ILP 2021 – W1S1
Getting started

Matthieu DE MARI – 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
- Office @ SUTD: 1.702.27

Outline (Week1, Session1 – W1S1)

- About this course: syllabus, objectives, Zoom, eDimension, practice, grading, 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 and mini-project #1
- W5S2: Everything about strings
- W5S3: Dictionaries and object-oriented thinking
- W6S1: Object-oriented programming (part 1)
- W6S2: Object-oriented programming (part 2) and mini-project #2
- W6S3: Final exam? (to be confirmed)

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

Practice?

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

Need some extra practice?

- Homeworks: nope.
- 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've played with are things you remember.”

- Jarrett Walker, international consultant in public transit network design.
(He also has a fantastic blog about using planning games, like MiniMetro and SimCity to explain and understand notions of transportation network design)
http://humantransit.org/2014/12/learning-how-transit-works-from-mini-metro.html

Plot twist: GAMIFICATION

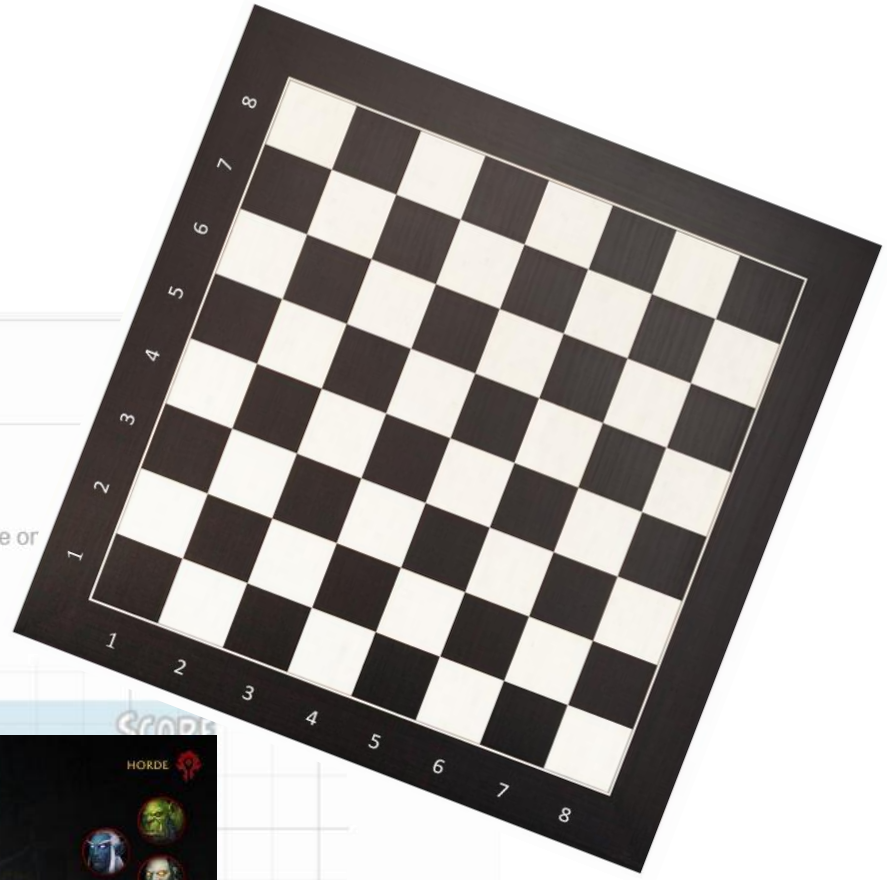
Activity 2 - Ballistics of an angry bird

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

oblem statement

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

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

12

Exams

Final (100%) – On W6S3 (to be confirmed).

- Covers notions in W1-W6.
- Theory MCQ.
- Maybe, a few practice questions.

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://docs.google.com/forms/d/e/1FAIpQLSejSKnRZYR-2kN1MlcEV0_5KNh0pu6V6JOaQsqS9Vo5iHEF1A/viewform?usp=pp_url

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

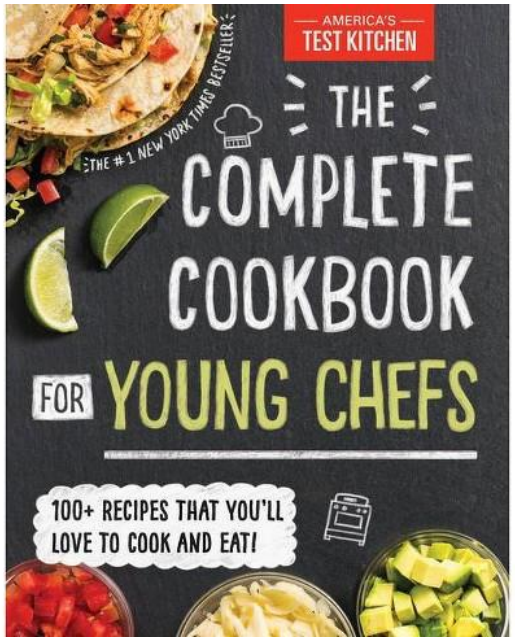
- **Definition (Programming):** **Programming** refers to the process of designing and building an executable computer program, to accomplish a specific computational task.

  It involves tasks such as
  - analysis,
  - designing **algorithms**,
  - and implementing said **algorithms** in a chosen programming language (a.k.a. **coding**).

- **Layman definition (Programming):** **Programming** consists of defining a sequence of instructions that the computer must follow to accomplish a task.

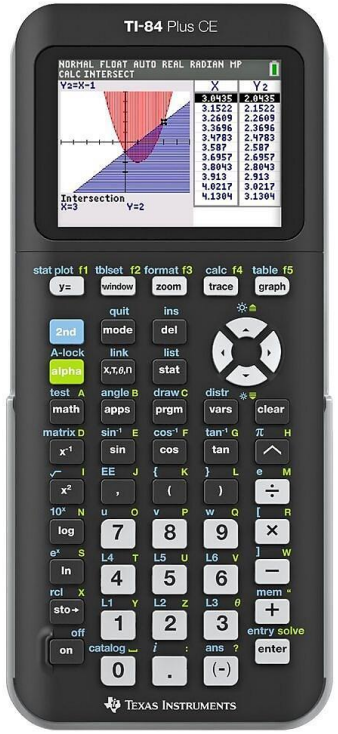
Programming: some analogies

- 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!



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 π,
  - Press equal key again,
  - You have reached your result.

Algorithm: definition

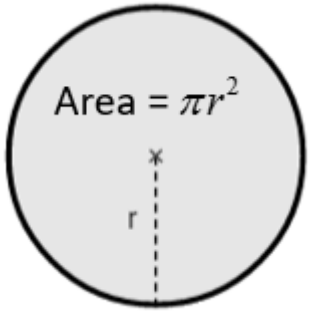
- **<u>Definition (Algorithm):</u>** An **algorithm** is a finite sequence of instructions, typically used to solve a class of problems or perform a computation task.

  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 $\mathbf{10}$,
  - Type multiply key,
  - Type $\mathbf{10}$,
  - Press equal/enter key,
  - Type multiply key,
  - Type π,
  - 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 π,
- Press equal key again,
- You have reached your result.

ALGORITHM

python™

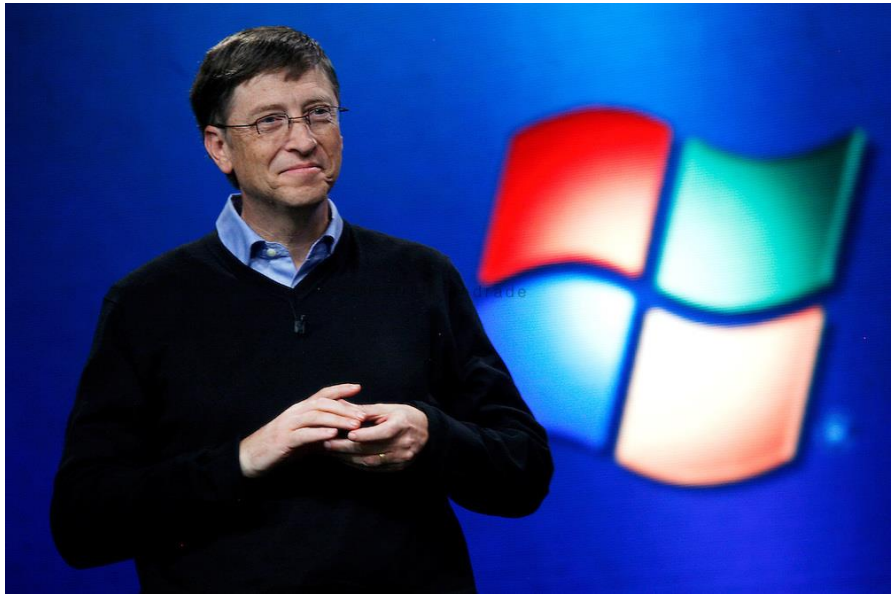
Quick question

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

Quick question

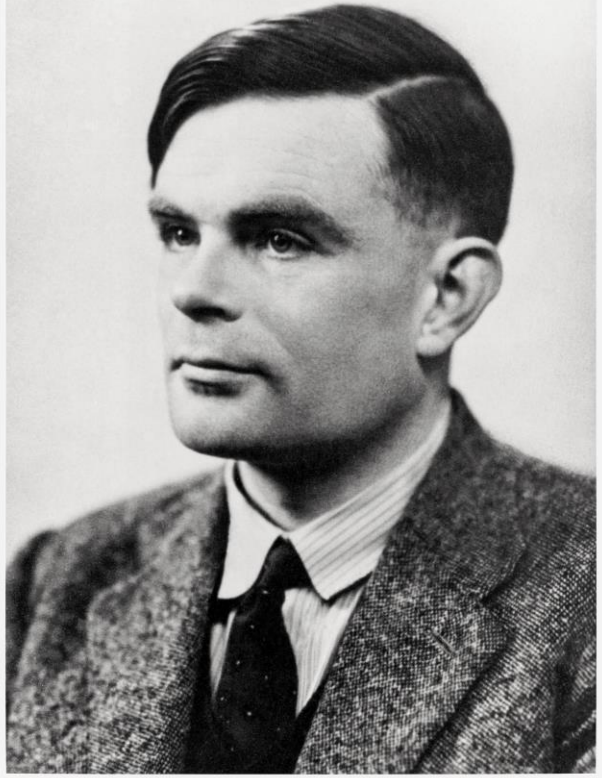
→ Who is considered the inventor of computers and “programming”?

Hint: it is NOT one of the persons below.



The “father” of computer science

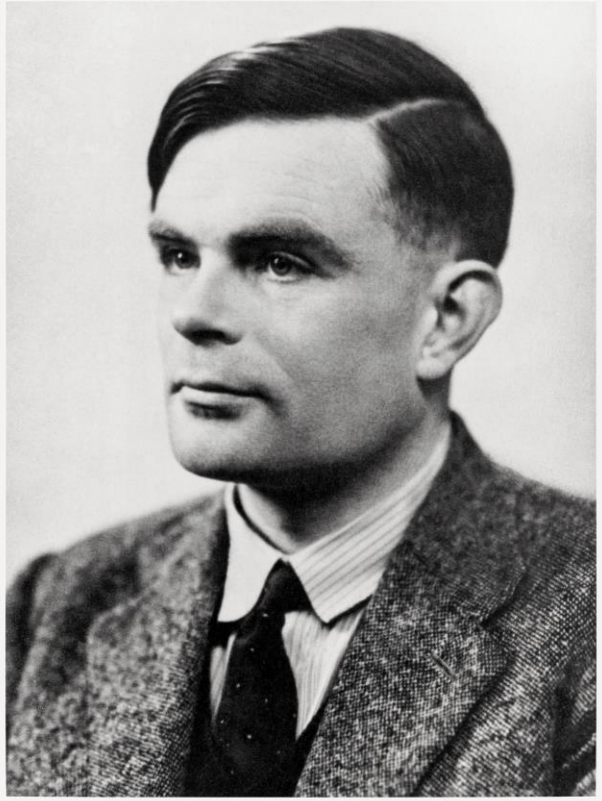
Alan Mathison Turing (1912–1954) was an English mathematician, computer scientist, logician, cryptanalyst, philosopher, and theoretical biologist.



The “father” of computer science

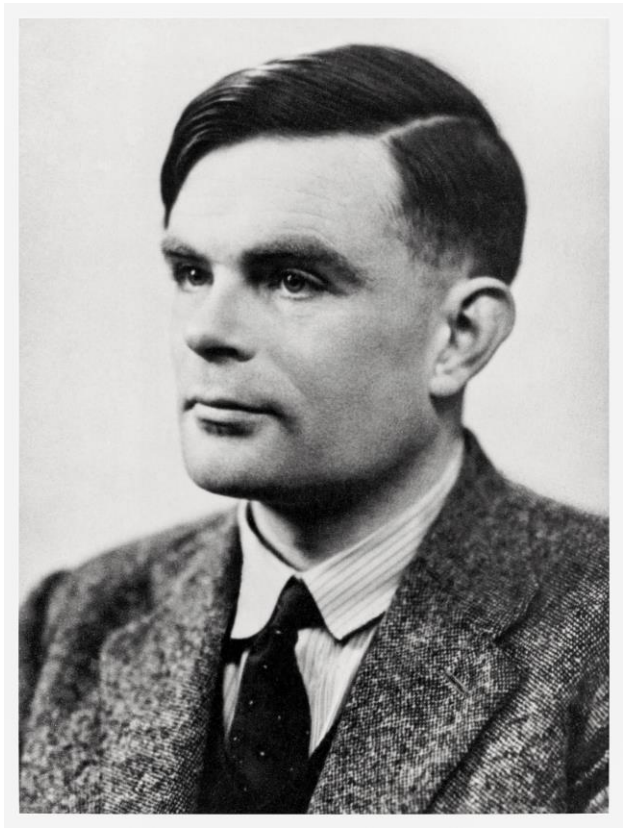
Alan Mathison Turing (1912–1954) was an English mathematician, computer scientist, logician, cryptanalyst, philosopher, and theoretical biologist.

- 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.



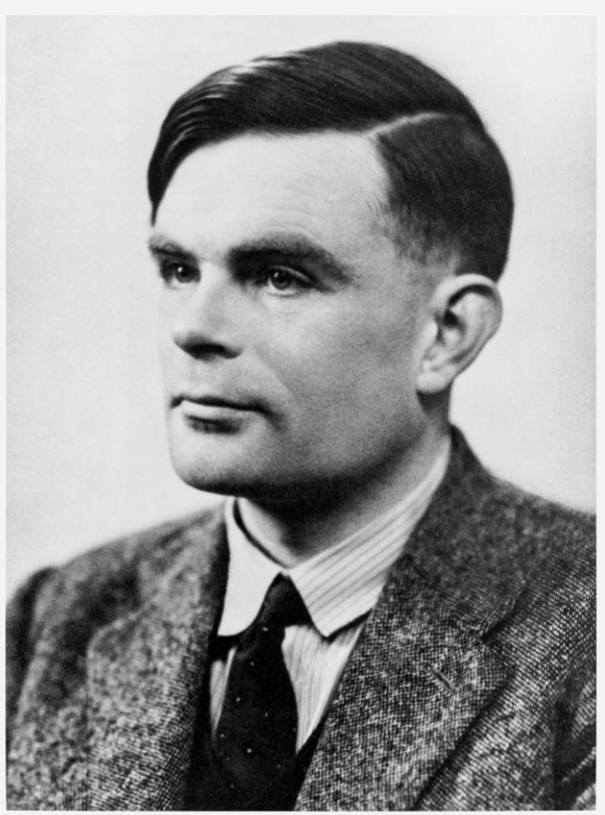
The “father” of computer science

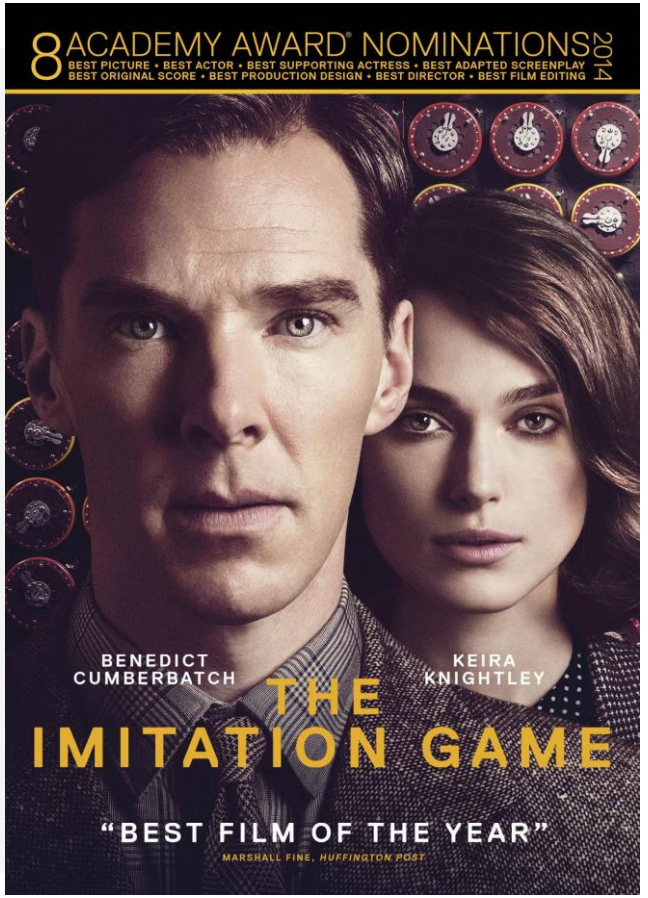
- During World War 2, he came up with an idea of a “computer-like” machine, to decode encrypted transmissions from the German army.
- Thanks to his decryption device, it has been estimated the war in Europe was shortened by more than two years and saved over 14 million lives.



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)

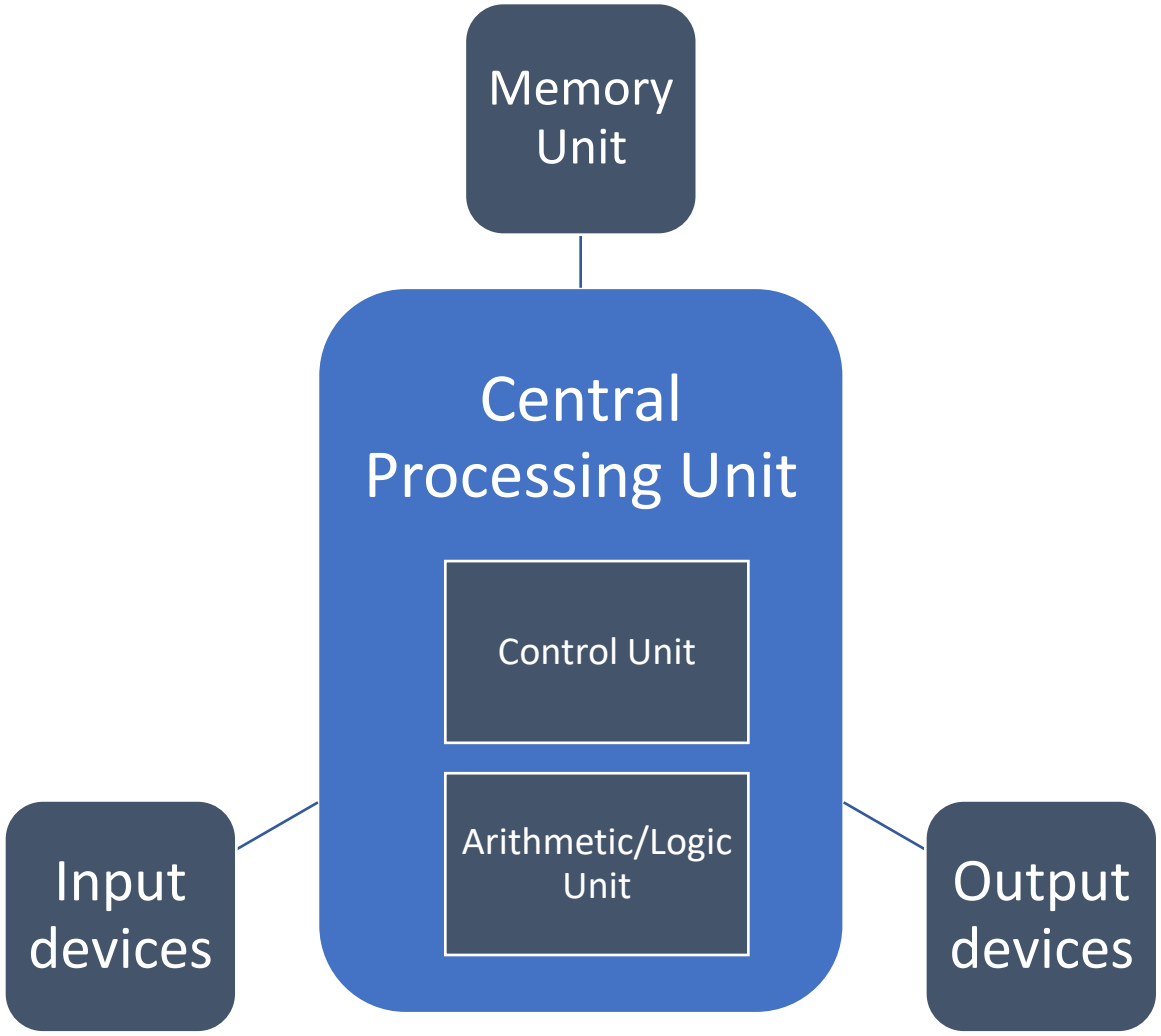
And that is it.

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

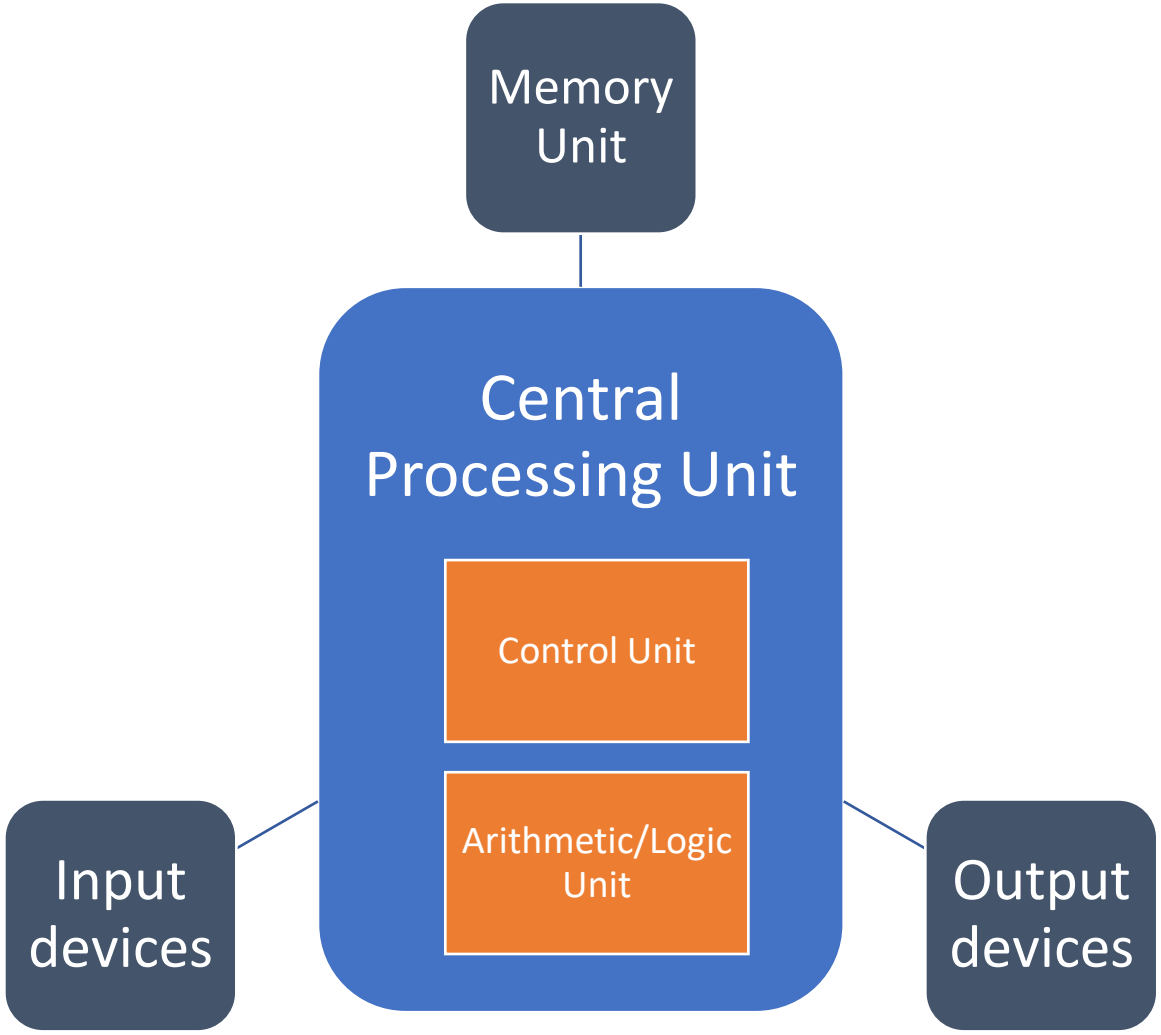
- **Definition (Von Neumann architecture):** The Von Neumann architecture describes one of the first architectures for a computer.



The Von Neumann architecture

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

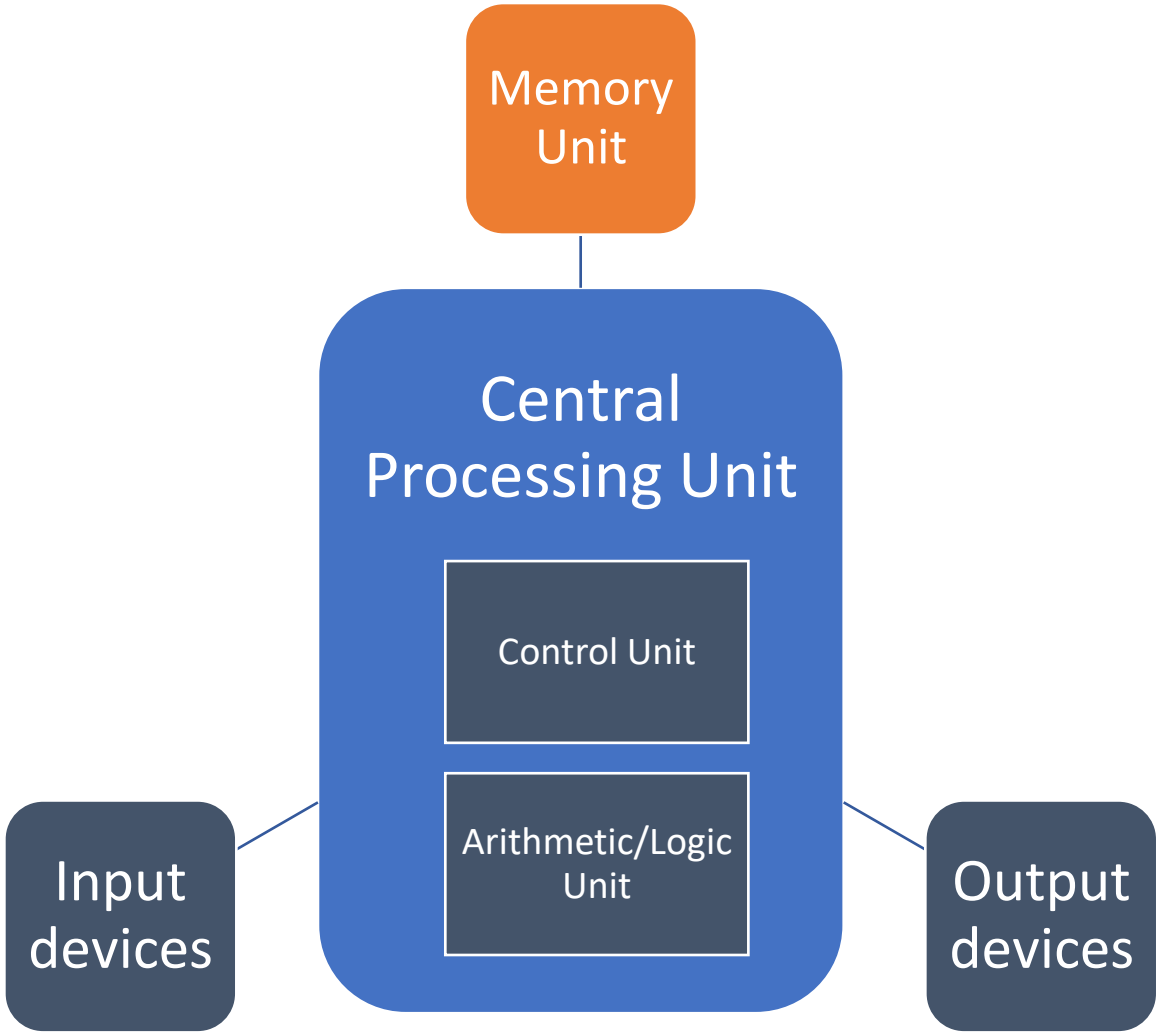
- An **Arithmetic/Logic Unit** (in charge of dealing with instructions, typically math operations, in binary 0/1),
- And **a Control Unit** (in charge of the hardware and communication between different hardware elements)



The Von Neumann architecture

It also contains…

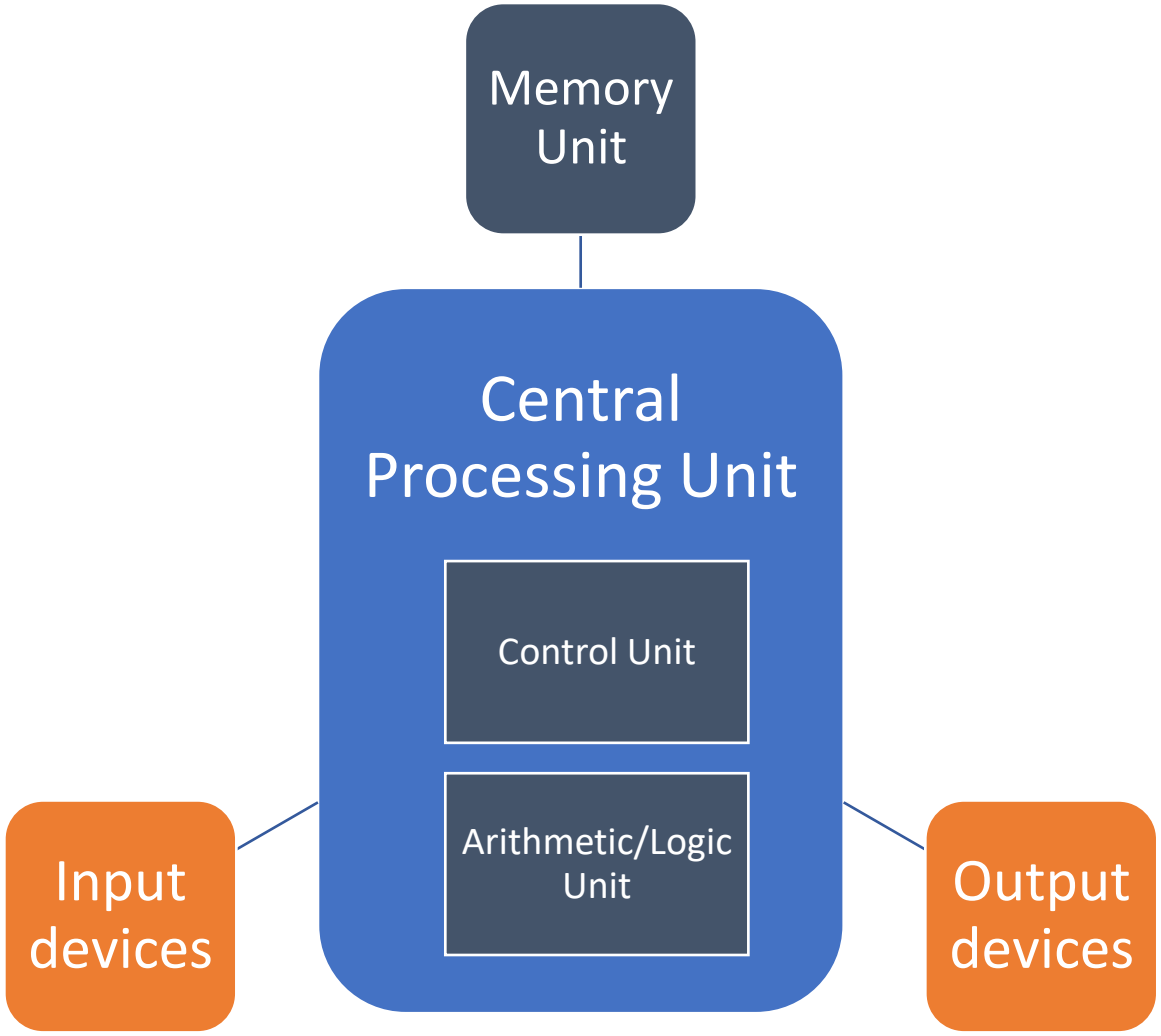
- A **Memory Unit** (for storing and retrieving results from previous computational tasks, using binary formatting 0/1),



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.).

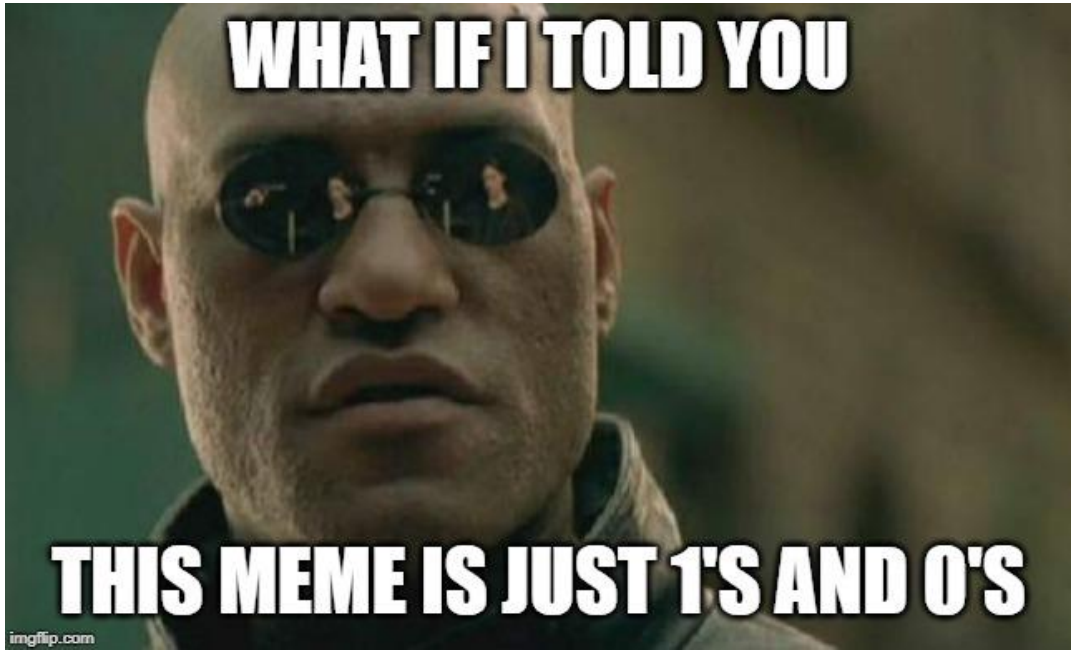


The problem of dealing with binary and the need for programming languages

- **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

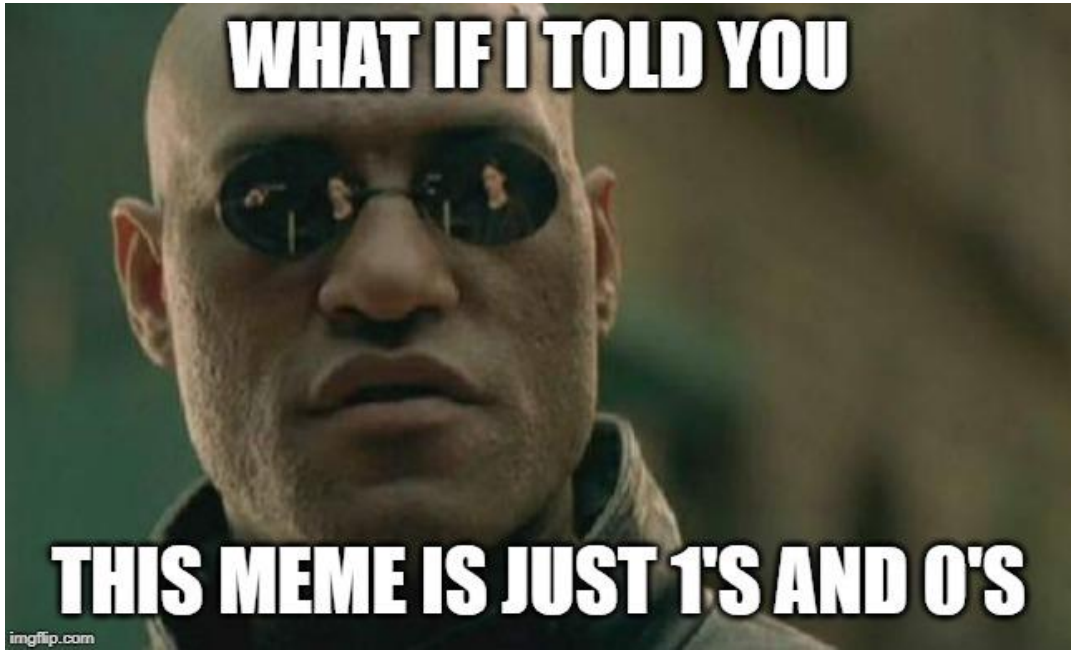
- **Observation:** in a computer, all operations (computational and memory tasks) and are performed in **binary**.



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
  - **Low-level language:** closer to binary

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:** close to human language (easy to learn)
  - **Low-level language:** close to binary (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.

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.

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.)

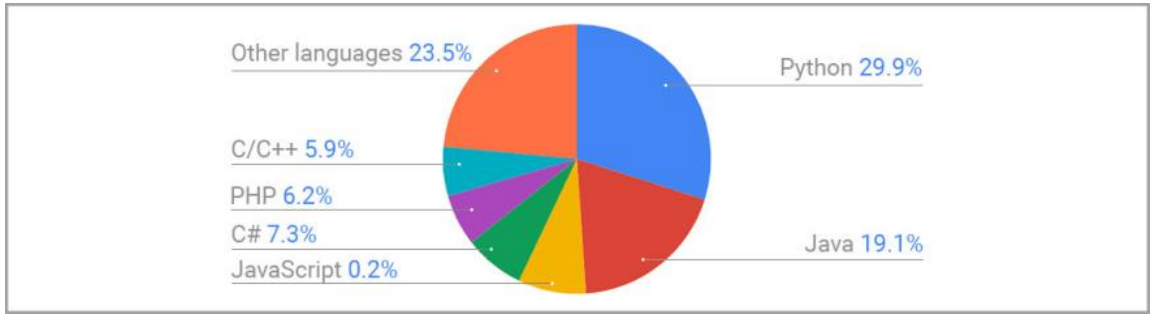
Why learn Python?

- **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.

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:** see Figure on the right.

Why learn Python?

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

Why learn Python?

- **Python is widely used in IT companies:** see Figure on the right.
- **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

- In this class, we will use **Python 3.9(.6)** (latest stable version as on the 19th of July 2021).
- 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/

Installing Python

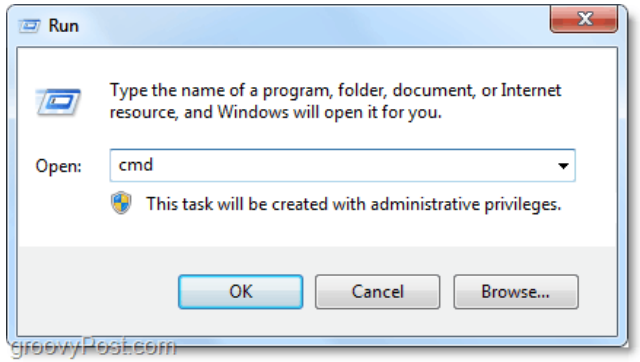
Install it now!

- In this class, we will use **Python 3.9(.6)** (latest stable version as on the 19th of July 2021).
- 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/

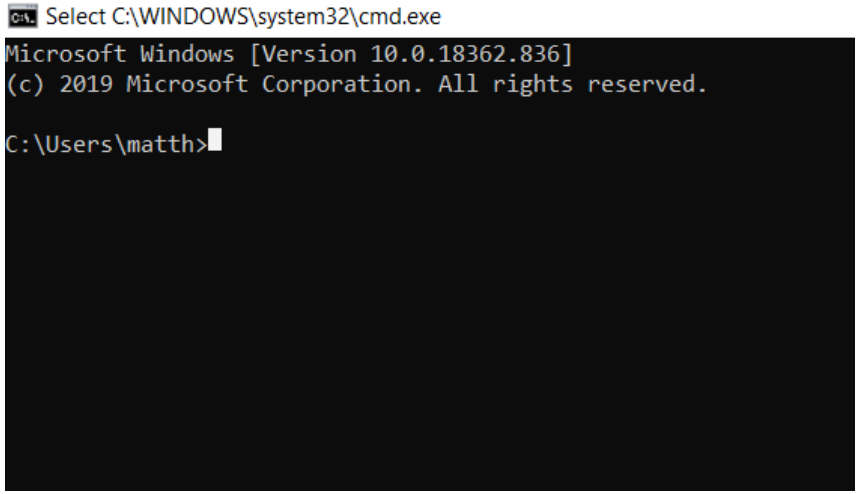
Opening a console (Windows)

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



- **Windows:** the cool way.

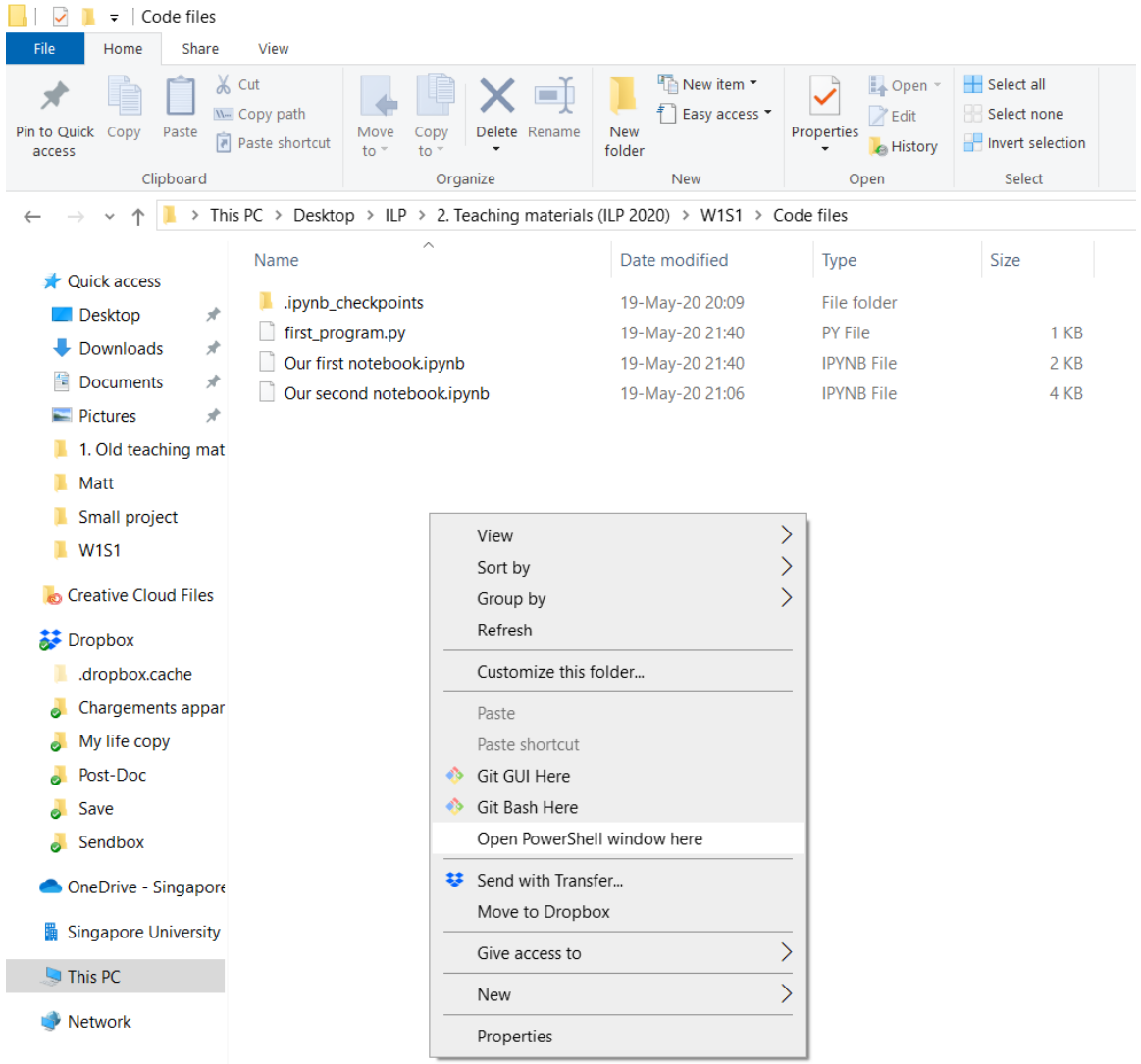
1. Press simultaneously **Windows key + R**,
2. then type **cmd**, and press **Enter.**



Opening a console (Windows – option 2)

- **Windows option 2:** faster and more convenient way, in my opinion

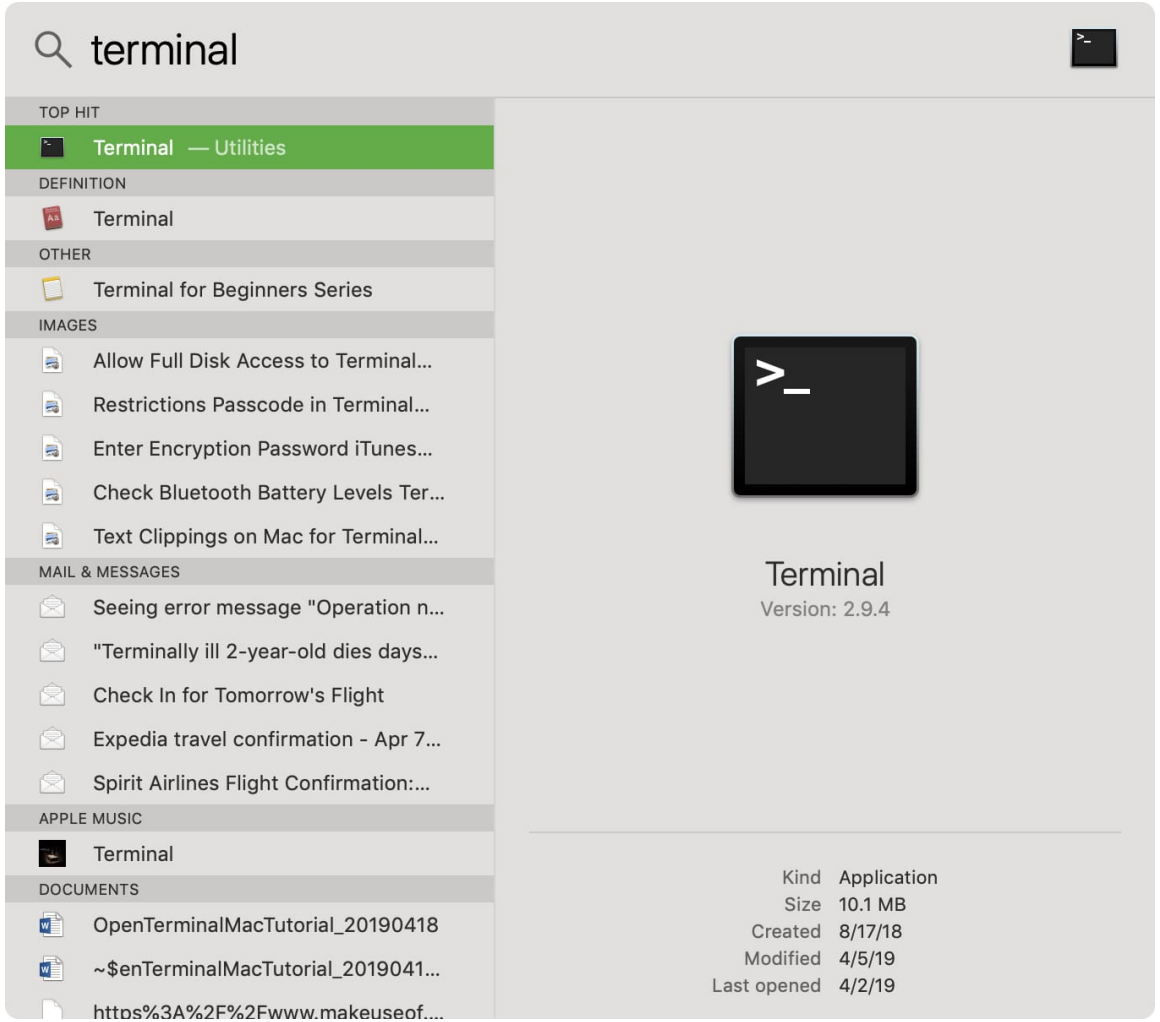
1. **Open an explorer** in the folder you attempt to work in.
2. **Hold shift** and **right-click** in an empty space of the explorer window.
3. **Choose "Open Powershell window here".**



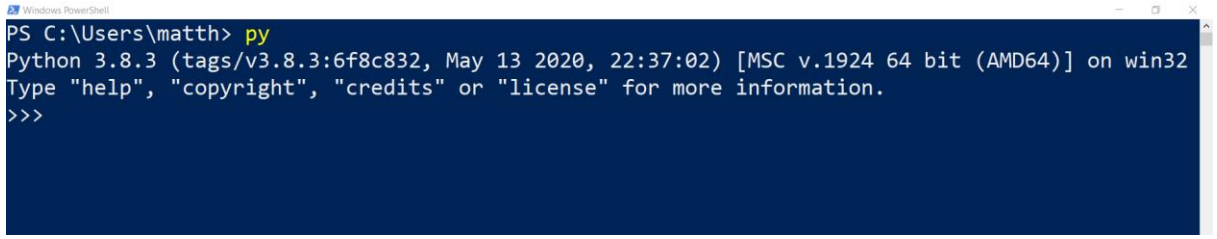
Opening a console (Mac OS)

- **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



Your first run of Python!



Start python by **typing one** of the following commands in the console and press **Enter!**

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

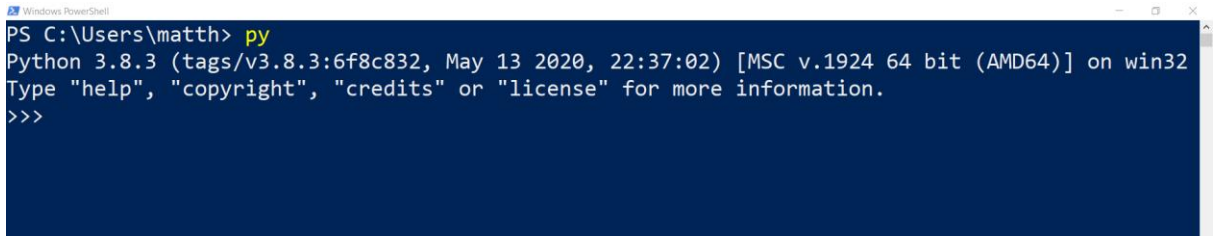
- **py (most frequent one, works in my case)**
- py3
- python
- python3

Your first run of Python!

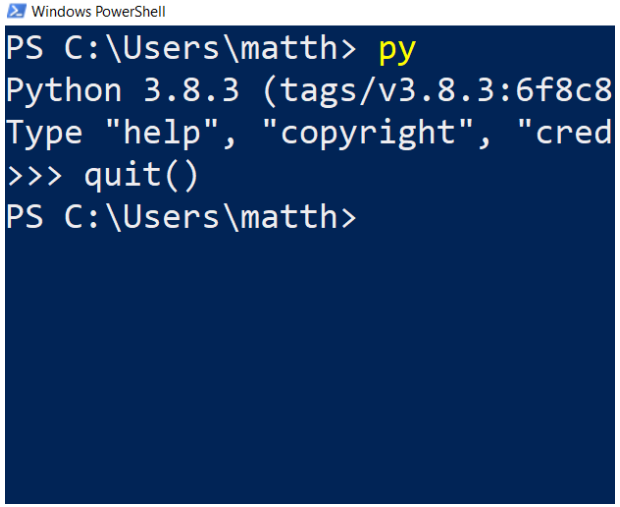
Start python by **typing one** of the following commands in the console and press **Enter!**

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

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

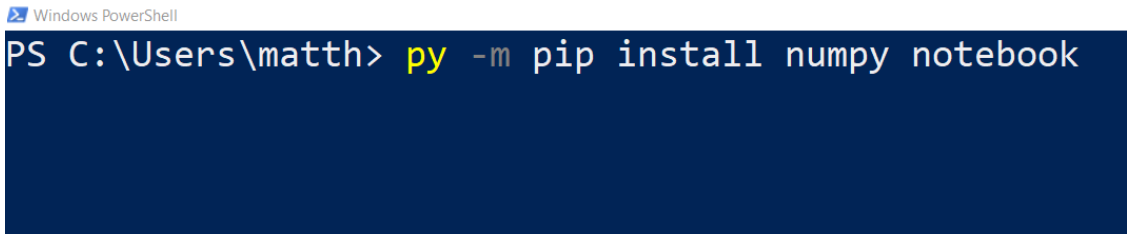


To exit Python, simply type quit() and press enter.

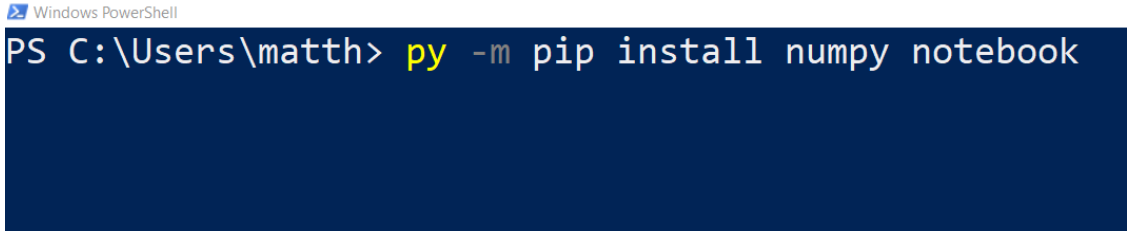


Installing & updating packages in Python

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



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.

py –m pip install numpy notebook

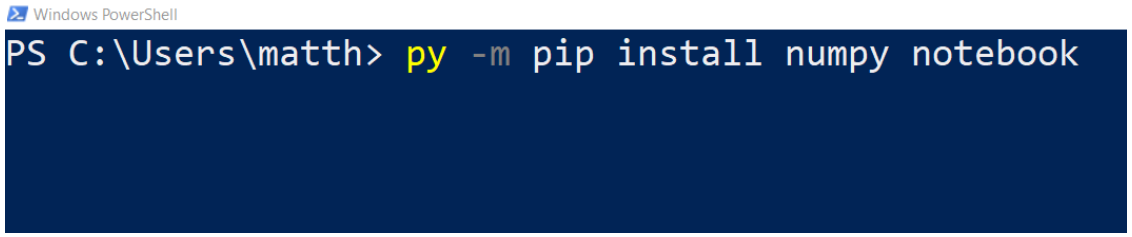
Note: if you are using **py3, python or python3** instead of **py**, adjust accordingly!

Installing & updating packages in Python

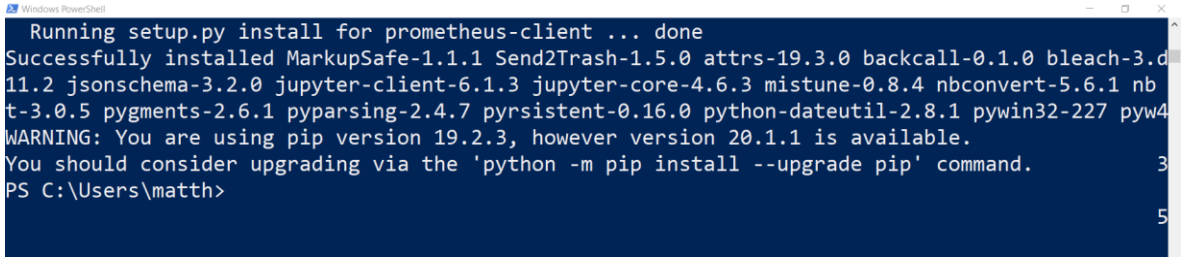
- Next, let us install **packages**. (**Packages** are extra functionalities for Python.)
- To do so, run the following command in your console.

py –m pip install numpy notebook

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



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!”.

Our first program!

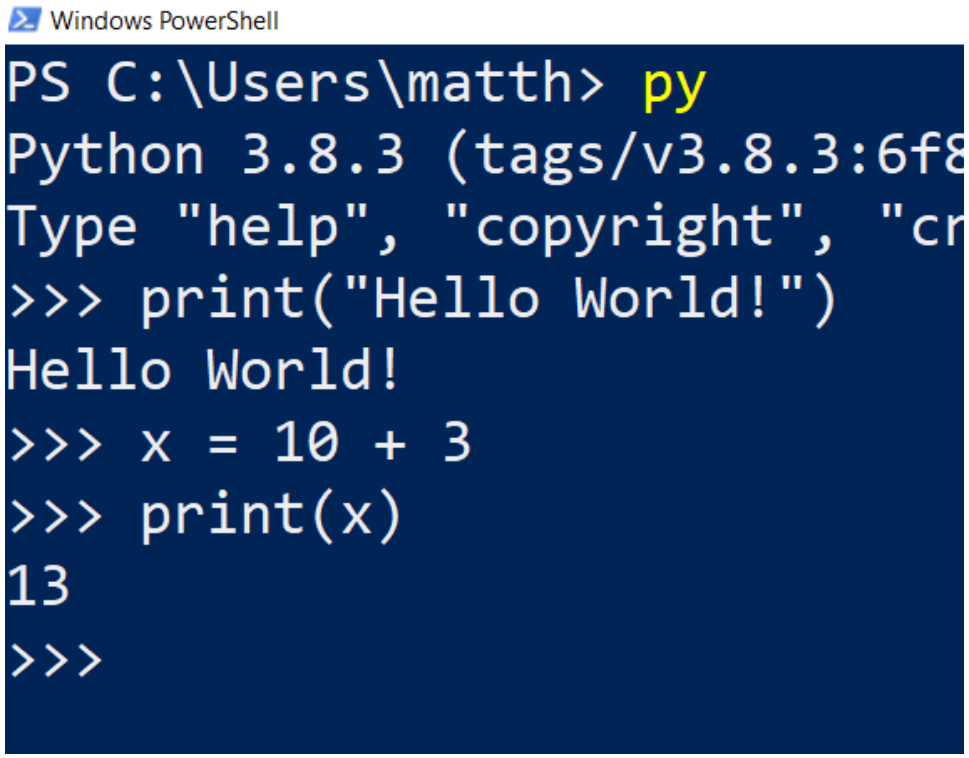
- Start python in a console, type **print(“Hello World!”)**, and submit by pressing Enter.
  It should display “Hello World!”.
- **Definition (the “Hello World” program):** The **“Hello World” program** is a computer program that outputs or displays the message "Hello World!". It is often used as a **sanity test** to make sure that a computer language is correctly installed.

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.

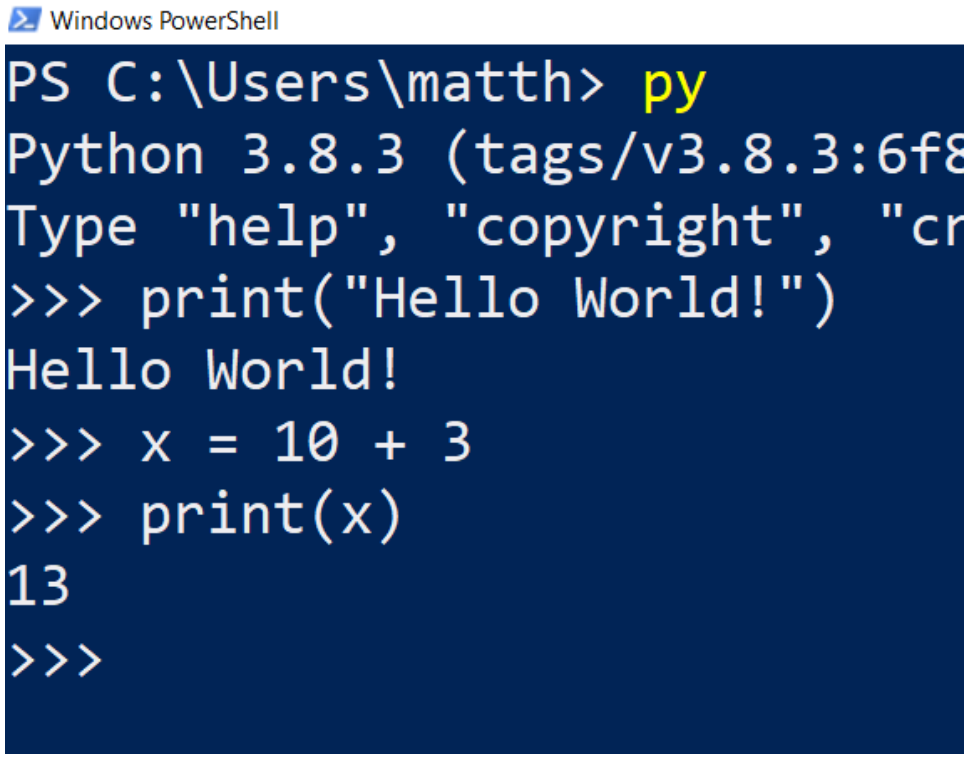
Our first program!

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



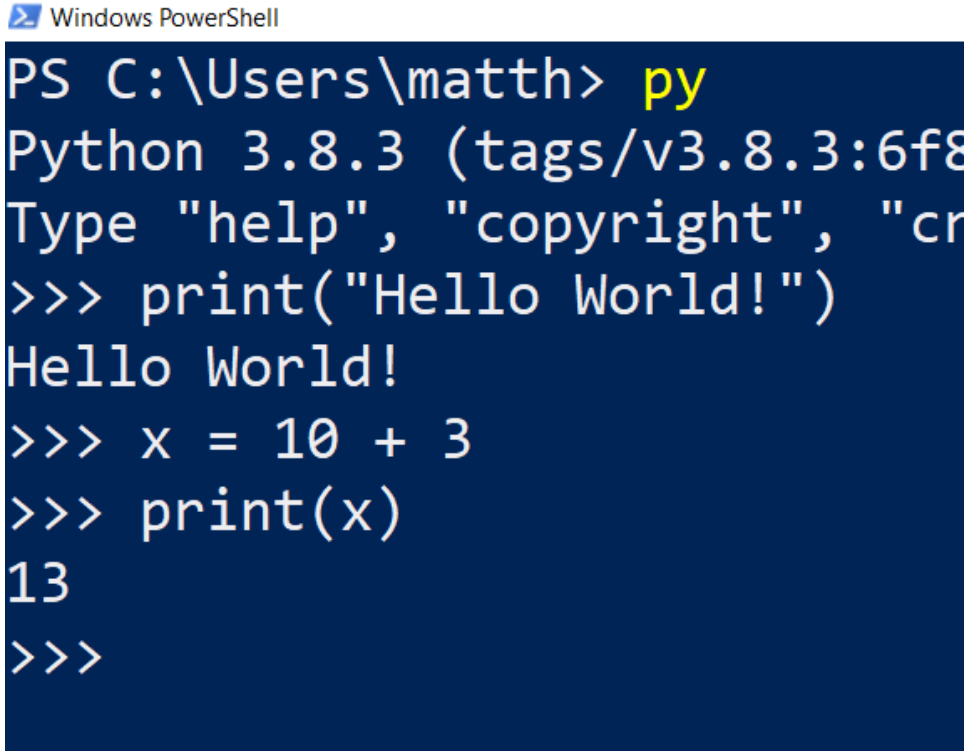
Our first program!

- Assigning something to memory is done with the = sign.
- **Note:** The = sign does not bear the same meaning as seen in mathematics.



Our first program!

- Assigning something to memory is done with the = sign.
- **Note:** The = sign does not bear the same meaning as seen 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.



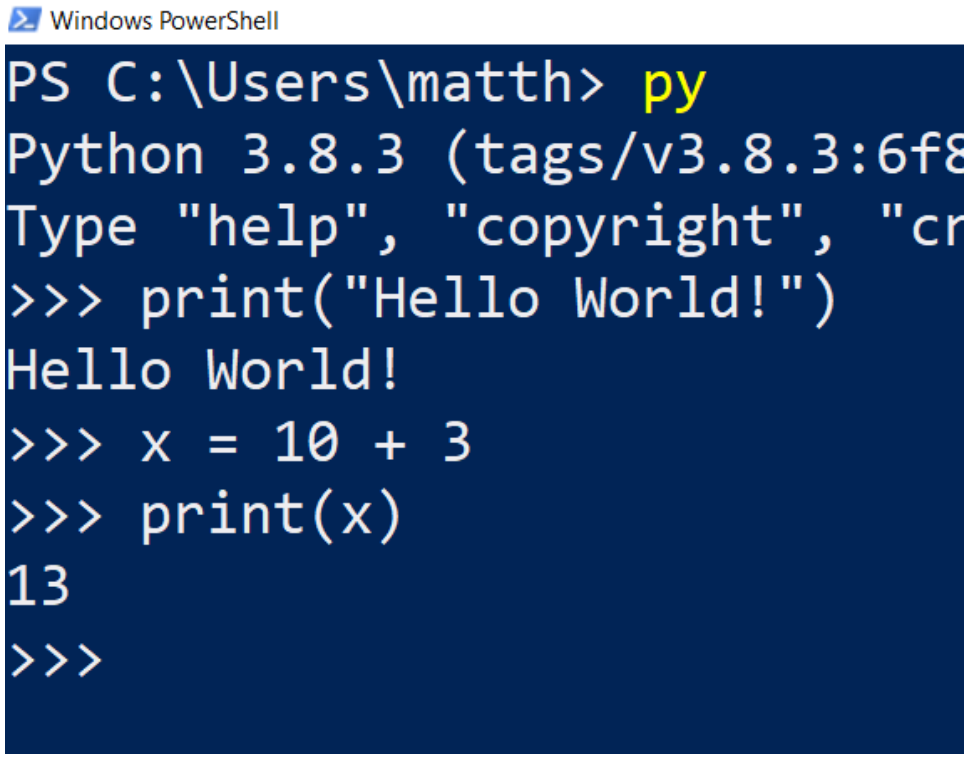
Our first program!

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

Our first program!

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

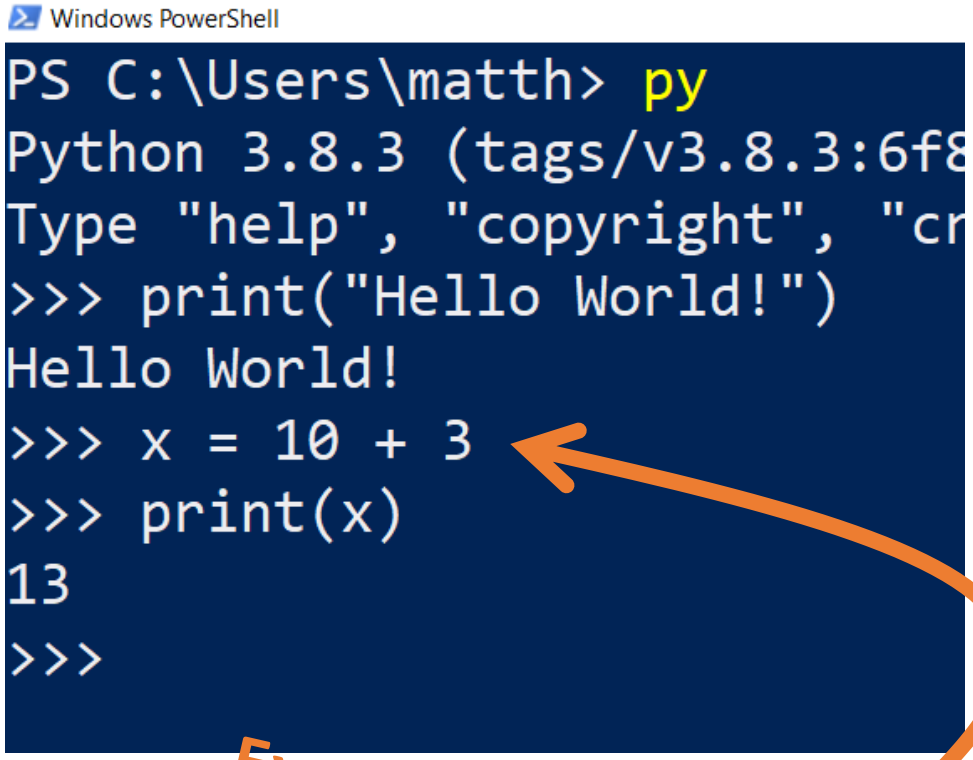
- Later on, we can retrieve the value stored in the variable, and – for instance – ask the computer to print it on screen for us, with the **print()** function.



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.



Executes in the background but shows nothing! Need to explicitly ask for a print().

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.

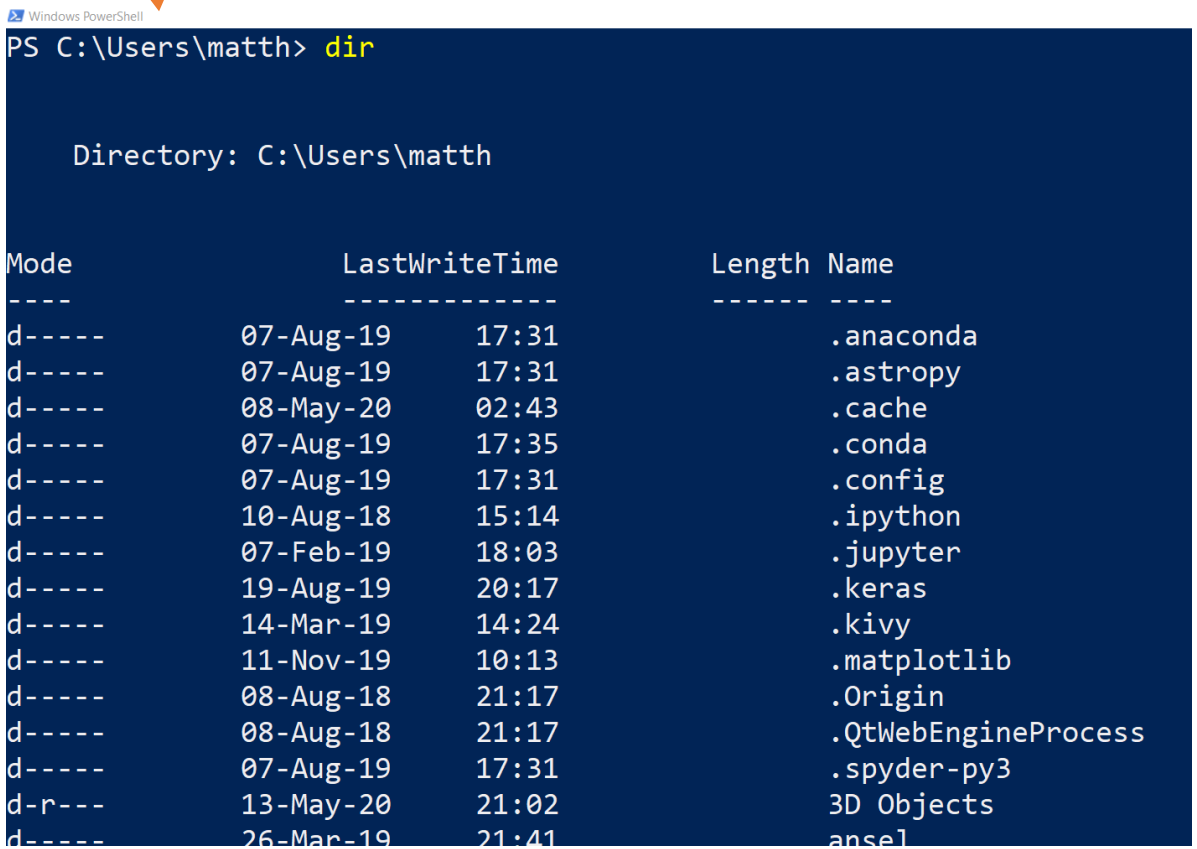
Executing a .py file in console mode

- 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.

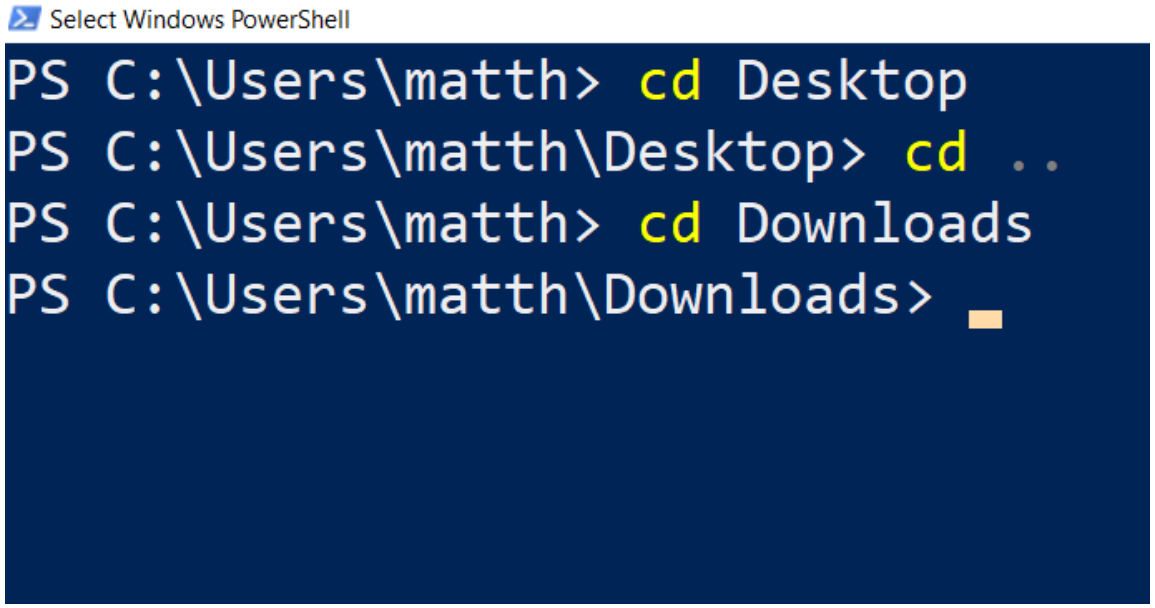
Current location is C:\Users\matth\



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**.

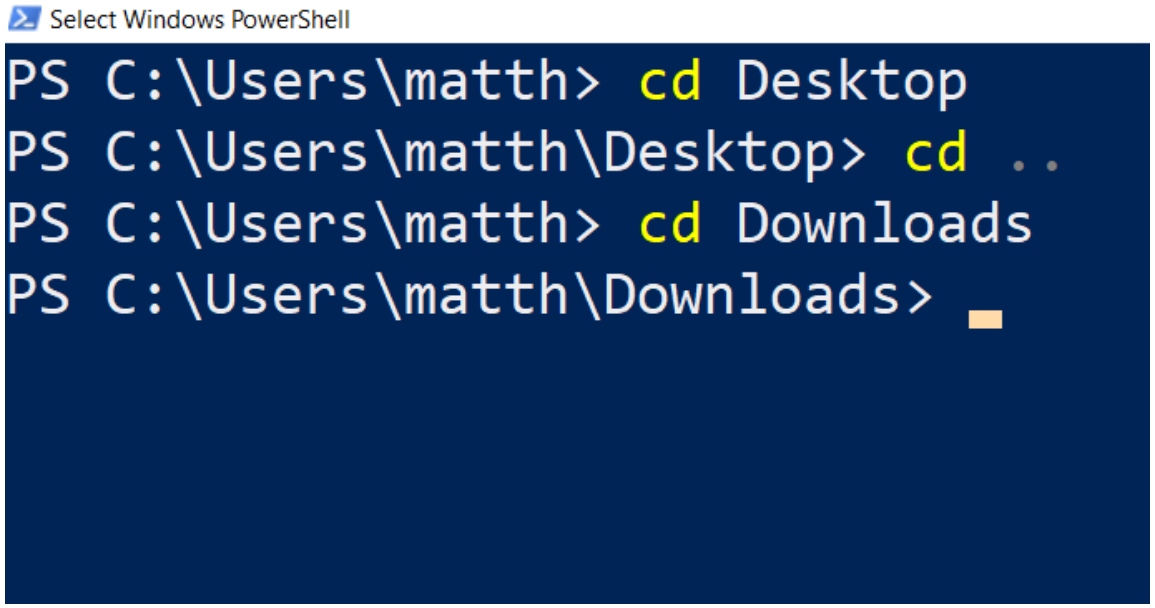
Observe how the current location is changing every time.

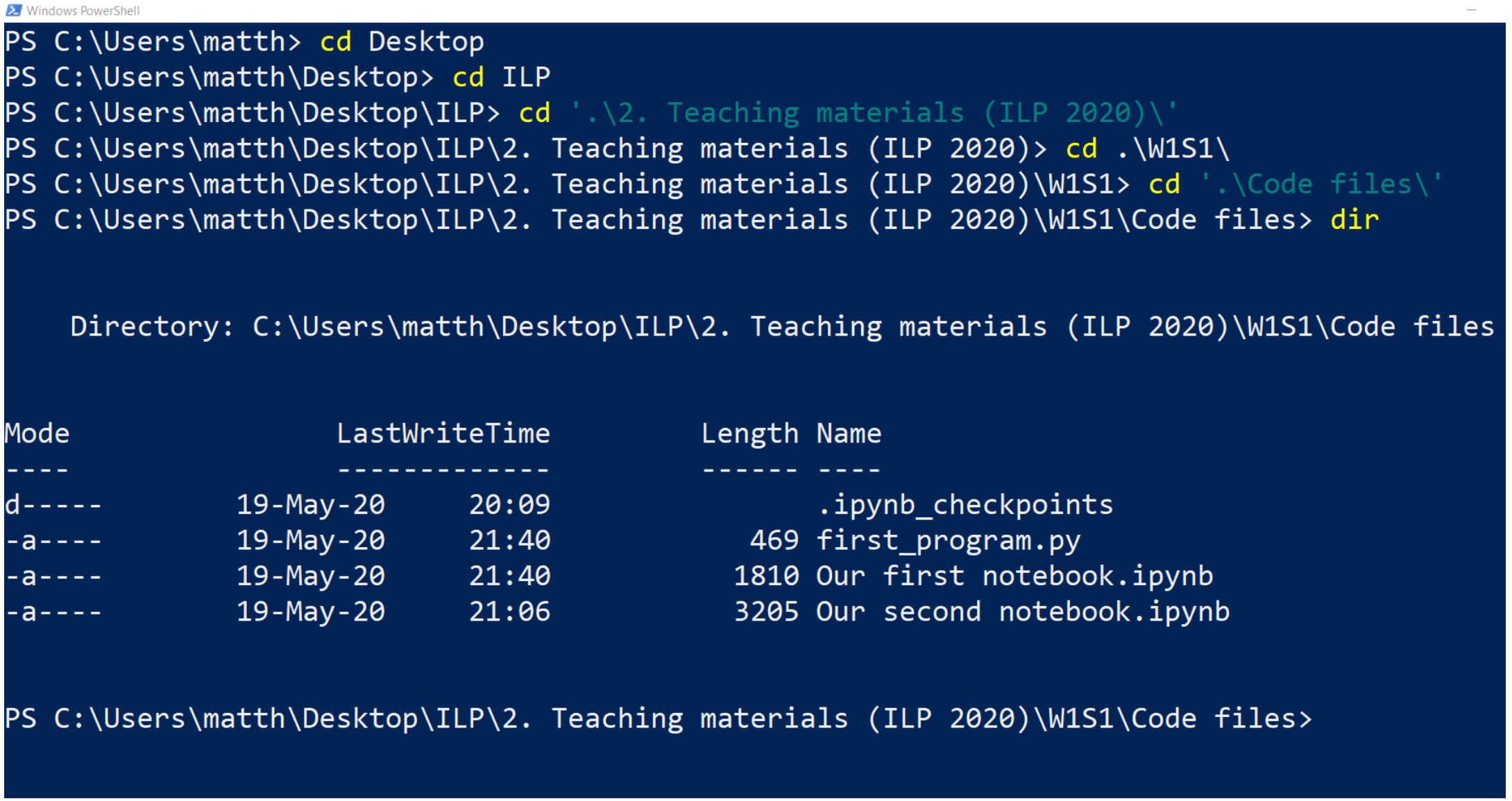


Executing a .py file in console mode

- **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.

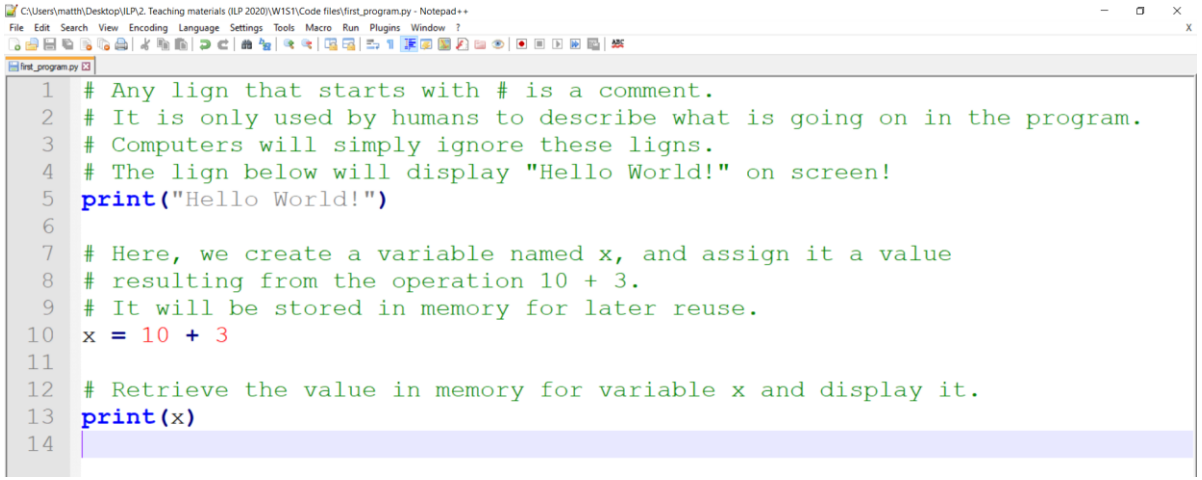




→ Now, use cd/dir/ls commands to move to the location of your **first_program.py** file !

Checking your .py file

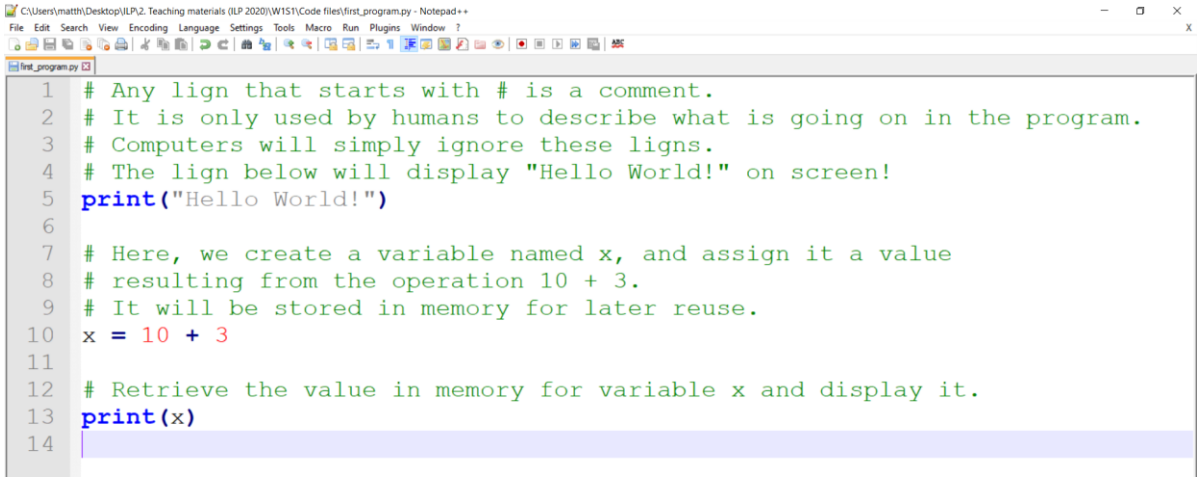
- Open your **first_program.py** file with any text editor (specifically do it by right clicking and asking to open with a text editor).
- Recognize the code we used earlier.

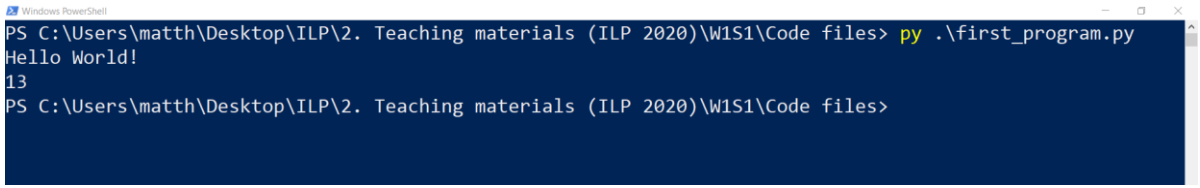


Checking your .py file

- 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





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

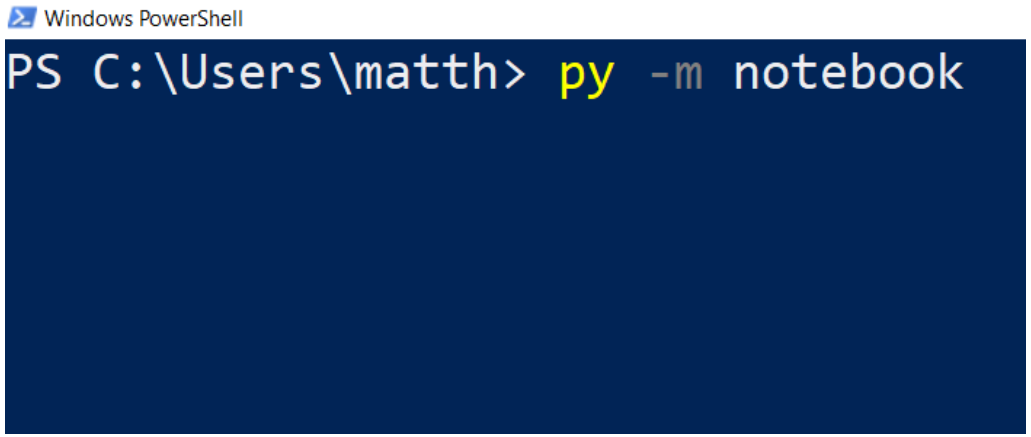
- **Problem:** Typing code in a text editor and running it from console is not exactly convenient…
- **Suggestion:** we should use an **Interactive Development Environment (IDE)**, which makes the coding easier for us.
- In this course, I suggest to use **Jupyter Notebook**, but you might look online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

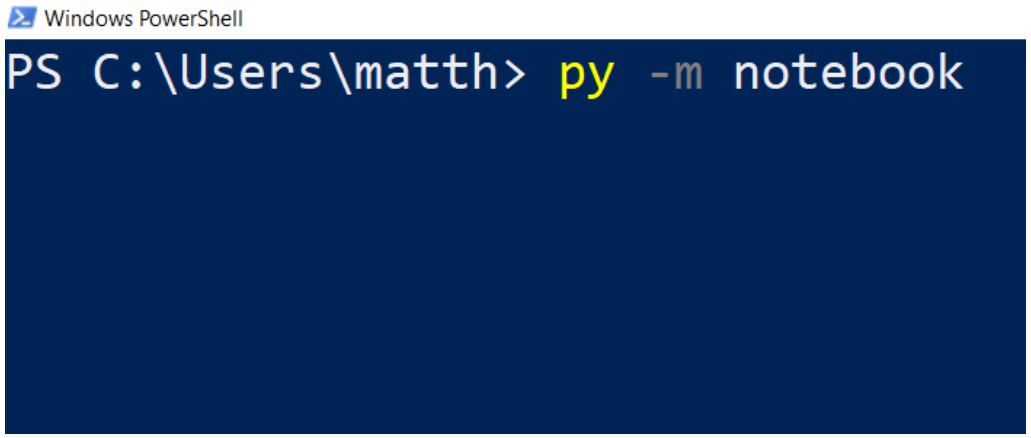
Running Python from an IDE, such as a Jupyter Notebook

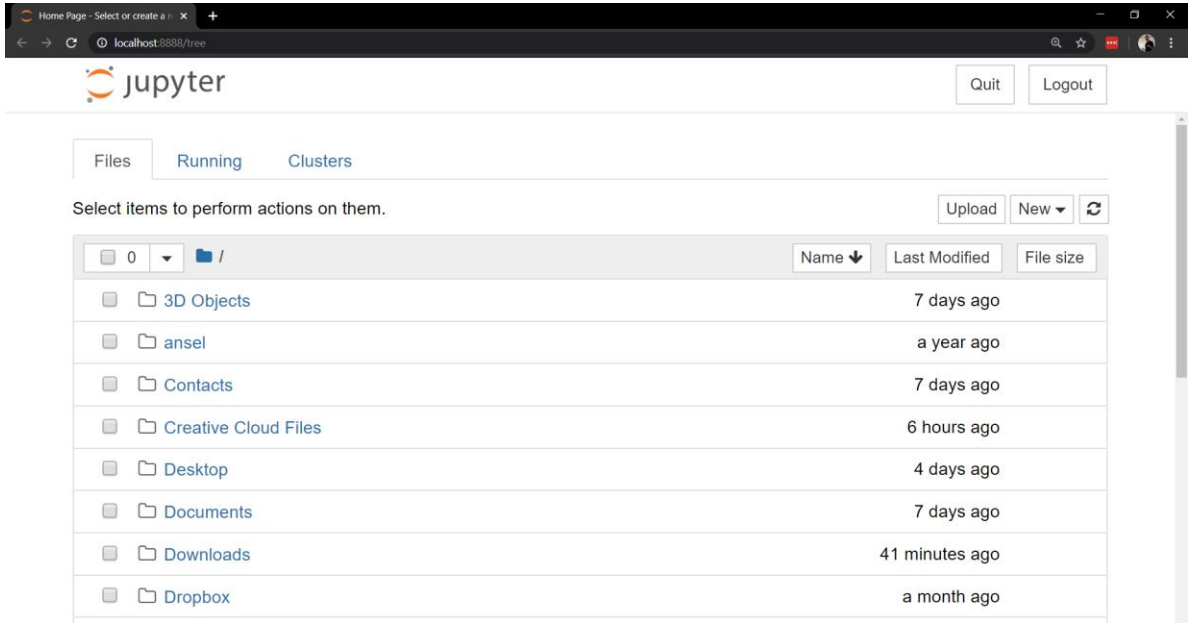
- Return to your console, **outside** of the Python environment (use **quit()** if needed).
- Type **py –m notebook**, and press enter to submit and call the **Python notebook module** (-m notebook)



Running Python from an IDE, such as a Jupyter Notebook

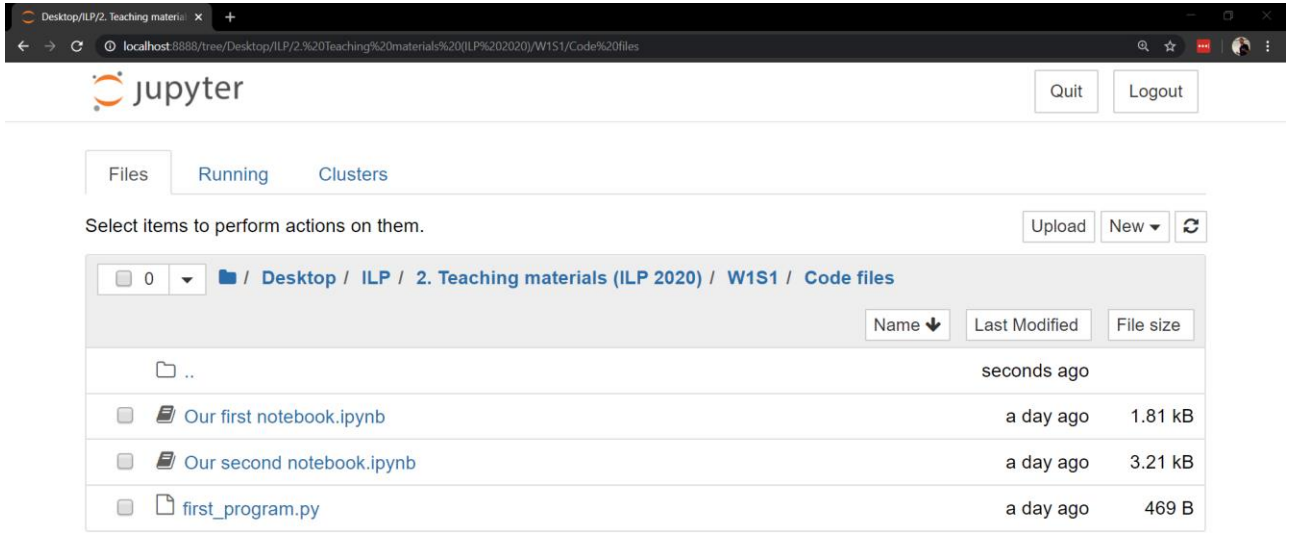
- Return to your console, **outside** of the Python environment (use **quit()** if needed).
- Type **py –m notebook**, and press enter to submit and call the **Python notebook module** (-m notebook)
- **It should open a notebook window/tab in your web browser.**





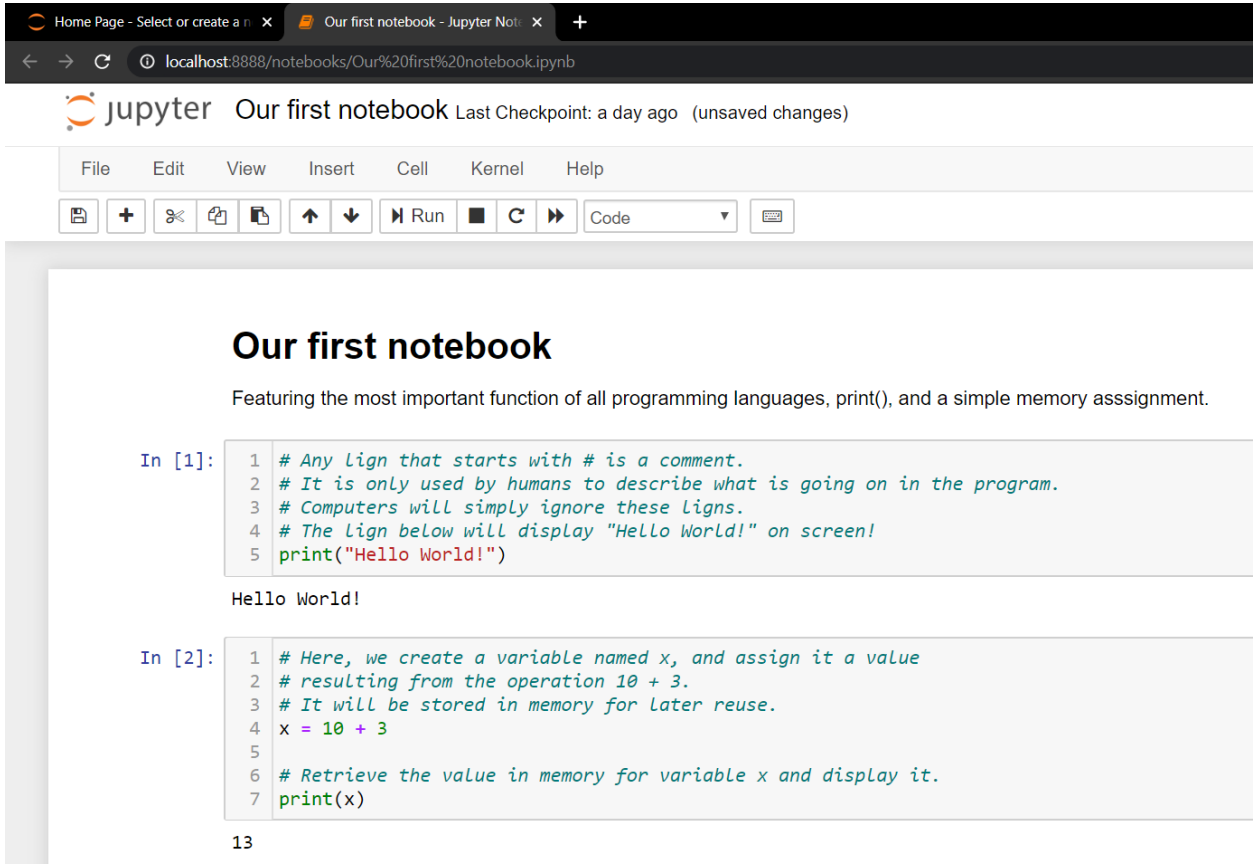
Running Python from an IDE, such as a Jupyter Notebook

- 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.



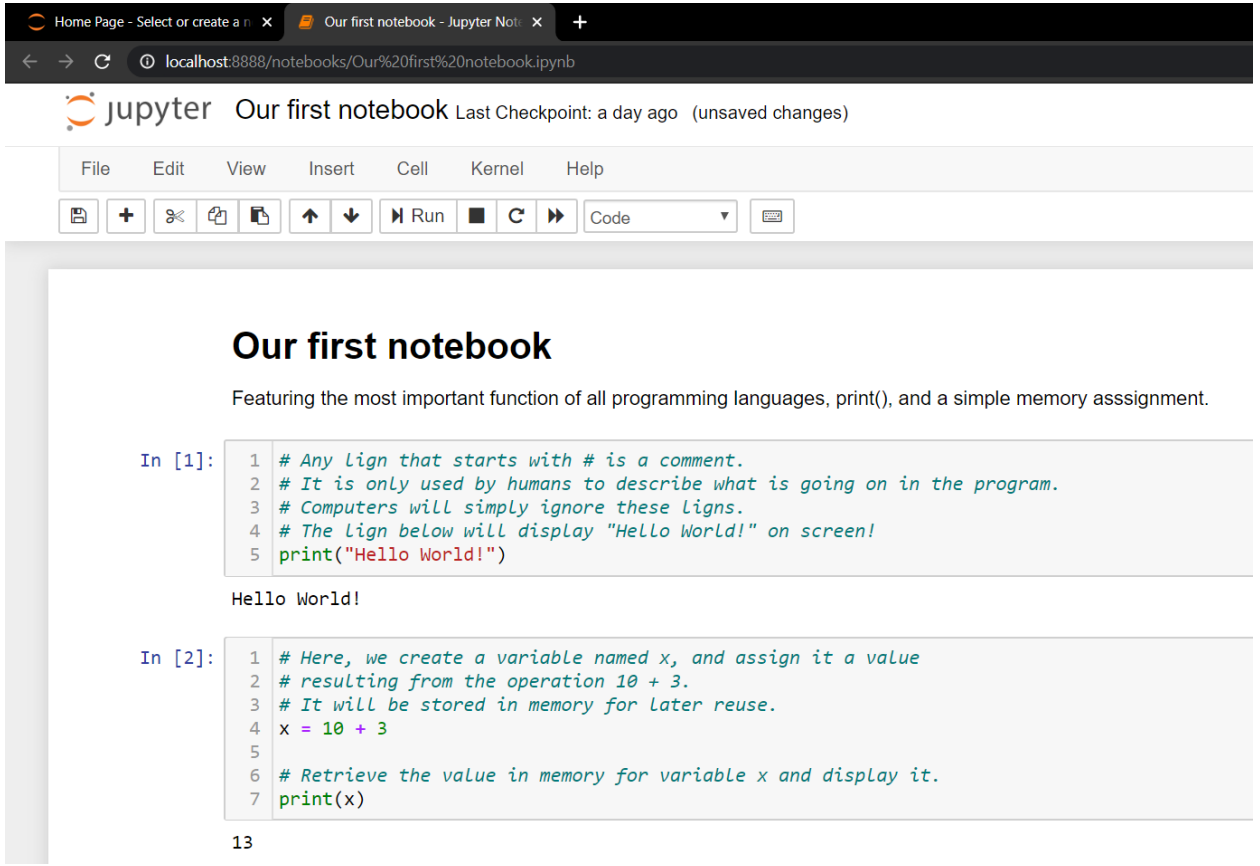
Running Python from an IDE, such as a Jupyter Notebook

- 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

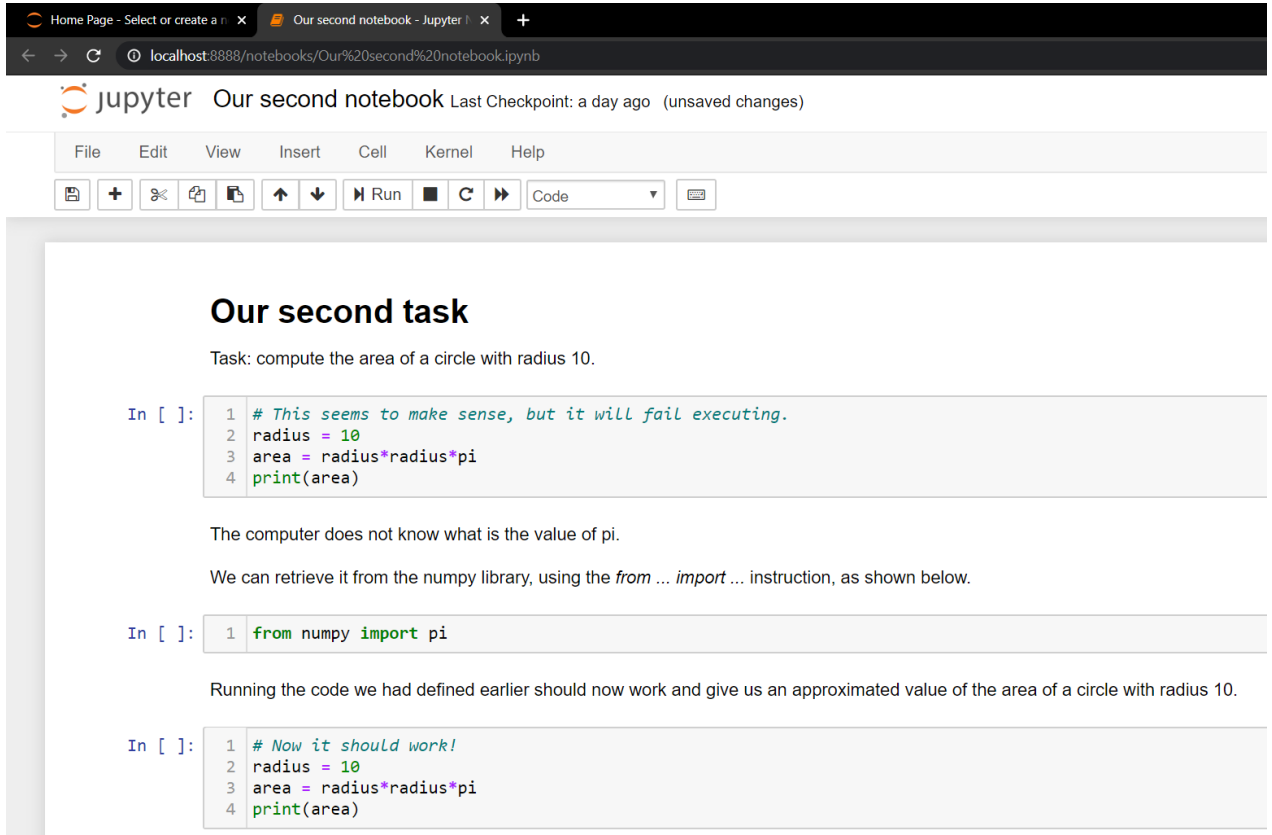
- Notebooks provide a mixed combination of
  - **text blocks** (in Markdown language)
  - and **code blocks** (in Python, these blocks have a “In […]” on their left side).
- Try executing a cell of code by selecting it and pressing **Shift+Enter**!
- A lot more convenient isn’t it?



Our second task

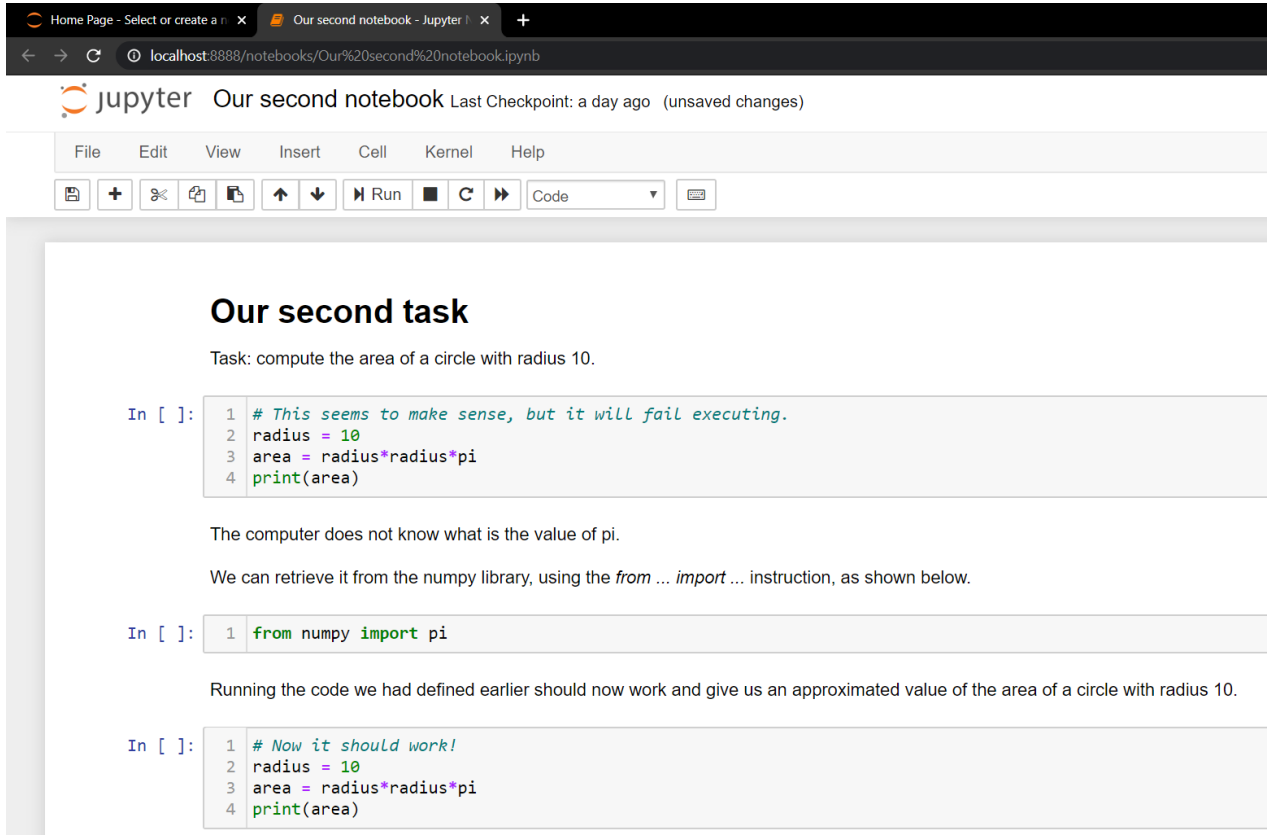
- Let us consider a **second task:** compute the area of a circle with radius 10.

- Open the second notebook.



Our second task

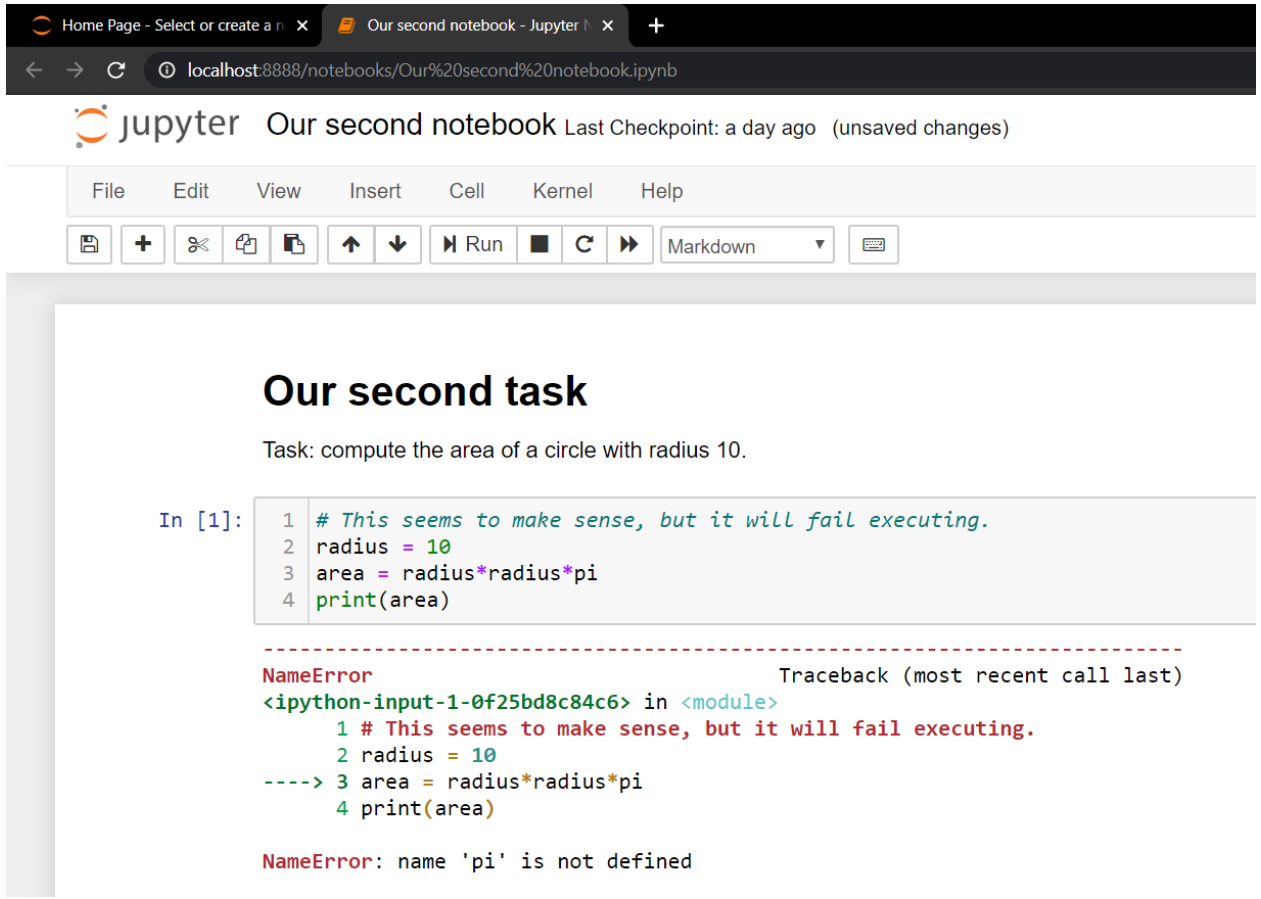
- Let us consider a **second task:** compute the area of a circle with radius 10.
- Open the second notebook.
- **Note:** in computer science, the multiplication operation is denoted $*$, not $\times$.



Our second task: problem

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

  → **We need the value of pi.**

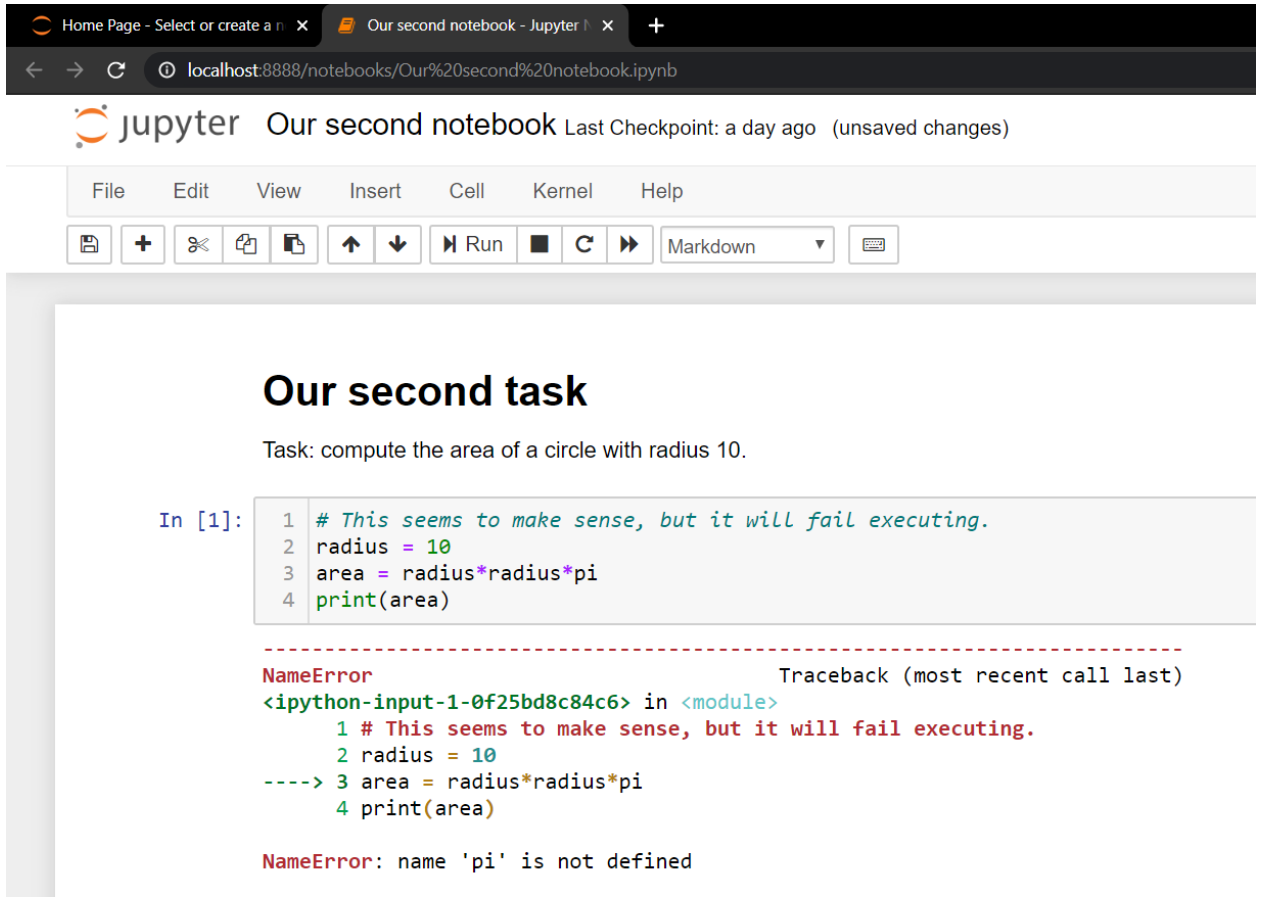


Our second task: problem

- 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 to retrieve it from a **package**.



Importing from a Python package

- By default, Python is pretty stupid and can only perform basic calculations (additions, multiplications, etc.)

Importing from a Python package

- By default, Python is pretty stupid and can only perform basic calculations (additions, multiplications, etc.)
- If we need more advanced concepts, we need to **import** them from a **package**.

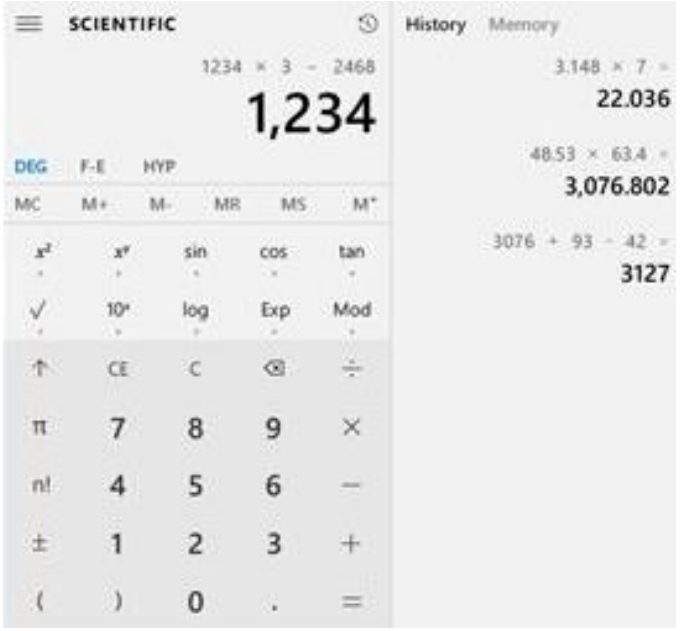
Importing from a Python package

- By default, Python is pretty stupid and can only perform basic calculations (additions, multiplications, etc.)
- If we need more advanced concepts, we need to **import** them from a **package**.

- **Import** ≈ adding a specific button to your calculator.

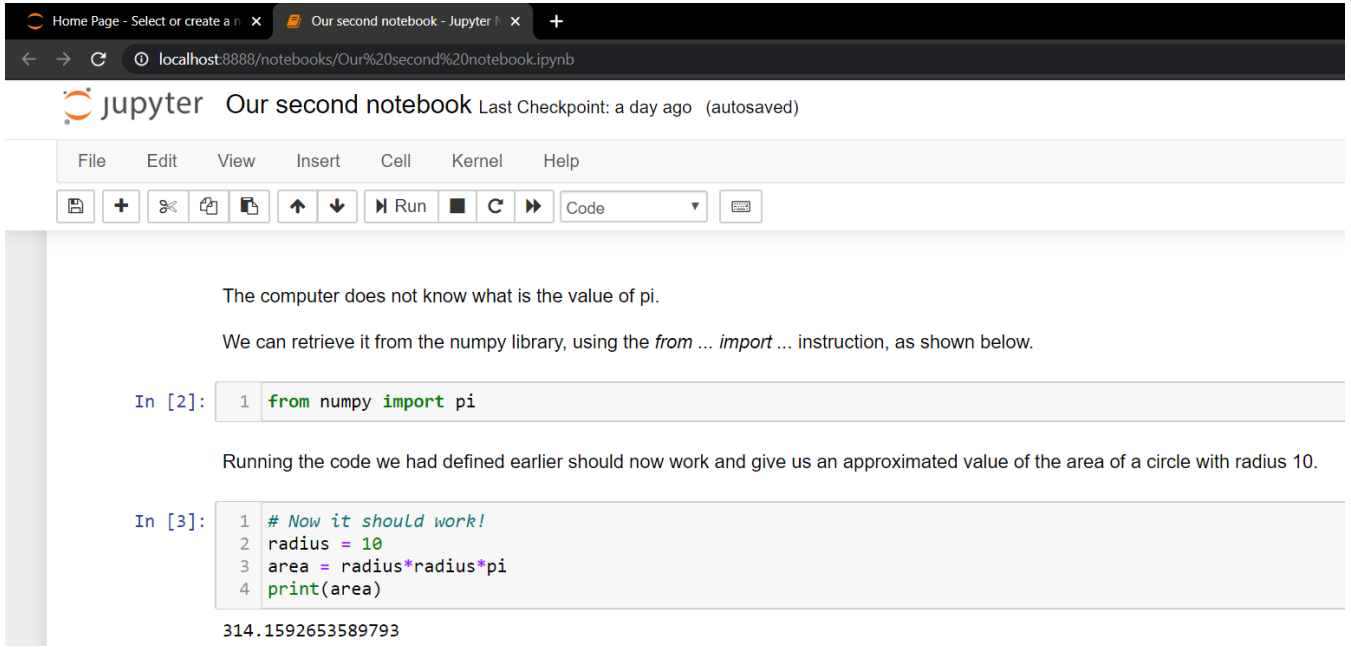
- By default, Python is pretty stupid and can only perform basic calculations (additions, multiplications, etc.)
- If we need more advanced concepts, we need to **import** them from a **package**.

- **Import** ≈ adding a specific button to your calculator.



Importing from a Python package

- By default, Python is pretty stupid and can only perform basic calculations (additions, multiplications, etc.)
- If we need more advanced concepts, we need to **import** them from a **package**.
- **Import** ≈ adding a specific button to your calculator.



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

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

python™

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.