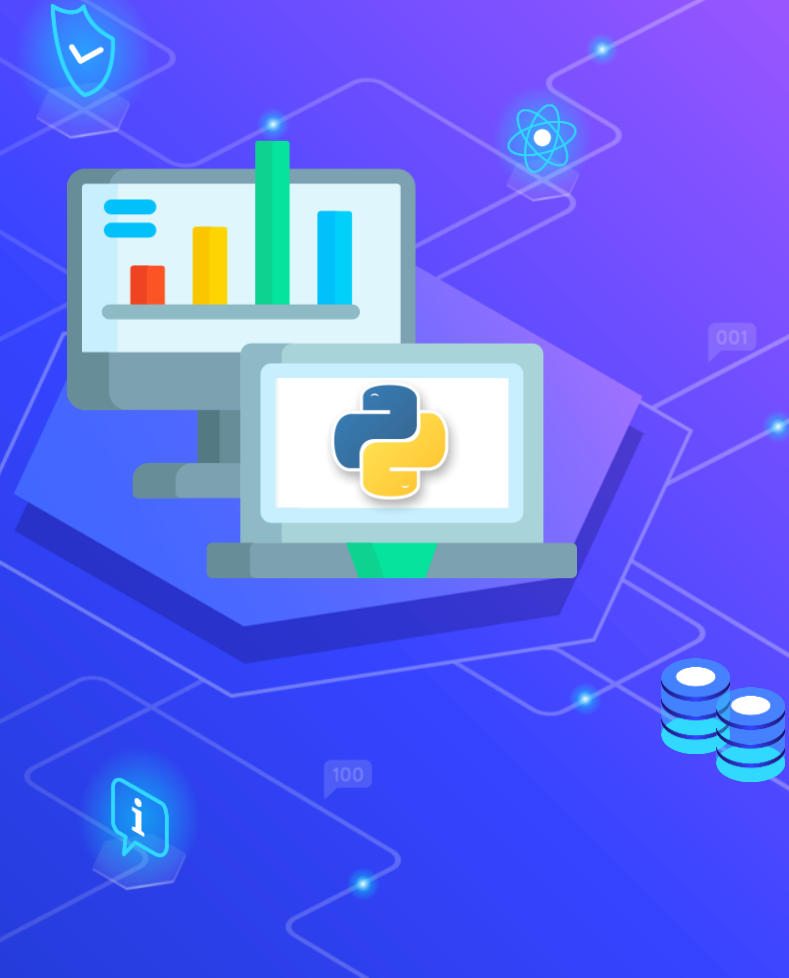


Data Science tools & Intro to python programming



Session Content

- ⬡ What we need to start ?
- ⬡ Why Python ?
- ⬡ How do we use Python ?
- ⬡ Code editor **VS** IDE
- ⬡ Jupyter Notebook
- ⬡ Visual Studio Code
- ⬡ Conda

- ⬡ Anaconda
- ⬡ Miniconda
- ⬡ Anaconda **VS** Miniconda
- ⬡ Virtual Environment in Python
- ⬡ Getting started with Python
 - Input and Output
 - Comments

Session Content

⬡ What we need to start ?

- ⬡ Why Python ?
- ⬡ How do we use Python ?
- ⬡ Code editor **VS** IDE
- ⬡ Jupyter Notebook
- ⬡ Visual Studio Code
- ⬡ Conda

- ⬡ Anaconda
- ⬡ Miniconda
- ⬡ Anaconda **VS** Miniconda
- ⬡ Virtual Environment in Python
- ⬡ Getting started with Python
 - Input and Output
 - Comments

What we need to Start ?

- As mentioned before, we need a programming language to communicate instructions to a machine, particularly a computer, **so which programming language we will choose ?**
- There are a lot of programming language, But every programming language has a specific function and therefore we must choose one of those that help us in the field of data science, **What programming languages are used in data science?**
- The most common programming languages are **(Python, R, JavaScript,)**
- In this diploma, we will work with Python, **But Why ?**

Session Content

- ⬡ What we need to start ?
- ⬡ Why Python ?
- ⬡ How do we use Python ?
- ⬡ Code editor **VS** IDE
- ⬡ Jupyter Notebook
- ⬡ Visual Studio Code
- ⬡ Conda

- ⬡ Anaconda
- ⬡ Miniconda
- ⬡ Anaconda **VS** Miniconda
- ⬡ Virtual Environment in Python
- ⬡ Getting started with Python
 - Input and Output
 - Comments

Why Python ?

1. multitasking : Python is a general-purpose, popular programming language and it is used in almost every technical field.

- Data Science
- Data Mining
- Desktop Applications
- Console-based Applications
- Mobile Applications
- Software Development
- Artificial Intelligence
- Web Applications
- Enterprise Applications
- Machine Learning
- Computer Vision or Image Processing Applications.
- Speech Recognitions

Why Python ?

1. multitasking : Python is a general-purpose, popular programming language and it is used in almost every technical field.



Why Python ?

2. Simplicity : Python is one of the easiest languages to start your journey. Also, its simplicity does not limit your functional possibilities, that's because

- This is a high-level programming
- Python is interpreted
- Python is fast in writing.

Java

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, world");  
    }  
}
```

Python

```
print("Hello, world")
```


Why Python ?

3. Huge Community : If you don't get support from other specialists, your learning path can be difficult .Even if you only have basic knowledge of the Python language, you can already use it for Machine Learning because of the huge number of libraries, resources, and tools available for you.

- Here you can get some help :

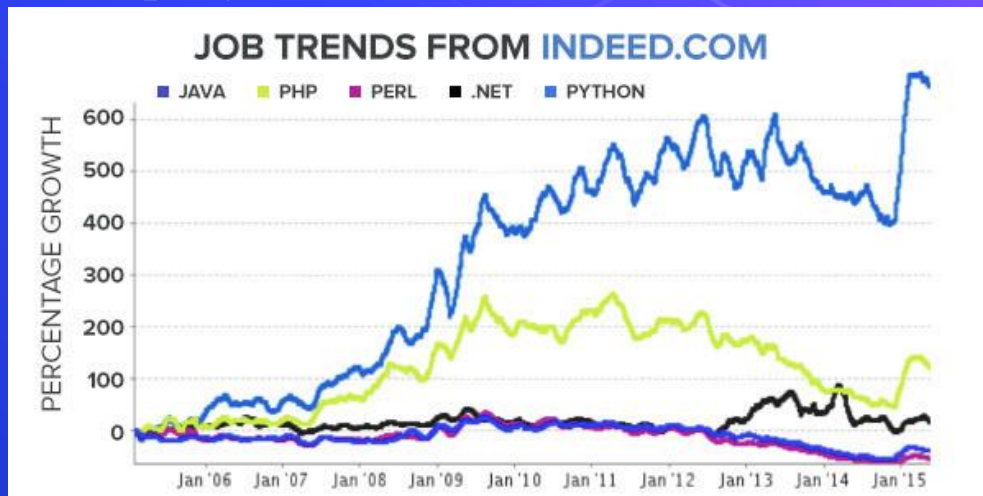
- **Irc Node:** <http://www.python.org/community/irc/>
- **StackOverflow:** <http://stackoverflow.com/questions/tagged/python?sort=newest>



Why Python ?

4. Jobs and Growth: Python is a unique language that has powerful growth and opens multiple career opportunities for Data Scientists. If you learn Python, you can consider multiple jobs:

- Python Developer
- Data Analyst
- Machine learning engineer
- Data Scientist



Why Python ?

5. Salary: If you are looking for high paying opportunities, Python has massive options for you.



Job	Average
Software Engineer / Developer / Programmer	\$79,629
Senior Data Scientist	\$128,906
Development Operations (DevOps) Engineer	\$94,410
Machine Learning Engineer	\$112,653
Junior Software Engineer	\$63,576
Test / Quality Assurance (QA) Engineer (Computer Software)	\$77,459
Principal Software Engineer	\$139,938

Session Content

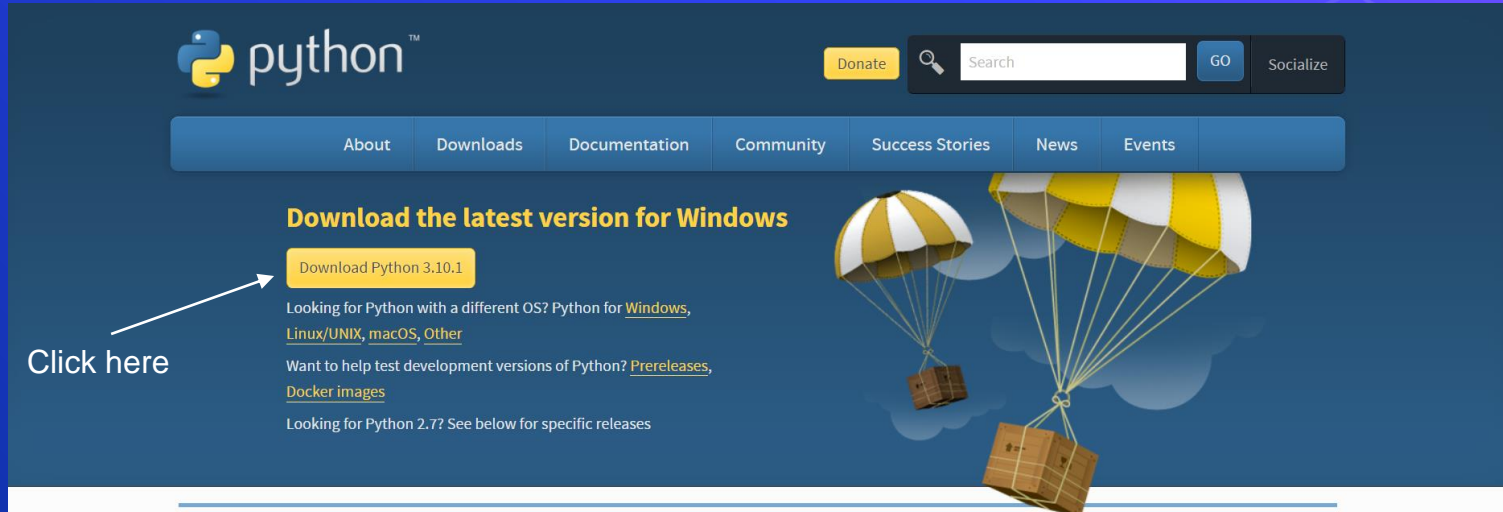
- ⬡ What we need to start ?
- ⬡ Why Python ?
- ⬡ How do we use Python ?
- ⬡ Code editor VS IDE
- ⬡ Jupyter Notebook
- ⬡ Visual Studio Code
- ⬡ Conda

- ⬡ Anaconda
- ⬡ Miniconda
- ⬡ Anaconda VS Miniconda
- ⬡ Virtual Environment in Python
- ⬡ Getting started with Python
 - Input and Output
 - Comments

How do we use Python ?

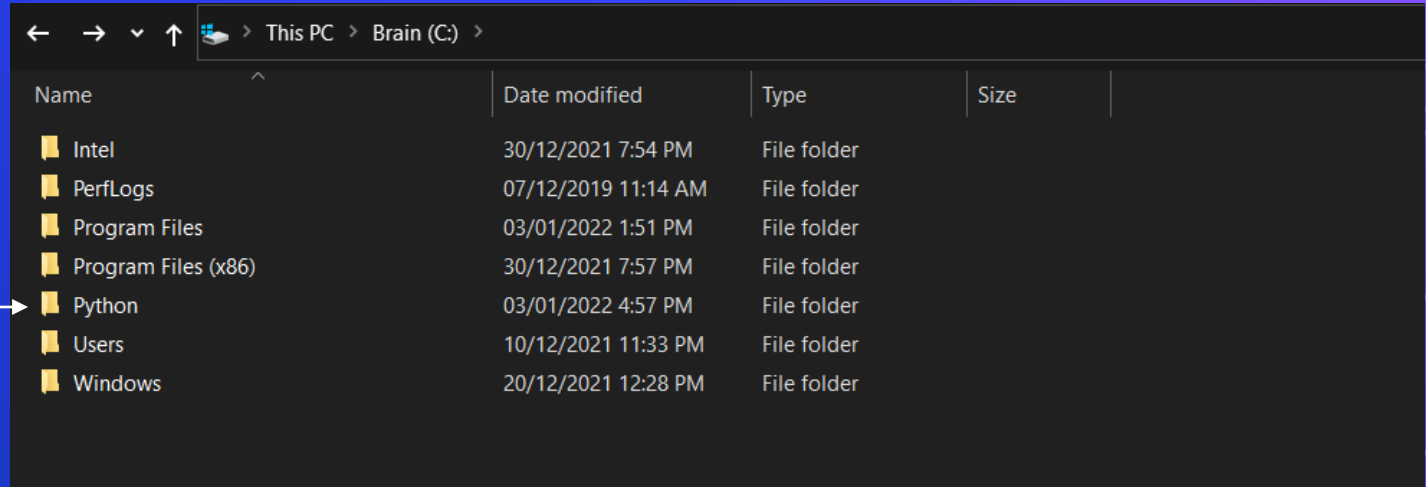
- To use python, we need to install interpreter to translate python code into machine language .

➤ Python official website : <https://www.python.org/downloads/>



How do we use Python ?

- **Step 1** : create the path where you want to install python :



Create file in C
partition to install
python in it

How do we use Python ?

- Step 2 : Run the installer



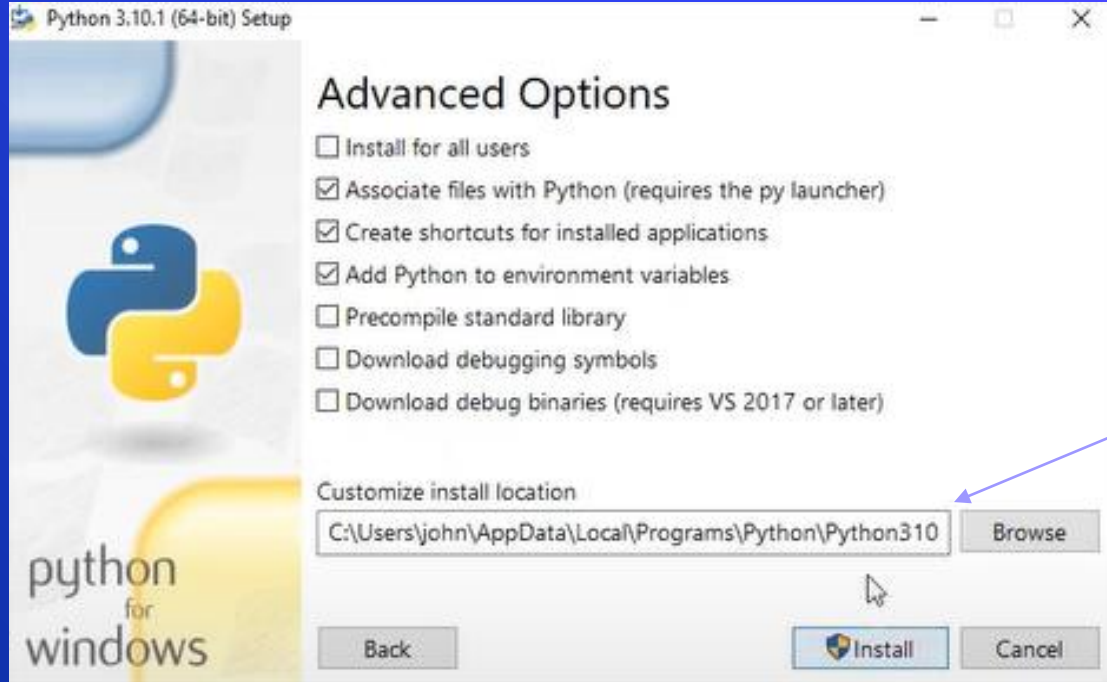
How do we use Python ?

- Step 2 : Run the installer



How do we use Python ?

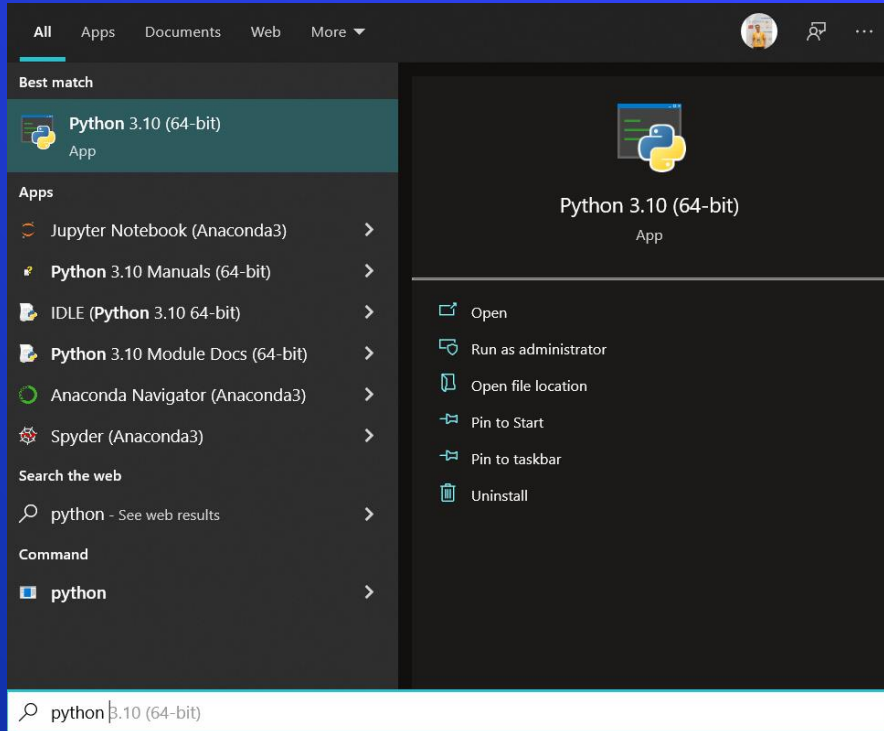
- Step 2 : Run the installer



Choose the path in the file which you created
Then, click install.

How do we use Python ?

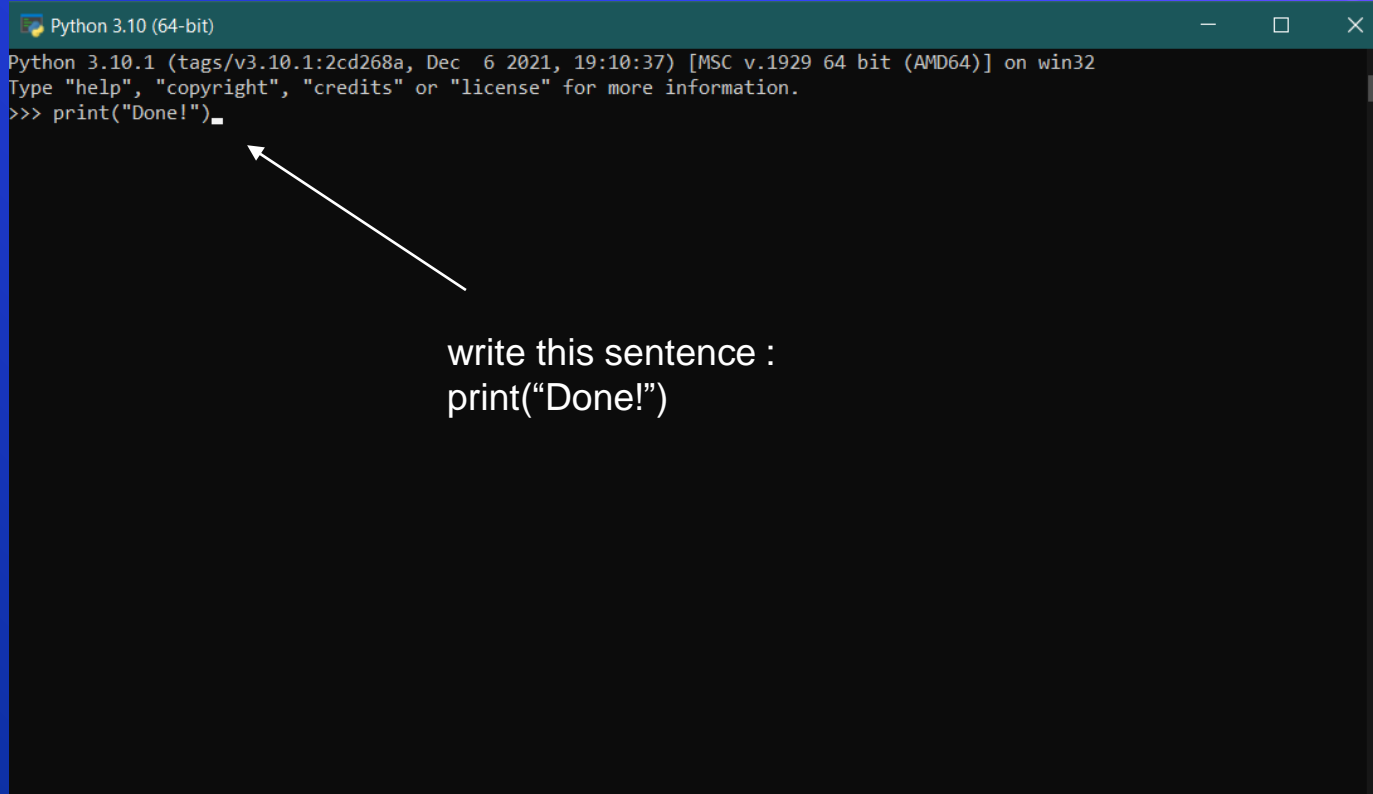
- Step 3 : Test the success of the operation



Search for
Python, then
click open

How do we use Python ?

- **Step 3** : Test the success of the operation

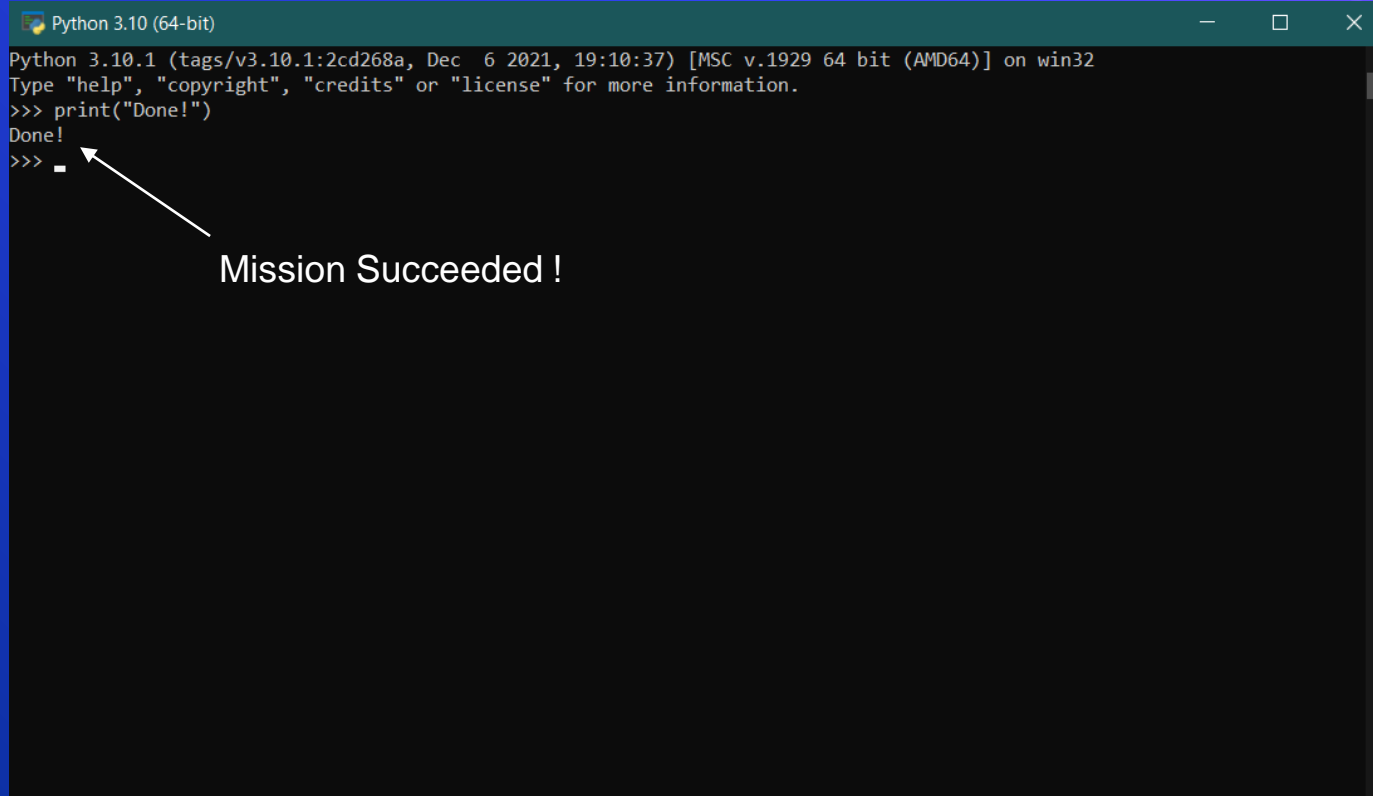
A screenshot of a Python 3.10 (64-bit) terminal window. The window title bar is dark green with the text 'Python 3.10 (64-bit)' and standard window controls. The terminal content shows the Python version and build information: 'Python 3.10.1 (tags/v3.10.1:2cd268a, Dec 6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32'. Below this, it says 'Type "help", "copyright", "credits" or "license" for more information.' and the prompt '>>>'. The command 'print("Done!")' is being typed at the prompt, with a white arrow pointing to it from the text 'write this sentence : print("Done!")' located to the right of the terminal window.

```
Python 3.10 (64-bit)
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec 6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Done!")
```

write this sentence :
`print("Done!")`

How do we use Python ?

- **Step 3** : Test the success of the operation



The image shows a terminal window titled "Python 3.10 (64-bit)". The window contains the following text:

```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec 6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Done!")
Done!
>>> _
```

An arrow points from the text "Mission Succeeded !" to the underscore character at the end of the last line of code.

Mission Succeeded !

How do we use Python ?

- Are we ready now ?

No, that's because python command line isn't easy to use and that's the reason why that we are need an IDE or Code Editor

- So, what is IDE or Code editor ?



Session Content

- ⬡ What we need to start ?
- ⬡ Why Python ?
- ⬡ How do we use Python ?
- ⬡ Code editor VS IDE
- ⬡ Jupyter Notebook
- ⬡ Visual Studio Code
- ⬡ Conda

- ⬡ Anaconda
- ⬡ Miniconda
- ⬡ Anaconda **VS** Miniconda
- ⬡ Virtual Environment in Python
- ⬡ Getting started with Python
 - Input and Output
 - Comments

IDE VS Code editor

- **IDE (Integrated development environment)**

is software for building applications that combines common developer tools into a single graphical user interface (GUI).

- **Code editor**

A text editor that can assist in writing software code with features such as syntax highlighting with visual cues, providing language specific auto-completion, and checking for bugs as code is being written.

IDE VS Code editor

- So why would or do you choose a traditional IDE instead of code editor ?

The answer would be practicality. For instance, imagine that you are coding in any text editor like Windows notepad. When your code is ready, you'll need to run it. You can't execute your program in a text editor like this, so you must use a prompt command to do it. Rather than use two different programs, wouldn't better have it all in just one place? That's what an IDE is ready for.

- What is most common IDEs and editors?

Jupyter Notebook, Spyder, PyCharm and Visual Studio Code

➤ In this diploma, we will focus on jupyter notebook and Visual Studio code, **But Why ?**

Session Content

- ⬡ What we need to start ?
- ⬡ Why Python ?
- ⬡ How do we use Python ?
- ⬡ Code editor **VS** IDE
- ⬡ Jupyter Notebook
- ⬡ Visual Studio Code
- ⬡ Conda

- ⬡ Anaconda
- ⬡ Miniconda
- ⬡ Anaconda **VS** Miniconda
- ⬡ Virtual Environment in Python
- ⬡ Getting started with Python
 - Input and Output
 - Comments

Jupyter Notebook

- Jupyter Notebook provides you with an easy-to-use, interactive data science environment across many programming languages
- Doesn't only work as an IDE, but also as a presentation or education tool.
- It's perfect for those who are just starting out with data science!



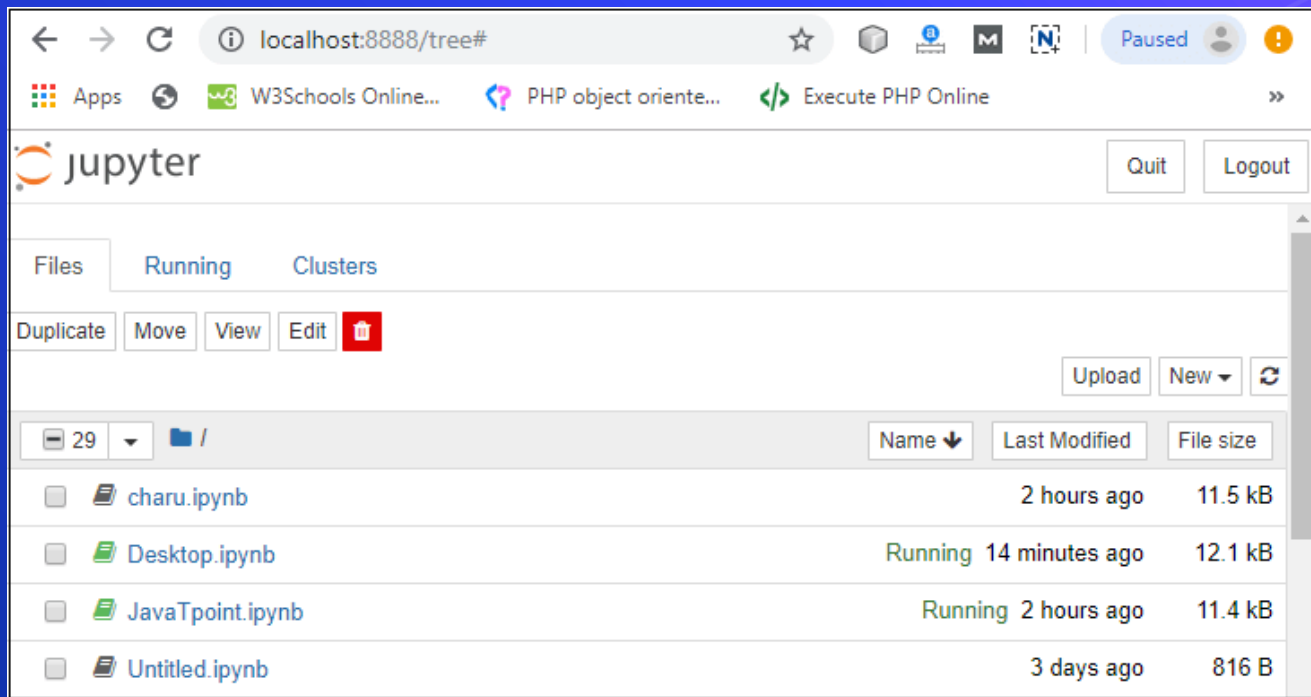
Jupyter Notebook

- Jupyter notebooks are now the base of important platforms as Google Collab.
- it is a Jupyter notebook that runs on Google servers with a free GPU and integrated with important libraries as PyTorch, TensorFlow, Keras and OpenCV. and you don't need to install anything.
- Moreover, the notebooks are saved to your Google Drive account.



Jupyter Notebook

- Jupyter notebook starts with the default web browser which shows the list of all python files.

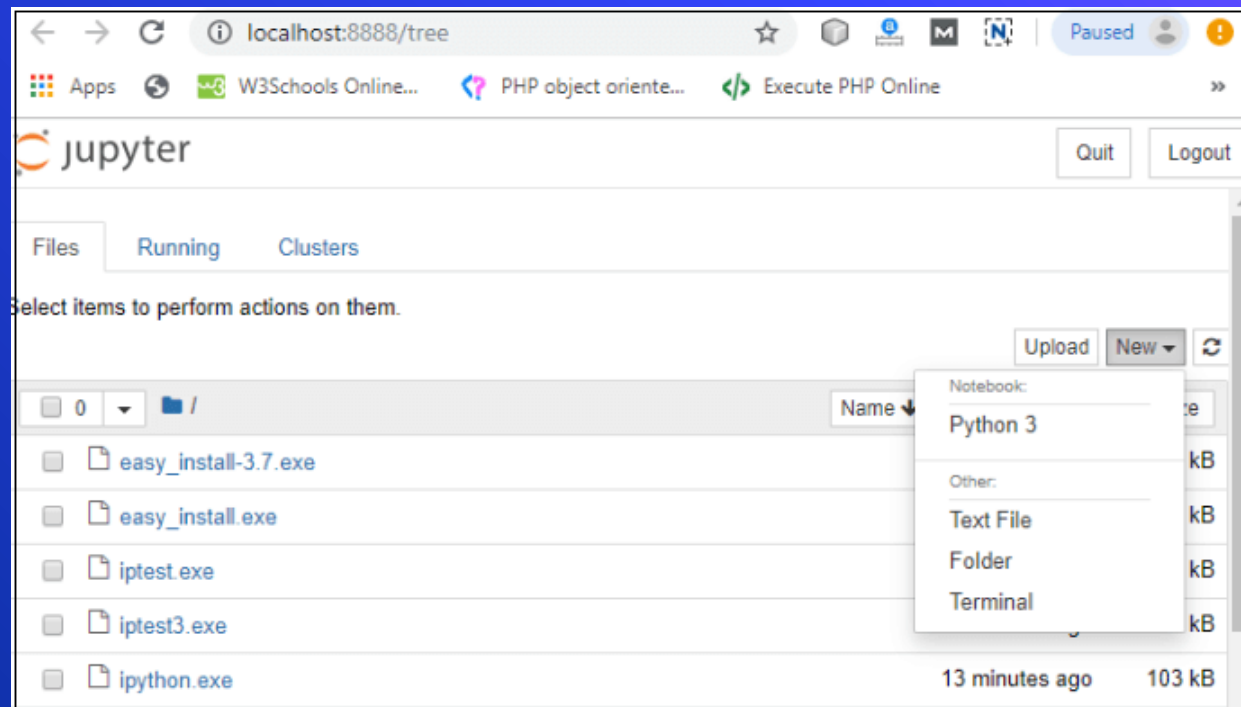


The screenshot shows a web browser window at the URL `localhost:8888/tree#`. The browser's address bar and tabs are visible at the top. The Jupyter interface includes a header with the Jupyter logo, a 'Quit' button, and a 'Logout' button. Below the header, there are tabs for 'Files', 'Running', and 'Clusters'. The 'Files' tab is active, displaying a file manager interface. At the top of the file manager, there are buttons for 'Duplicate', 'Move', 'View', 'Edit', and a trash icon. On the right side of the file manager, there are buttons for 'Upload', 'New', and a refresh icon. The main area of the file manager shows a list of files and folders. The list has columns for 'Name', 'Last Modified', and 'File size'. The files listed are:

	Name	Last Modified	File size
<input type="checkbox"/>	charu.ipynb	2 hours ago	11.5 kB
<input type="checkbox"/>	Desktop.ipynb	Running 14 minutes ago	12.1 kB
<input type="checkbox"/>	JavaTpoint.ipynb	Running 2 hours ago	11.4 kB
<input type="checkbox"/>	Untitled.ipynb	3 days ago	816 B

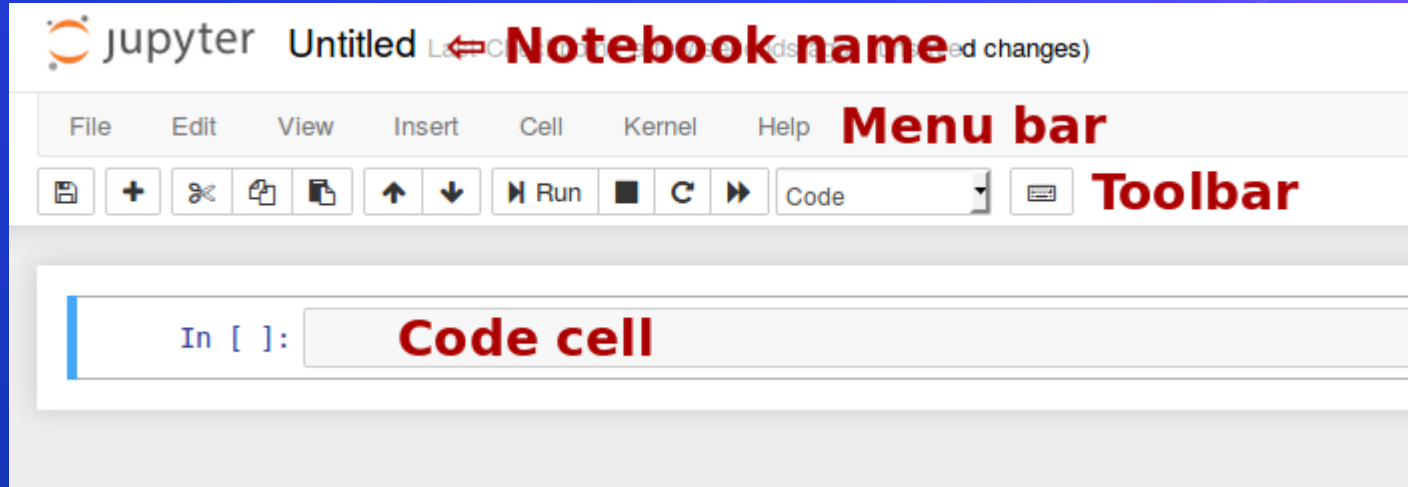
Jupyter Notebook

- To create a Notebook in Jupyter, go to New and select Python3.



Jupyter Notebook

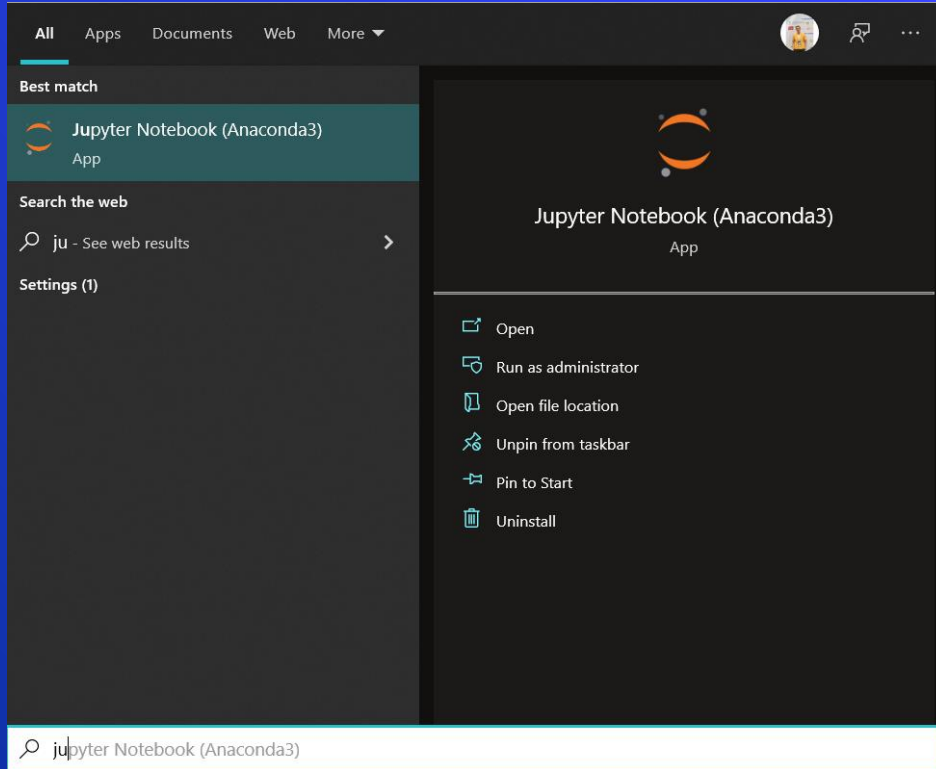
- When you create a new notebook.



- Full tutorial for Jupyter Notebook : <https://www.javatpoint.com/jupyter-notebook>

Jupyter Notebook

- for easy opening, write **jupyter notebook** in start menu.



Session Content

- ⬡ What we need to start ?
- ⬡ Why Python ?
- ⬡ How do we use Python ?
- ⬡ Code editor **VS** IDE
- ⬡ Jupyter Notebook
- ⬡ **Visual Studio Code**
- ⬡ Conda

- ⬡ Anaconda
- ⬡ Miniconda
- ⬡ Anaconda **VS** Miniconda
- ⬡ Virtual Environment in Python
- ⬡ Getting started with Python
 - Input and Output
 - Comments

Visual Studio Code

- Visual Studio Code combines the simplicity of a source code editor with powerful developer tools
- At its heart, Visual Studio Code features a lightning-fast source code editor, perfect for day-to-day use. With support for hundreds of languages.
- VS Code has support for Git so you can work with source control without leaving the editor

- Is there a something that brings together all the IDEs for data science in one place ?
 - **yes, *Anaconda is here for you.***
 - But first we should know what is Conda ?***



Session Content

- ⬡ What we need to start ?
- ⬡ Why Python ?
- ⬡ How do we use Python ?
- ⬡ Code editor **VS** IDE
- ⬡ Jupyter Notebook
- ⬡ Visual Studio Code
- ⬡ Conda

- ⬡ Anaconda
- ⬡ Miniconda
- ⬡ Anaconda **VS** Miniconda
- ⬡ Virtual Environment in Python
- ⬡ Getting started with Python
 - Input and Output
 - Comments

What is Conda ?

- Conda is a language-agnostic tool for package management and environment management. As a package manager, Conda can install, update and remove packages. As an environment manager, it can manage virtual environments.



How can we use it ?

Conda & pip package managers

<https://anaconda.org/>

```
1 conda install --package name--
```

<https://pypi.org/>

```
1 pip install --package name--
```



Conda & pip differences

- Pip installs Python packages whereas Conda installs packages which may contain software written in any language.
- For example, before using pip, a Python interpreter must be installed via a system package manager or by downloading and running an installer.
- Conda on the other hand can install Python packages as well as the Python interpreter directly.
- Another key difference between the two tools is that Conda can create isolated environments
- Pip has no built-in support for environments but rather depends on other tools like virtualenv or venv to create isolated environments.
- Tools such as pipenv, poetry, and hatch wrap pip and virtualenv to provide a unified method for working with these environments.



Session Content

- ⬡ What we need to start ?
- ⬡ Why Python ?
- ⬡ How do we use Python ?
- ⬡ Code editor **VS** IDE
- ⬡ Jupyter Notebook
- ⬡ Visual Studio Code
- ⬡ Conda

⬡ Anaconda

- ⬡ Miniconda
- ⬡ Anaconda **VS** Miniconda
- ⬡ Virtual Environment in Python
- ⬡ Getting started with Python
 - Input and Output
 - Comments

ANACONDA

- Anaconda is a tool that offers all the required tools involved in data science at once. The programmers choose Anaconda for its ease of use.
- Anaconda aims at simplifying the data management process
- **Download Now :**
<https://www.anaconda.com/products/individual>



ANACONDA



Individual Edition

Your data science toolkit

With over 25 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.

Anaconda Individual Edition

Download 

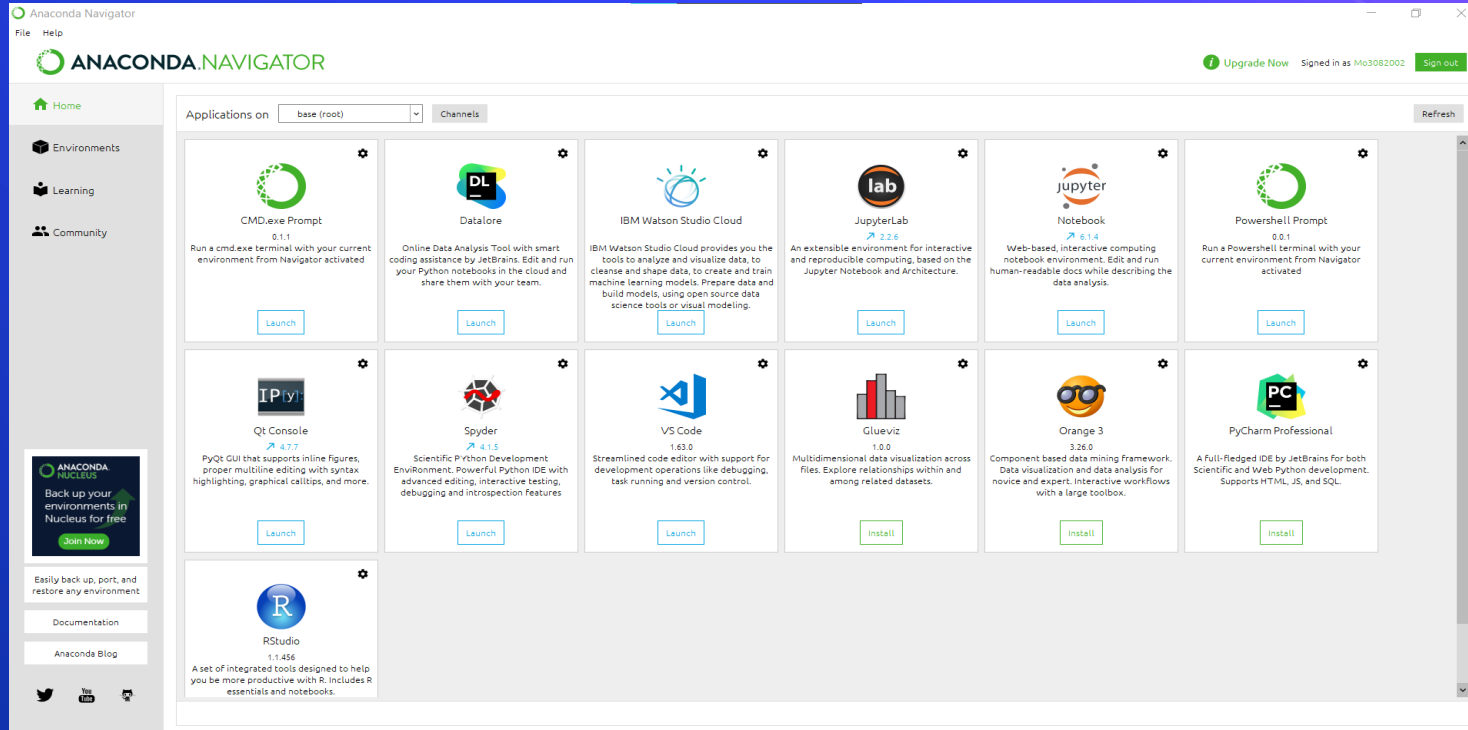
For Windows

Python 3.9 • 64-Bit Graphical Installer • 510 MB

Get Additional Installers



ANACONDA



➤ Here are all the tools a data scientist needs

Session Content

- ⬡ What we need to start ?
- ⬡ Why Python ?
- ⬡ How do we use Python ?
- ⬡ Code editor **VS** IDE
- ⬡ Jupyter Notebook
- ⬡ Visual Studio Code
- ⬡ Conda

- ⬡ Anaconda
- ⬡ Miniconda
- ⬡ Anaconda **VS** Miniconda
- ⬡ Virtual Environment in Python
- ⬡ Getting started with Python
 - Input and Output
 - Comments

Miniconda

- Miniconda is a mini version of Anaconda
- Miniconda is essentially an installer for an empty Conda environment, containing only Conda, its dependencies, and Python.

- **Download Now :**

<https://docs.conda.io/en/latest/miniconda.html#miniconda>

The Miniconda logo is displayed on a blue, 3D-style hexagonal background. The word "MINI" is in black, and "CONDA" is in green. A green snake icon, representing Anaconda, is integrated into the letter 'C' of "CONDA". The background of the slide features a network of white lines and hexagons with binary code (010, 011, 001, 100) and blue dots, suggesting a digital or network theme.

MINI CONDA®

Session Content

- ⬡ What we need to start ?
- ⬡ Why Python ?
- ⬡ How do we use Python ?
- ⬡ Code editor **VS** IDE
- ⬡ Jupyter Notebook
- ⬡ Visual Studio Code
- ⬡ Conda

- ⬡ Anaconda
- ⬡ Miniconda
- ⬡ **Anaconda VS Miniconda**
- ⬡ Virtual Environment in Python
- ⬡ Getting started with Python
 - Input and Output
 - Comments

Anaconda or Miniconda?

Choose Anaconda if you:



- ✓ Are new to Conda or Python.
- ✓ Like the convenience of having Python and over 1,500 scientific packages automatically installed at once.
- ✓ Have the time and disk space---a few minutes and 3 GB.
- ✓ Do not want to individually install each of the packages you want to use.
- ✓ Wish to use a curated and vetted set of packages.

Choose Miniconda if you:

MINICONDA®

- ✓ Do not mind installing each of the packages you want to use individually.
- ✓ Do not have time or disk space to install over 1,500 packages at once.
- ✓ Want fast access to Python and the conda commands and you wish to sort out the other programs later.

Session Content

- ⬡ What we need to start ?
- ⬡ Why Python ?
- ⬡ How do we use Python ?
- ⬡ Code editor **VS** IDE
- ⬡ Jupyter Notebook
- ⬡ Visual Studio Code
- ⬡ Conda

- ⬡ Anaconda
- ⬡ Miniconda
- ⬡ Anaconda **VS** Miniconda
- ⬡ **Virtual Environment in Python**
- ⬡ Getting started with Python
 - Input and Output
 - Comments

What is virtual environment?



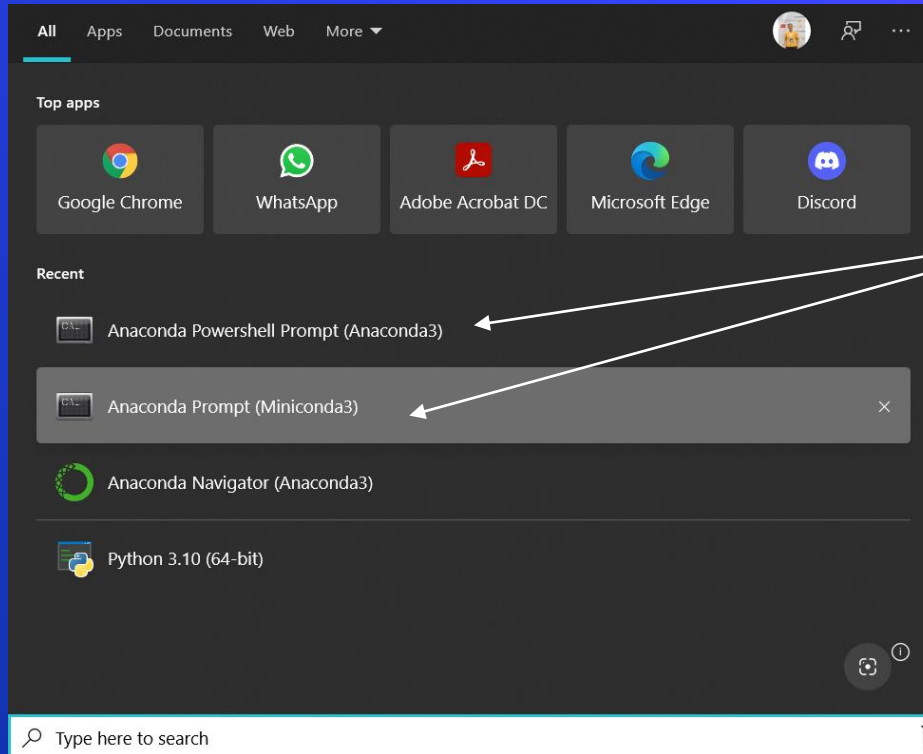
- A virtual environment is a tool that helps to keep dependencies required by different projects separate by creating isolated python virtual environments for them.

Why virtual environment is important ?

- Imagine a scenario where you are working on two python projects and one of them uses a Python 2.0 and the other uses Python 3.0 and so on. In such situations virtual environment can be really useful to maintain dependencies of both the projects.

How to setup virtual environment with Conda ?

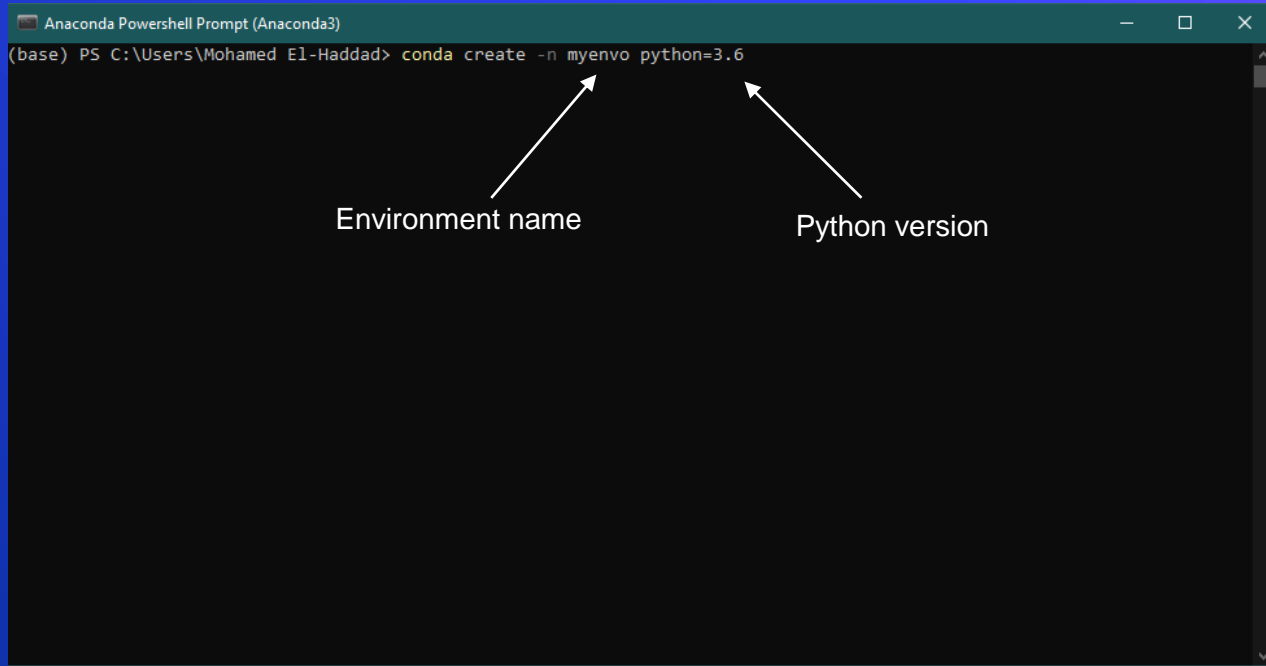
Step1 : Open Start menu and search for Anaconda prompt



Choose one of them
depend on what
you installed.

How to setup virtual environment with Conda ?

Step2 : Write this Command



```
Anaconda Powershell Prompt (Anaconda3)
(base) PS C:\Users\Mohamed El-Haddad> conda create -n myenvo python=3.6
```

The screenshot shows a terminal window titled "Anaconda Powershell Prompt (Anaconda3)". The command entered is `conda create -n myenvo python=3.6`. Two white arrows point from text labels to parts of the command: one from "Environment name" to `-n myenvo` and another from "Python version" to `python=3.6`.

How to setup virtual environment with Conda ?

Step3 : Choose Y

```
Anaconda Powershell Prompt (Anaconda3)

(base) PS C:\Users\Mohamed El-Haddad> conda create -n myenv python=3.6
WARNING: A space was detected in your requested environment path
'C:\Users\Mohamed El-Haddad\.conda\envs\myenvo'
Spaces in paths can sometimes be problematic.
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\Mohamed El-Haddad\.conda\envs\myenvo

  added / updated specs:
    - python=3.6

The following NEW packages will be INSTALLED:

 certifi           pkgs/main/noarch::certifi-2020.6.20-pyhd3eb1b0_3
 pip               pkgs/main/win-64::pip-21.2.2-py36haa95532_0
 python            pkgs/main/win-64::python-3.6.13-h3758d61_0
 setuptools        pkgs/main/win-64::setuptools-58.0.4-py36haa95532_0
 sqlite            pkgs/main/win-64::sqlite-3.37.0-h2bbff1b_0
 vc                pkgs/main/win-64::vc-14.2-h21ff451_1
 vs2015_runtime    pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
 wheel             pkgs/main/noarch::wheel-0.37.0-pyhd3eb1b0_1
 wincertstore      pkgs/main/win-64::wincertstore-0.2-py36h7fe50ca_0

Proceed ([y]/n)?
```

Press Y

How to setup virtual environment with Conda ?

Step2 : Active your environment

```
Anaconda Powershell Prompt (Anaconda3)

The following NEW packages will be INSTALLED:

certifi      pkgs/main/noarch::certifi-2020.6.20-pyhd3eb1b0_3
pip          pkgs/main/win-64::pip-21.2.2-py36haa95532_0
python       pkgs/main/win-64::python-3.6.13-h3758d61_0
setuptools   pkgs/main/win-64::setuptools-58.0.4-py36haa95532_0
sqlite       pkgs/main/win-64::sqlite-3.37.0-h2bfff1b_0
vc           pkgs/main/win-64::vc-14.2-h21ff451_1
vs2015_runtime pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
wheel        pkgs/main/noarch::wheel-0.37.0-pyhd3eb1b0_1
wincertstore pkgs/main/win-64::wincertstore-0.2-py36h7fe50ca_0

Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#   $ conda activate myenvo
#
# To deactivate an active environment, use
#
#   $ conda deactivate

(base) PS C:\Users\Mohamed El-Haddad> conda activate myenvo
```

Write this commandd

Session Content

- ⬡ What we need to start ?
- ⬡ Why Python ?
- ⬡ How do we use Python ?
- ⬡ Code editor **VS** IDE
- ⬡ Jupyter Notebook
- ⬡ Visual Studio Code
- ⬡ Conda

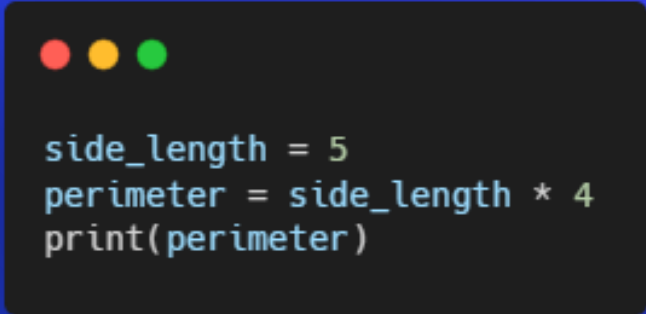
- ⬡ Anaconda
- ⬡ Miniconda
- ⬡ Anaconda **VS** Miniconda
- ⬡ Virtual Environment in Python
- ⬡ Getting started with Python
 - **Input and Output**
 - Comments

Data input

- Programs are written to solve problems. To solve a problem, a program needs **data input** to make a process on it.
- Data can be input in different ways:
 - Written directly into the program. **This is called hard coding.**
 - By the user when the program is running.

Data input

- Written directly into the program. **This is called hard coding.**
- Consider this **Python** program for calculating the perimeter of a square:



```
side_length = 5
perimeter = side_length * 4
print(perimeter)
```

- The data in the variable 'side_length' has been hard coded, ie it has been written directly into the program.

Data input

- By the user when the program is running.
- Consider this **Python** program for calculating the perimeter of a square:

```
side_length = input("Type in a side length: ")  
side_length = int(side_length)  
perimeter = side_length * 4  
print("The perimeter of the square is: ")  
print(perimeter)
```

- This time, the data for the variable 'side_length' is input by the user when the program is running.

Data Output

- Once data has been processed, programs often need to output the data they have generated. In Python, the 'print' statement is used to output data.

```
side_length = input("Type in a side length: ")
side_length = int(side_length)
perimeter = side_length * 4
print("The perimeter of the square is: ")
print(perimeter)
```

- Display a message explaining what information is being output. Text is placed within quotes.
- Output the contents of the variable 'perimeter'. Variables are not placed within quotes.

Quiz

Q. Which of the following is considered a hard coding

A -



```
side_length = 5  
perimeter = side_length * 4  
print(perimeter)
```

B -



```
side_length = input("Type in a side length: ")  
side_length = int(side_length)  
perimeter = side_length * 4  
print("The perimeter of the square is: ")  
print(perimeter)
```



Quiz

Q. To get the users input what function do we use?

A - input

B - input()

C - input<>

D - INPUT()



Multiple Choice

Session Content

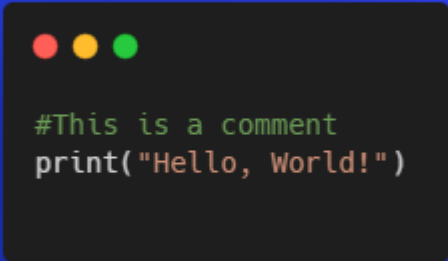
- ⬡ What we need to start ?
- ⬡ Why Python ?
- ⬡ How do we use Python ?
- ⬡ Code editor **VS** IDE
- ⬡ Jupyter Notebook
- ⬡ Visual Studio Code
- ⬡ Conda

- ⬡ Anaconda
- ⬡ Miniconda
- ⬡ Anaconda **VS** Miniconda
- ⬡ Virtual Environment in Python
- ⬡ Getting started with Python
 - Input and Output
 - **Comments**

Comments

- Comments are the lines in the code that are ignored by the compiler during the execution of the program.

- Comments can be used to explain Python code.
- Comments can be used to make the code more readable.



```
#This is a comment  
print("Hello, World!")
```

Comments

- Comments are the lines in the code that are ignored by the compiler during the execution of the program.

- Comments can be used to explain Python code.
- Comments can be used to make the code more readable.

```
"""  
This is a comment  
written in  
more than just one line  
"""  
print("Hello, World!")
```

Quiz

Q. What symbol do you use to make a comment in Python?

A - '''

B - #

C - !

D - @



Multiple Choice

Questions ?!



Thanks!

