

PYTHON QUICK REFRESHER



Introduction to Python

What is Python?

- Python is one of the most popular programming languages, created by **Guido van Rossum** and released in 1991.
- Python is an **interpreted** programming language as opposed to a compiled programming language.



Why python?

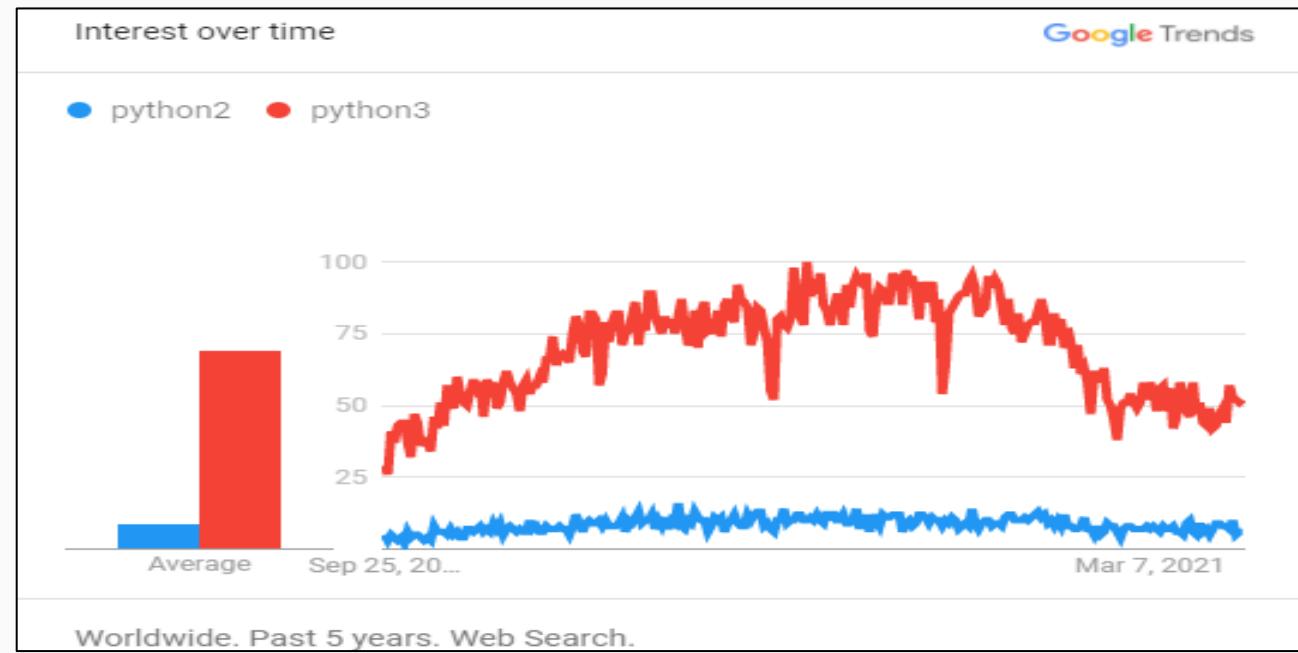
- Python is a beginner-friendly language due to its simplified syntax which is close to natural language.
- It has been designed to write clear, logical code for small and large-scale projects.
- It is used extensively in the industry for software development, data science, artificial intelligence, and machine learning.



Setting Up Python

Python3 vs Python2

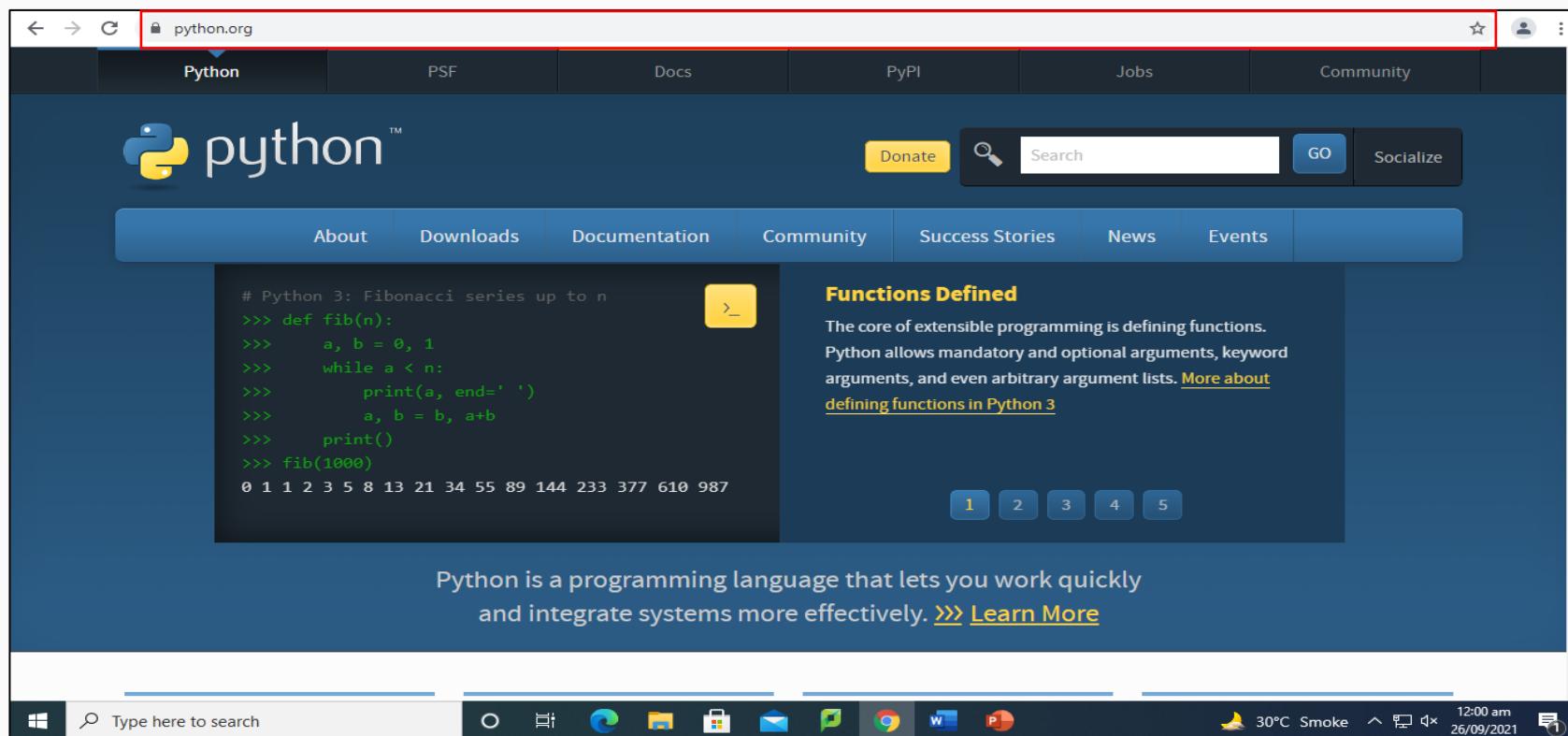
- Python2 is an outdated version of python. Python3 is the current version and is in-demand.
- In this course, we will be learning Python3 .





Setting Up Python – Windows (1/8)

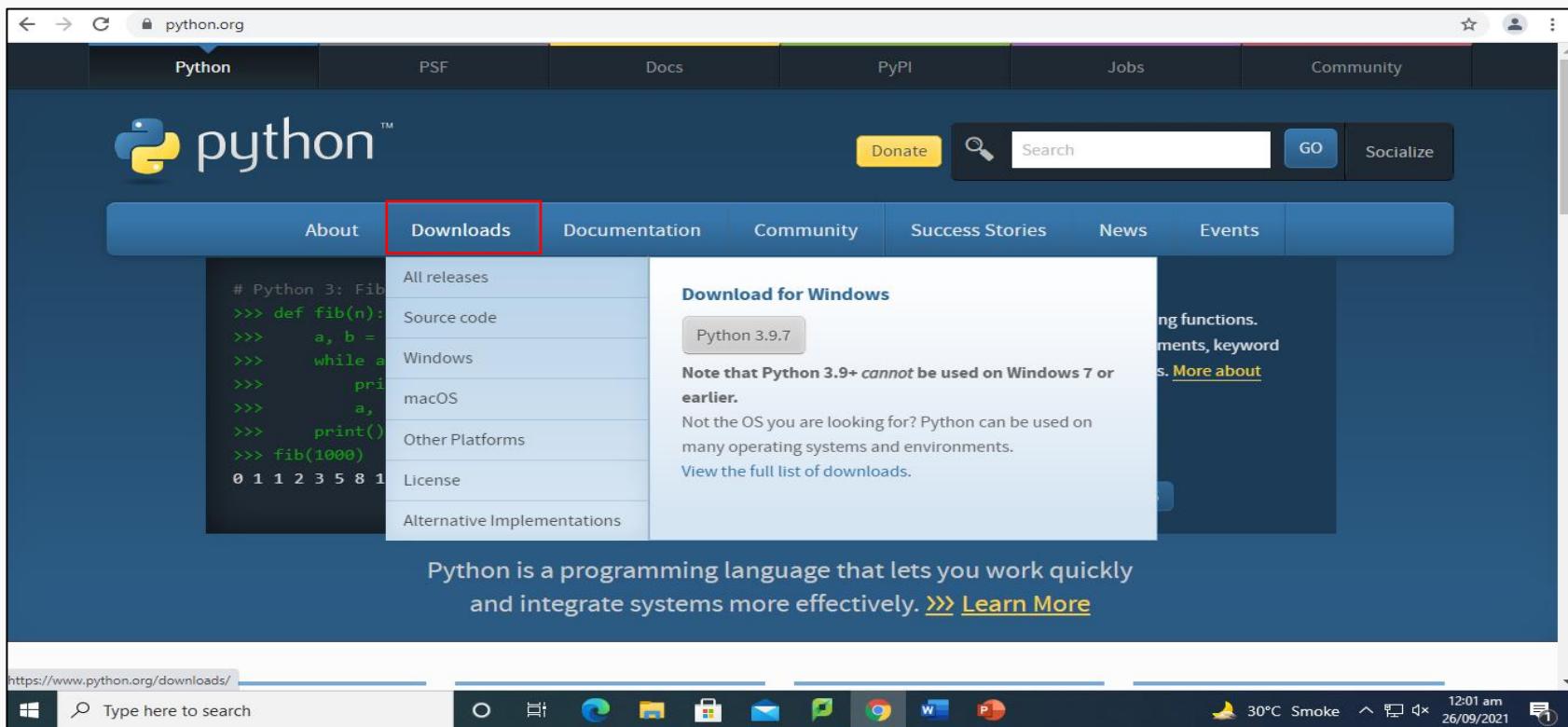
- Visit: <https://www.python.org/>





Setting Up Python – Windows (2/8)

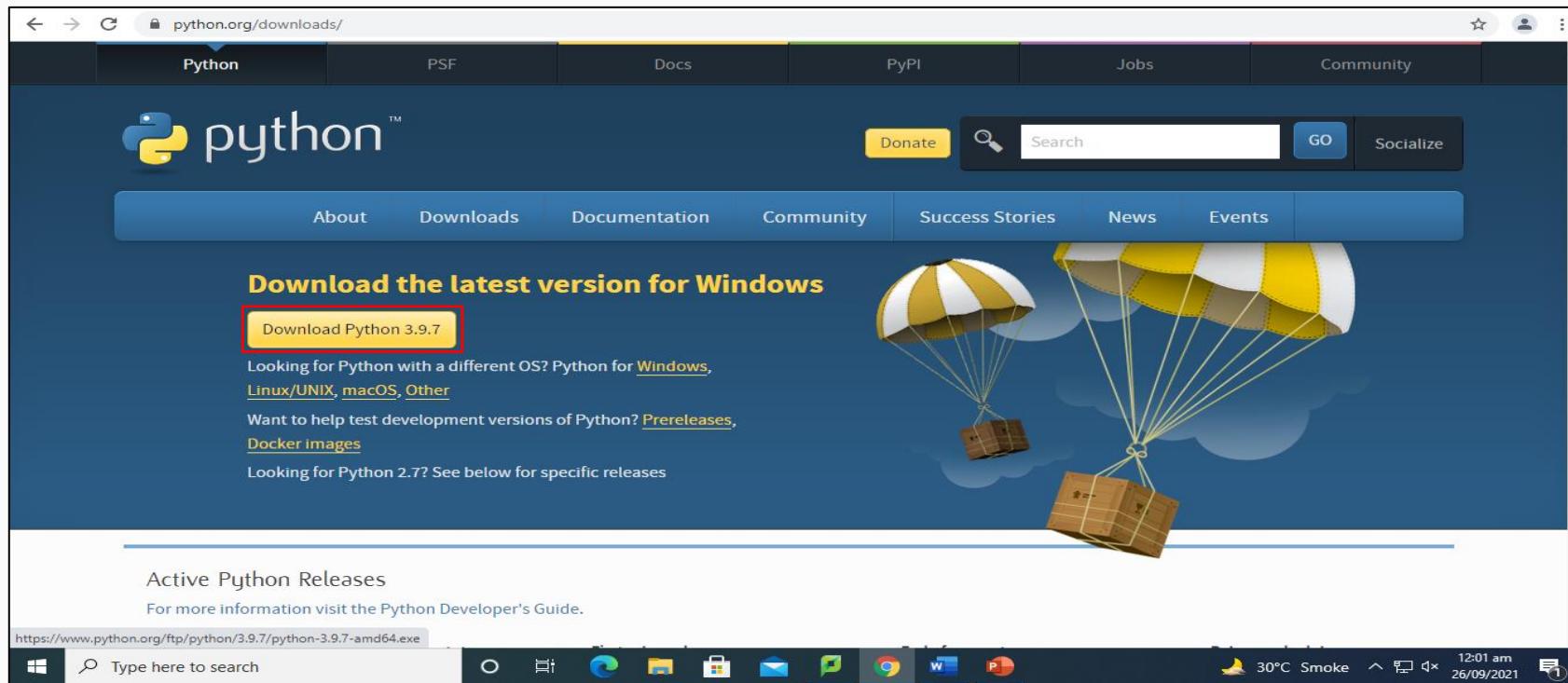
- Click on Downloads.





Setting Up Python – Windows (3/8)

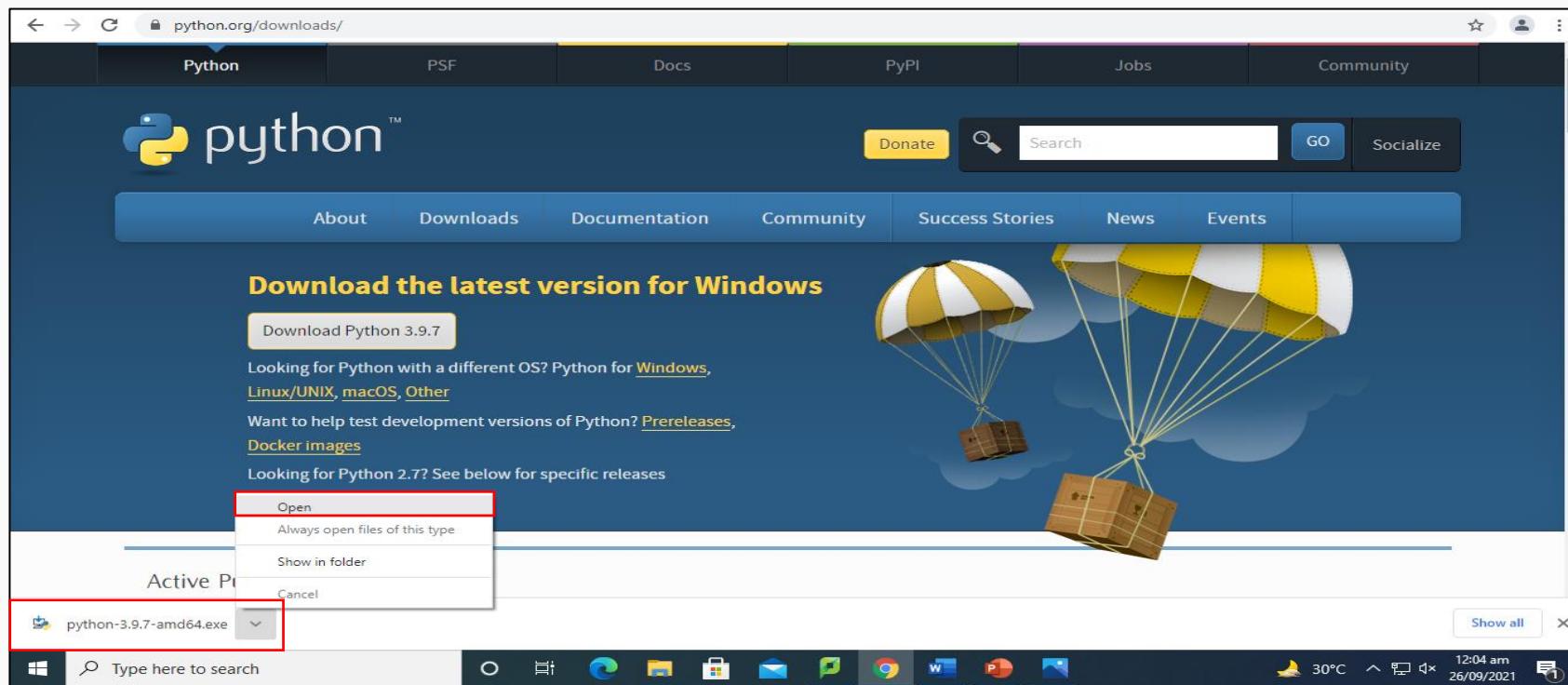
- This will show you the latest version of Python3 for your operating system. Download it.





Setting Up Python – Windows (4/8)

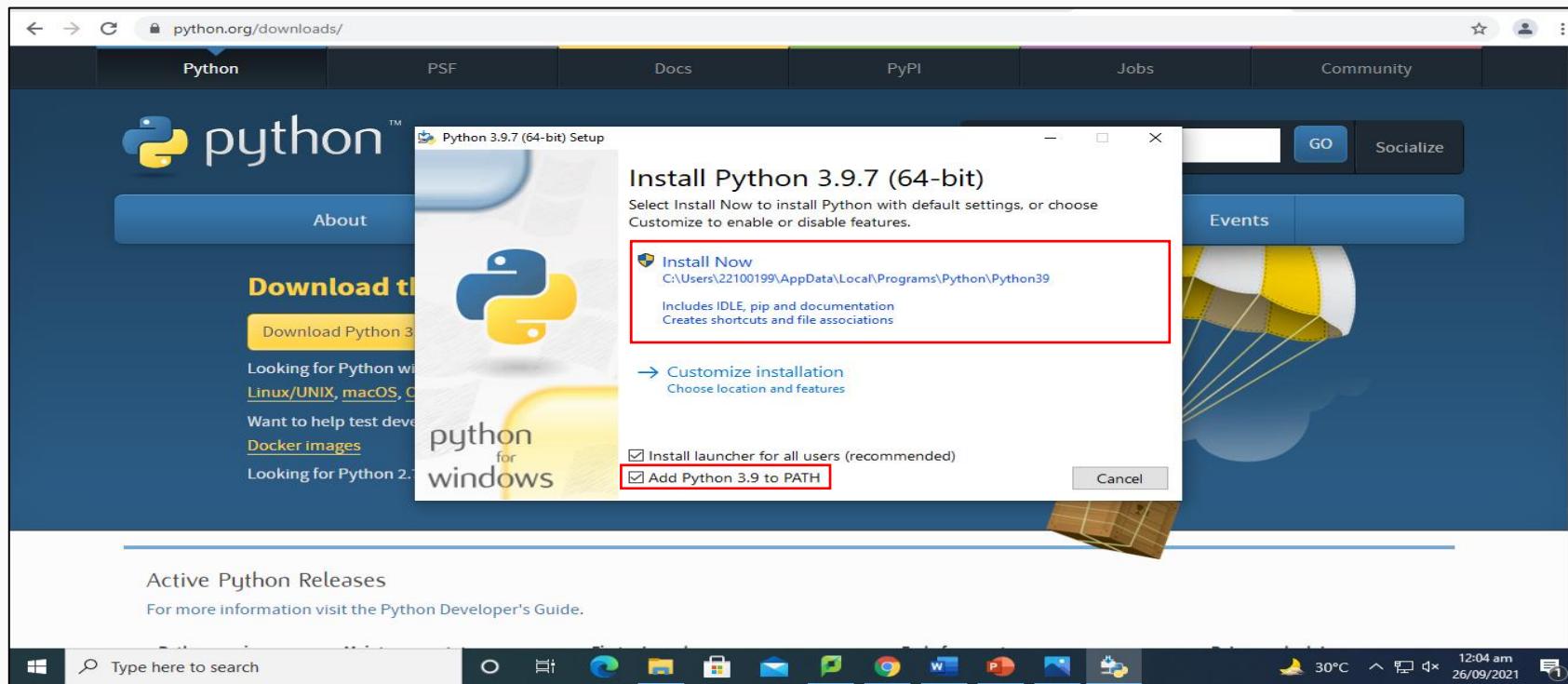
- Once downloaded, click Open.





Setting Up Python – Windows (5/8)

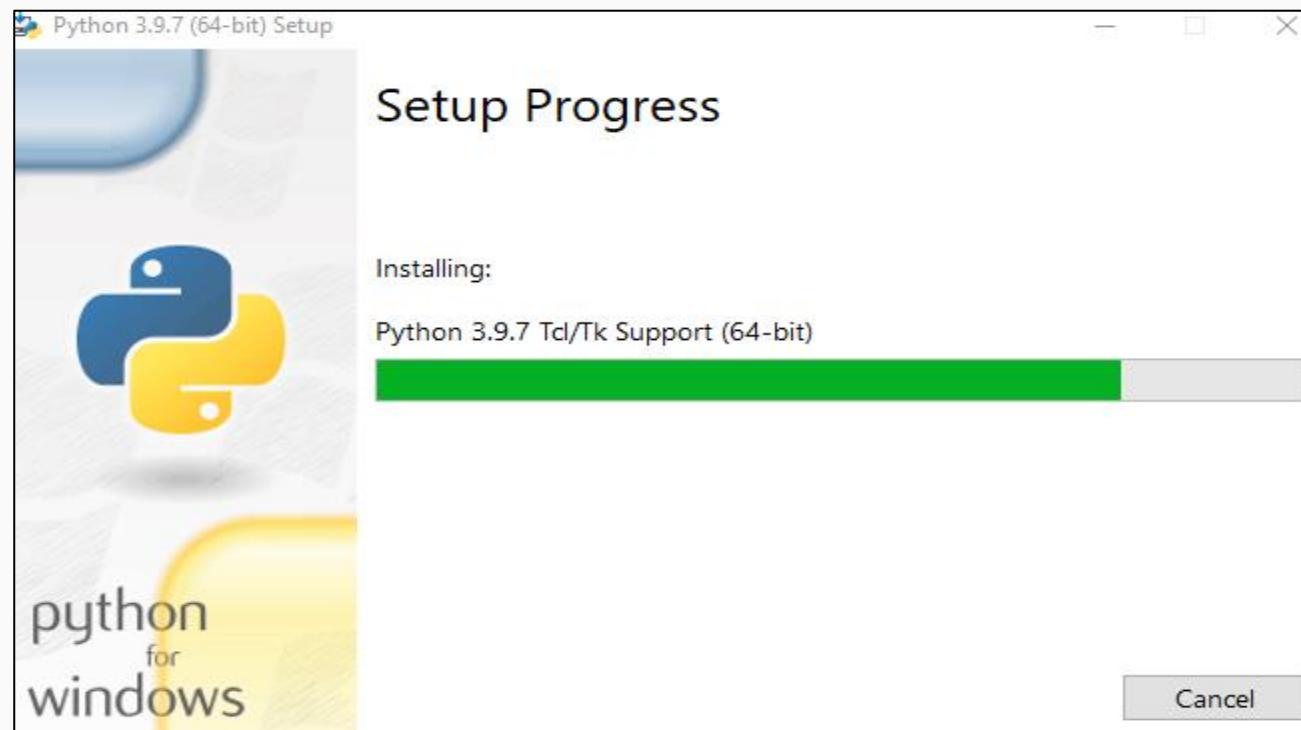
- Installation wizard will open up. Make sure that 'Add Python 3.9 to Path' is checked. Click on Install Now.





Setting Up Python – Windows (6/8)

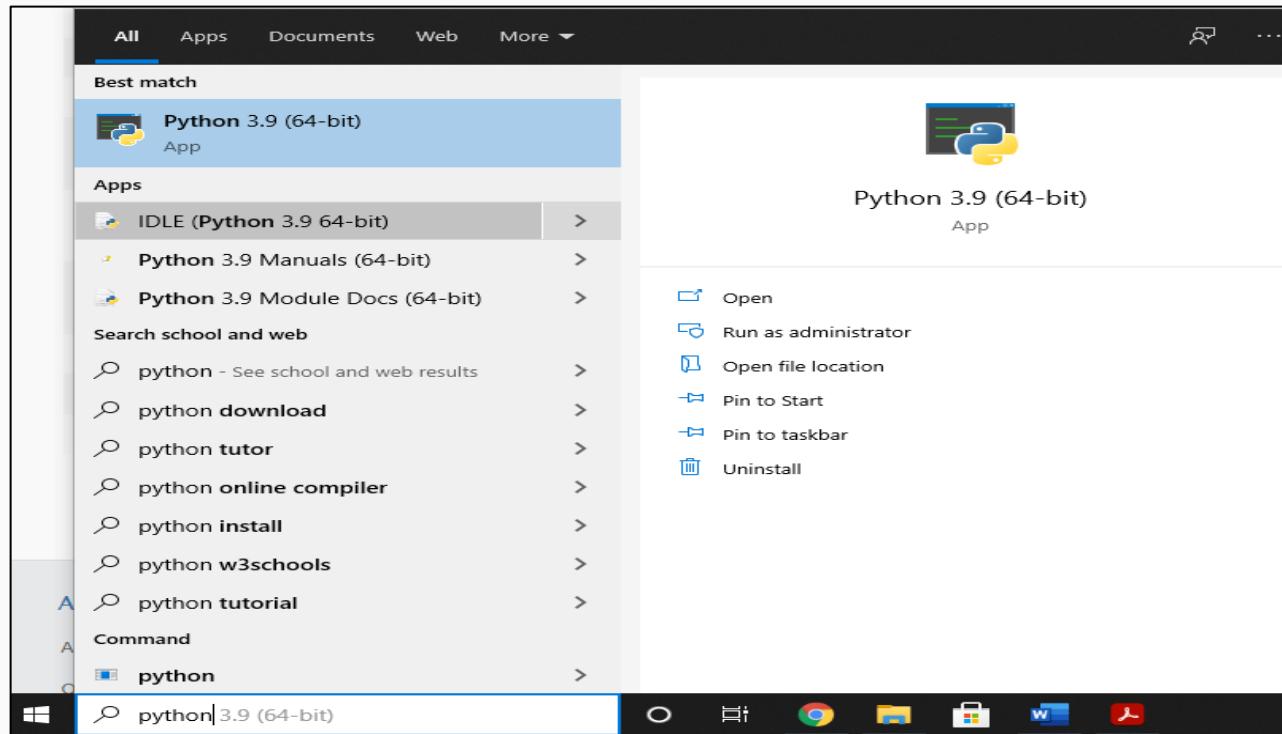
- It will take sometime to install.





Setting Up Python – Windows (7/8)

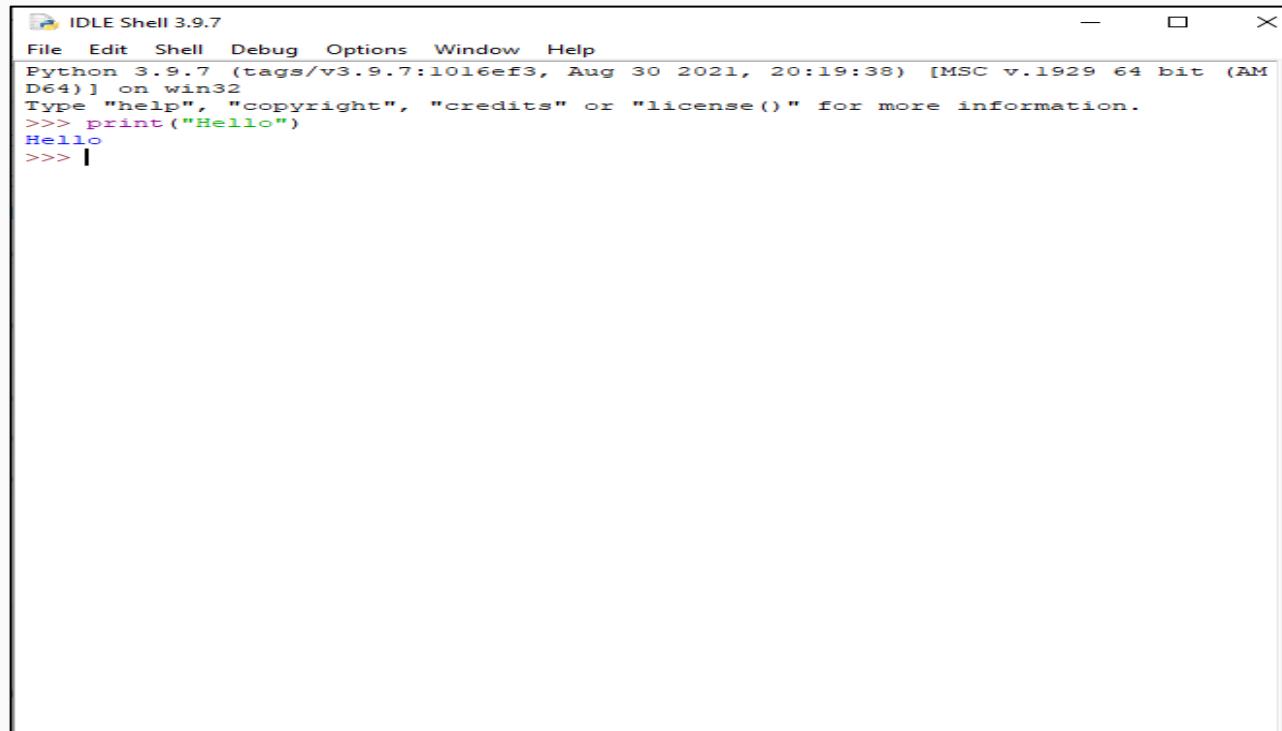
- Once installed, type python in your search bar and click on ‘IDLE (Python 3.9 64-bit)’.





Setting Up Python – Windows (8/8)

- This will open up Python IDLE. Type print("Hello") and press Enter to execute.



The screenshot shows the Python IDLE Shell window. The title bar reads "IDLE Shell 3.9.7". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The status bar at the bottom indicates "Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32". The main window displays the following text:
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello")
Hello
>>> |



Setting Up Python – Mac (1/13)

- Visit: <https://www.python.org/>

The screenshot shows the Python.org homepage. A red box highlights the browser's address bar, which displays the URL <https://www.python.org/>. The page features a dark blue header with the Python logo and navigation links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the header is a search bar with a 'Search' button and a 'Socialize' button. The main content area includes a code snippet demonstrating a Fibonacci series, a section titled 'Functions Defined' with text about defining functions in Python 3, and a footer with a 'Learn More' link.

Python 3: Fibonacci series up to n

```
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
987
```

Functions Defined

The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. [More about defining functions in Python 3](#)

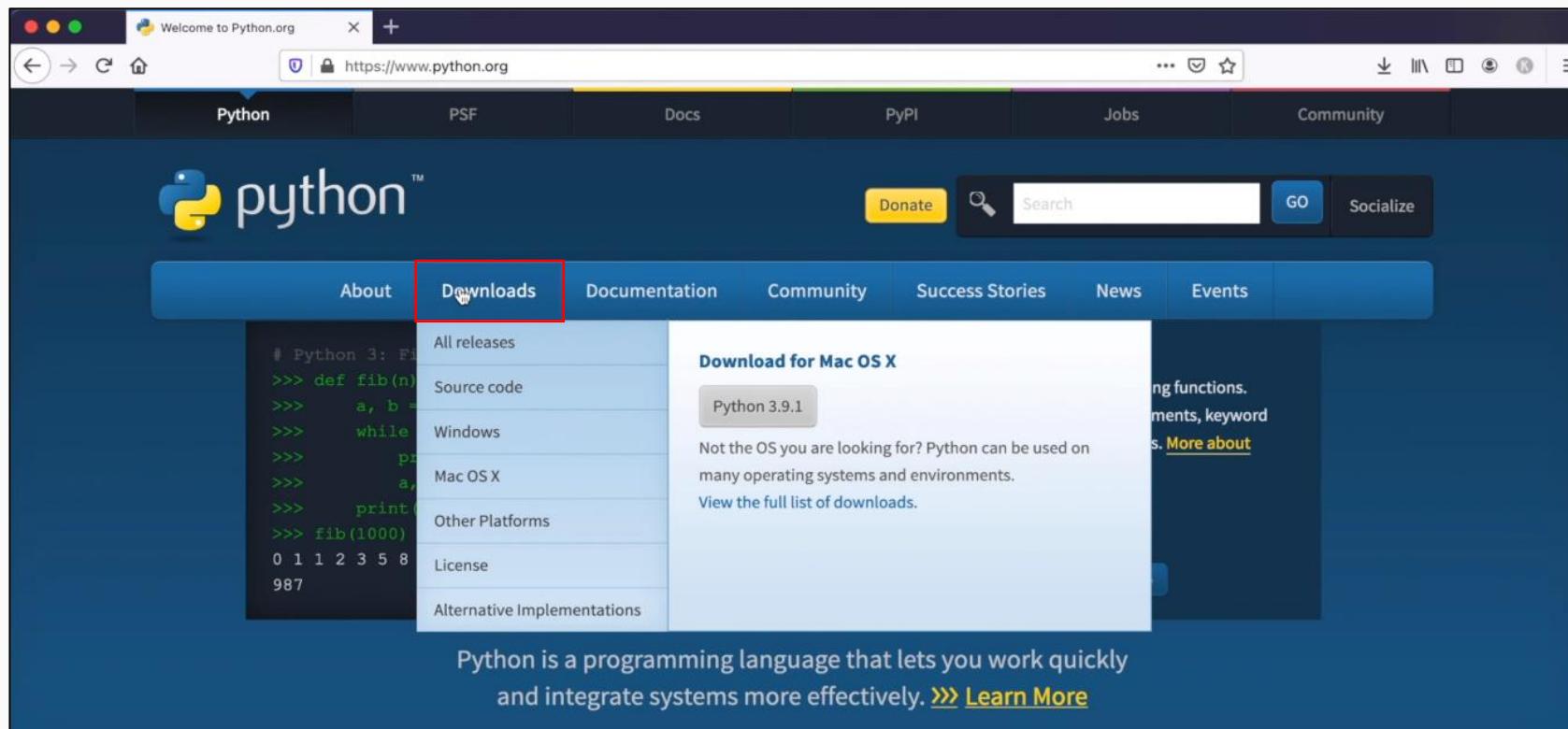
1 2 3 4 5

Python is a programming language that lets you work quickly and integrate systems more effectively. [» Learn More](#)



Setting Up Python – Mac (2/13)

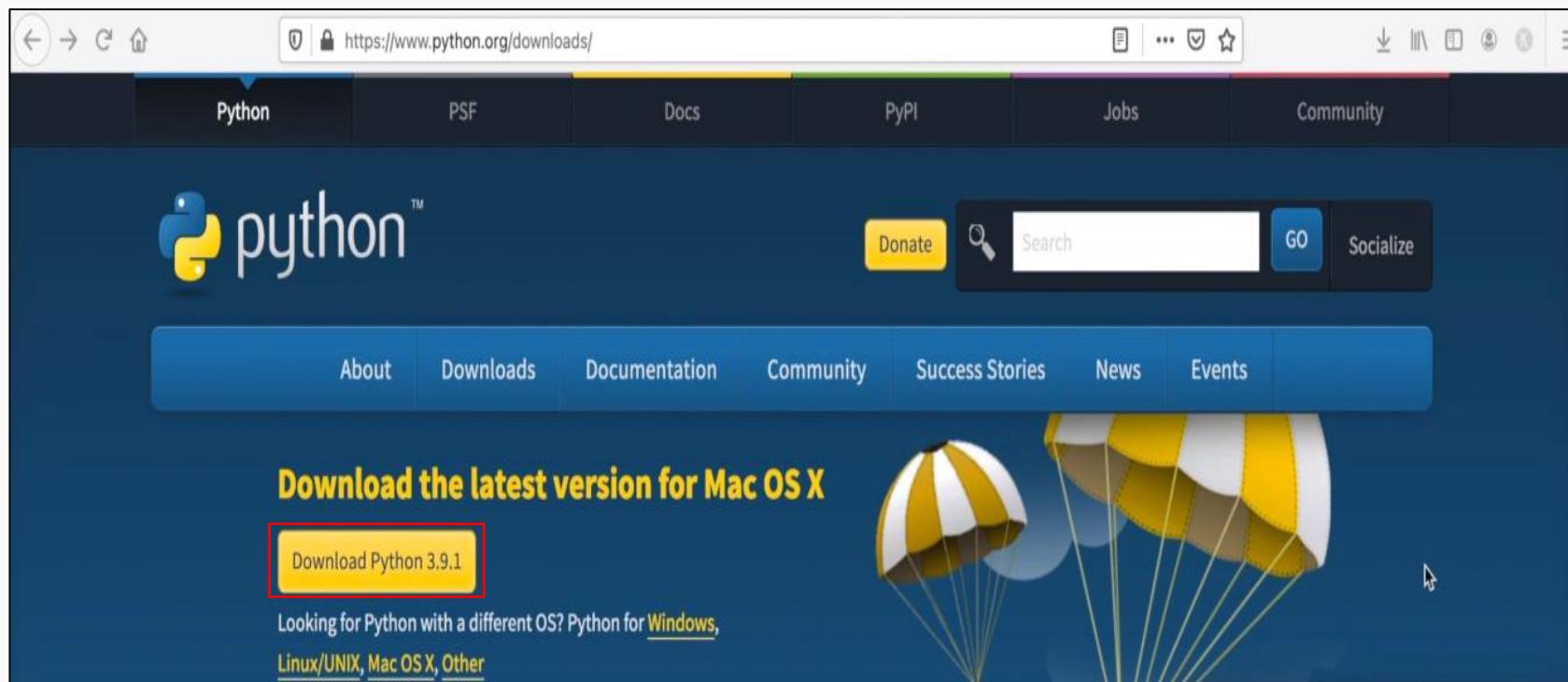
- Click on Downloads.





Setting Up Python – Mac (3/13)

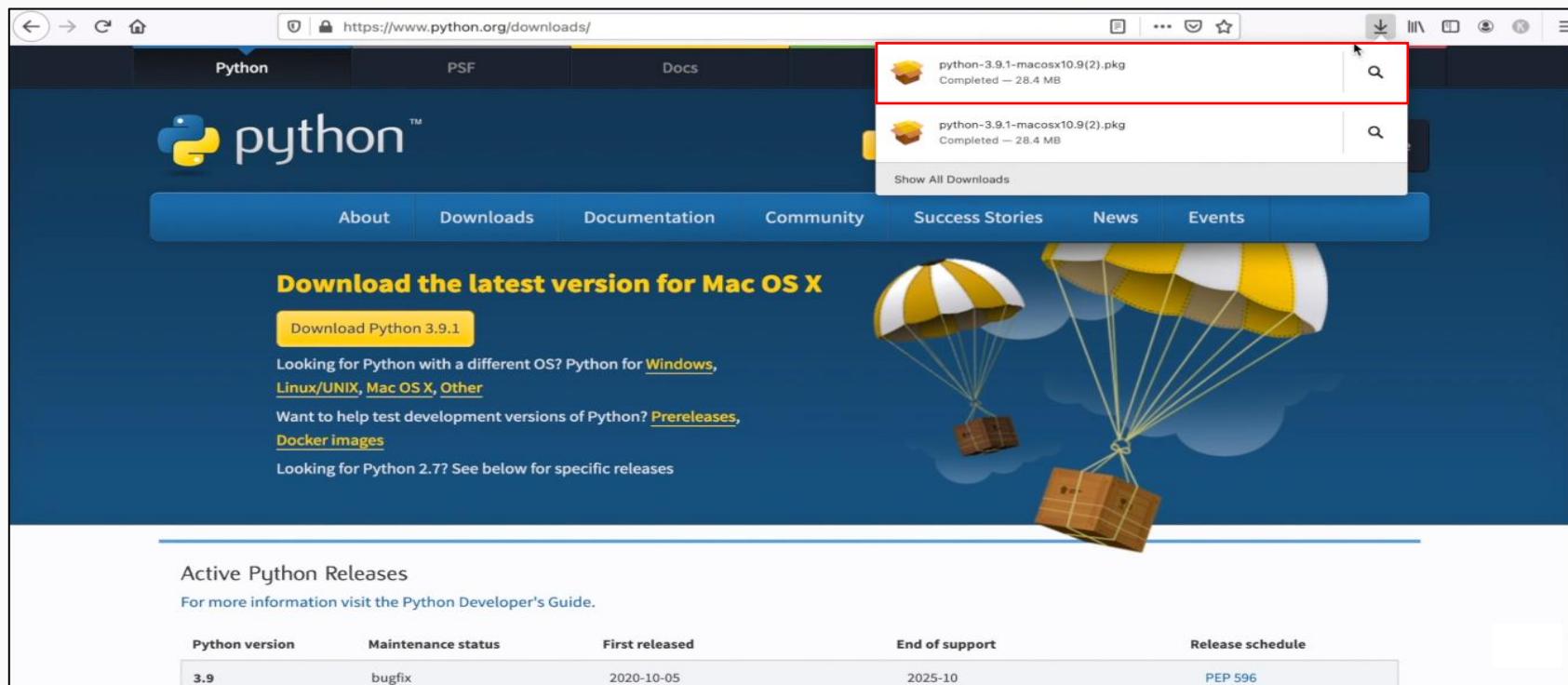
- This will show you the latest version of Python3 for your operating system. Download it.





Setting Up Python – Mac (4/13)

- Once downloaded, click on the file.





Setting Up Python – Mac (5/13)

- Click continue.





Setting Up Python – Mac (6/13)

- Click continue.





Setting Up Python – Mac (7/13)

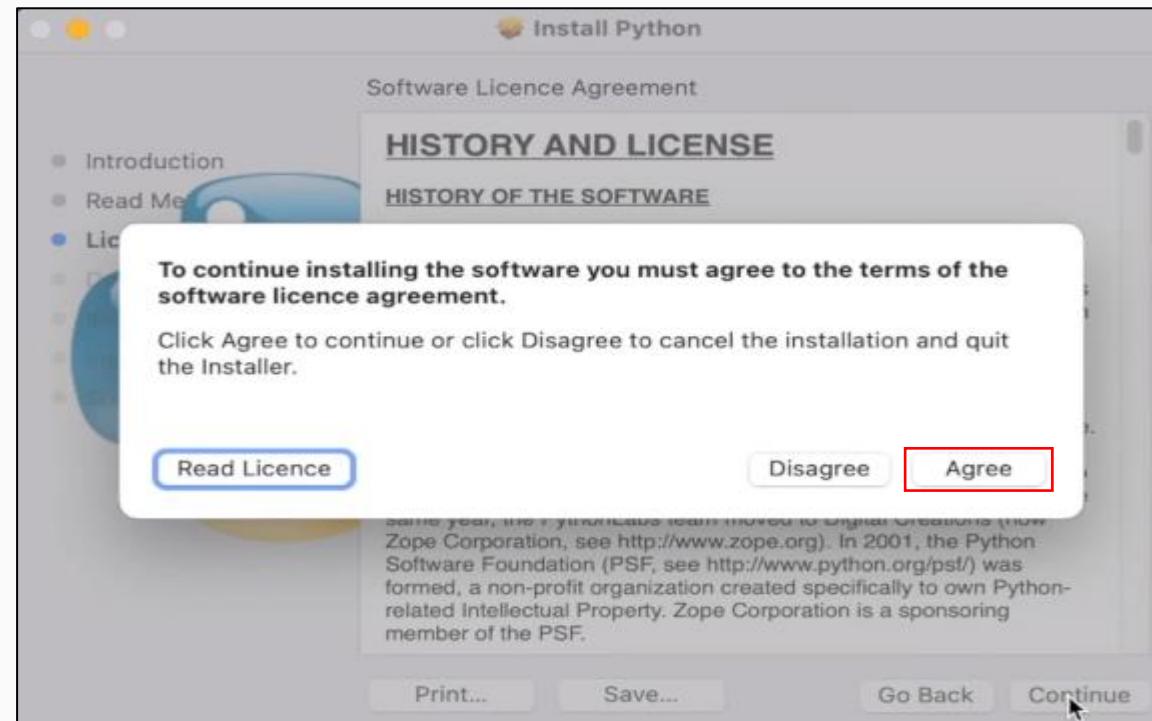
- Click continue.





Setting Up Python – Mac (8/13)

- Read and accept the license agreement.





Setting Up Python – Mac (9/13)

- Click continue.





Setting Up Python – Mac (10/13)

- Click install. If prompted, enter your password.





Setting Up Python – Mac (11/13)

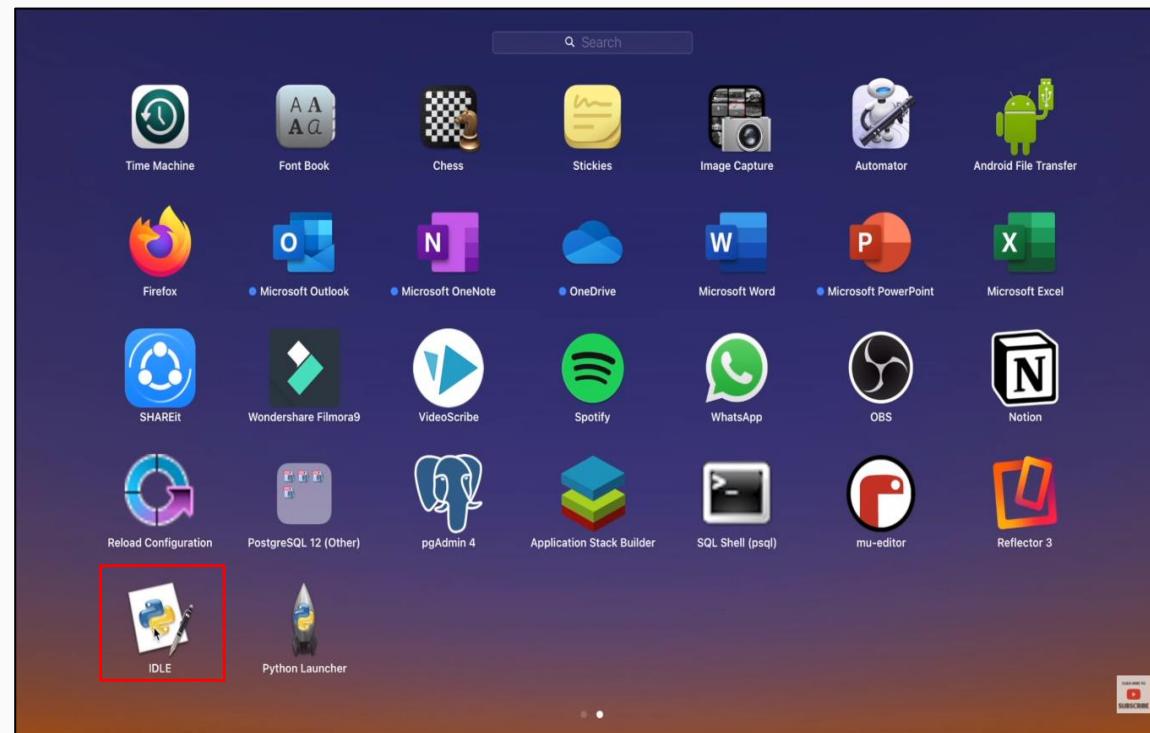
- Wait for the installation to finish.





Setting Up Python – Mac (12/13)

- Once installed, search for IDLE in your spotlight and click on it.





Setting Up Python – Mac (13/13)

- This will open up Python IDLE. Type print("Hello World") and press Enter to execute.

The screenshot shows the Python IDLE Shell 3.9.1 window. The title bar reads "IDLE Shell 3.9.1". The main area displays the following text:

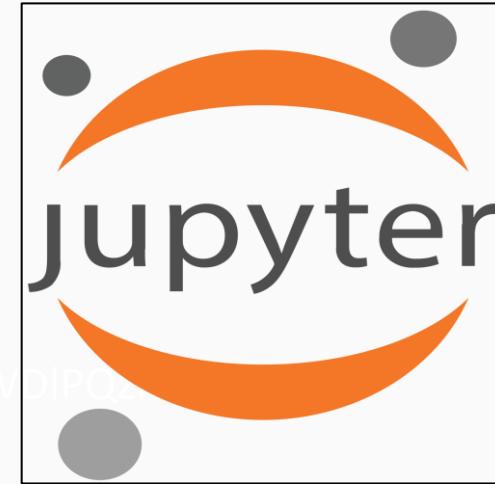
```
Python 3.9.1 (v3.9.1:1e5d33e9b9, Dec  7 2020, 12:10:52)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
WARNING: The system preference "Prefer tabs when opening documents" is set to
"Always". This will cause various problems with IDLE. For the best experience,
change this setting when running IDLE (via System Preferences -> Dock).
>>> print('Hello World')
Hello World
>>>
>>>
>>> |
```

A cursor arrow is visible in the bottom-left area of the window. At the bottom right, there is a status bar with the text "Ln: 11 Col: 4".



Jupyter Notebook

- Jupyter Notebook is a web application that allows you to create documents that may contain code, visualizations, and narrative text.
- It is very popular and is used extensively in data science, artificial intelligence, and machine learning.
<https://uniho.zoom.us/j/83422537997?pwd=Z3VENmI0QTV2Q2xIWDP0ZUJ3dGc9>
- A notebook has the .ipynb extension



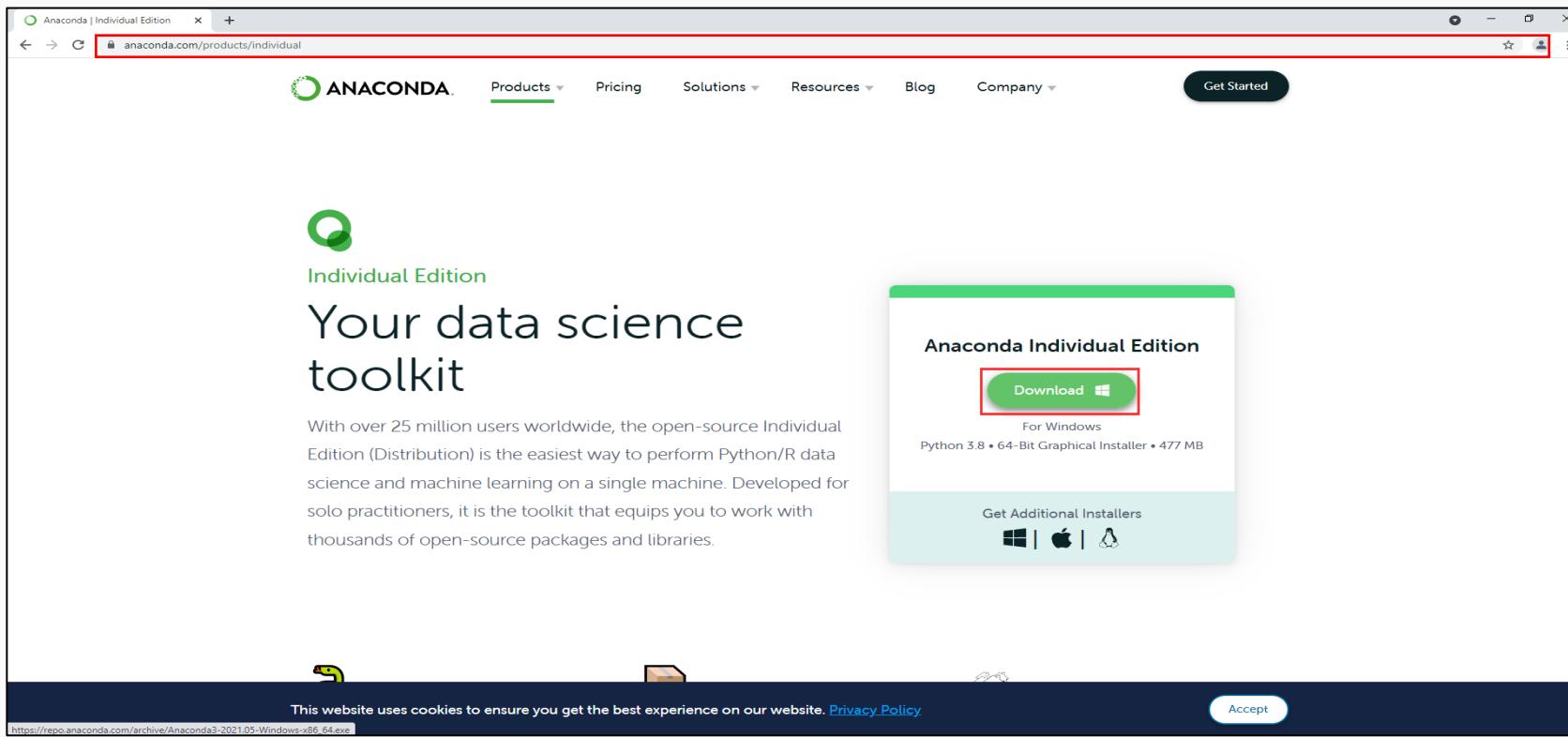
Alternatives to Jupyter Notebook

- Google Colaboratory is an alternative to Jupyter Notebook and does not require downloading anything.



Anaconda for Windows (1/13)

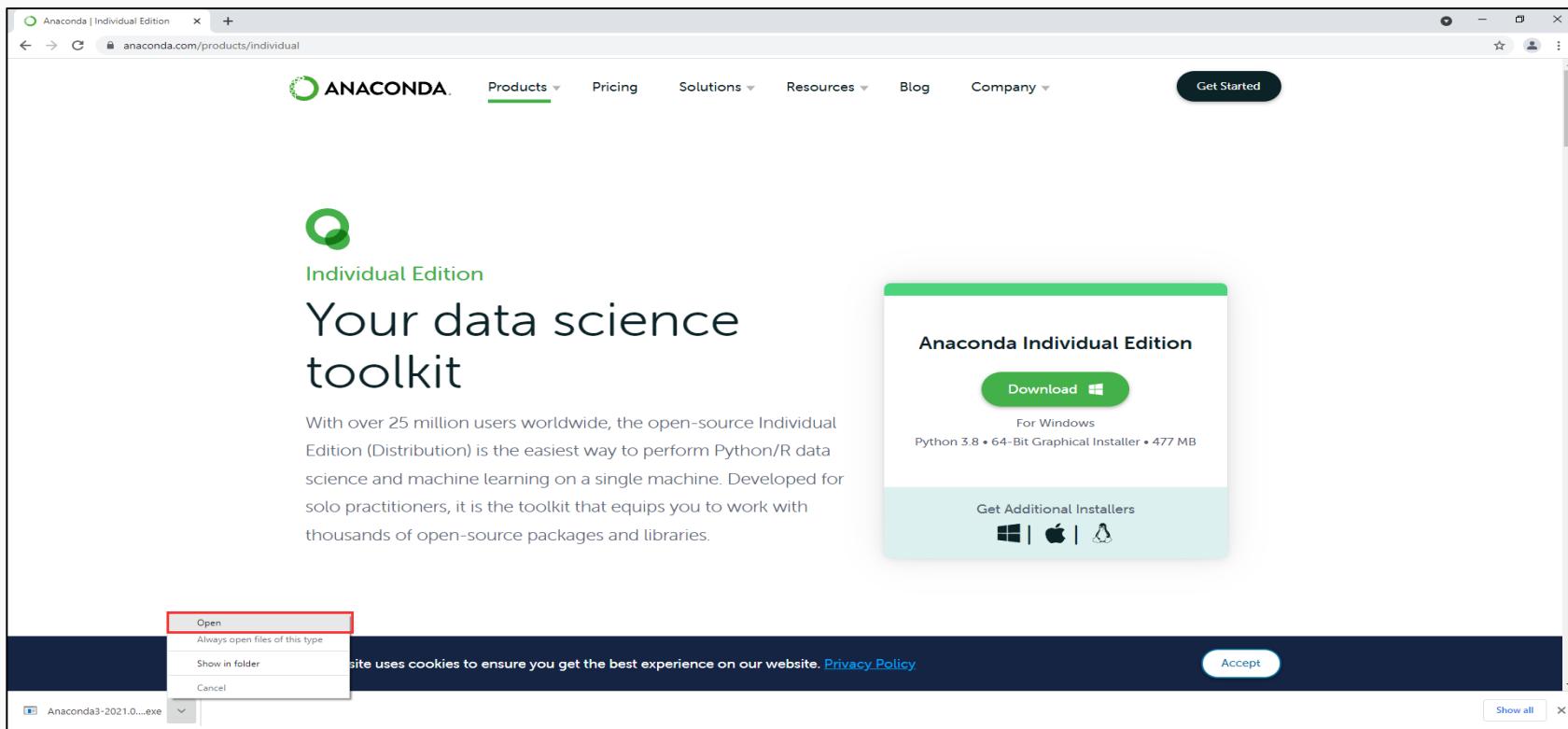
- Visit: <https://www.anaconda.com/products/individual>
- Click on the Download button.





Anaconda for Windows (2/13)

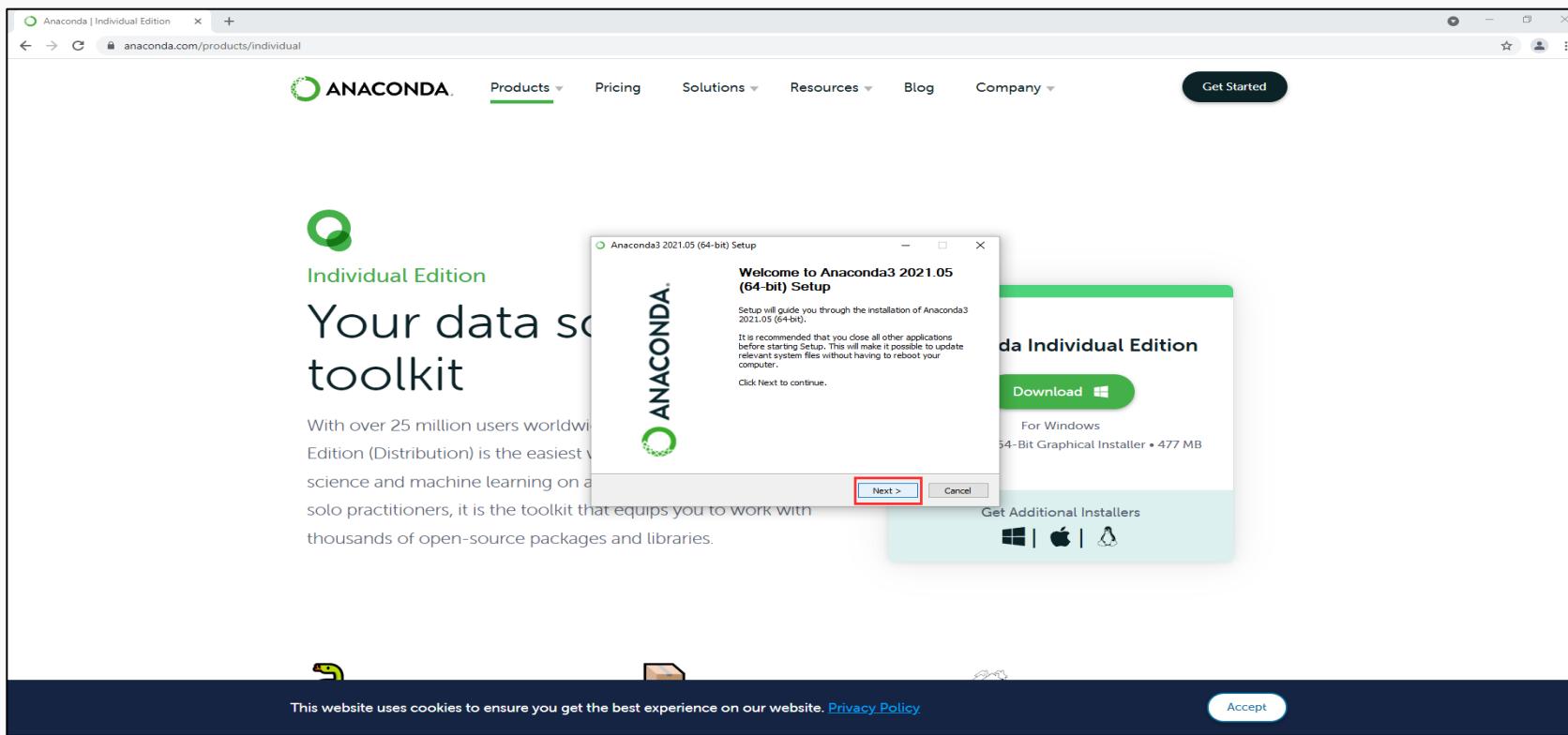
- Once Downloaded, click on Open.





Anaconda for Windows (3/13)

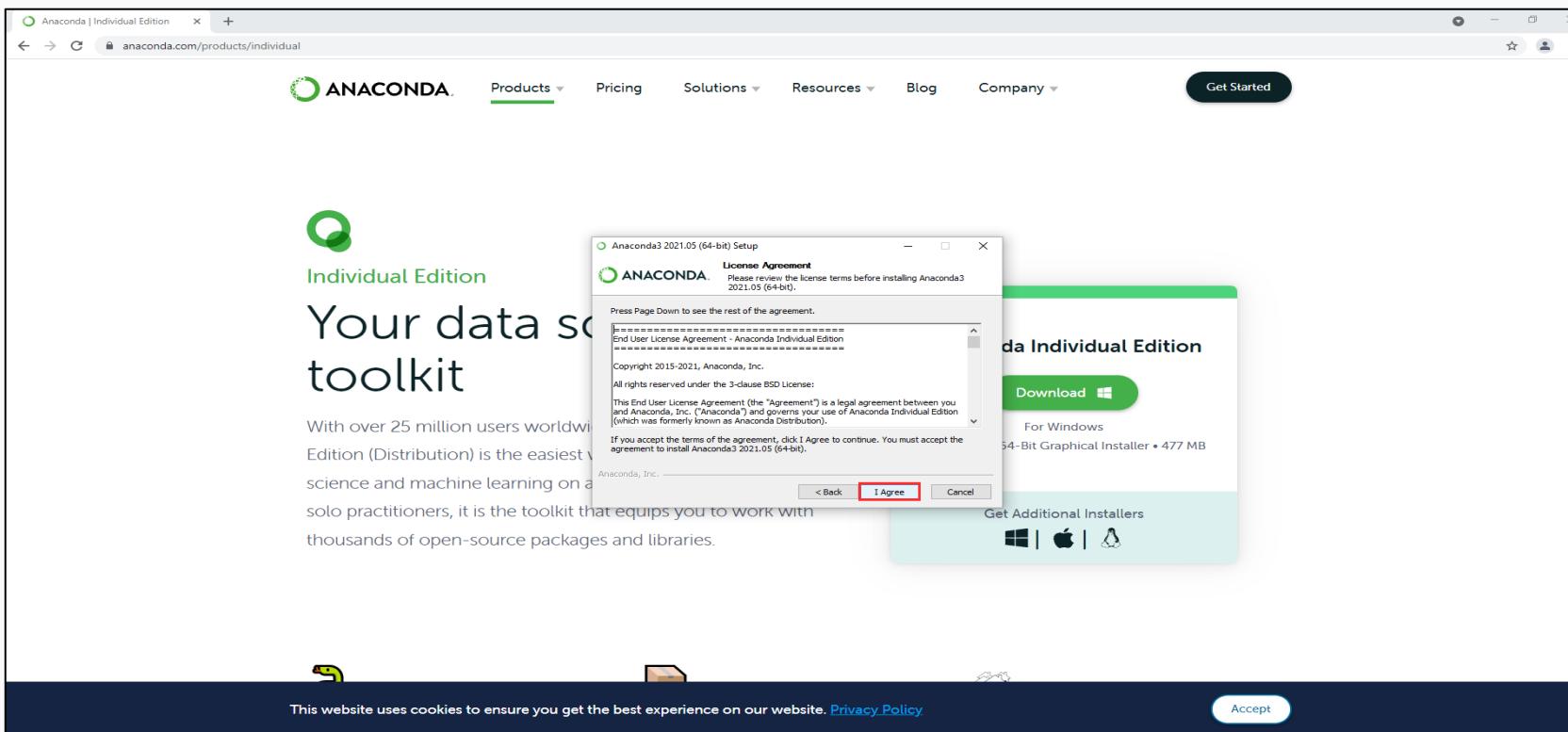
- An installation Wizard will open up.
- Click on Next.





Anaconda for Windows (4/13)

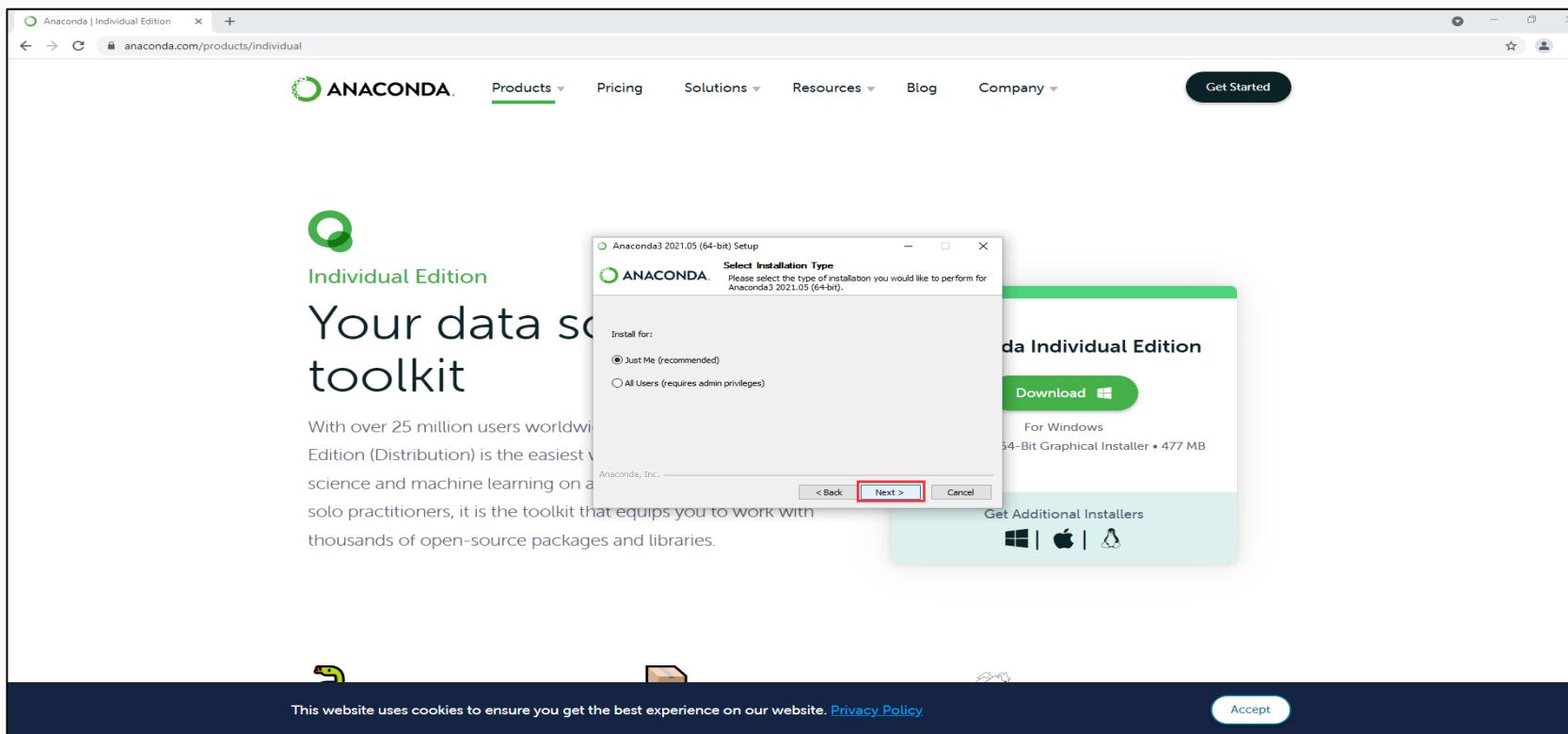
- Read the License Agreement and click I Agree.





Anaconda for Windows (5/13)

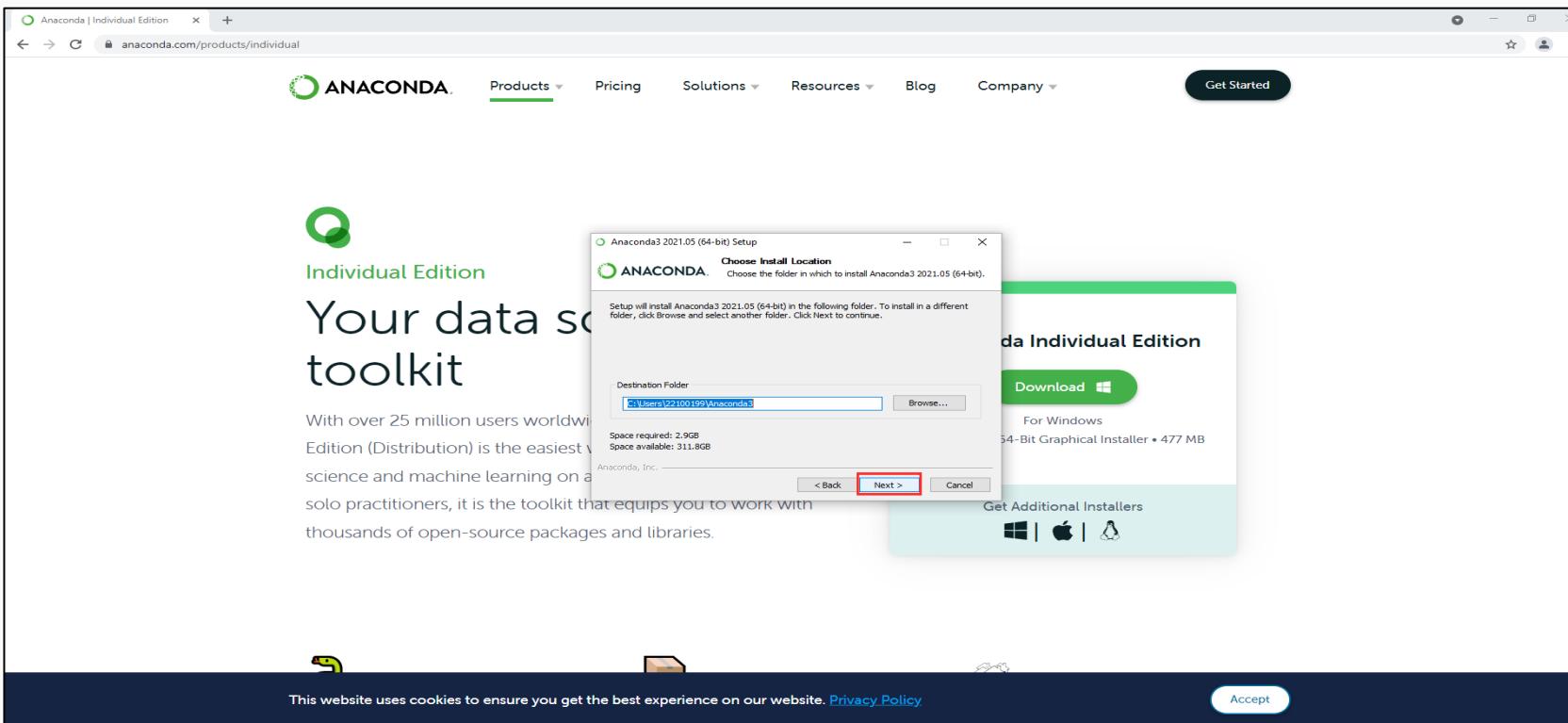
- Click Next.





Anaconda for Windows (6/13)

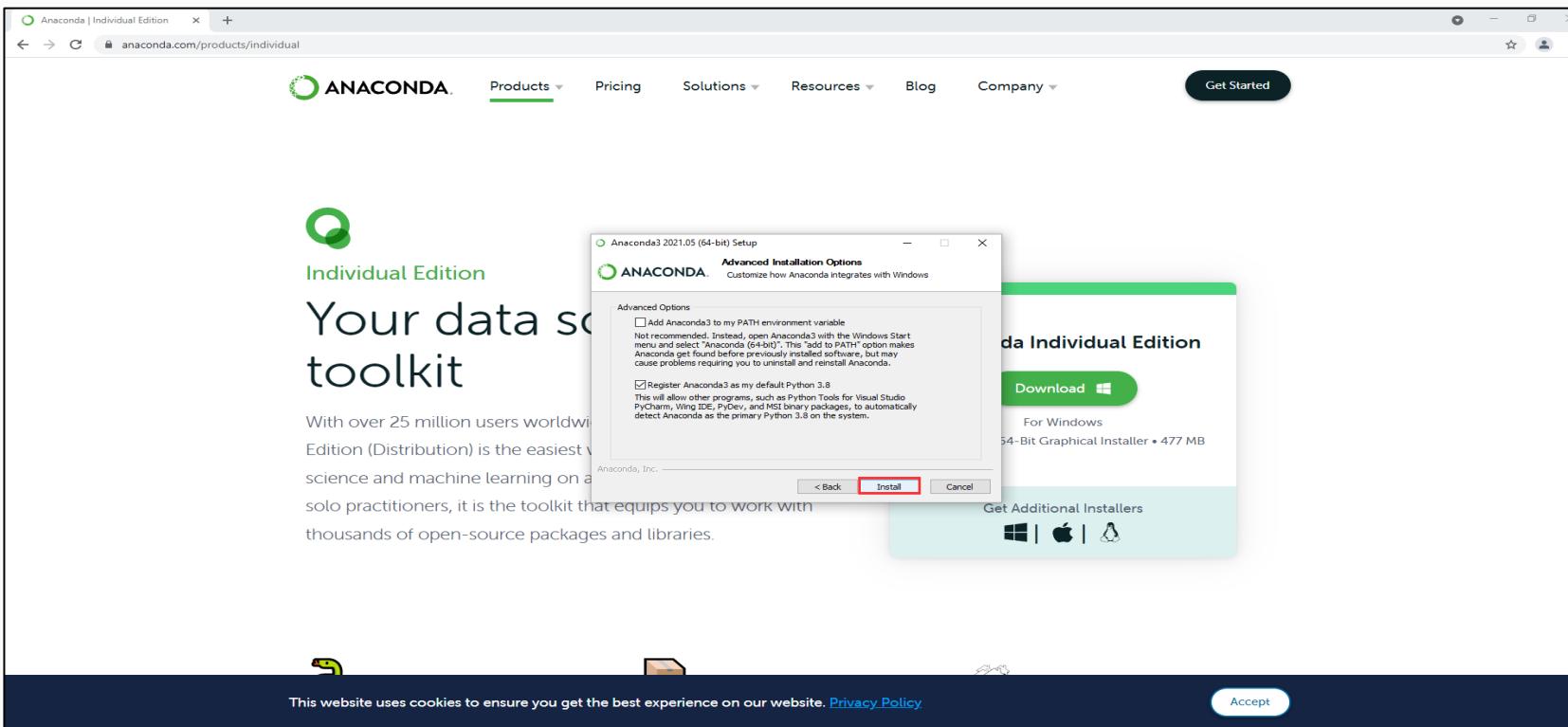
- Choose the destination and click Next.





Anaconda for Windows (7/13)

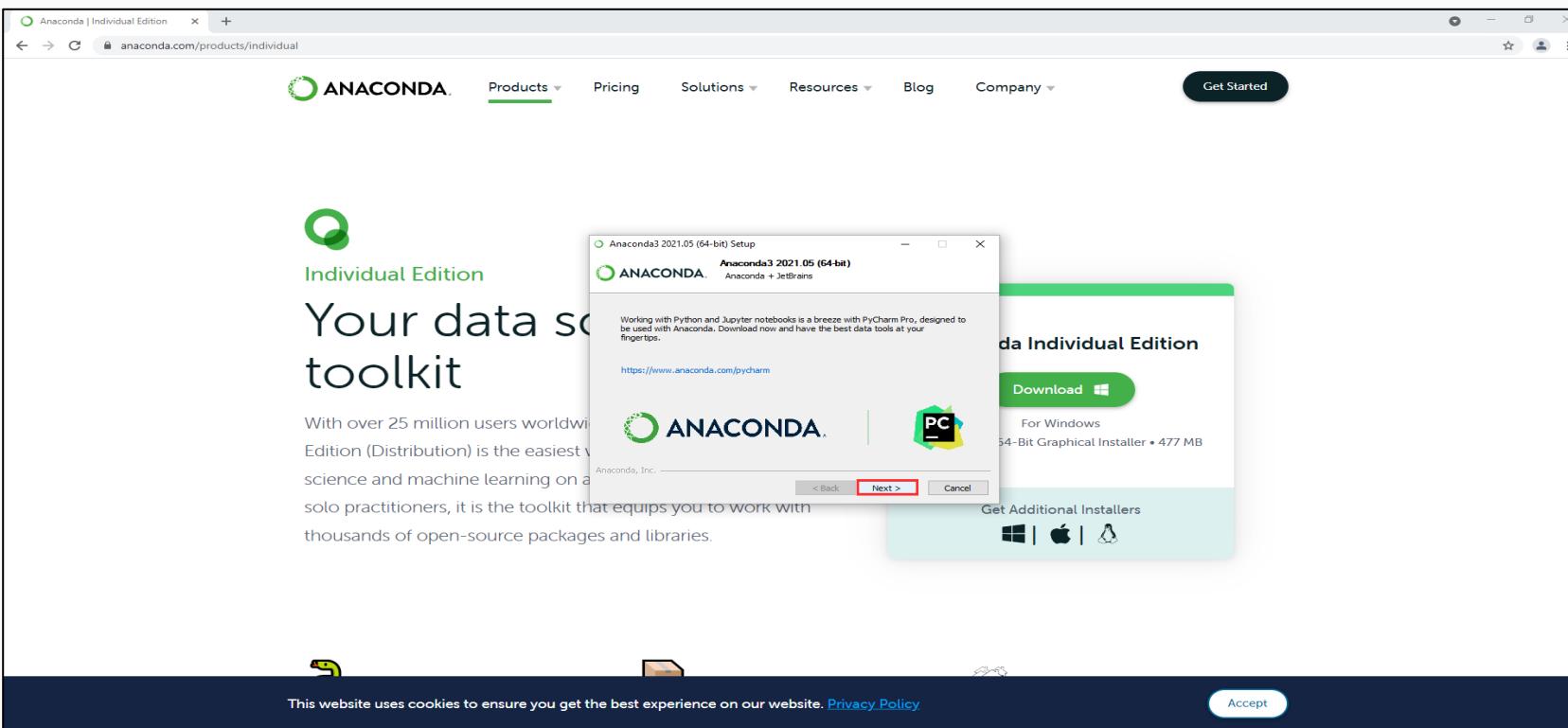
- Click Install.





Anaconda for Windows (8/13)

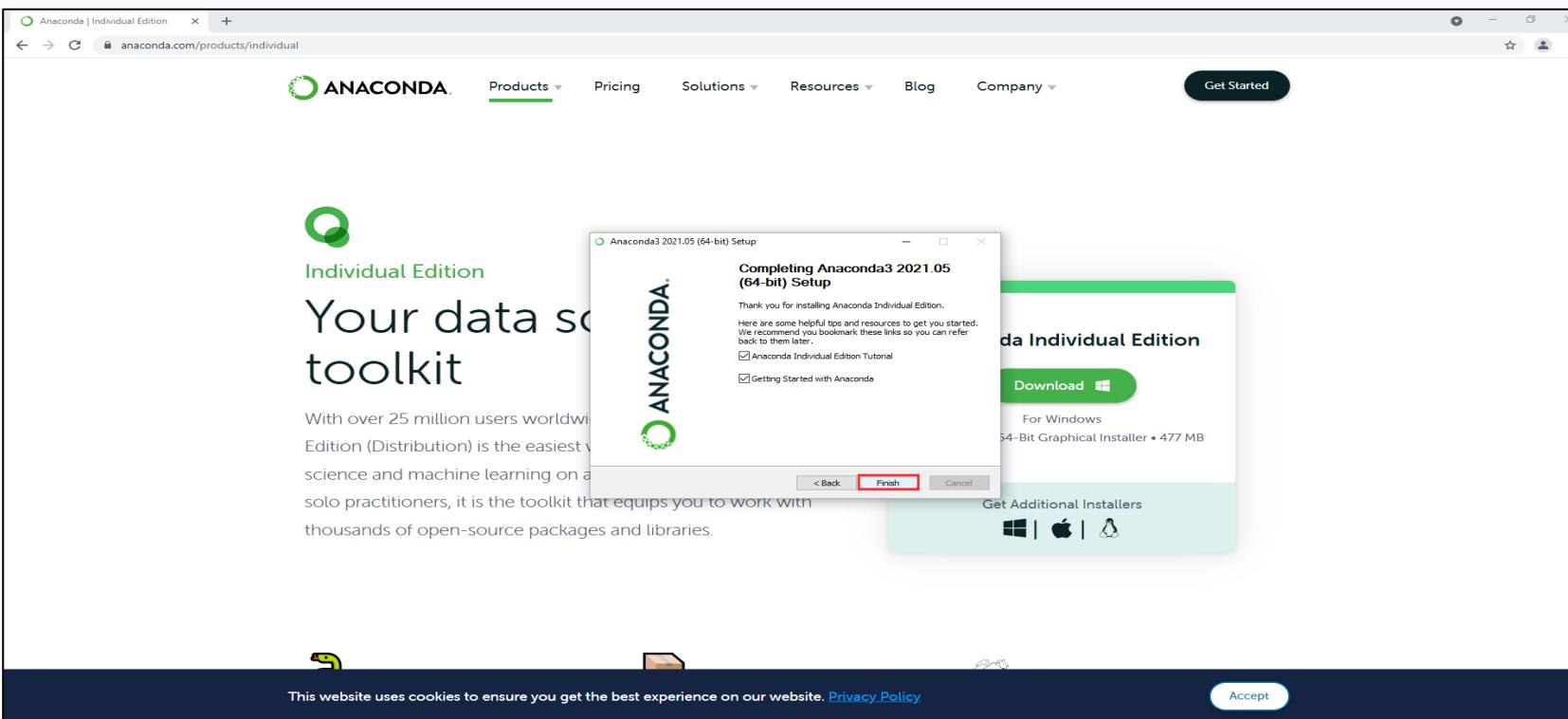
- Wait for the installation to finish and click Next.





Anaconda for Windows (9/13)

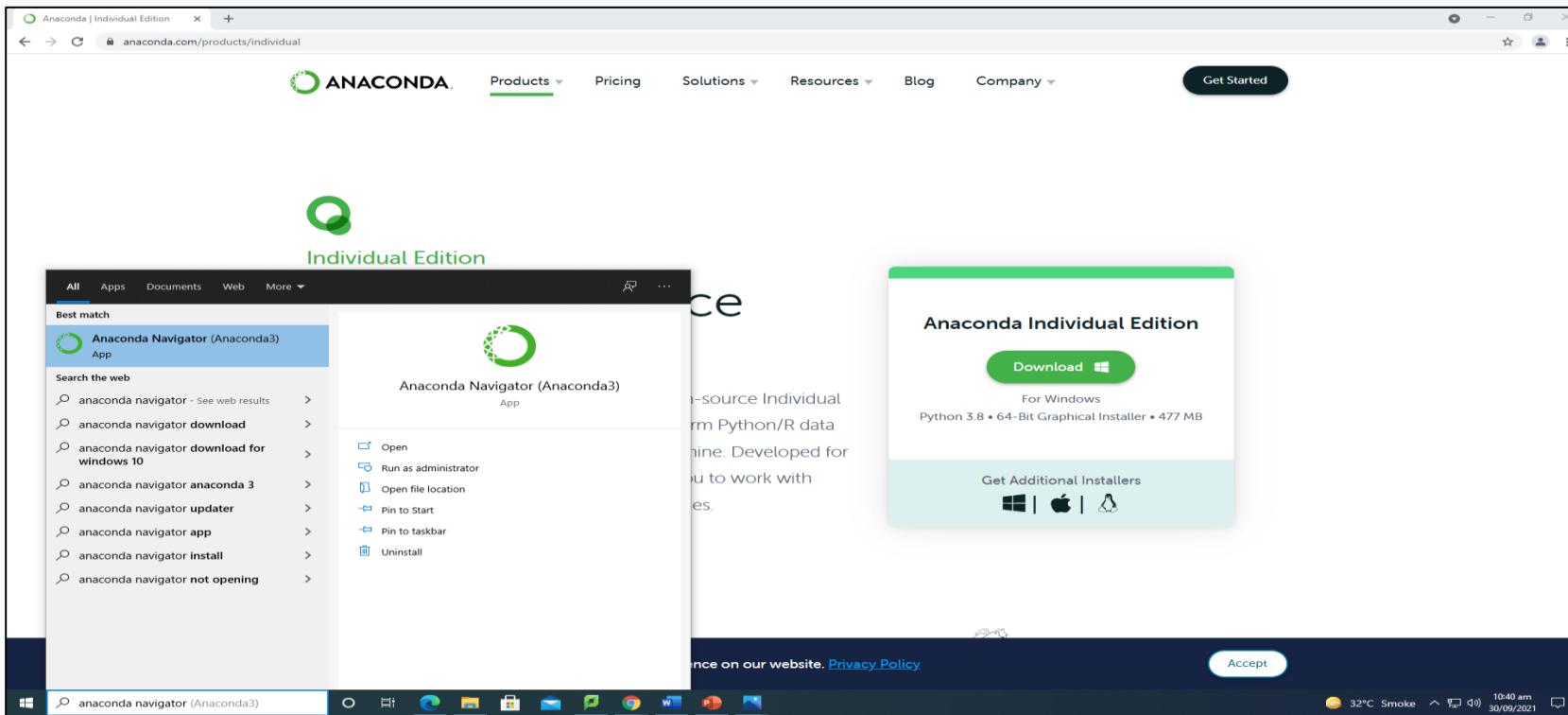
- Click Finish.





Anaconda for Windows (10/13)

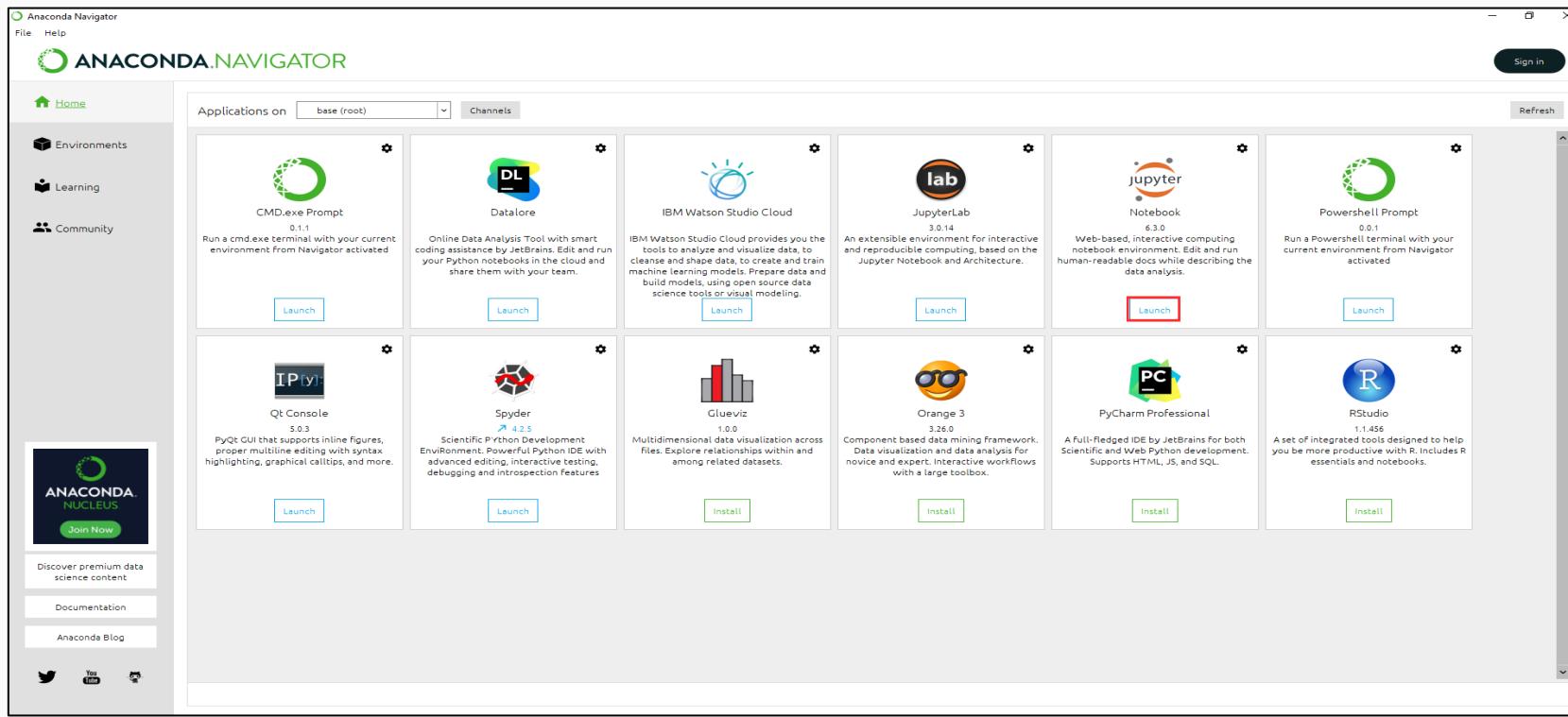
- To launch Jupyter Notebook, search for anaconda navigator and open it.





Anaconda for Windows (11/13)

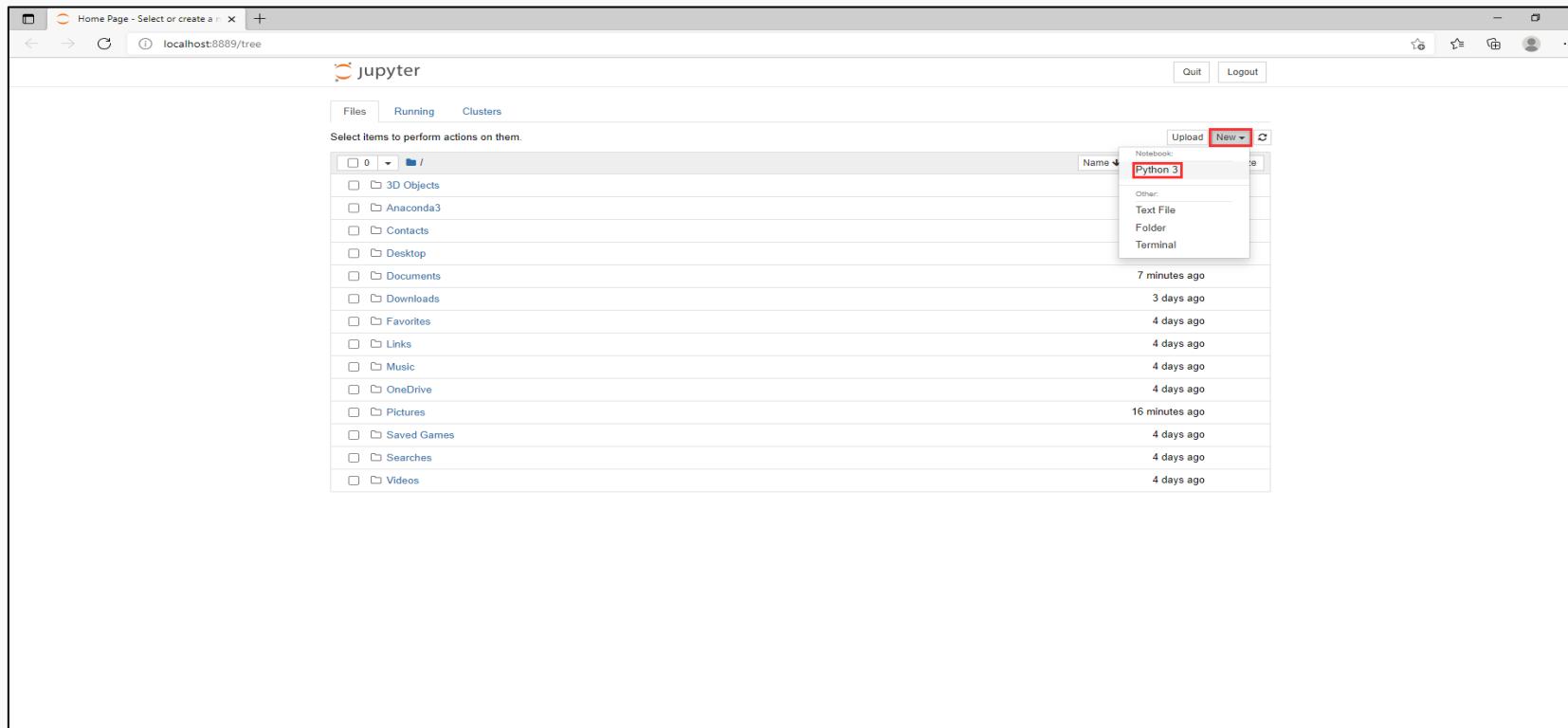
- Following screen will open up.
- Launch Jupyter Notebook from here.





Anaconda for Windows (12/13)

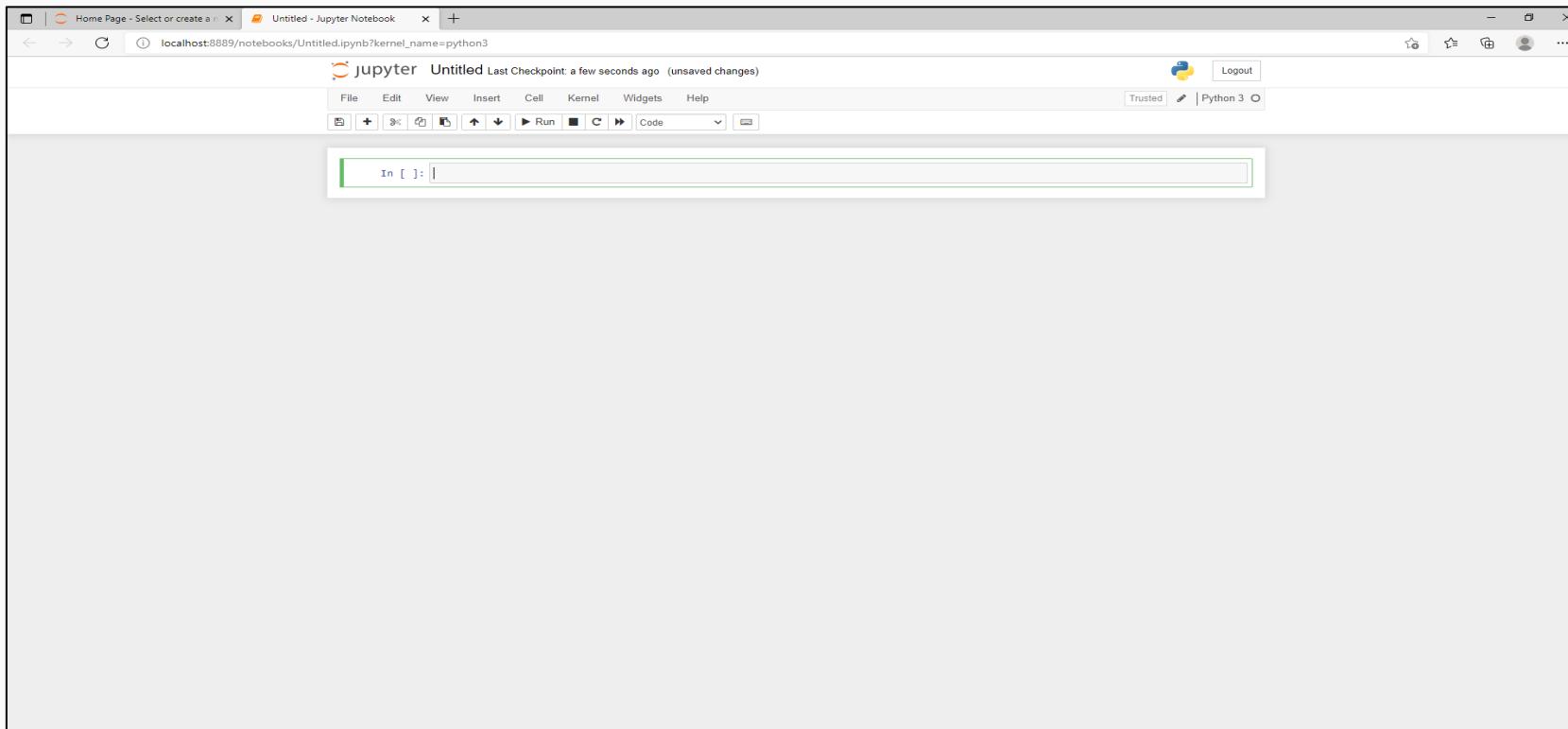
- This will launch Jupyter Notebook in your default browser.
- Click on New -> Python3 to create a new Notebook.





Anaconda for Windows (13/13)

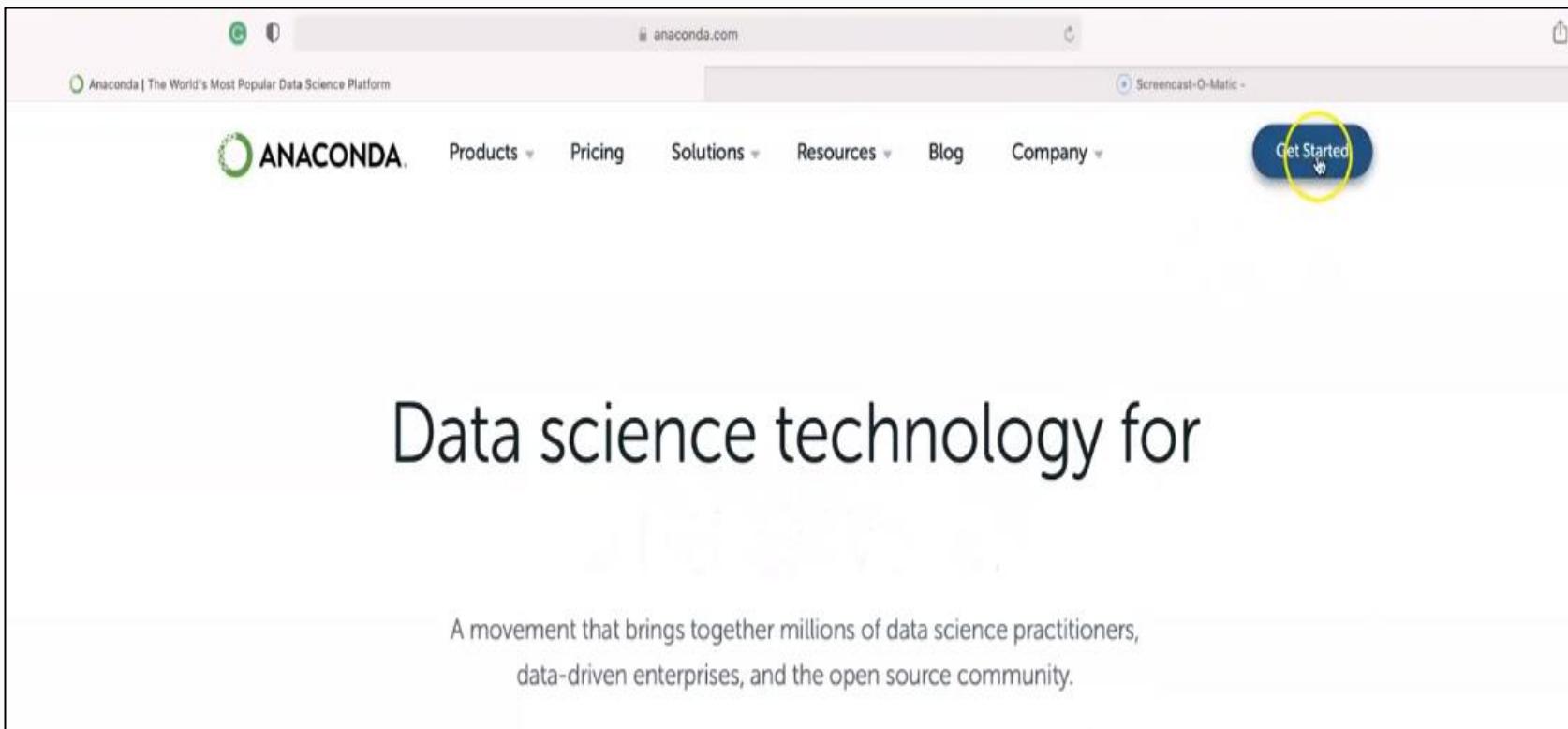
- This is how a Notebook looks like.





Anaconda for Mac (1/14)

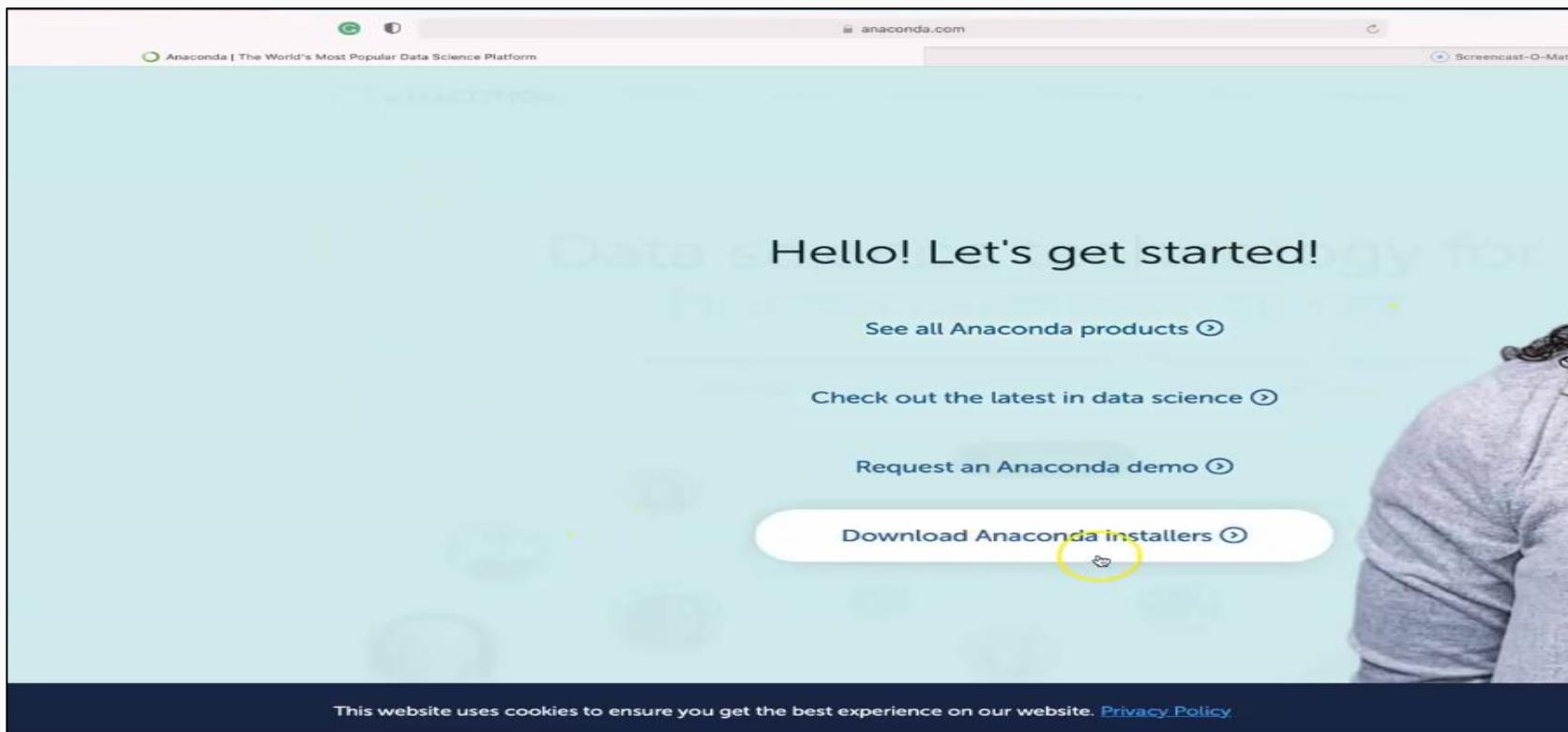
- Visit: <https://www.anaconda.com/>
- Click on Get Started.





Anaconda for Mac (2/14)

- Click Download Anaconda Installers.





Anaconda for Mac (3/14)

- Download the graphical installer for Mac.

The screenshot shows the 'Anaconda Installers' page. It has three main sections: 'Windows' (with icons for Python 3.8 64-bit Graphical and 32-bit Graphical installers), 'MacOS' (with icons for Python 3.8 64-bit Graphical, 64-bit Command Line, and 32-bit Command Line installers, where the 64-bit Graphical one is highlighted with a yellow circle), and 'Linux' (with icons for Python 3.8 64-bit (x86), 64-bit (Power8 and Power9), 64-bit (AWS Graviton2 / ARM64), and 64-bit (Linux on IBM Z & LinuxONE) installers). Below these sections is a 'ADDITIONAL INSTALLERS' box containing text about older versions of the Individual Edition installers and a link to the Miniconda installer homepage. At the bottom, there's a dark footer bar with a cookie consent message, an 'Accept' button, and a 'SUBSCRIBE' button.

Windows

MacOS

Linux

ADDITIONAL INSTALLERS

The archive has older versions of Anaconda Individual Edition installers. The [Miniconda installer homepage](#) can be found here.

This website uses cookies to ensure you get the best experience on our website. [Privacy Policy](#)

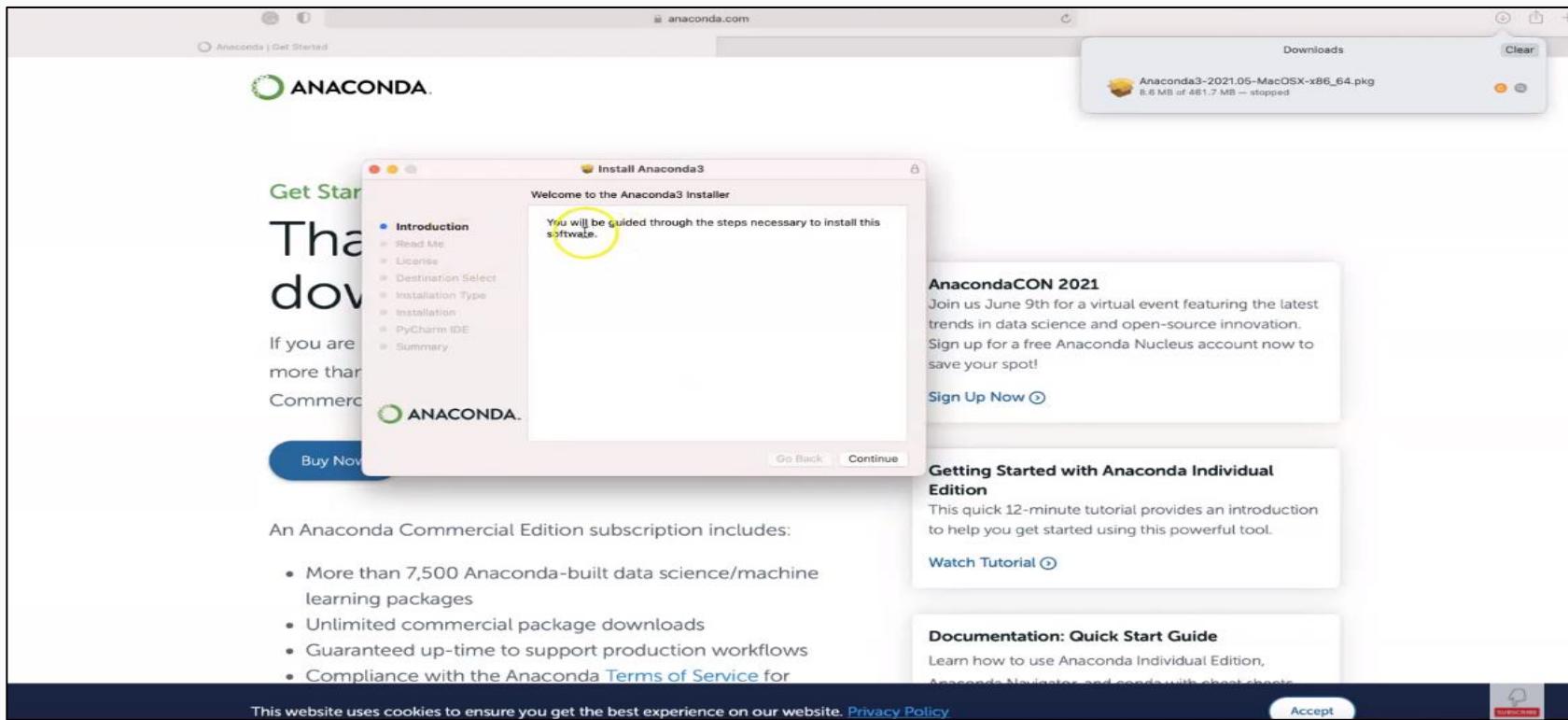
Accept

SUBSCRIBE



Anaconda for Mac (4/14)

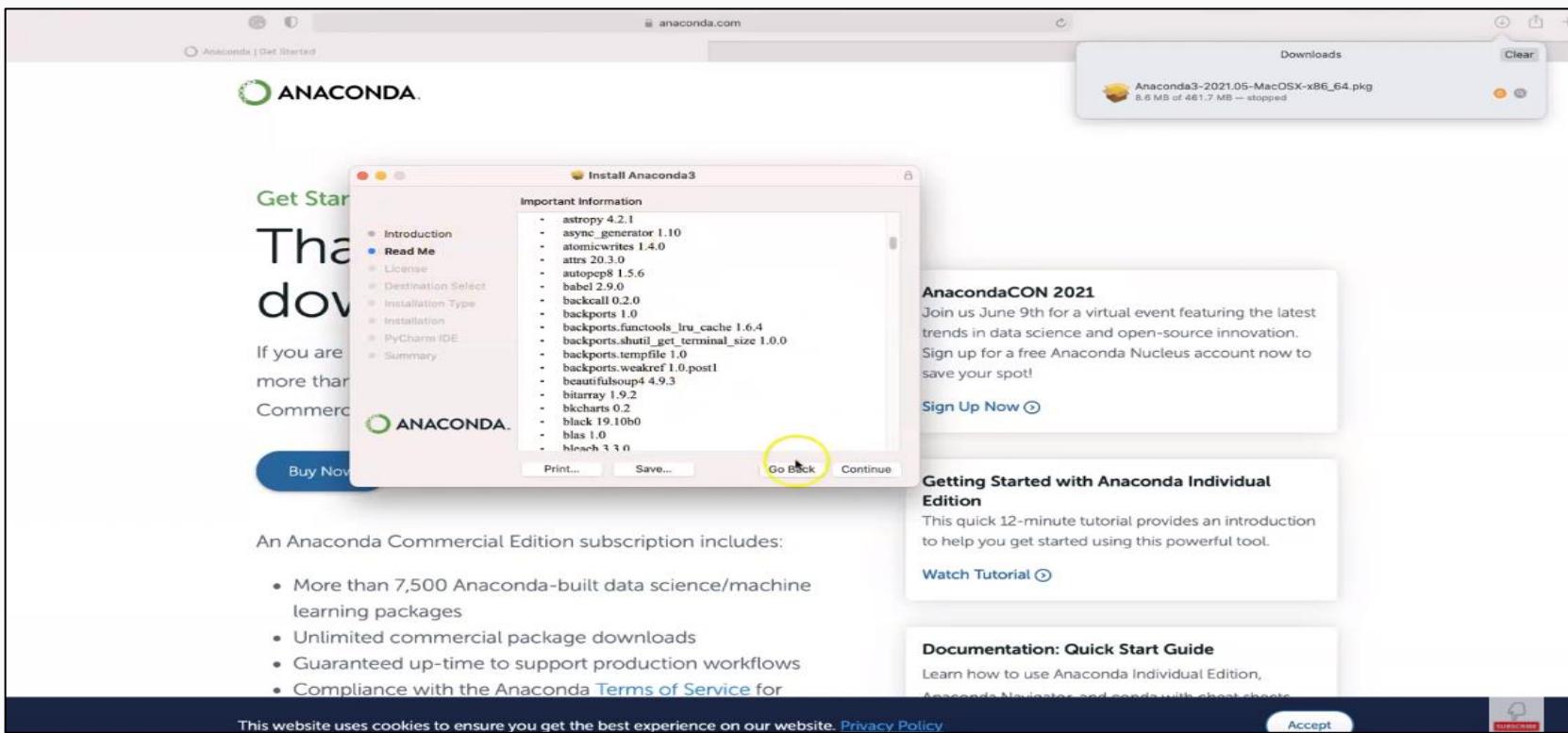
- Once download, launch the installation wizard by double clicking on the file.
- Click Continue.





Anaconda for Mac (5/14)

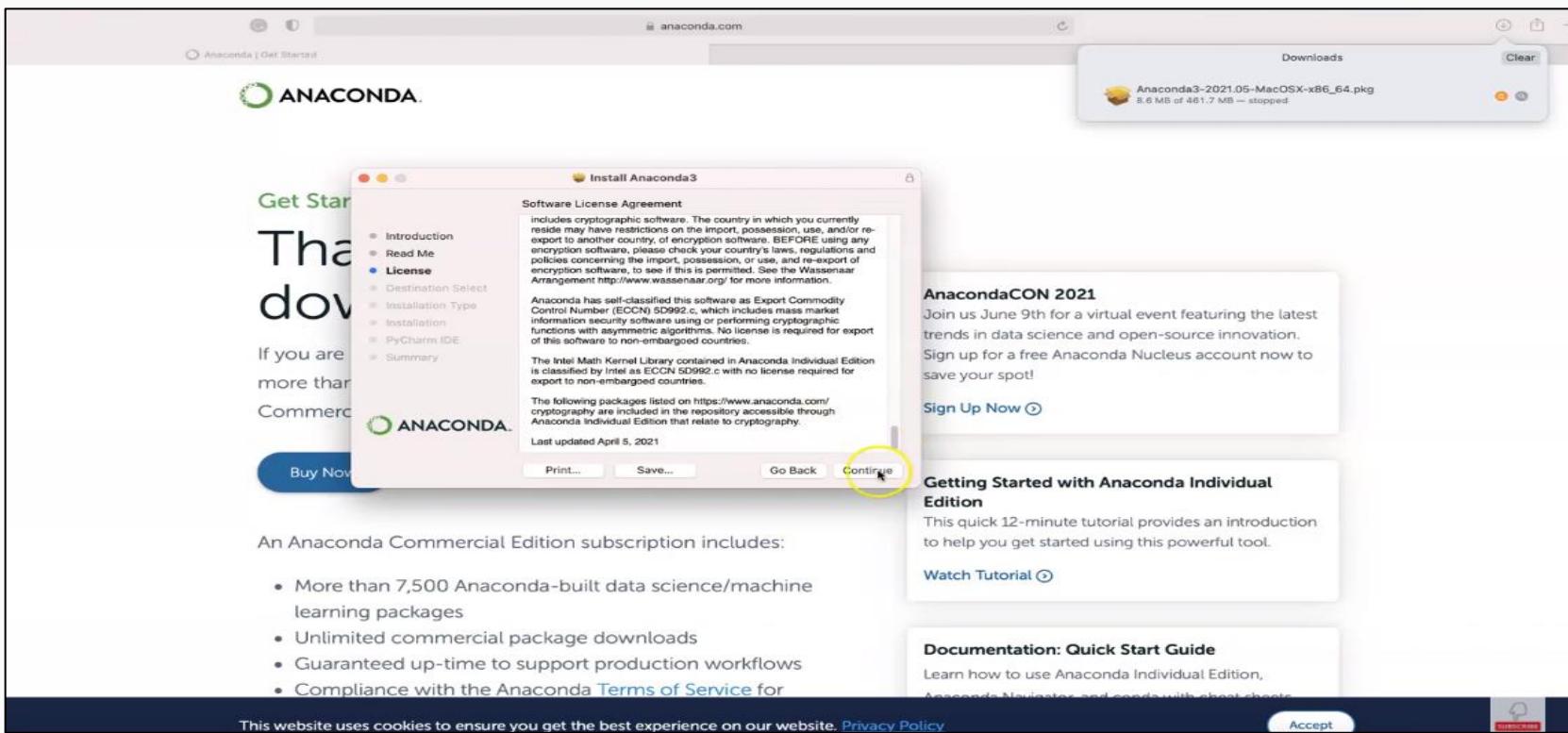
- Read the Read Me file and click continue.





Anaconda for Mac (6/14)

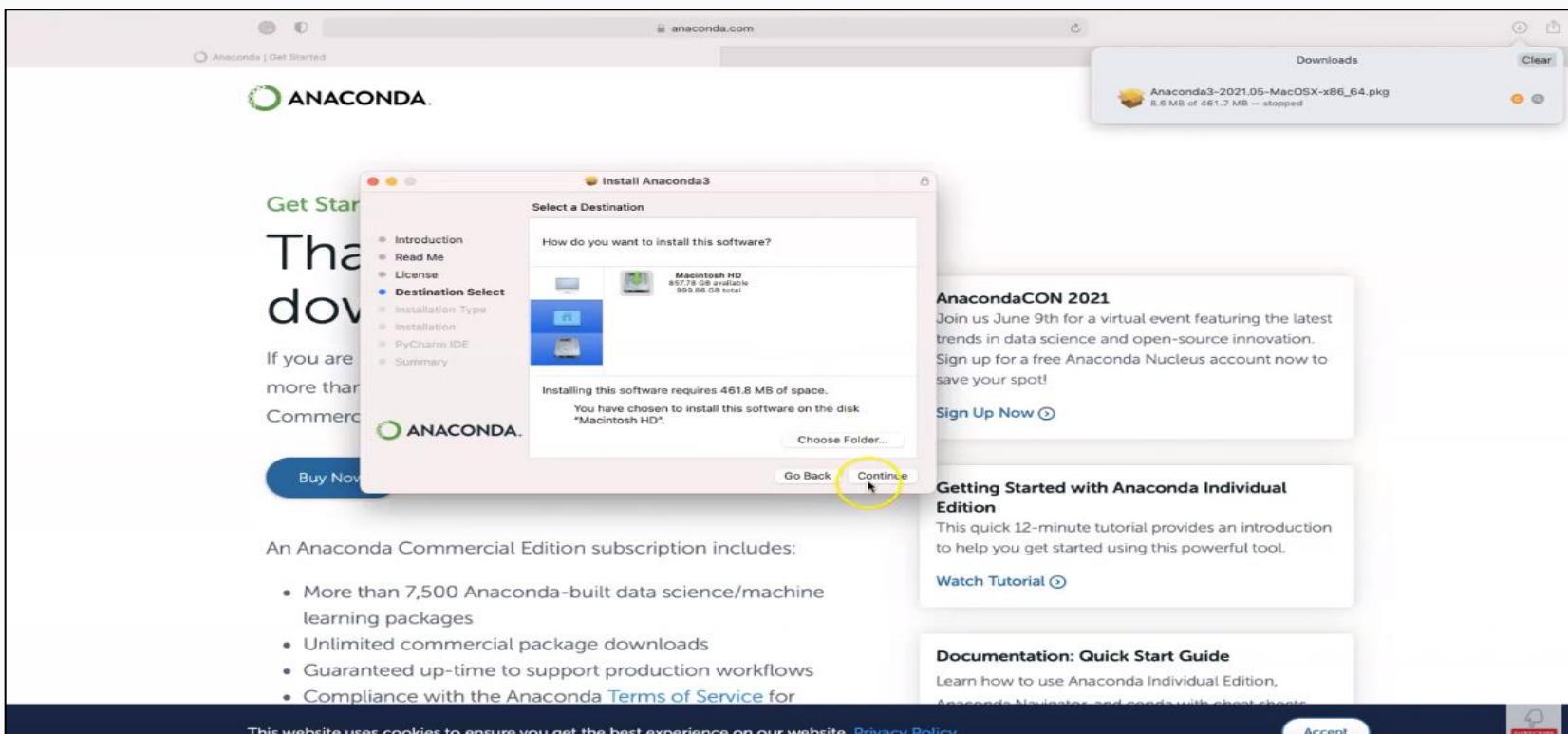
- Read the License Agreement and click continue.





Anaconda for Mac (7/14)

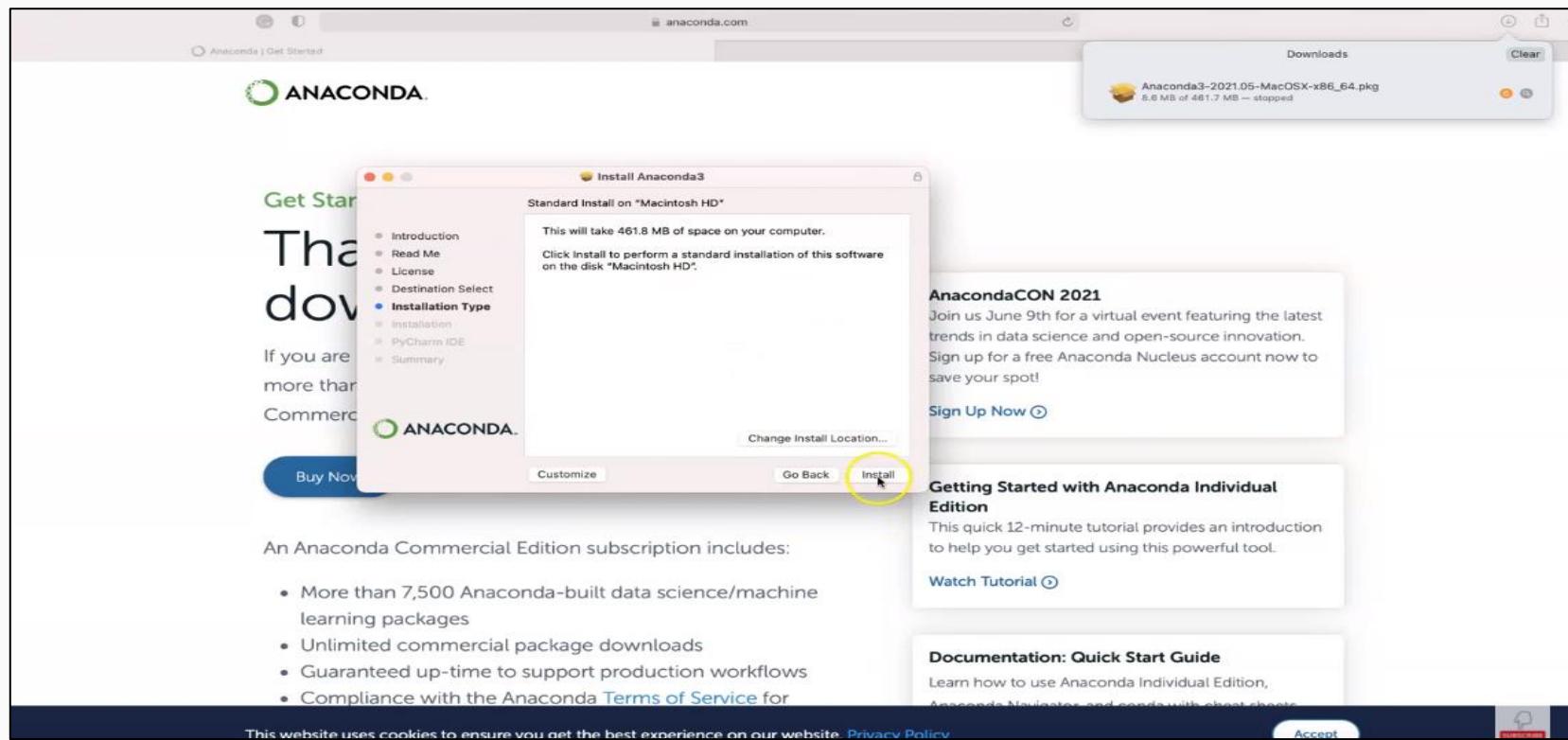
- Select the destination and click Continue.





Anaconda for Mac (8/14)

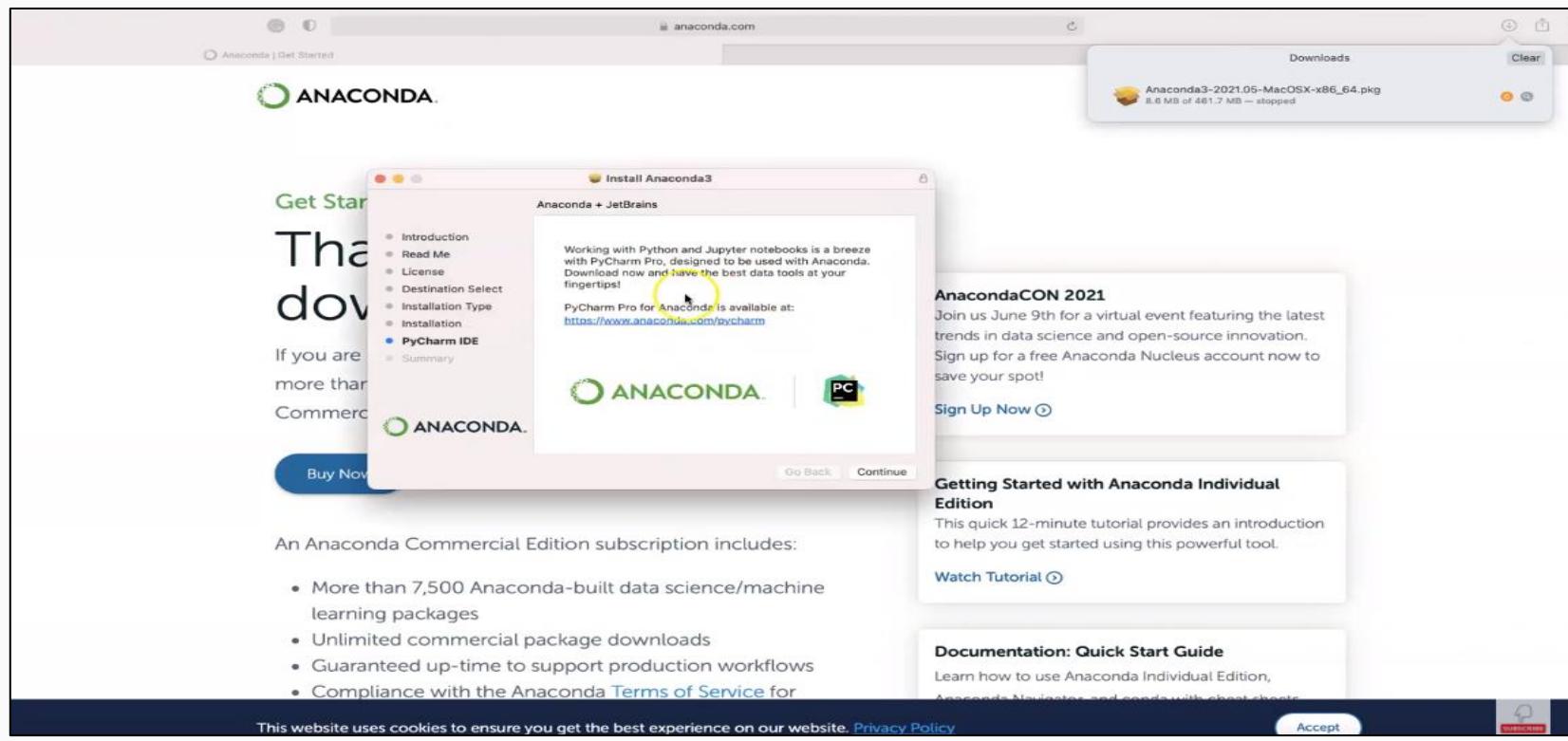
- Click Install.





Anaconda for Mac (9/14)

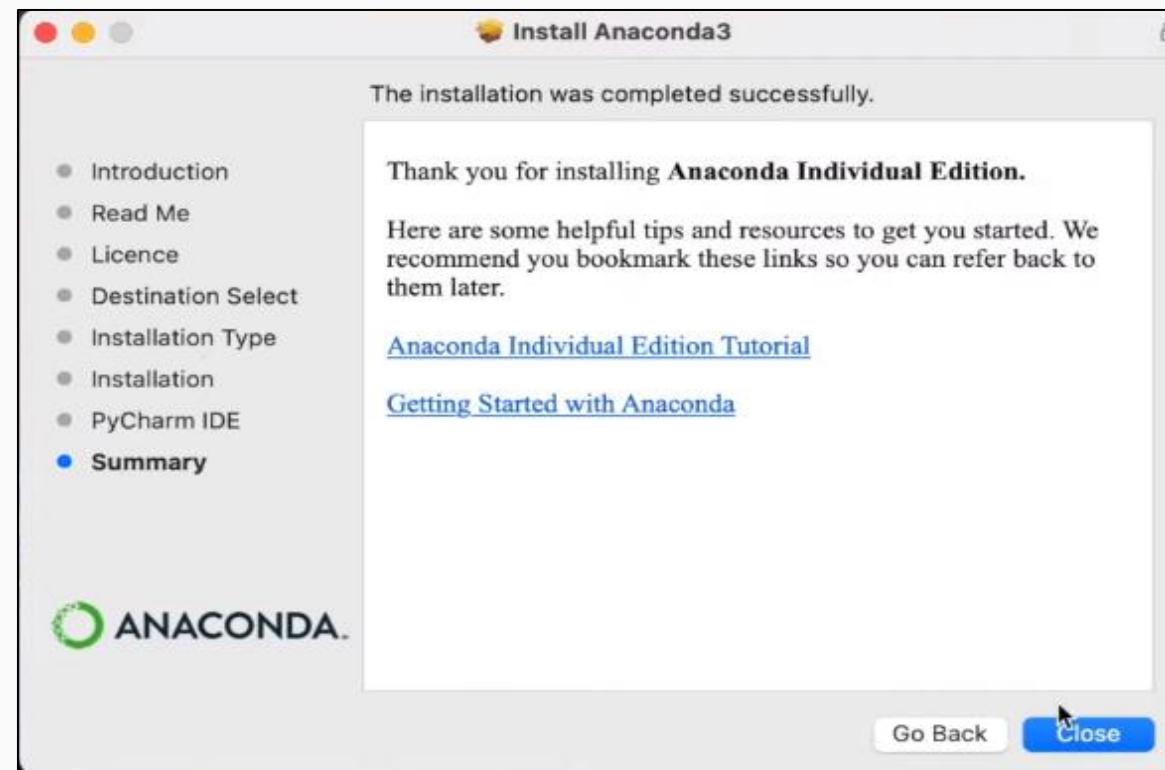
- Click Continue.





Anaconda for Mac (10/14)

- Once the installation finishes, click Close.





Anaconda for Mac (11/14)

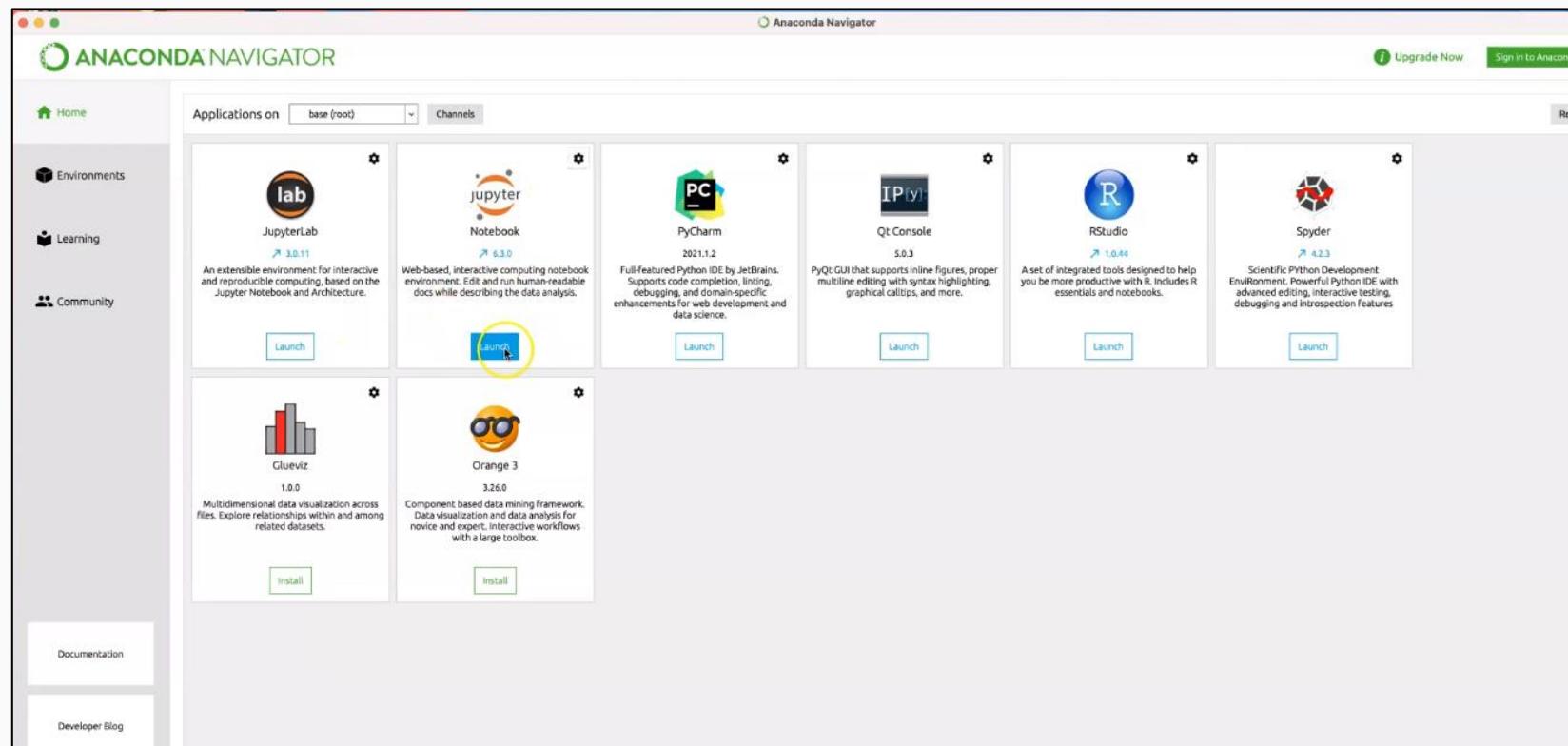
- Open your spotlight and click on Anaconda Navigator to launch it.





Anaconda for Mac (12/14)

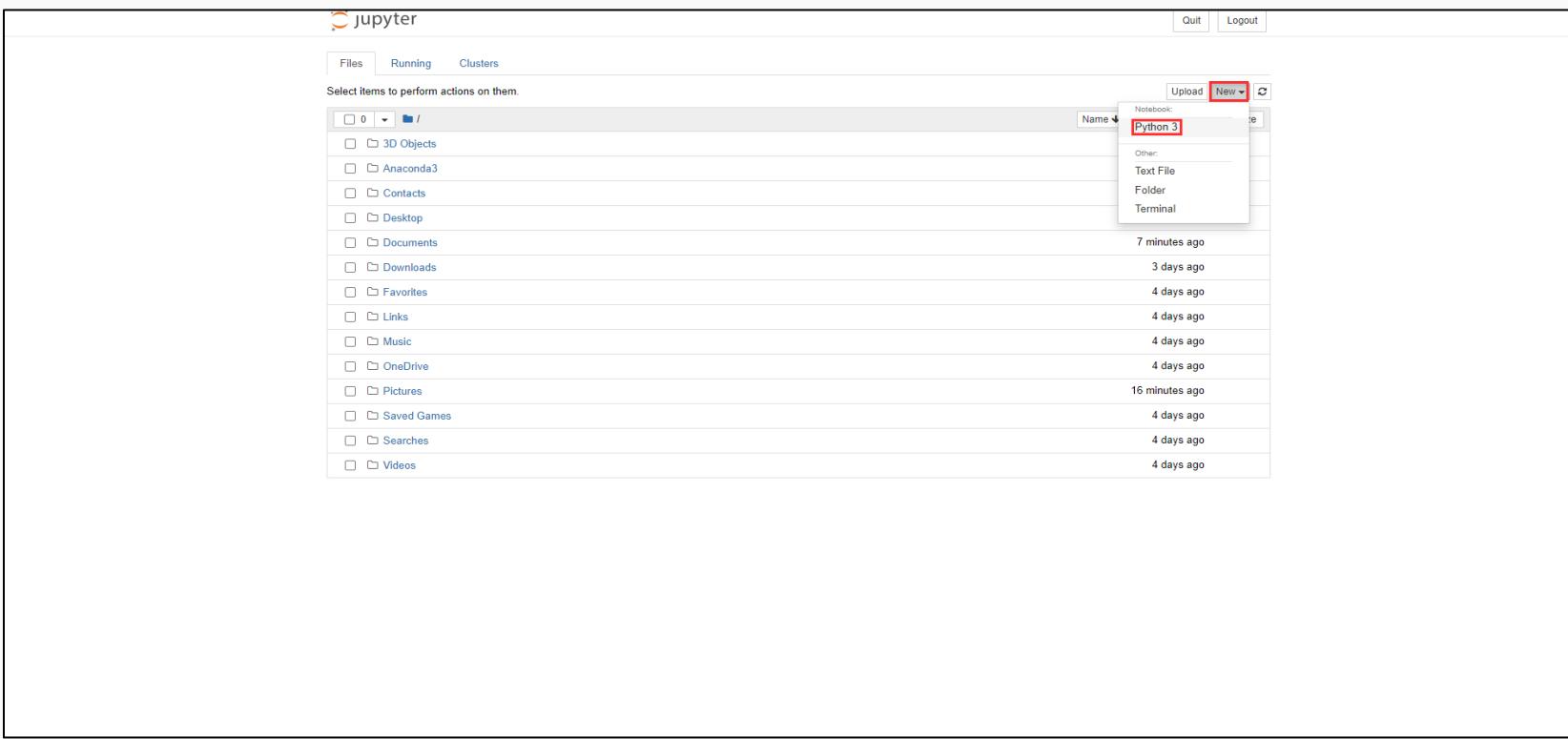
- Launch Jupyter Notebook. It will take sometime to launch.





Anaconda for Mac (13/14)

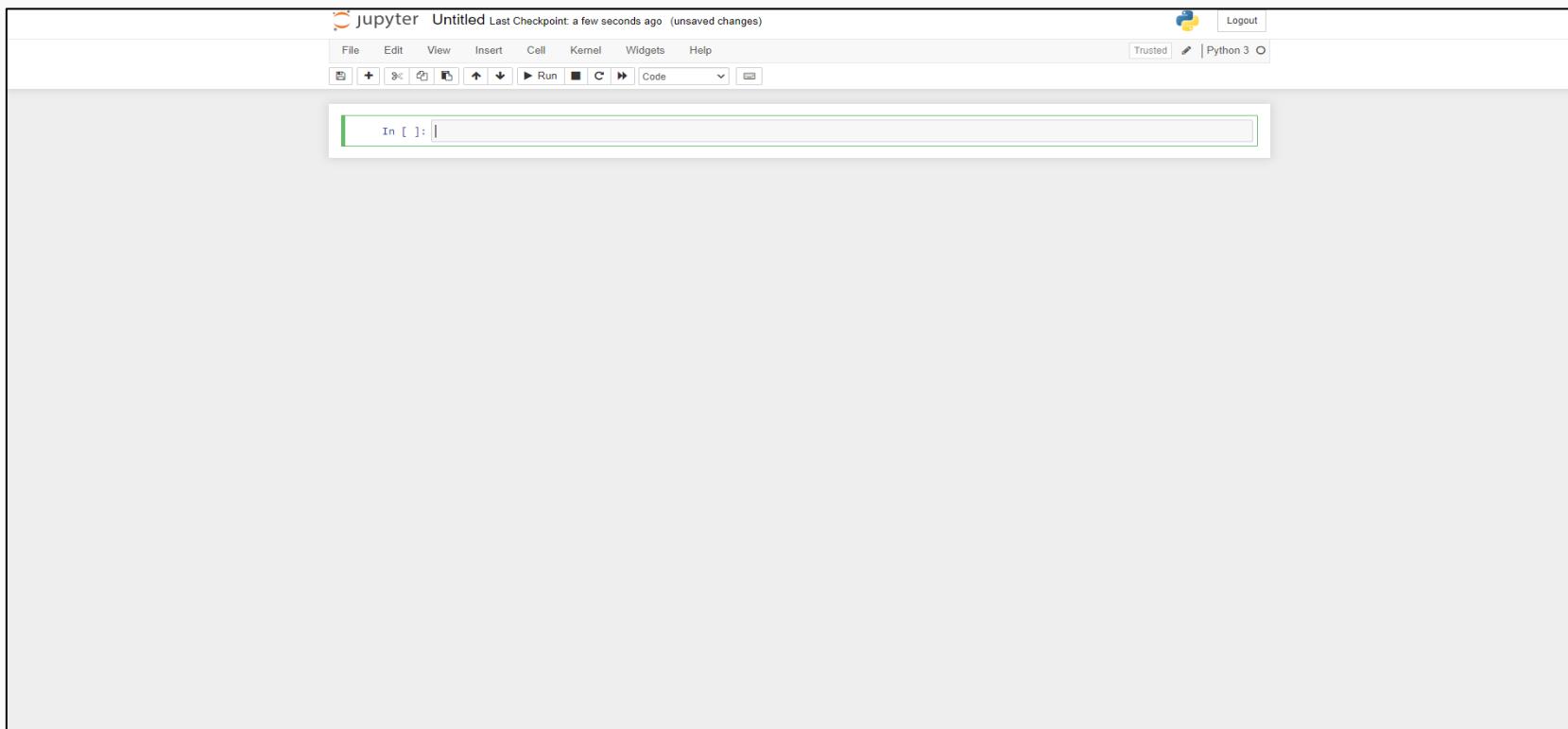
- This will launch Jupyter Notebook in your default browser.
- Click on New -> Python3 to create a new Notebook.





Anaconda for Mac (14/14)

- This is how a Notebook looks like.





Anaconda for Ubuntu (1/11)

- Open your terminal, paste the following command, and hit enter
sudo apt install libgl1-mesa-glx libegl1-mesa libxrandr2 libxrandr2 libxss1 libxcursor1 libcomposite1 libasound2 libxi6 libxtst6
- Type your password and hit Enter.

A screenshot of a terminal window titled "waqar@waqar-N750BU: ~". The window contains the following text:
waqar@waqar-N750BU:~\$ sudo apt install libgl1-mesa-glx libegl1-mesa libxrandr2 libxrandr2 libxss1 libxcursor1 libcomposite1 libasound2 libxi6 libxtst6
[sudo] password for waqar:

The terminal has a dark background and light-colored text. The cursor is located at the end of the password prompt line.



Anaconda for Ubuntu (2/11)

- Next give the following command and hit enter

```
wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-x86_64.sh -O  
~/Downloads/Anaconda3-2020.11-Linux-x86_64.sh
```

The screenshot shows a terminal window titled "waqar@waqar-N750BU: ~". The window displays the output of a command that first installs several Debian packages (libasound2, libasound2-data, libegl1-mesa, libegl1-mesa-glx) and then downloads the Anaconda installer script. The terminal shows the progress of the wget command, which is currently at 16% completion with a speed of 7.34MB/s and an estimated time remaining of 61s.

```
Preparing to unpack .../libasound2_1.2.2-2.1ubuntu2.4_amd64.deb ...
Unpacking libasound2:amd64 (1.2.2-2.1ubuntu2.4) over (1.2.2-2.1ubuntu2.3) ...
Preparing to unpack .../libasound2-data_1.2.2-2.1ubuntu2.4_all.deb ...
Unpacking libasound2-data (1.2.2-2.1ubuntu2.4) over (1.2.2-2.1ubuntu2.3) ...
Preparing to unpack .../libegl1-mesa_21.0.3-0ubuntu0.3~20.04.2_amd64.deb ...
Unpacking libegl1-mesa:amd64 (21.0.3-0ubuntu0.3~20.04.2) over (20.2.6-0ubuntu0.20.0
4.1) ...
Preparing to unpack .../libegl1-mesa-glx_21.0.3-0ubuntu0.3~20.04.2_amd64.deb ...
Unpacking libegl1-mesa-glx:amd64 (21.0.3-0ubuntu0.3~20.04.2) over (20.2.6-0ubuntu0.2
0.04.1) ...
Setting up libegl1-mesa:amd64 (21.0.3-0ubuntu0.3~20.04.2) ...
Setting up libasound2-data (1.2.2-2.1ubuntu2.4) ...
Setting up libegl1-mesa-glx:amd64 (21.0.3-0ubuntu0.3~20.04.2) ...
Setting up libasound2:amd64 (1.2.2-2.1ubuntu2.4) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
waqar@waqar-N750BU:~$ wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linu
x-x86_64.sh -O ~/Downloads/Anaconda3-2020.11-Linux-x86_64.sh
--2021-10-03 12:51:56-- https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-
x86_64.sh
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.130.3, 104.16.131.3, 2606
:4700::6810:8303, ...
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.130.3|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 554535580 (529M) [application/x-sh]
Saving to: '/home/waqar/Downloads/Anaconda3-2020.11-Linux-x86_64.sh'

x86_64.sh          16%[==>]   85.64M  7.34MB/s    eta 61s
```



Anaconda for Ubuntu (3/11)

- Next give the following command and hit enter
cd ~/Downloads

The screenshot shows a terminal window with the following output:

```
waqar@waqar-N750BU: ~/Downloads
4.1) ...
Preparing to unpack .../libgl1-mesa-glx_21.0.3-0ubuntu0.3~20.04.2_amd64.deb ...
Unpacking libgl1-mesa-glx:amd64 (21.0.3-0ubuntu0.3~20.04.2) over (20.2.6-0ubuntu0.2
0.04.1) ...
Setting up libegl1-mesa:amd64 (21.0.3-0ubuntu0.3~20.04.2) ...
Setting up libasound2-data (1.2.2-2.1ubuntu2.4) ...
Setting up libgl1-mesa-glx:amd64 (21.0.3-0ubuntu0.3~20.04.2) ...
Setting up libasound2:amd64 (1.2.2-2.1ubuntu2.4) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
waqar@waqar-N750BU:~$ wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linu
x-x86_64.sh -O ~/Downloads/Anaconda3-2020.11-Linux-x86_64.sh
--2021-10-03 12:51:56-- https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-
x86_64.sh
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.130.3, 104.16.131.3, 2606
:4700::6810:8303, ...
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.130.3|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 554535580 (529M) [application/x-sh]
Saving to: '/home/waqar/Downloads/Anaconda3-2020.11-Linux-x86_64.sh'

/home/waqar/Downloads/Anaconda3-2020.11-Linux-x86_64.sh
/home/waqar/Download 100%[=====] 528.85M 7.25MB/s in 75s
2021-10-03 12:53:12 (7.08 MB/s) - '/home/waqar/Downloads/Anaconda3-2020.11-Linux-x8
6_64.sh' saved [554535580/554535580]

waqar@waqar-N750BU:~$ cd ~/Downloads
waqar@waqar-N750BU:~/Downloads$
```



Anaconda for Ubuntu (4/11)

- Next give the following command and hit enter

```
sudo chmod +x Anaconda3-2020.11-Linux-x86_64.sh
```

The screenshot shows a terminal window with the following output:

```
Preparing to unpack .../libgl1-mesa-glx_21.0.3-0ubuntu0.3~20.04.2_amd64.deb ...
Unpacking libgl1-mesa-glx:amd64 (21.0.3-0ubuntu0.3~20.04.2) over (20.2.6-0ubuntu0.2
0.04.1) ...
Setting up libegl1-mesa:amd64 (21.0.3-0ubuntu0.3~20.04.2) ...
Setting up libasound2-data (1.2.2-2.1ubuntu2.4) ...
Setting up libgl1-mesa-glx:amd64 (21.0.3-0ubuntu0.3~20.04.2) ...
Setting up libasound2:amd64 (1.2.2-2.1ubuntu2.4) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
waqar@waqar-N750BU:~$ wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linu
x-x86_64.sh -O ~/Downloads/Anaconda3-2020.11-Linux-x86_64.sh
--2021-10-03 12:51:56-- https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-
x86_64.sh
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.130.3, 104.16.131.3, 2606
:4700::6810:8303, ...
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.130.3|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 554535580 (529M) [application/x-sh]
Saving to: '/home/waqar/Downloads/Anaconda3-2020.11-Linux-x86_64.sh'

/home/waqar/Download 100%[=====] 528.85M 7.25MB/s   in 75s

2021-10-03 12:53:12 (7.08 MB/s) - '/home/waqar/Downloads/Anaconda3-2020.11-Linux-x8
6_64.sh' saved [554535580/554535580]

waqar@waqar-N750BU:~$ cd ~/Downloads
waqar@waqar-N750BU:~/Downloads$ sudo chmod +x Anaconda3-2020.11-Linux-x86_64.sh
waqar@waqar-N750BU:~/Downloads$
```



Anaconda for Ubuntu (5/11)

- Next give the following command and hit enter
./Anaconda3-2020.11-Linux-x86_64.sh
- When prompted, press enter to continue.

The screenshot shows a terminal window titled "waqar@waqar-N750BU: ~/Downloads". The window contains the following text:

```
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
waqar@waqar-N750BU:~$ wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linu
x-x86_64.sh -O ~/Downloads/Anaconda3-2020.11-Linux-x86_64.sh
--2021-10-03 12:51:56-- https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-
x86_64.sh
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.130.3, 104.16.131.3, 2606
:4700::6810:8303, ...
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.130.3|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 554535580 (529M) [application/x-sh]
Saving to: '/home/waqar/Downloads/Anaconda3-2020.11-Linux-x86_64.sh'

/home/waqar/Download 100%[=====] 528.85M 7.25MB/s   in 75s

2021-10-03 12:53:12 (7.08 MB/s) - '/home/waqar/Downloads/Anaconda3-2020.11-Linux-x8
6_64.sh' saved [554535580/554535580]

waqar@waqar-N750BU:~$ cd ~/Downloads
waqar@waqar-N750BU:~/Downloads$ sudo chmod +x Anaconda3-2020.11-Linux-x86_64.sh
waqar@waqar-N750BU:~/Downloads$ ./Anaconda3-2020.11-Linux-x86_64.sh

Welcome to Anaconda3 2020.11

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>> [ ]
```



Anaconda for Ubuntu (6/11)

- Press the Enter key to scroll through the License Agreement.
- After reading the Agreement, type yes in the prompt and hit enter.

A network authentication protocol designed to provide strong authentication for client/server applications by using secret-key cryptography.

cryptography
A Python library which exposes cryptographic recipes and primitives.

pycryptodome
A fork of PyCrypto. It is a self-contained Python package of low-level cryptographic primitives.

pycryptodomex
A stand-alone version of pycryptodome.

libsodium
A software library for encryption, decryption, signatures, password hashing and more.

pynacl
A Python binding to the Networking and Cryptography library, a crypto library with the stated goal of improving usability, security and speed.

Last updated September 28, 2020

Do you accept the license terms? [yes|no]
[no] >>> yes



Anaconda for Ubuntu (7/11)

- Anaconda installer will ask you where do you want to install Anaconda. We suggest pressing Enter key to install it in the home directory.
- After sometime, Anaconda will be installed on your machine.

The screenshot shows a terminal window titled "waqar@waqar-N750BU: ~/Downloads". The window contains the following text:

```
A stand-alone version of pycryptodome.  
libsodium  
A software library for encryption, decryption, signatures, password hashing and more.  
pynacl  
A Python binding to the Networking and Cryptography library, a crypto library with the stated goal of improving usability, security and speed.  
  
Last updated September 28, 2020  
  
Do you accept the license terms? [yes|no]  
[no] >>> yes  
  
Anaconda3 will now be installed into this location:  
/home/waqar/anaconda3  
  
- Press ENTER to confirm the location  
- Press CTRL-C to abort the installation  
- Or specify a different location below  
  
[/home/waqar/anaconda3] >>>  
PREFIX=/home/waqar/anaconda3
```



Anaconda for Ubuntu (8/11)

- Once installation is finished, you will be asked “Do you wish the installer to initialize Anaconda3 by running conda init?” type “yes” and hit Enter. Setup will finish the installation process.

The screenshot shows a terminal window titled "waqar@waqar-N750BU: ~/Downloads". The window displays a list of packages installed, followed by transaction summary messages, and a final prompt asking if the user wishes to initialize Anaconda3.

```
waqar@waqar-N750BU: ~/Downloads
wheel                  pkgs/main/noarch::wheel-0.35.1-py_0
widgetsnbextension    pkgs/main/linux-64::widgetsnbextension-3.5.1-py38_0
wrapt                  pkgs/main/linux-64::wrapt-1.11.2-py38h7b6447c_0
wurlitzer              pkgs/main/linux-64::wurlitzer-2.0.1-py38_0
xlrd                   pkgs/main/noarch::xlrd-1.2.0-py_0
xlsxwriter             pkgs/main/noarch::xlsxwriter-1.3.7-py_0
xlwt                   pkgs/main/linux-64::xlwt-1.3.0-py38_0
xmltodict              pkgs/main/noarch::xmltodict-0.12.0-py_0
xz                     pkgs/main/linux-64::xz-5.2.5-h7b6447c_0
yaml                   pkgs/main/linux-64::yaml-0.2.5-h7b6447c_0
yapf                   pkgs/main/noarch::yapf-0.30.0-py_0
zeromq                 pkgs/main/linux-64::zeromq-4.3.3-he6710b0_3
zict                   pkgs/main/noarch::zict-2.0.0-py_0
zipp                   pkgs/main/noarch::zipp-3.4.0-pyhd3eb1b0_0
zlib                   pkgs/main/linux-64::zlib-1.2.11-h7b6447c_3
zope                   pkgs/main/linux-64::zope-1.0-py38_1
zope.event              pkgs/main/linux-64::zope.event-4.5.0-py38_0
zope.interface          pkgs/main/linux-64::zope.interface-5.1.2-py38h7b6447c_0
zstd                   pkgs/main/linux-64::zstd-1.4.5-h9ceee32_0

Preparing transaction: done
Executing transaction: done
installation finished.
Do you wish the installer to initialize Anaconda3
by running conda init? [yes|no]
[no] >>> yes
```



Anaconda for Ubuntu (9/11)

- To launch Jupyter Notebook, restart your terminal.
- You will see (base) written before your home directory name.
- Type “jupyter notebook” without the quotation marks and hit enter.
- After a few minutes, Jupyter Notebook will run on your default browser.

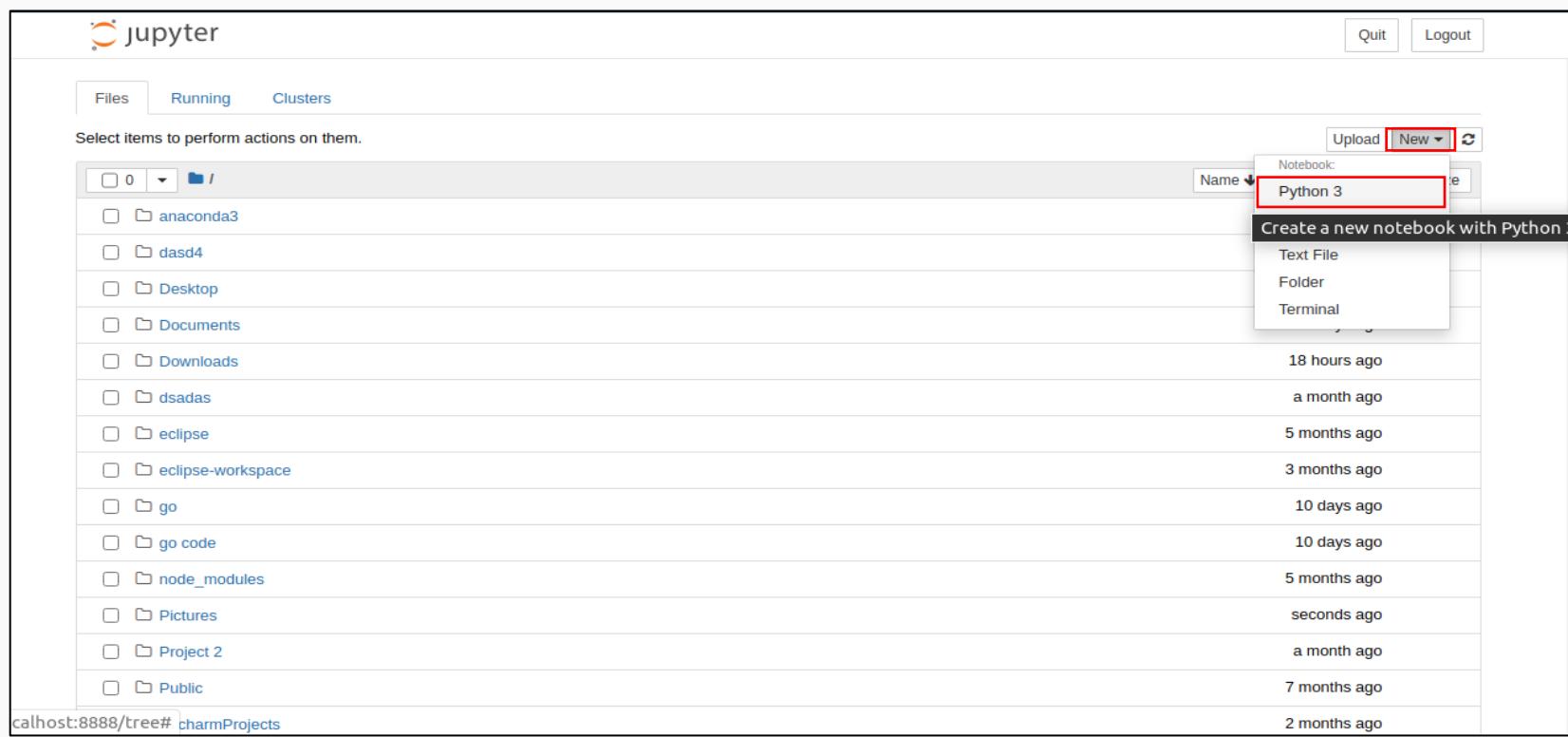
```
(base) waqar@waqar-N750BU:~$ jupyter notebook
[I 13:04:48.077 NotebookApp] JupyterLab extension loaded from /home/waqar/anaconda3/lib/python3.8/site-packages/jupyterlab
[I 13:04:48.077 NotebookApp] JupyterLab application directory is /home/waqar/anaconda3/share/jupyter/lab
[I 13:04:48.080 NotebookApp] Serving notebooks from local directory: /home/waqar
[I 13:04:48.080 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 13:04:48.080 NotebookApp] http://localhost:8888/?token=d5facfe1ebd41a65d6db22c79d447b3f1dd0fae9522ceb0a
[I 13:04:48.080 NotebookApp] or http://127.0.0.1:8888/?token=d5facfe1ebd41a65d6db22c79d447b3f1dd0fae9522ceb0a
[I 13:04:48.081 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 13:04:48.351 NotebookApp]

To access the notebook, open this file in a browser:
    file:///home/waqar/.local/share/jupyter/runtime/nbserver-9875-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=d5facfe1ebd41a65d6db22c79d447b3f1dd0fae9522ceb0a
    or http://127.0.0.1:8888/?token=d5facfe1ebd41a65d6db22c79d447b3f1dd0fae9522ceb0a
```



Anaconda for Mac (10/11)

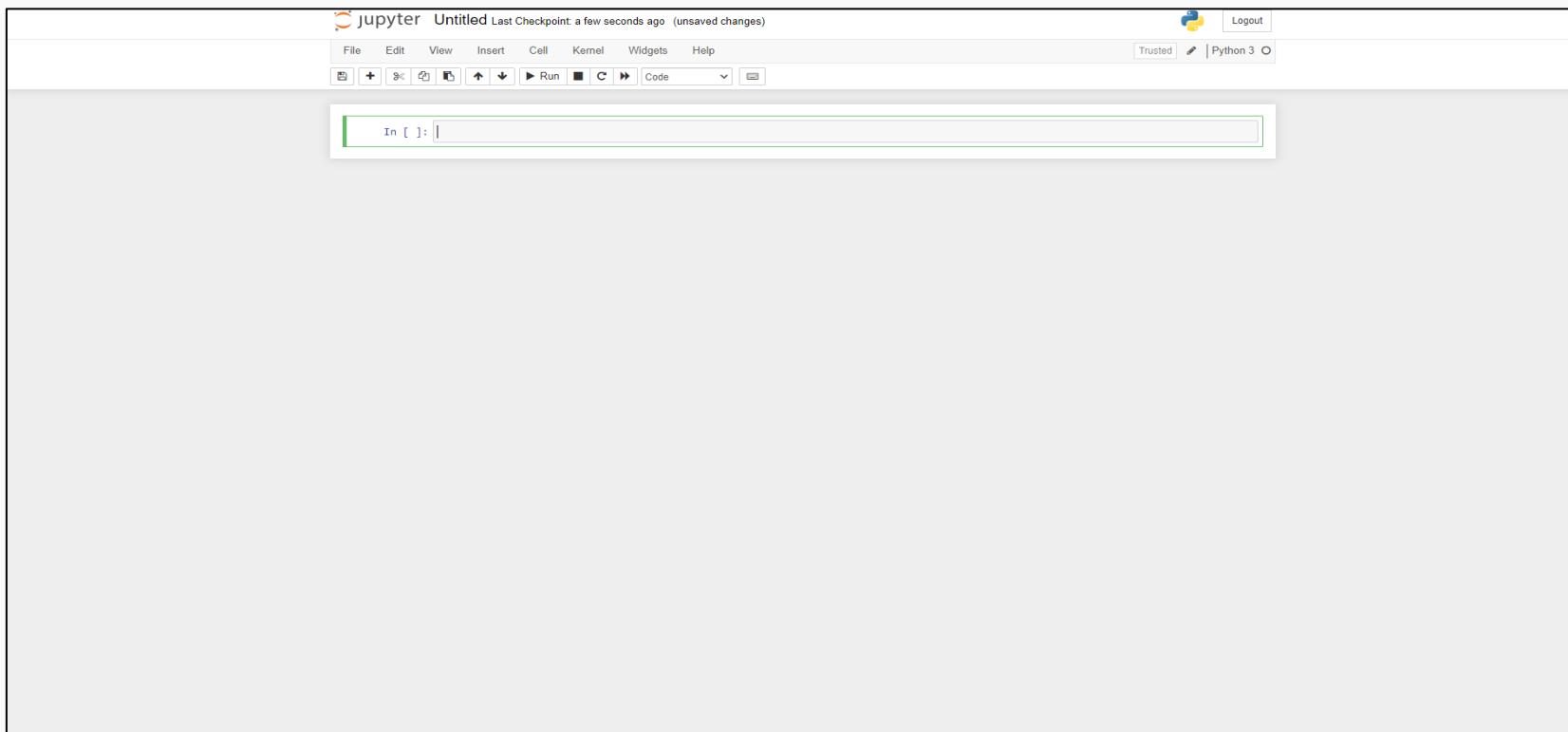
- Click on New -> Python3 to create a new Notebook.





Anaconda for Ubuntu (11/11)

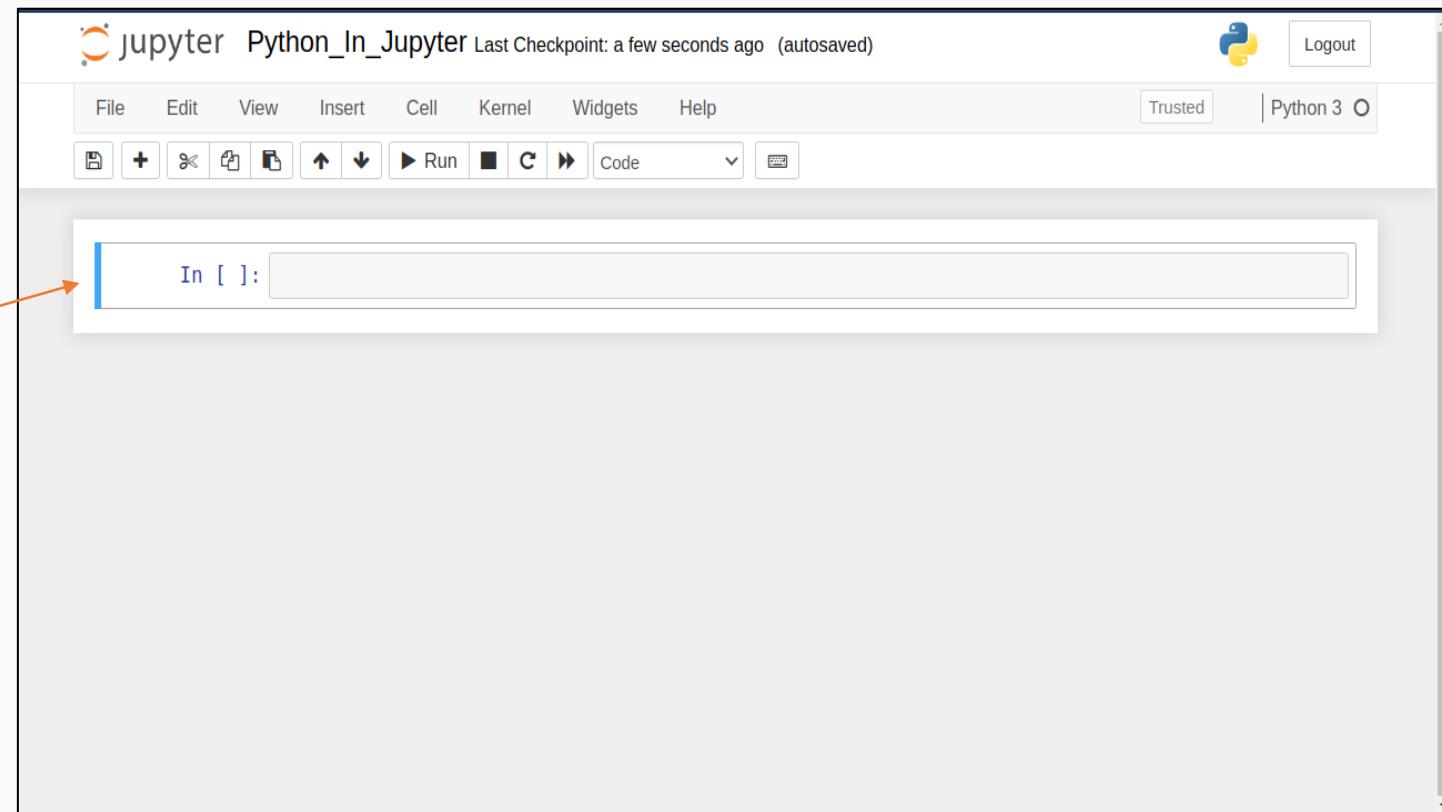
- This is how a Notebook looks like.





Implementing Python in Jupyter Notebook (1/2)

Upon launching Jupyter Notebook, this is what you will see!



This is where we write our code, called a cell. You can have as many of them as you want.



Implementing Python in Jupyter Notebook (2/2)

- Write your Python code inside a cell.
- Select the cell by clicking on it and then click on the Run button to execute the code.
- Output of a cell is produced right below it.

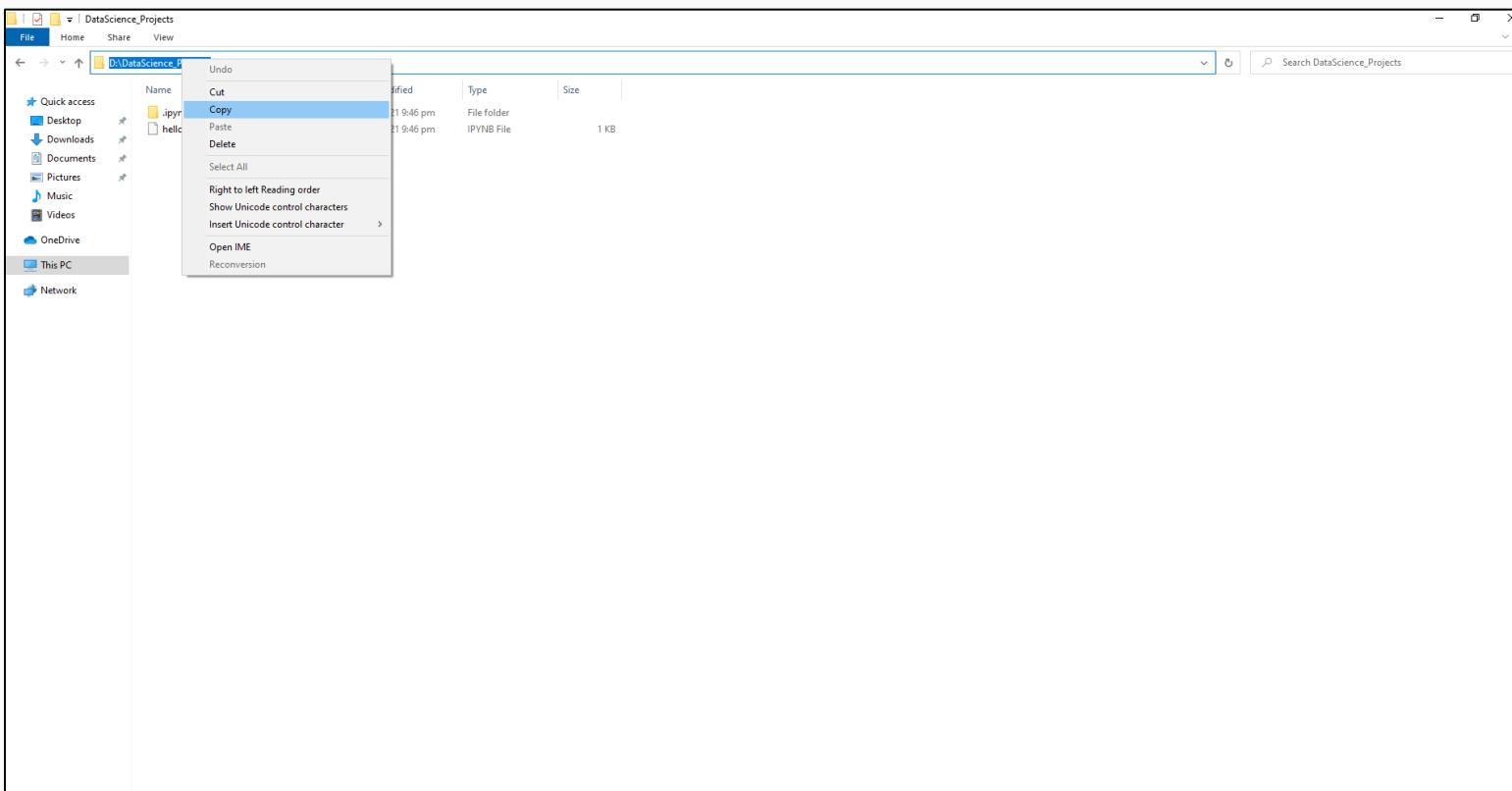
The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** jupyter Python_In_Jupyter Last Checkpoint: 2 minutes ago (unsaved changes) Logout
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help. The "Cell" button is highlighted.
- Status Bar:** Trusted | Python 3
- Cell Content:** In [1]: `print("Hello World")`
- Output:** Hello World
- Annotations:** A red arrow labeled "output" points to the "Hello World" text in the output cell. A mouse cursor is hovering over the "Run" button in the toolbar.



Managing Directories in Jupyter Notebook – Windows (1/7)

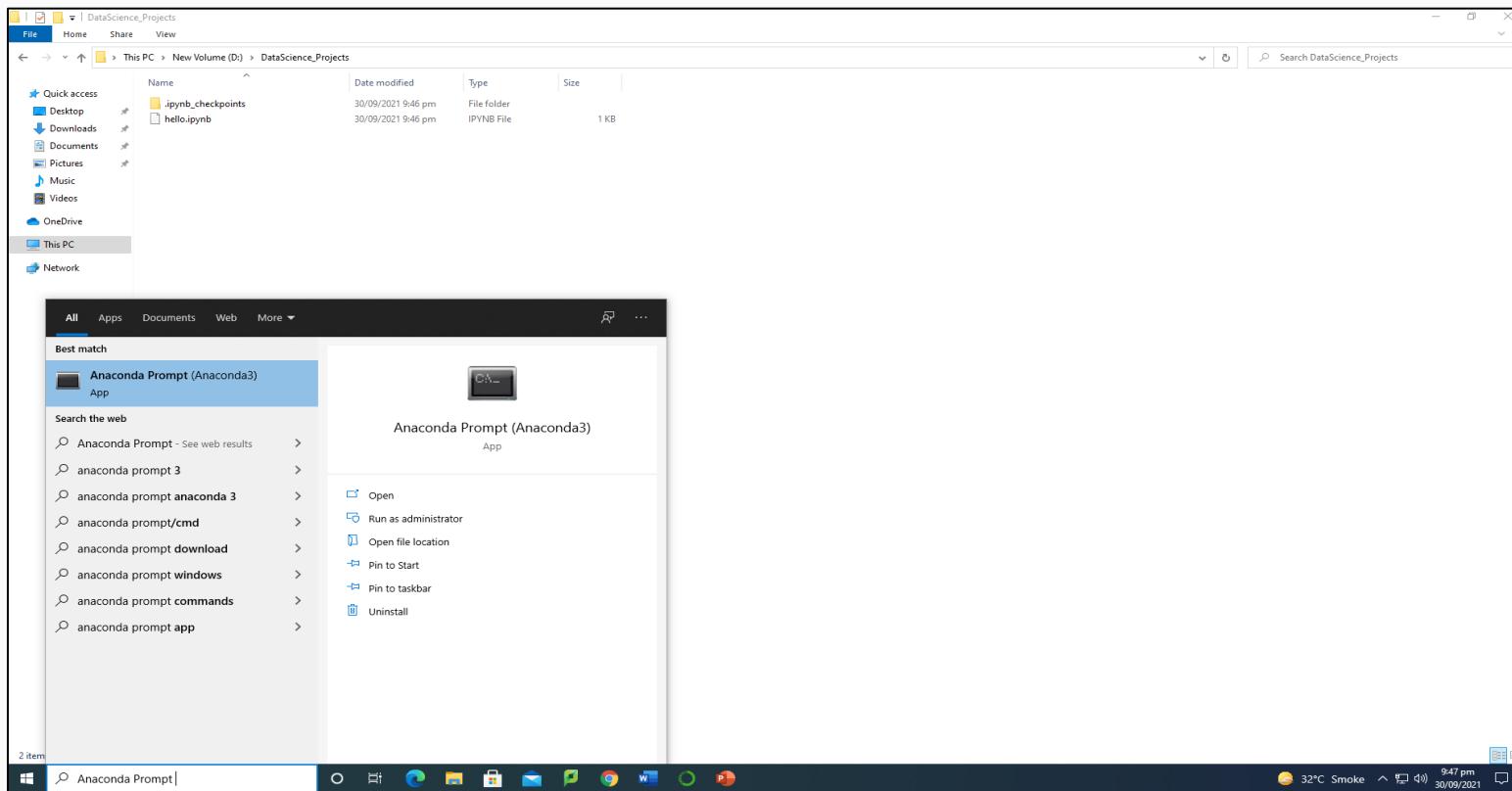
- To open Jupyter Notebook in a particular directory/folder in windows, copy the path of the directory.





Managing Directories in Jupyter Notebook – Windows (2/7)

- In your search bar type Anaconda Prompt and launch it
 - Anaconda Prompt is the CLI of Anaconda





Managing Directories in Jupyter Notebook – Windows (3/7)

- Anaconda Prompt opens up in your C drive. Since we want to move to the D drive, we type d: and hit enter.

The screenshot shows a Windows-style terminal window titled "Anaconda Prompt (Anaconda3)". The command line displays the following text:
(base) C:\Users\22100199>d:
(base) D:\>
The window has standard minimize, maximize, and close buttons at the top right. A vertical scroll bar is visible on the right side of the terminal area.



Managing Directories in Jupyter Notebook – Windows (4/7)

- Next, to switch to the desired directory we type **cd** and then **paste the path of the directory**.

A screenshot of the Anaconda Prompt window titled "Anaconda Prompt (Anaconda3)". The window has a black background and white text. It shows the following command sequence:

```
(base) C:\Users\22100199>d:  
(base) D:\>cd D:\DataScience_Projects  
(base) D:\DataScience_Projects>
```



Managing Directories in Jupyter Notebook – Windows (5/7)

- Finally, we launch Jupyter Notebook by typing jupyter notebook.
- Jupyter Notebook takes a while to open.

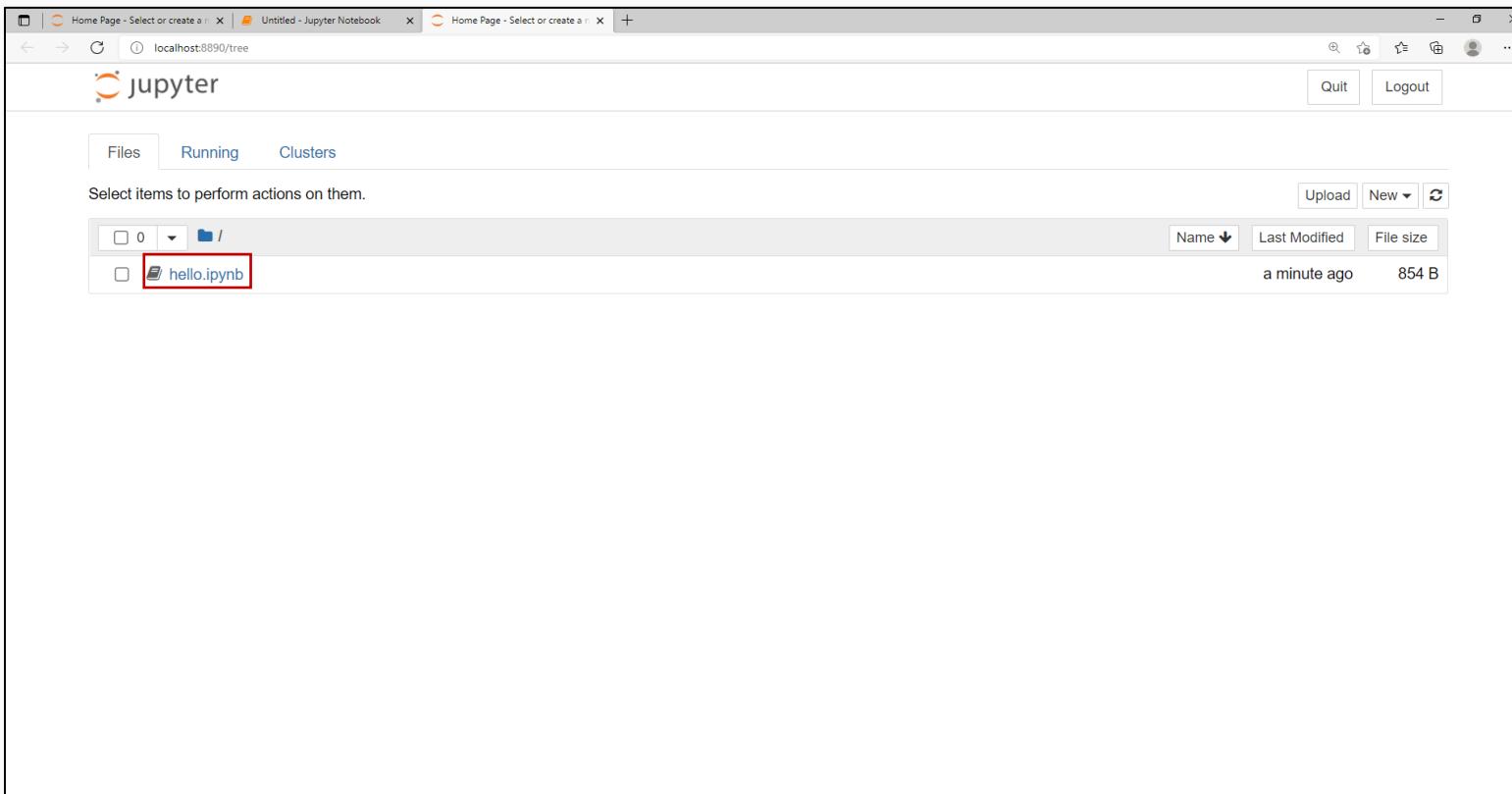
Anaconda Prompt (Anaconda3) - jupyter notebook

```
(base) C:\Users\22100199>d:  
(base) D:\>cd D:\DataScience_Projects  
(base) D:\DataScience_Projects>jupyter notebook  
[I 21:47:44.481 NotebookApp] The port 8888 is already in use, trying another port.  
[I 21:47:44.481 NotebookApp] The port 8889 is already in use, trying another port.  
[I 2021-09-30 21:47:45.068 LabApp] JupyterLab extension loaded from C:\Users\22100199\Anaconda3\lib\site-packages\jupyterlab  
[I 2021-09-30 21:47:45.068 LabApp] JupyterLab application directory is C:\Users\22100199\Anaconda3\share\jupyter\lab  
[I 21:47:45.068 NotebookApp] Serving notebooks from local directory: D:\DataScience_Projects  
[I 21:47:45.068 NotebookApp] Jupyter Notebook 6.3.0 is running at:  
[I 21:47:45.068 NotebookApp] http://localhost:8890/?token=14d64985db6992c640f1d073c32482fc04aa213dc8262852  
[I 21:47:45.068 NotebookApp] or http://127.0.0.1:8890/?token=14d64985db6992c640f1d073c32482fc04aa213dc8262852  
[I 21:47:45.068 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).  
-
```



Managing Directories in Jupyter Notebook – Windows (6/7)

- Once it opens, we can either open an existing notebook or create a new one.
- In this case we open the existing notebook named hello.ipynb.





Managing Directories in Jupyter Notebook – Windows (7/7)

- This is how hello.ipynb looks like.

The screenshot shows a Jupyter Notebook interface running in a web browser. The title bar indicates the window is titled 'hello - Jupyter Notebook'. The browser address bar shows the URL 'localhost:8890/notebooks/hello.ipynb'. The notebook menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. On the right side of the menu bar, there are buttons for 'Not Trusted' and 'Python 3'. Below the menu bar is a toolbar with various icons for file operations. The main workspace contains one code cell:

```
In [1]: print("Hello World")
Hello World
```

The cell output 'Hello World' is displayed below the code. There is also an empty cell placeholder 'In []:' below the first cell.



Managing Directories in Jupyter Notebook – Mac (1/3)

- To open Jupyter Notebook in a particular directory/folder in Mac, open that directory in terminal.
- Type ‘jupyter notebook’ without the quotation marks and hit Enter.

A screenshot of a Mac OS X terminal window. The window has red, yellow, and green close buttons at the top left. The title bar is dark. The main area shows the following text:

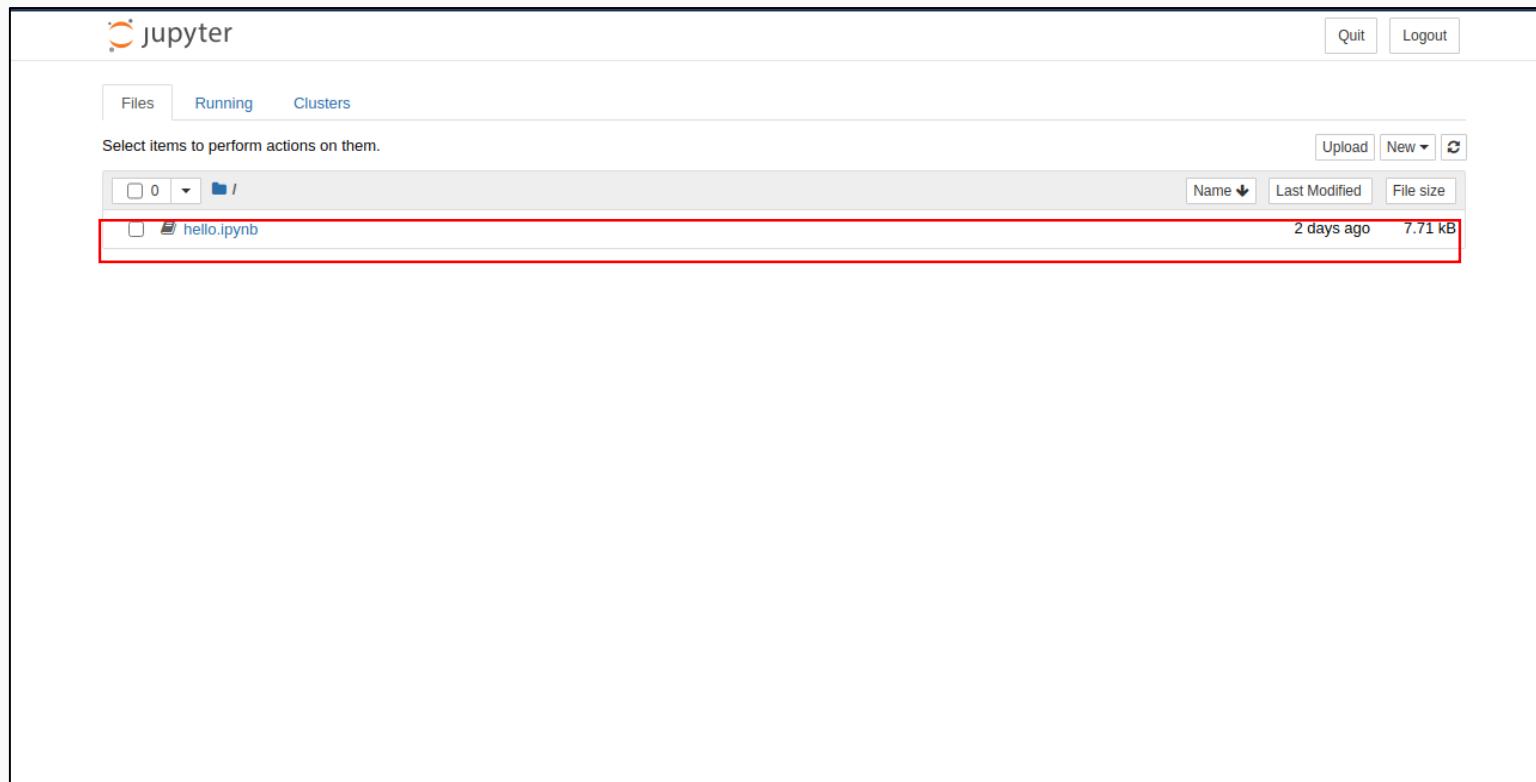
```
Last login: Fri Mar 27 23:32:59 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base)      $ jupyter notebook
```

The cursor is visible at the end of the command line.



Managing Directories in Jupyter Notebook – Mac (2/3)

- Jupyter Notebook will launch in your browser.
- You can create a new Notebook or open an existing one.
- Let's open this 'hello.ipynb' file.





Managing Directories in Jupyter Notebook – Mac (3/3)

- ‘hello.ipynb’ contains a simple print statement.

The screenshot shows a Jupyter Notebook interface titled "jupyter hello". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar below the menu has icons for file operations, cell selection, and execution. On the right, there are buttons for Trusted, Logout, and Python 3. The main area displays a code cell with the following content:

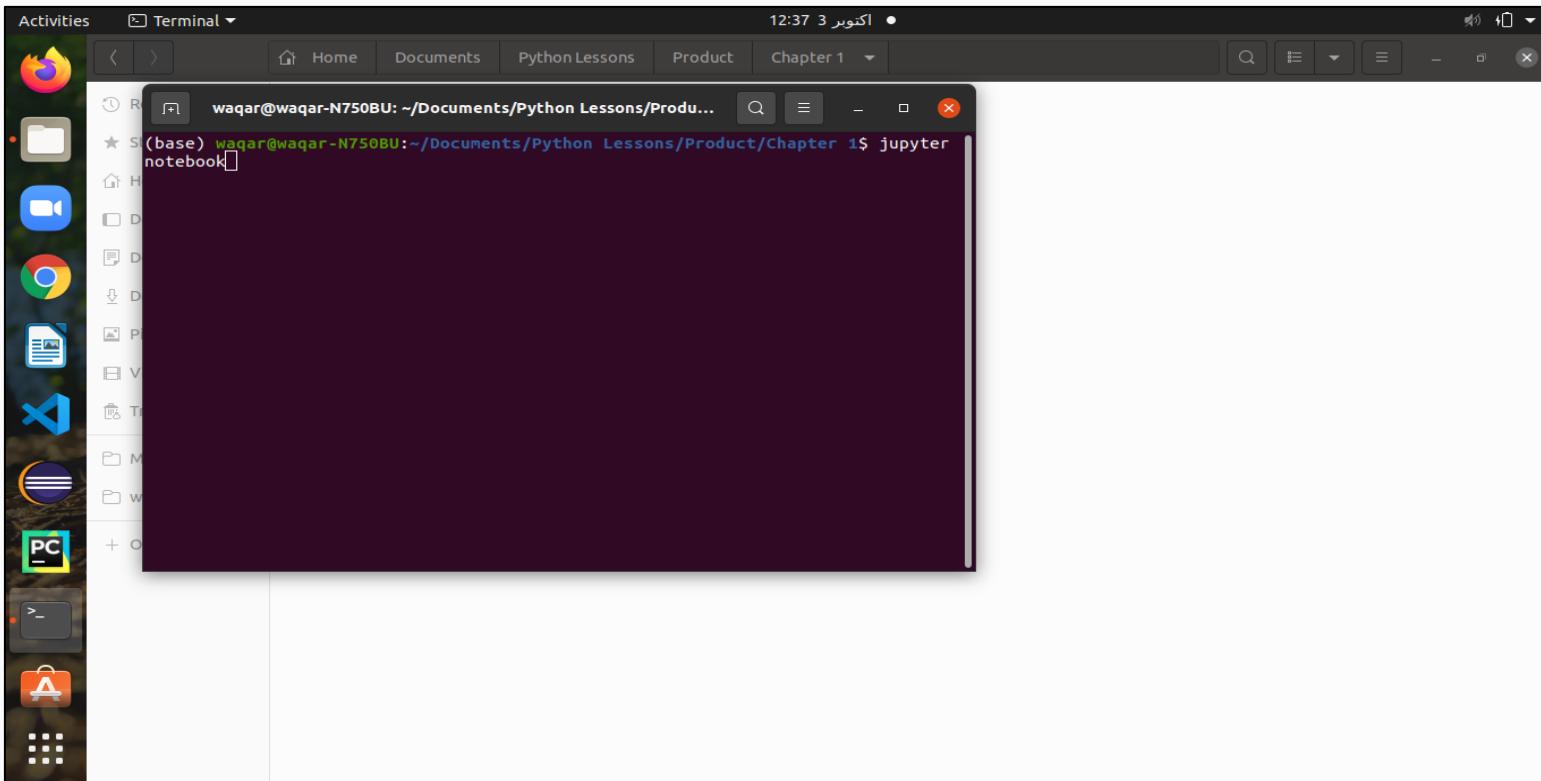
```
In [1]: print("Hello World")
Hello World
```

The output "Hello World" is displayed in a monospaced font. Below the cell, there is another input field labeled "In []:" with a cursor, indicating where the next code can be entered.



Managing Directories in Jupyter Notebook – Ubuntu (1/4)

- To open Jupyter Notebook in a particular directory/folder in Ubuntu, go to that directory, right click and select ‘open in terminal’.
- Type ‘jupyter notebook’ without quotation marks and hit Enter.





Managing Directories in Jupyter Notebook – Ubuntu (2/4)

- Wait for sometime.

```
waqar@waqar-N750BU: ~/Documents/Python Lessons/Product/Chapter 1$ jupyter notebook
[I 12:37:48.012 NotebookApp] JupyterLab extension loaded from /home/waqar/anaconda3/lib/python3.8/site-packages/jupyterlab
[I 12:37:48.013 NotebookApp] JupyterLab application directory is /home/waqar/anaconda3/share/jupyter/lab
[I 12:37:48.017 NotebookApp] Serving notebooks from local directory: /home/waqar/Documents/Python Lessons/Product/Chapter 1
[I 12:37:48.017 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 12:37:48.017 NotebookApp] http://localhost:8888/?token=5ba3e23ed05c480fe4a767b1dbb853ae0c163b31090a34c4
[I 12:37:48.017 NotebookApp] or http://127.0.0.1:8888/?token=5ba3e23ed05c480fe4a767b1dbb853ae0c163b31090a34c4
[I 12:37:48.017 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 12:37:48.180 NotebookApp]

To access the notebook, open this file in a browser:
    file:///home/waqar/.local/share/jupyter/runtime/nbserver-2797-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=5ba3e23ed05c480fe4a767b1dbb853ae0c163b31090a34c4
    or http://127.0.0.1:8888/?token=5ba3e23ed05c480fe4a767b1dbb853ae0c163b31090a34c4
```



Managing Directories in Jupyter Notebook – Ubuntu (3/4)

- Jupyter Notebook will launch in your default browser.
- You can create a new Notebook or open an existing one.
- Let's open this 'hello.ipynb' file.





Managing Directories in Jupyter Notebook – Mac (4/4)

- ‘hello.ipynb’ contains a simple print statement.

The screenshot shows a Jupyter Notebook interface on a Mac. The title bar reads "jupyter hello (unsaved changes)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. On the right, there are buttons for Trusted, Logout, and Python 3. The main area displays a code cell with the following content:

```
In [1]: print("Hello World")
Hello World
```

The "Hello World" output is highlighted with a green border, indicating it is selected or the current cell. The status bar at the bottom shows "In []:" followed by a cursor.



Output

Use the print() function to produce output.

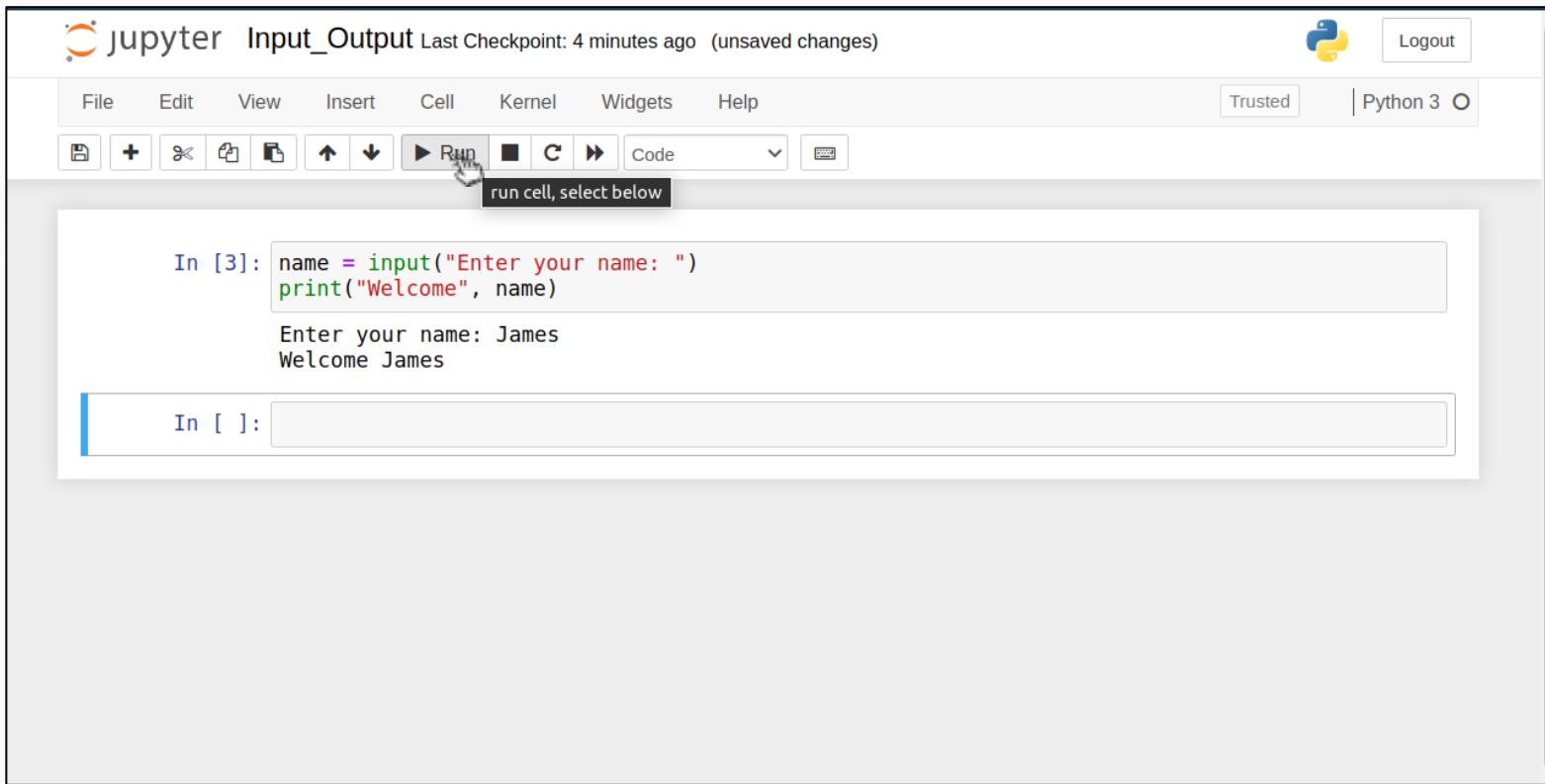
The screenshot shows a Jupyter Notebook interface with the title "jupyter Input_Output" and a status bar indicating "Last Checkpoint: 5 minutes ago (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, and Python 3. The "Run" button is highlighted with a mouse cursor, and a tooltip says "run cell, select below". Below the toolbar, the code cell "In [4]: print("Hello World")" is run, resulting in the output "Hello World". A new code cell "In []:" is visible at the bottom.

```
In [4]: print("Hello World")
Hello World
In [ ]:
```



Input

- Use the `input()` function to take input from the user.
- Here we are providing James as input.



The screenshot shows a Jupyter Notebook interface with the title "jupyter Input_Output". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, and Python 3. A "Run" button is highlighted with a mouse cursor. Below the toolbar, a code cell contains the following Python code:

```
In [3]: name = input("Enter your name: ")
print("Welcome", name)
```

The output area shows the result of running the code:

```
Enter your name: James
Welcome James
```

A new input cell is visible at the bottom:

```
In [ ]: 
```



Quiz Time

1. Which of the following is the correct input statement?

- a) Input("Enter your name: ")
- b) input("Enter your name: ")
- c) Input(Enter your name:)
- d) input(Enter your name:)



Quiz Time

1. Which of the following is the correct input statement?

- a) Input("Enter your name: ")
- b) **input("Enter your name: ")**
- c) Input(Enter your name:)
- d) input(Enter your name:)



Working with different data types

- Python has following primitive data types
 - int
 - float
 - string
 - bool
- Python also has following built-in non-primitive data types
 - Lists
 - Dictionaries
 - Tuples



Working with different data types

int

An int (integer) is a whole number such as 1, 5, 65, 1000

float

A float is a decimal point number such as 1.5, 7.8, 100.0

string <https://unibo.zoom.us/j/83422537997?pwd=Z3VENmI0QTV2Q2xjWDIPQzhrVU5PZz09>

A string is a sequence of characters which can include alphabets, numbers, whitespaces, and special characters. In python, we use single, double or triple quotes to denote a string.

- E.g., ‘Hello World’, “Hello World”, and “”Hello World”” are all valid strings

bool

A bool is either True or False



Working with different data types

type()

type() function is used to get the type of a value/variable.

The screenshot shows a Jupyter Notebook interface with the title "jupyter Data_Types". The notebook has a Python 3 kernel and is in a trusted state. The toolbar includes buttons for file operations, cell creation, and execution. A tooltip "run cell, select below" is visible over the Run button. The code cells show the following outputs:

```
In [7]: print(type(5))
<class 'int'>

In [8]: print(type(2.5))
<class 'float'>

In [9]: print(type("Hello"))
<class 'str'>

In [10]: print(type(True))
<class 'bool'>
```



Variables

- Variables are placeholders for values.
- Python is a dynamically typed language, which means that the data types of variables are checked at runtime.
 - Hence, we do not need to declare the data type of a variable at the time of creating the variable.
- To print the value of a variable, pass the variable name to print() statement without quotation marks.

Rules for Variable Names

- Variable names can contain alphabets, integers, and/or underscore (_) character.
- Variable names can not begin with a number. They can only begin with a letter or an underscore (_).
- Variable names can not contain whitespaces and special characters.



Variables

Example in Jupyter

```
In [11]: x = 1
print(x)
1

In [13]: name = "James"
print(name)
James
```



Quiz Time

1. Which of the following is a valid variable name?

- a) _name
- b) @name
- c) 1name
- d) n a m e

2. What is the correct way of printing the value of a variable called num?

- a) print("num")
- b) print('num')
- c) print("“num”")
- d) print(num)



Quiz Time

1. Which of the following is a valid variable name?

- a) `_name`
- b) `@name`
- c) `1name`
- d) `n a m e`

2. What is the correct way of printing the value of a variable called num?

- a) `print("num")`
- b) `print('num')`
- c) `print("")num""")`
- d) `print(num)`



Arithmetic Operators (1/3)

Addition

Use the plus (+) operator for addition

Subtraction

Use the minus (-) operator for subtraction

<https://ombo.zoom.us/j/83422537997?pwd=Z3VENmI0QTV2Q2xJWDIPQzhrVU5PZz09>

Multiplication

Use the asterisk (*) operator for multiplication



Arithmetic Operators (2/3)

Division

There are two types of division in Python

- Float Division: Use (/) for float division. If the result is an integer, it is converted to a float by adding .0
- Integer Division: Use (//) for integer division. If the result of is a float, it is converted to an integer by truncating the decimal part

Modulo

- Modulo operator returns the remainder of a division
- Use (%) operator modulo



Arithmetic Operators (3/3)

The screenshot shows a Jupyter Notebook interface with the title "jupyter Arithmetic_Operators" and a status bar indicating "Last Checkpoint: 13 minutes ago (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, and Python 3. The code editor displays the following code and output:

```
In [14]: print(5 + 3)
8

In [15]: print(6.5-2)
4.5

In [16]: print(2*4)
8

In [17]: print(5/2)
2.5

In [19]: print(5//2)
2

In [20]: print(5 % 2)
1
```



Quiz Time

1. What is the correct output of the float division $4/2$?

- a) 2
- b) 2.0
- c) 0
- d) 1

2. What is the correct output of the integer division $4//2$?

- a) 2
- b) 2.0
- c) 0
- d) 1

3. What is the correct output of the expression $9\%2$?

- a) 0
- b) 1
- c) 4.5
- d) 4



Quiz Time

1. What is the correct output of the float division $4/2$?

- a) 2
- b) 2.0**
- c) 0
- d) 1

2. What is the correct output of the integer division $4//2$?

- a) 2**
- b) 2.0
- c) 0
- d) 1

3. What is the correct output of the expression $9\%2$?

- a) 0
- b) 1**
- c) 4.5
- d) 4



Comparison Operators

- Comparison operators are used to compare two values with each other.
- The result of a comparison operator is either True or False.
- We have the following comparison operators
 - **Greater than (>)**
 - returns True if the value on the left is bigger than that on the right
 - **Less than (<)**
 - returns True if the value on the left is smaller than that on the right
 - **Greater than or equal to (>=)**
 - returns True if the value on the left is either bigger than or equal to the value on the right
 - **Less than or equal to (<=)**
 - returns True if the value on the left is either smaller than or equal to the value on the right
 - **Not equal to (!=)**
 - returns True if the value on the left is not equal to the value on the right
 - **Is equal to (==)**
 - returns True if the value on the left is equal to the value on the right



Logical Operators

- We can use logical operators to combine two or more conditions.
- Python has three logical operators
 - And operator (`and`)
 - Or operator (`or`)
 - Not operator (`not`)
- and/or operators are binary as they take two operands.
- Not is a unary operator as it takes only one operand.



Logical Operators

and operator

- and returns a True only if both the operands are True.
- In python, we use the keyword ‘and’ to perform and operation.

The screenshot shows a Jupyter Notebook interface with the title "jupyter and_Operator (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3, and Logout. The "Run" button is highlighted with a mouse cursor. The notebook contains three code cells:

- In [37]: `print(20 > 3)`
True
- In [38]: `print(100 < 7)`
False
- In [40]: `print((20 > 3) and (100 < 7))`
False



Logical Operators

or operator

- or returns a True if at least one of the operands is True.
- In python, we use the keyword 'or' to perform or operation.

The screenshot shows a Jupyter Notebook interface with the title "jupyter or_Operator (autosaved)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3, and Logout. The toolbar also features icons for file operations, cell selection, and execution. A tooltip "run cell, select below" is visible over the Run button. The notebook contains three code cells:

- In [37]: `print(20 > 3)`
True
- In [38]: `print(100 < 7)`
False
- In [39]: `print((20 > 3) or (100 < 7))`
True



Logical Operators

not operator

- not is a unary operator.
- not changes True to False and vice versa.
- In python, we use the keyword 'not' to perform not operation.

The screenshot shows a Jupyter Notebook interface with the title "jupyter not_Operator (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3, and Logout. Below the toolbar is a toolbar with icons for file operations and cell execution. A tooltip "run cell, select below" is visible over the "Run" button. The main area displays a code cell labeled "In [41]:" containing the following Python code:

```
In [41]: x = True
print(x)
print(not x)
```

The output of the cell is:

```
True
False
```



Quiz Time

1. What is the correct output of the expression; $(4 > 3)$ and $(5 > 3)$?

- a) True
- b) False

2. What is the correct output of the expression; $(7 < 2)$ or $(8 < 3)$?

- a) True
- b) False

3. What is the correct output of the expression; not True?

- a) True
- b) False



Quiz Time

1. What is the correct output of the expression; (4 >3) and (5>3)?

- a) **True**
- b) False

2. What is the correct output of the expression; (7 < 2) or (8 < 3)?

- a) True
- b) **False**

3. What is the correct output of the expression; not True?

- a) True
- b) **False**



Conditional Statements (1/4)

- Conditional statements are used to make decisions.
- In Python, we make decisions using if-else and elif statements.
- elif and else are both optional, however they must follow an if statement.
- We can have as many elif statements as we want. But there is only one if and one else statement.



Conditional Statements (2/4)

Control Flow

- Here is the general syntax of conditional statements

```
if condition:
```

```
    block of statements to execute
```

```
elif condition:
```

```
    block of statements to execute
```

```
else:
```

```
    block of statements to execute
```

- Conditions are evaluated in a top-down manner.



Conditional Statements (3/4)

Example in Jupyter

The screenshot shows a Jupyter Notebook interface with the title "jupyter Conditional_Statements (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3, and Logout. Below the toolbar is a toolbar with various icons for file operations like Open, Save, and Run. A tooltip "run cell, select below" points to the Run button. The code cell contains the following Python code:

```
In [46]: if 5 == 6:  
    print(1)  
elif 6 == 6:  
    print(2)  
else:  
    print(3)
```

The output cell shows the number 2, indicating that the condition 6 == 6 was true.



Conditional Statements (4/4)

Combining Conditions

- We can combine two or more conditions together using logical operators.

```
In [49]: if 7 > 3 and 10 < 5:  
    print(1)  
elif 7 > 3 and 5 < 10:  
    print(2)  
elif 3 > 7 and 10 < 5:  
    print(3)  
elif 3 > 7 and 5 < 10:  
    print(4)  
else:  
    print(5)
```

2



Loops

- Loops are used for repeating a set of statements.
- There are two types of loops in Python
 - For loops
 - While loops

<https://unibo.zoom.us/j/83422537997?pwd=Z3VENmI0QTV2Q2xjWDIPQzhrVU5PZz09>



for Loop (1/2)

- for loop takes a range() function.
- range() takes three arguments; start, stop, step.
- loop runs from start to stop - 1 with an increment equal to step after every iteration.

The screenshot shows a Jupyter Notebook interface with the title "jupyter for_Loop (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3, and Logout. Below the toolbar is a toolbar with icons for file operations, cell selection, and execution. A tooltip says "run cell, select below". The code cell contains the following Python code:

```
In [50]: for i in range(0, 21, 2):
    print(i)
```

The output of the code is:

```
0
2
4
6
8
10
12
14
16
18
20
```



for Loop (2/2)

- If we omit the step value, it is set to 1 by default.

The screenshot shows a Jupyter Notebook interface with a teal header bar. The title bar reads "jupyter for_Loop Last Checkpoint: 34 minutes ago (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, and Python 3. A "Run" button is highlighted with a mouse cursor. The code cell content is "In [51]: for i in range(0, 21): print(i)". The output cell displays the numbers from 0 to 20, each on a new line.

```
In [51]: for i in range(0, 21):
    print(i)
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```



while Loop (1/2)

- while loop takes a condition and runs as long as the condition is True.

The screenshot shows a Jupyter Notebook interface with the title "jupyter while_Loop (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3, and Logout. The toolbar buttons include New, Open, Save, Run, Cell Type, and Cell Options. A tooltip "run cell, select below" points to the Run button. The code cell (In [58]) contains the following Python code:

```
In [58]: i = 1
while i <= 10:
    print(i)
    i = i + 1
```

The output of the code is displayed below the cell, showing the numbers 1 through 10, each on a new line.



while Loop (2/2)

Infinite Loop

- We can create an infinite loop using a while loop.
- We give a terminating condition inside the infinite loop and use the keyword ‘break’ to break out of the infinite loop.

The screenshot shows a Jupyter Notebook interface with the title "jupyter while_Loop Last Checkpoint: 39 minutes ago (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3, and Logout. A tooltip "run cell, select below" points to the Run button. The code cell (In [57]) contains the following Python code:

```
while True:
    userName = input("Enter your username: ")
    if userName == "James":
        print("Welcome James")
        break
    else:
        print("Wrong username")
```

The output window shows the following interaction:

```
Enter your username: Anderson
Wrong username
Enter your username: james
Wrong username
Enter your username: James
Welcome James
```

The bottom cell (In []) is empty.



Sequences

- Python has following built-in sequences (non-primitive data types)
 - Lists
 - Dictionaries
 - Tuples
- We can use the `len()` function to find the total number of elements in any sequence.

The screenshot shows a Jupyter Notebook interface titled "Sequences (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3, and Logout. A "Run" button is highlighted with a cursor. The code cell contains:

```
In [59]: flowersList = ["Rose", "Lily", "Tulip"]
print(len(flowersList))
```

The output cell shows the result:

```
3
```



Lists (1/5)

- Lists are used to store multiple values.
- A list can have values of different data types.
- Lists are ***mutable*** i.e., they can be modified.

<https://unibo.zoom.us/j/83422537997?pwd=Z3VENmI0QTV2Q2xjWDIPQzhrVU5PZz09>



Lists (2/5)

List Indexing

- We can get the element at a particular position in the list using list indexing.
 - Remember, indices start at 0 and go until `len(list) - 1`.

Rose	Lily	Tulip
0	1	2

<https://us0600.zoom.us/j/8342253797?pwd=z3V6NnIuQ1V2Q2XW01PQ2tIVU5PZz09>

- Consider the list above that contains names of three flowers. Hence, it's length is 3.
- If we call this list `flowersList`, then;
 - `flowersList[0]` will give us Rose
 - `flowersList[1]` will give us Lily
 - `flowersList[2]` will give us Tulip



Lists (3/5)

List Slicing

- We can also slice a list.
- For slicing a list, we give a starting-index (inclusive) and an end-index (exclusive).

Rose	Lily	Tulip
0	1	2

<https://unido.zoom.us/j/8342253797?pwd=z5VENmIUQIV2Q2XJWUfPQz0tVU5PZz09>

- If we call this list flowersList then;
 - flowersList[0:2] will give us [Rose, Lily]
 - flowersList[0:3] will give us [Rose, Lily, Tulip]
 - flowersList[1:2] will give us [Lily]



Lists (4/5)

Example in Jupyter

The screenshot shows a Jupyter Notebook interface with the title "jupyter Sequences (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3, and Logout. The "Cell" menu is currently selected. A tooltip "run cell, select below" points to the "Run" button. The code cell contains the following Python code:

```
In [63]: flowersList = ["Rose", "Lily", "Tulip"]
firstElement = flowersList[0]
print(firstElement)

lastTwoElements = flowersList[1:3]
print(lastTwoElements)
```

The output cell displays the results:

```
Rose
['Lily', 'Tulip']
```



Lists (5/5)

Iterating Over List

- We can loop over a list using a modified form of for loop.

The screenshot shows a Jupyter Notebook interface with the title "jupyter Sequences (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3, and Logout. A "Run" button is highlighted with a cursor, and a tooltip says "run cell, select below". The code cell contains the following Python code:

```
In [65]: flowersList = ["Rose", "Lily", "Tulip"]
for i in flowersList:
    print(i)
```

The output of the code is displayed below the cell:

```
Rose
Lily
Tulip
```



Dictionaries (1/3)

- Dictionaries are used to store key, value pairs.
- A key and a value are separated by a colon (:).
- Each key, value pair is separated by a comma (,).
- Dictionaries are **mutable** i.e., they can be modified.
- We can use `.keys()` function to get all the keys and `.values()` function to get all the values in the dictionary.

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** jupyter Sequences (unsaved changes), Python 3
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Logout
- In [3]:**

```
gradesDictionary = {'English': 'A', 'Maths': 'B', 'History': 'C'}  
print(gradesDictionary)
```

Output: {'English': 'A', 'Maths': 'B', 'History': 'C'}
- In [4]:**

```
print(gradesDictionary.keys())
```

Output: dict_keys(['English', 'Maths', 'History'])
- In [5]:**

```
print(gradesDictionary.values())
```

Output: dict_values(['A', 'B', 'C'])

On the right side of the interface, there are two colored circles with labels:

- Blue circle: Keys
- Green circle: Values



Dictionaries (2/3)

- We can get a value associated with a key by passing the key.

The screenshot shows a Jupyter Notebook interface with the title "jupyter Sequences (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, and Python 3. A "Run" button is highlighted with a tooltip "run cell, select below". In the code cell, the following Python code is written:

```
In [69]: data = {"Name": "James", "Age": 30, "Country": "US"}  
name = data["Name"]  
age = data["Age"]  
country = data["Country"]  
  
print(name)  
print(age)  
print(country)
```

The output cell displays the results of the print statements:

```
James  
30  
US
```



Dictionaries (3/3)

Iterating Over Dictionary

- We can get all the key, value pairs in a Dictionary using a modified form of for loop.

The screenshot shows a Jupyter Notebook interface with the title "jupyter Sequences (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3, and Logout. Below the toolbar is a toolbar with icons for file operations and cell execution. A tooltip "run cell, select below" is visible over the "Run" button. The code cell contains the following Python code:

```
In [70]: data = {"Name": "James", "Age": 30, "Country": "US"}  
for key in data:  
    print(data[key])
```

The output cell shows the results of the execution:

```
James  
30  
US
```



Tuples (1/5)

- Tuples are also used to store multiple values.
- Tuples can have values of different data types.
- Tuples are *immutable* i.e., they can not be modified.

<https://unibo.zoom.us/j/83422537997?pwd=Z3VENmI0QTV2Q2xjWDIPQzhrVU5PZz09>



Tuples (2/5)

Tuple Indexing

- We can index a tuple just as we can index a list.

Rose	Lily	Tulip
0	1	2

<https://unido.zoom.us/j/83422537997?pwd=z3VENmI0Q1V2Q2XJWDRUZnVU5PZz09>

- Consider the tuple above that contains names of three flowers. Hence, its length is 3.
- If we call this tuple flowersTuple, then;
 - flowersTuple[0] will give us Rose
 - flowersTuple[1] will give us Lily
 - flowersTuple[2] will give us Tulip



Tuples (3/5)

Tuple Slicing

- Tuple slicing works the same way as list slicing.

Rose	Lily	Tulip
0	1	2

<https://us06web.zoom.us/j/83422537997?pwd=z5VENmIUQIV2Q2XJWUfPQz0tVU5PZz09>

- If we call this tuple flowersTuple then;
 - flowersTuple[0:2] will give us (Rose, Lily)
 - flowersTuple[0:3] will give us (Rose, Lily, Tulip)
 - flowersTuple[1:2] will give us (Lily)



Tuples (4/5)

Example in Jupyter

The screenshot shows a Jupyter Notebook interface titled "jupyter Sequences (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3, and Logout. The "Run" button is highlighted with a cursor. Below the toolbar, a code cell is displayed with the following Python code:

```
In [73]: flowersTuple = ("Rose", "Lily", "Tulip")
firstElement = flowersTuple[0]
print(firstElement)

lastTwoElements = flowersTuple[1:3]
print(lastTwoElements)
```

The output of the code is:

```
Rose
('Lily', 'Tulip')
```



Tuples (5/5)

Iterating Over Tuple

- We can loop over a tuple using a modified form of for loop.

The screenshot shows a Jupyter Notebook interface titled "jupyter Sequences (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3, and Logout. Below the toolbar is a toolbar with icons for file operations, cell creation, and run. A tooltip "run cell, select below" is visible over the run button. The code cell contains the following Python code:

```
In [74]: flowersTuple = ("Rose", "Lily", "Tulip")
for item in flowersTuple:
    print(item)
```

The output of the code is displayed below the cell:

```
Rose
Lily
Tulip
```



Quiz Time

1. Which of the following is the immutable sequence?

- a) List
- b) Dictionary
- c) Tuple

2. Consider a list `x = ["red", 1, True]`. What will `x[1]` give?

- a) "red"
- b) 1
- c) True

3. Consider a dictionary `y = {"name": "James", "age": 1}`. What is "age" in this dictionary?

- a) Key
- b) Value
- c) None



Quiz Time

1. Which of the following is the immutable sequence?

- a) List
- b) Dictionary
- c) **Tuple**

2. Consider a list `x = ["red", 1, True]`. What will `x[1]` give?

- a) "red"
- b) **1**
- c) True

3. Consider a dictionary `y = {"name": "James", "age": 1}`. What is "age" in this dictionary?

- a) **Key**
- b) Value
- c) None



Built-in Functions

- Python has a lot of built-in functions e.g.,
 - `len()`, which returns the length of a sequence
 - `abs()`, which returns the absolute value
 - `max()`, which returns the maximum value in a sequence
- You can find a detailed list of built-in functions in Python at the following link:
<https://docs.python.org/3/library/functions.html>

https://unibo.zoom.us/j/83422537997?pwd=Z3VENmI0QTV2Q2xjWDIPQzhrVU5PZz09



User-defined Functions (1/2)

- Apart from built-in functions, users can create their own functions as well.
- We use the ‘def’ keyword to define a function.
- A function should be defined before it is called.
- We do not need to declare the return type while defining a function.

The screenshot shows a Jupyter Notebook interface with the title "jupyter Sequences (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3, and Logout. The "Run" button is highlighted with a mouse cursor. Below the toolbar, there are buttons for file operations like Open, Save, and New, as well as cell selection and execution controls. Two code cells are visible:

```
In [75]: def AddNumbers(x, y):
    sum = x + y
    return sum
```

```
In [76]: result = AddNumbers(5, 7)
print(result)
12
```



User-defined Functions (2/2)

- We can also return more than one value from a function.
- In this case we are returning 2 values from the functions, hence we need to store them in 2 variables, namely num1 and num2.

The screenshot shows a Jupyter Notebook interface with the title "jupyter Sequences (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, and Python 3. A "Run" button is highlighted with a mouse cursor. Below the toolbar, two code cells are visible:

```
In [78]: def IncrementByOne(x, y):
    x = x + 1
    y = y + 1
    return x, y
```

```
In [81]: num1, num2 = IncrementByOne(5, 7)
print(num1)
print(num2)
```

The output for cell In [81] shows the numbers 6 and 8.



Quiz Time

1. A function should be defined

- a) After it is called
- b) Before it is called
- c) Doesn't matter

2. A function in Python can only return 1 value. Is this statement True or False?

- a) True
- b) False



Quiz Time

1. A function should be defined

- a) After it is called
- b) Before it is called**
- c) Doesn't matter

2. A function in Python can only return 1 value. Is this statement True or False?

- a) True
- b) False**



Resources

Here are a few resources for you to revise Python syntax

- <https://www.w3schools.com/python/>
- <https://www.youtube.com/watch?v=rfscVS0vtbw>
- <https://www.tutorialspoint.com/python/index.htm>



References

- How to install python on Mac.

<https://www.youtube.com/watch?v=M323OL6K5vs>

- How to install Anaconda in Ubuntu

https://www.youtube.com/watch?v=dGm10q_y3xw

- How to install Anaconda in Mac

<https://www.youtube.com/watch?v=PJWnGNWQ1Dc>

- Managing Directories in Jupyter Noteook – Mac

<https://www.youtube.com/watch?v=G1Vm6wR0XIQ>