\W1S1\Code files> py .\first_program.py
Hello World!
13
PS C:\Users\matth\Desktop\ILP\2. Teaching materials (ILP 2020)\W1S1\Code files>
```



# Running Python from an IDE

- **Problem:** Typing code in a text editor and running it from console is not exactly convenient...

# Running Python from an IDE

- **Problem:** Typing code in a text editor and running it from console is not exactly convenient...
- **Suggestion:** we should use an **Interactive Development Environment (IDE)**, which makes the coding easier for us.

# Running Python from an IDE

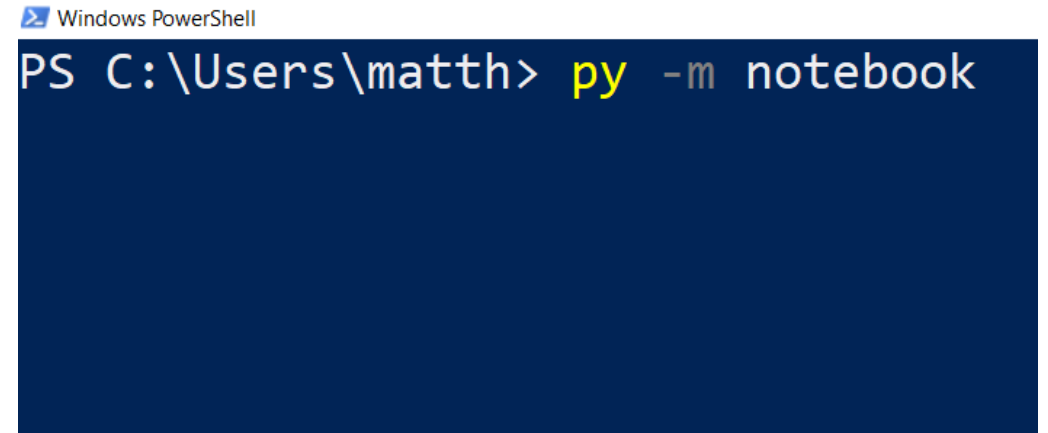
- **Problem:** Typing code in a text editor and running it from console is not exactly convenient...
- **Suggestion:** we should use an **Interactive Development Environment (IDE)**, which makes the coding easier for us.
- In this course, I suggest to use **Jupyter Notebook**, but you might look online for other IDEs if you want!

# Running Python from an IDE, such as a Jupyter Notebook

- Return to your console, **outside** of the Python environment (use **quit()** if needed).

# Running Python from an IDE, such as a Jupyter Notebook

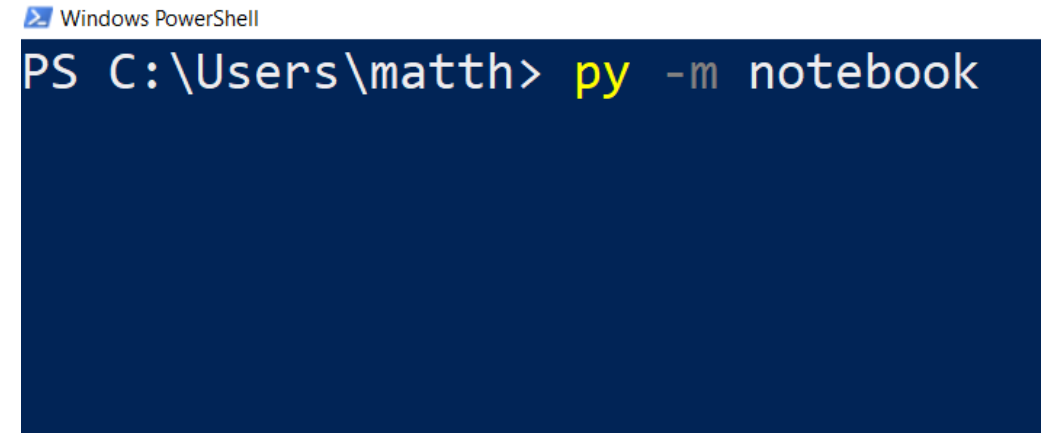
- Return to your console, **outside** of the Python environment (use **quit()** if needed).
- Type **py -m notebook**, and press enter to submit and call the **Python notebook module** (-m notebook)



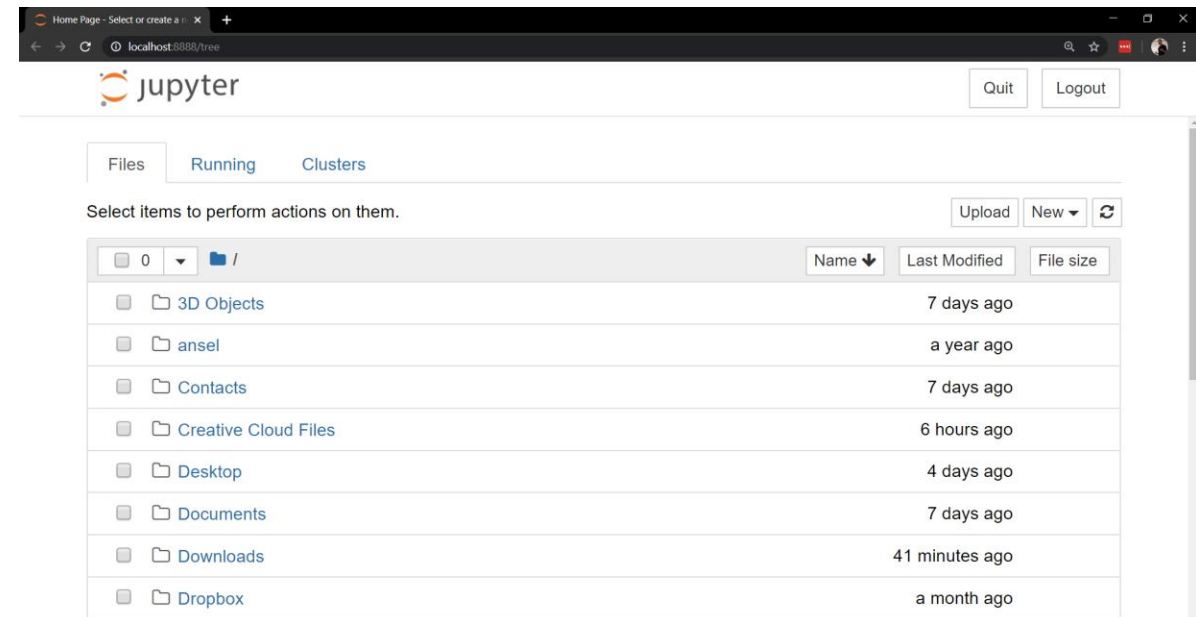
```
Windows PowerShell
PS C:\Users\matth> py -m notebook
```

# Running Python from an IDE, such as a Jupyter Notebook

- Return to your console, **outside** of the Python environment (use **quit()** if needed).
- Type **py -m notebook**, and press enter to submit and call the **Python notebook module (-m notebook)**
- **It should open a notebook window/tab in your web browser.**

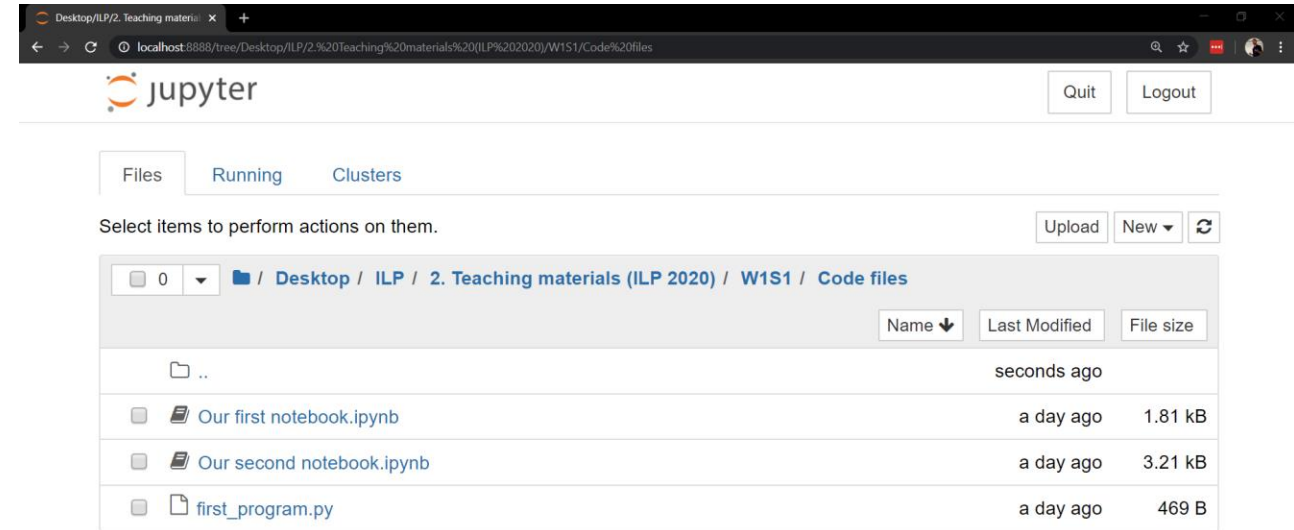


```
Windows PowerShell
PS C:\Users\matth> py -m notebook
```



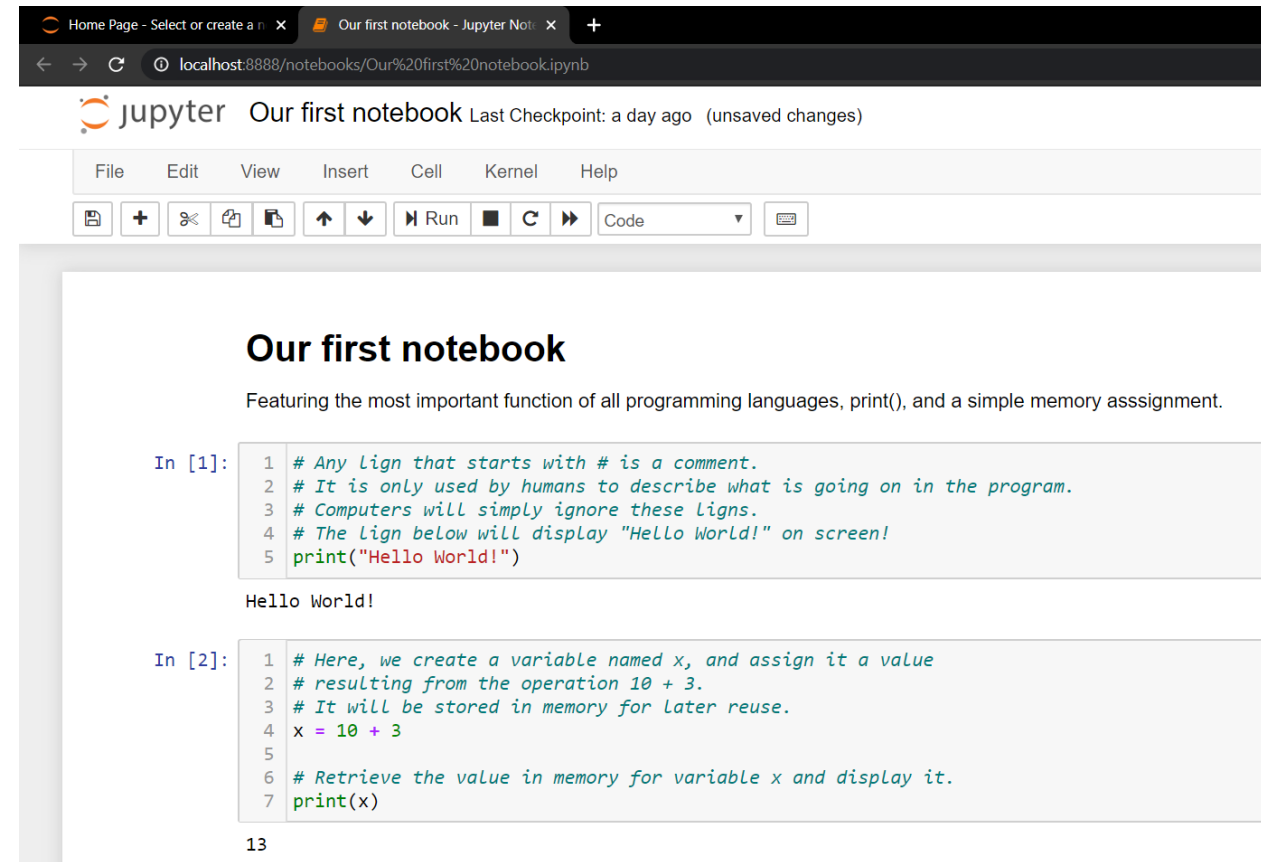
# Running Python from an IDE, such as a Jupyter Notebook

- Notebooks are a more convenient way to program on Python.
- They provide an explorer to navigate to a folder of your choice.
- Move to the folder where the code you downloaded is.



# Running Python from an IDE, such as a Jupyter Notebook

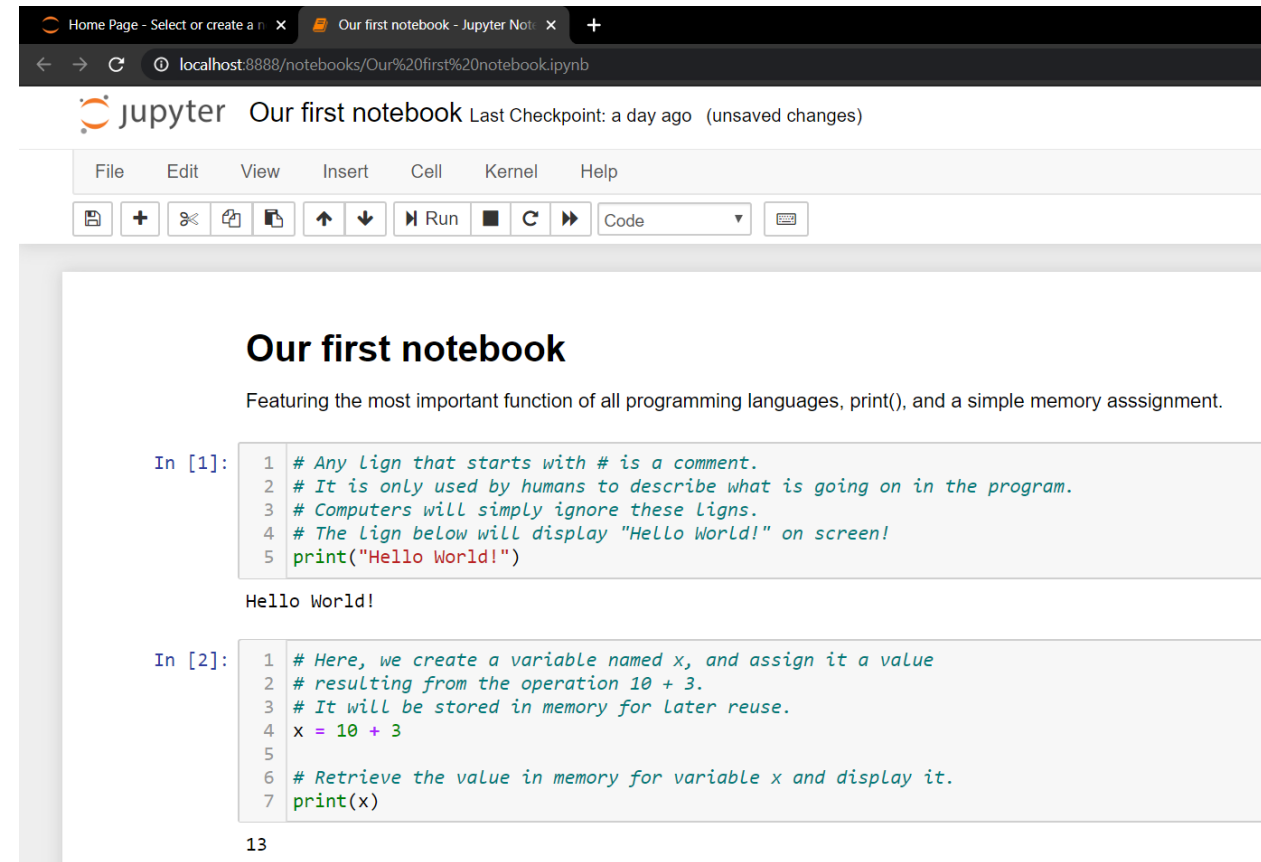
- Notebooks provide a mixed combination of
  - **text blocks** (in Markdown language)
  - and **code blocks** (in Python, these blocks have a “In [...]” on their left side).





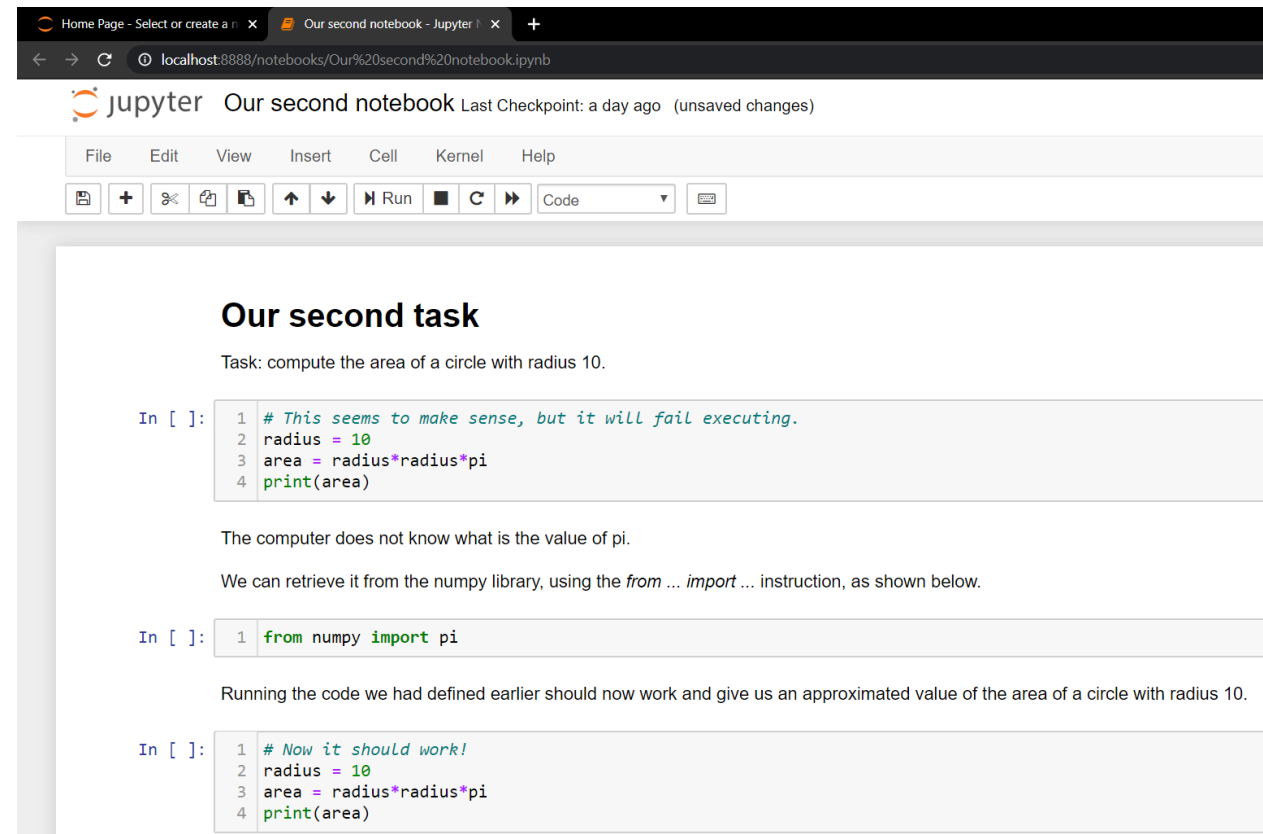
# Running Python from an IDE, such as a Jupyter Notebook

- Notebooks provide a mixed combination of
  - **text blocks** (in Markdown language)
  - 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

- Let us consider a **second task**: compute the area of a circle with radius 10.
- Open the second notebook.



Home Page - Select or create a notebook | Our second notebook - Jupyter | +

localhost:8888/notebooks/Our%20second%20notebook.ipynb

jupyter Our second notebook Last Checkpoint: a day ago (unsaved changes)

File Edit View Insert Cell Kernel Help

+ %< > Run [ ] Code [ ]

### Our second task

Task: compute the area of a circle with radius 10.

```
In [ ]: 1 # This seems to make sense, but it will fail executing.
        2 radius = 10
        3 area = radius*radius*pi
        4 print(area)
```

The computer does not know what is the value of pi.

We can retrieve it from the numpy library, using the *from ... import ...* instruction, as shown below.

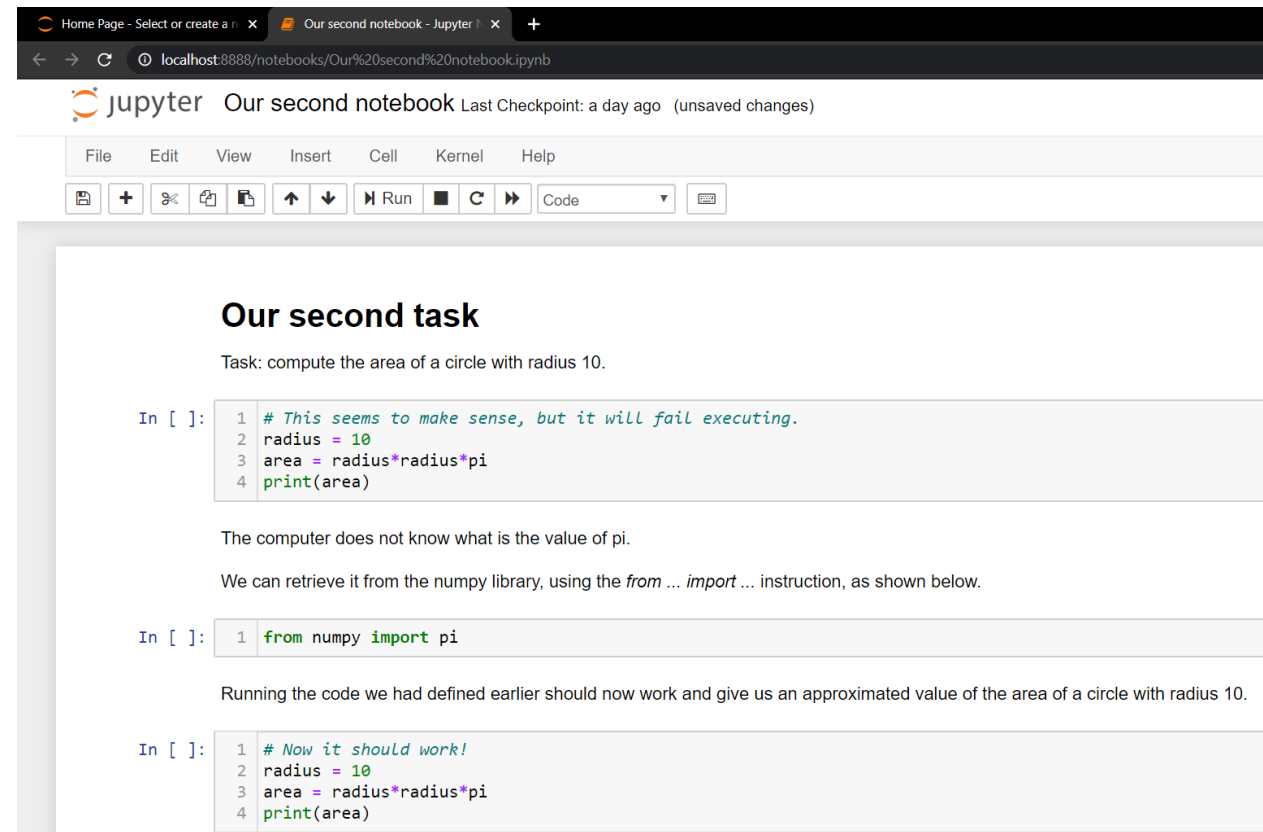
```
In [ ]: 1 from numpy import pi
```

Running the code we had defined earlier should now work and give us an approximated value of the area of a circle with radius 10.

```
In [ ]: 1 # Now it should work!
        2 radius = 10
        3 area = radius*radius*pi
        4 print(area)
```

# Our second task

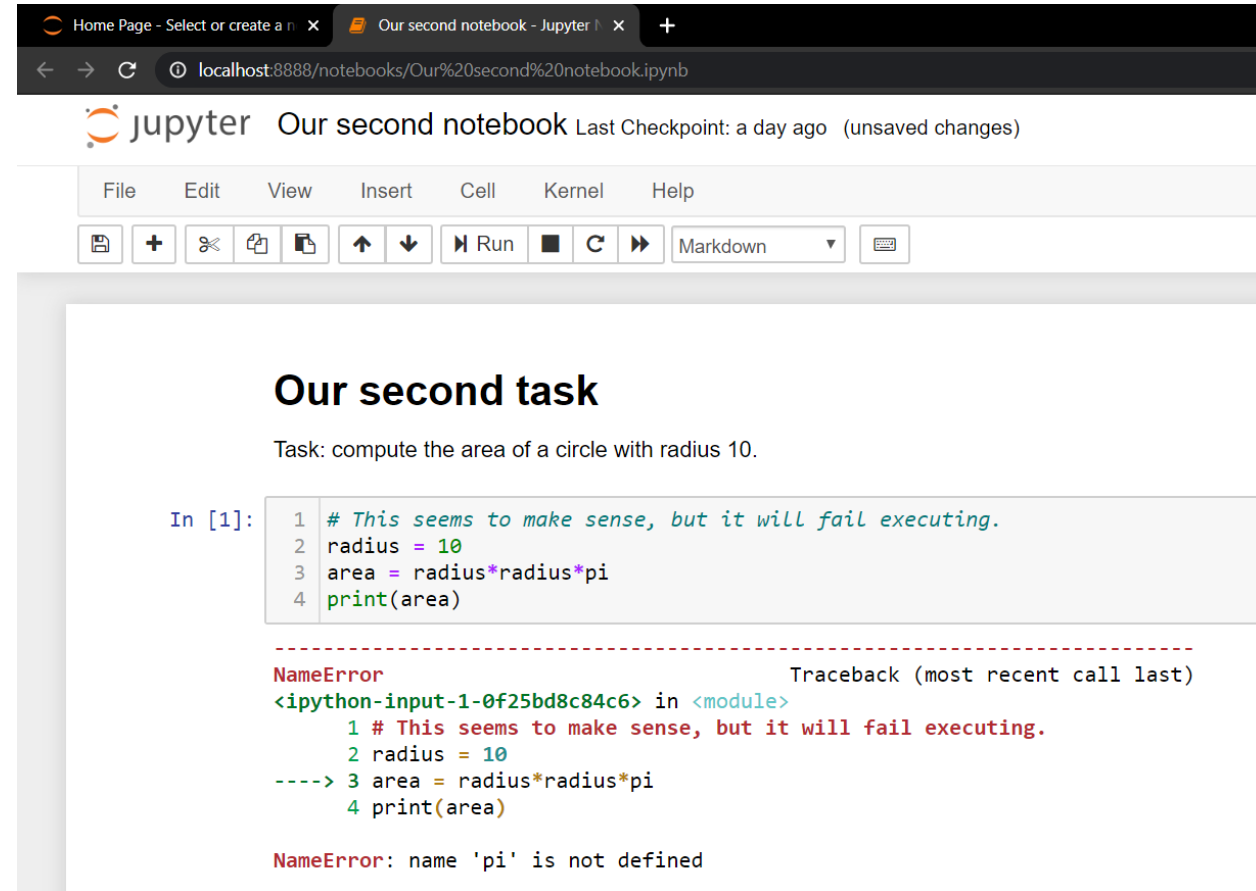
- Let us consider a **second task**: compute the area of a circle with radius 10.
- Open the second notebook.
- **Note:** in computer science, the multiplication operation is denoted `*`, not `×`.



# Our second task: problem

- While the task seems easy mathematically speaking, we have a problem...

→ **We need the value of pi.**



The screenshot shows a Jupyter Notebook browser interface. The browser tabs include 'Home Page - Select or create a n...' and 'Our second notebook - Jupyter'. The address bar shows 'localhost:8888/notebooks/Our%20second%20notebook.ipynb'. The notebook title is 'Our second notebook' with a subtitle 'Last Checkpoint: a day ago (unsaved changes)'. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. Below the menu is a toolbar with icons for saving, adding cells, and running. The main content area has a heading 'Our second task' followed by the text 'Task: compute the area of a circle with radius 10.' Below this is a code cell labeled 'In [1]:' containing the following Python code:

```
1 # This seems to make sense, but it will fail executing.
2 radius = 10
3 area = radius*radius*pi
4 print(area)
```

Below the code cell is a red dashed line and a traceback message:

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-1-0f25bd8c84c6> in <module>
      1 # This seems to make sense, but it will fail executing.
      2 radius = 10
----> 3 area = radius*radius*pi
      4 print(area)

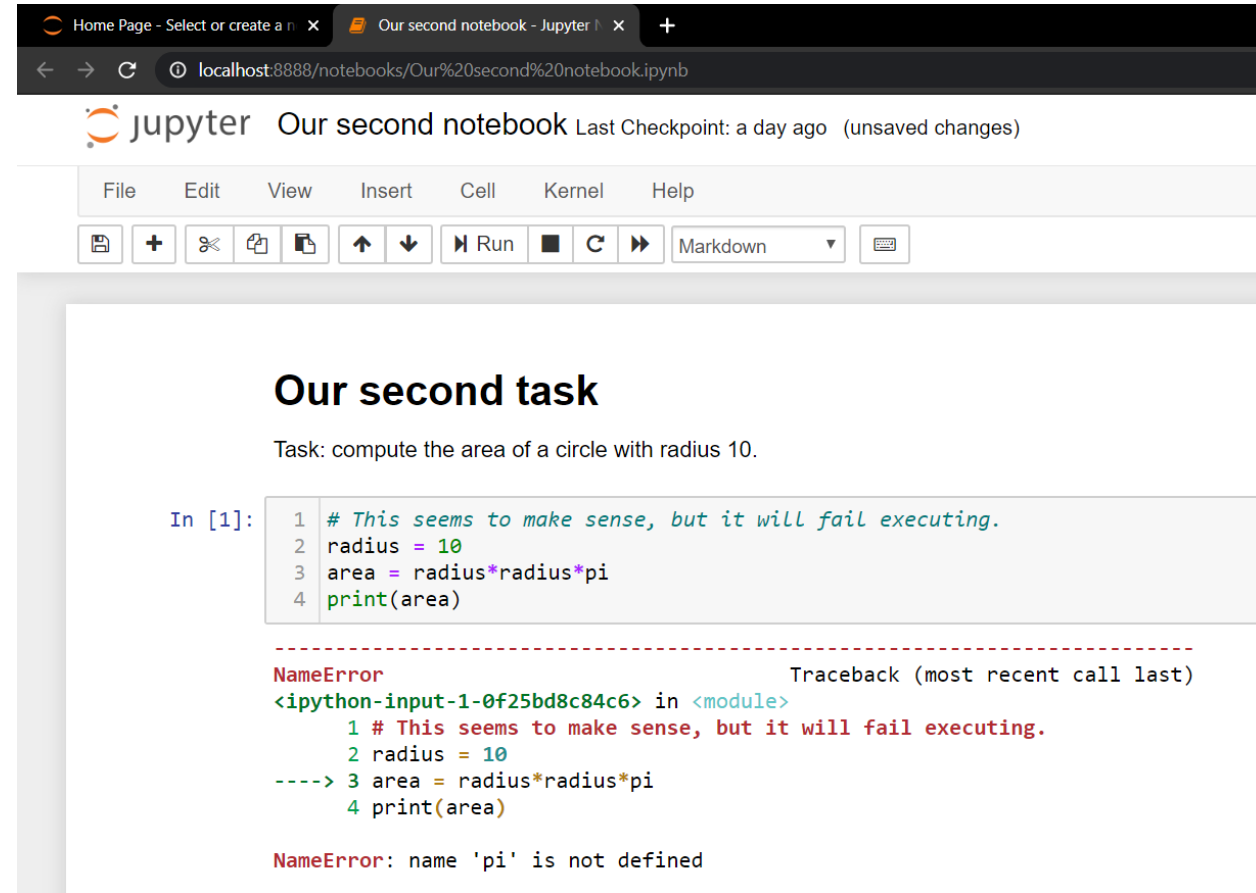
NameError: name 'pi' is not defined
```

# Our second task: problem

- While the task seems easy mathematically speaking, we have a problem...

→ **We need the value of pi.**

- 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**.



The screenshot shows a Jupyter Notebook browser interface. The browser tabs include 'Home Page - Select or create a n...' and 'Our second notebook - Jupyter'. The address bar shows 'localhost:8888/notebooks/Our%20second%20notebook.ipynb'. The notebook title is 'Our second notebook' with a subtitle 'Last Checkpoint: a day ago (unsaved changes)'. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. Below the menu is a toolbar with icons for saving, adding cells, and running. The main content area has a heading 'Our second task' followed by the text 'Task: compute the area of a circle with radius 10.' Below this is a code cell with the following Python code:

```
In [1]: 1 # This seems to make sense, but it will fail executing.
        2 radius = 10
        3 area = radius*radius*pi
        4 print(area)
```

The code cell has a red border and a red 'NameError' message at the bottom: 'NameError: name 'pi' is not defined'. Above the error message is a 'Traceback (most recent call last)' section showing the same code lines as the cell.

# Importing from a Python package

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



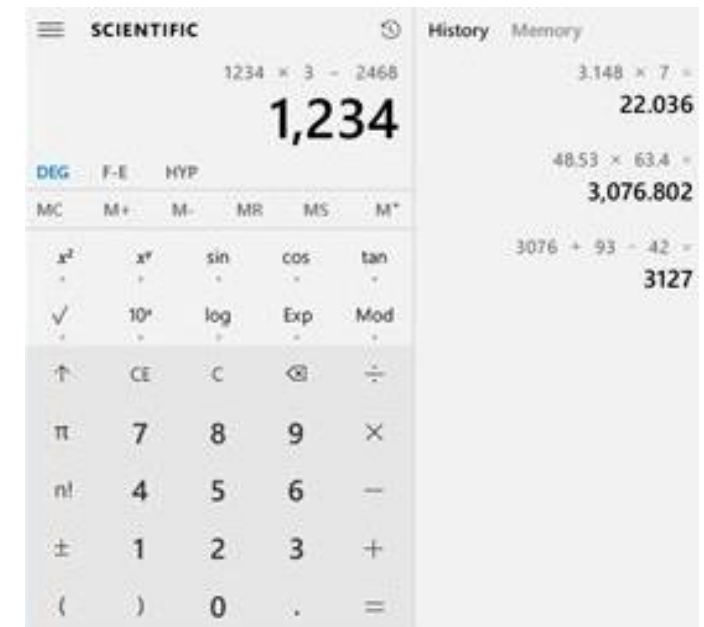
# Importing from a Python package

- 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 from a Python package

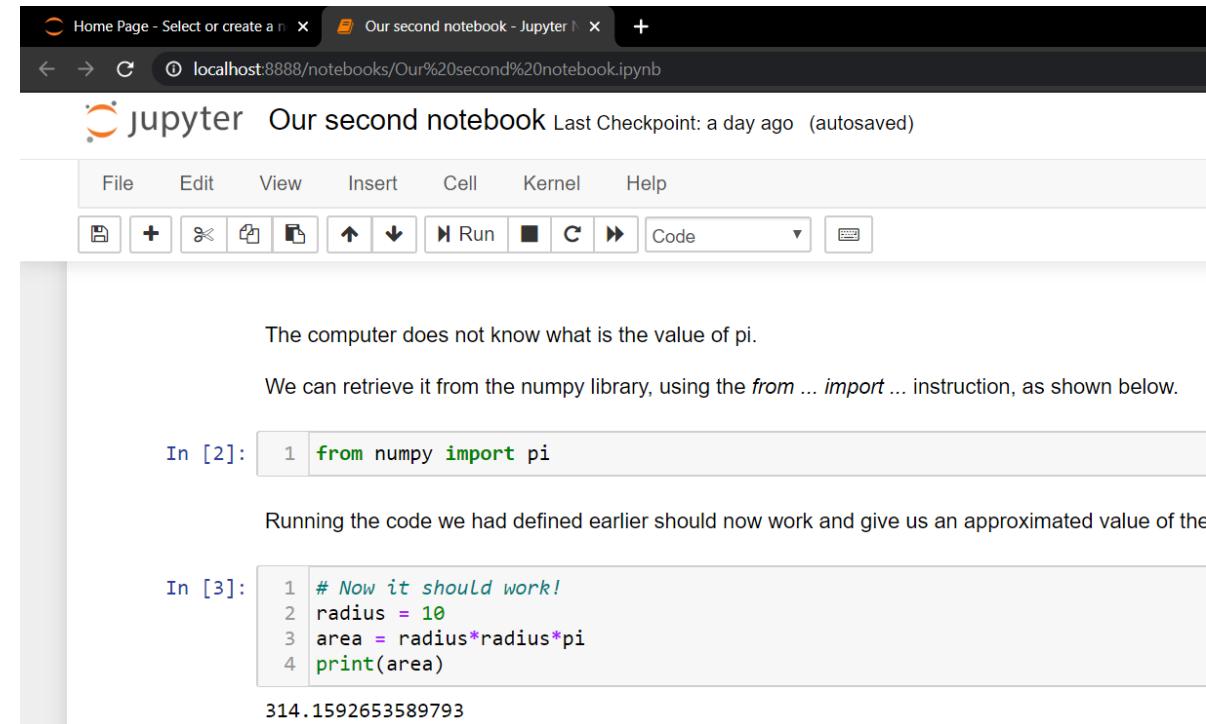
- By default, Python is a simple yet powerful calculator, which can only perform basic calculations (additions, multiplications, etc.)
- If we need more advanced concepts, we need to **import** them from a **package**.
- **Import**  $\approx$  adding a specific button to your calculator.





# 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**.
- **Import**  $\approx$  adding a set of buttons/functions to your calculator.



The screenshot shows a Jupyter Notebook browser interface. The browser tabs include 'Home Page - Select or create a n...' and 'Our second notebook - Jupyter'. The address bar shows 'localhost:8888/notebooks/Our%20second%20notebook.ipynb'. The notebook title is 'Our second notebook' with a note 'Last Checkpoint: a day ago (autosaved)'. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. The toolbar contains icons for file operations, running, and code execution. The notebook content shows two code cells. The first cell, labeled 'In [2]:', contains the code 'from numpy import pi' and is followed by text explaining that the computer does not know the value of pi and that it can be retrieved from the numpy library. The second cell, labeled 'In [3]:', contains the code to calculate the area of a circle with radius 10 using the imported pi constant. The output of the second cell is '314.1592653589793'.

```
In [2]: 1 from numpy import pi
```

The computer does not know what is the value of pi.  
We can retrieve it from the numpy library, using the *from ... import ...* instruction, as shown below.

```
In [3]: 1 # Now it should work!  
2 radius = 10  
3 area = radius*radius*pi  
4 print(area)
```

314.1592653589793

By the way, this means that you do not need your calculator anymore, welcome to the real world.



scientific calculator

All of Singapore



Sell

Home > Search results for "scientific calculator"

333 search results for 'scientific calculator' in Singapore

Save this search ☐

Category ▾

Sort: Best Match ▾

Condition ▾

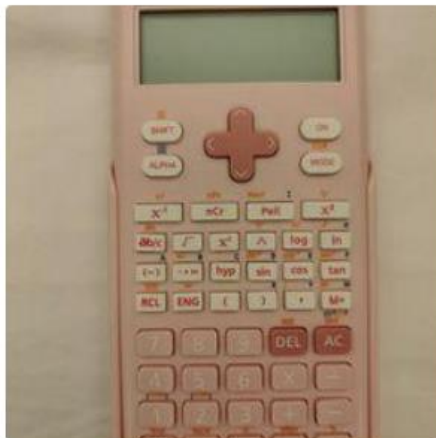
Price ▾

Deal Option ▾

More filters



nie\_hhhh  
16 hours ago



scientific calculator

**S\$9**

Like new



tanakof  
2 days ago



Casio scientific calculator

**S\$5**

Well used



smookid  
11 hours ago



Casio Scientific Calculator fx-95MS

**S\$15**

Well used



wss  
1 day ago



Casio fx-991EX Classwiz Scientific Calculator

**S\$31**

Like new



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

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



# Conclusion

## What we have seen

- What is programming?
  - Programming Languages and why we will use Python.
  - Installing Python, extra packages and IDEs.
  - Test run to confirm everything works.
- If you were able to execute the two codes (in console and Jupyter Notebooks): you are officially done for today.
- Let me know in Zoom chat if you have encountered technical issues and we will try to fix them together.**