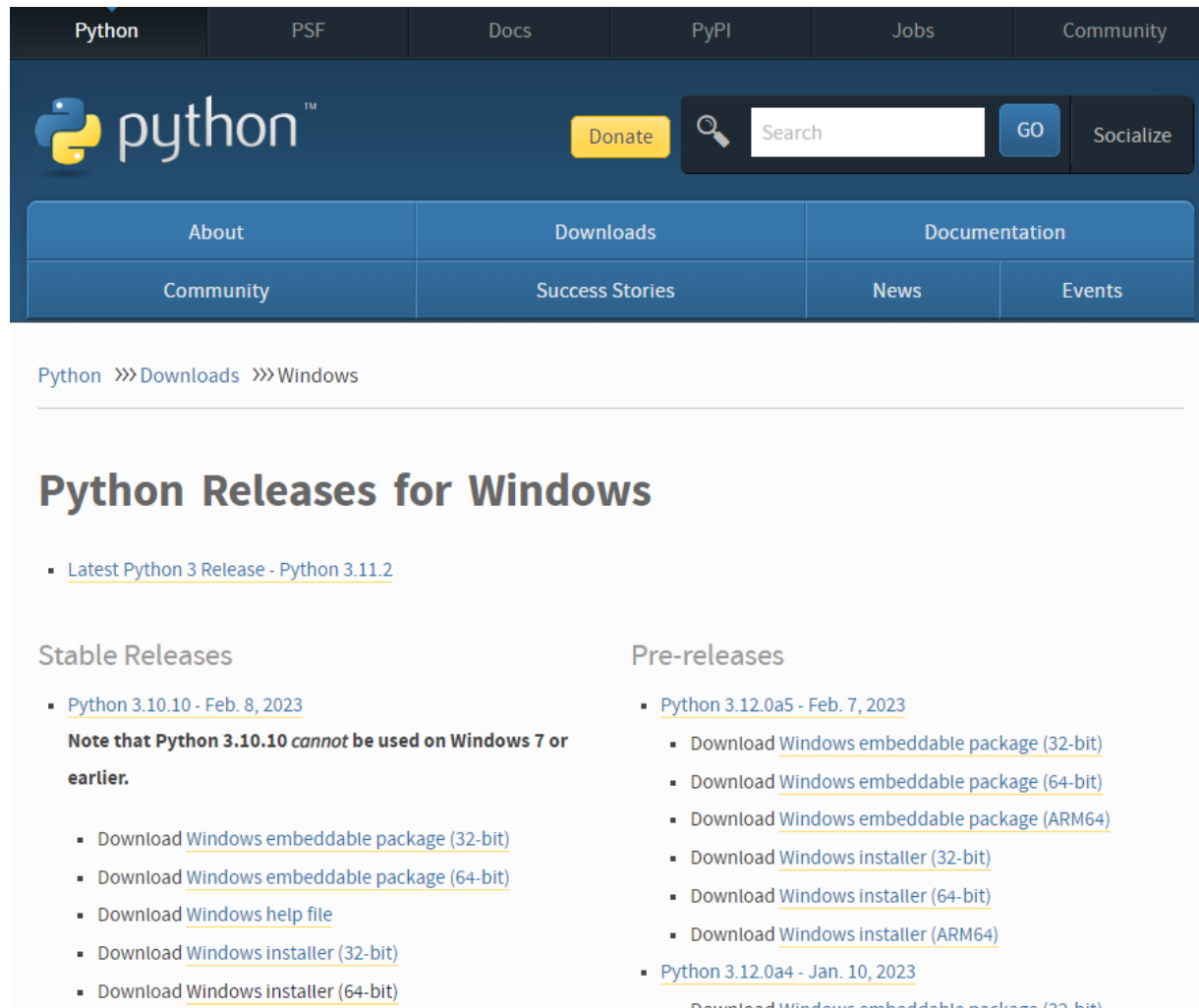


Step 1 — Downloading the Python Installer

1. Go to the official [Python download page for Windows](#).
2. Find a stable Python 3 release. This tutorial was tested with Python version 3.10.10.
3. Click the appropriate link for your system to download the executable file: **Windows installer (64-bit)** or **Windows installer (32-bit)**.

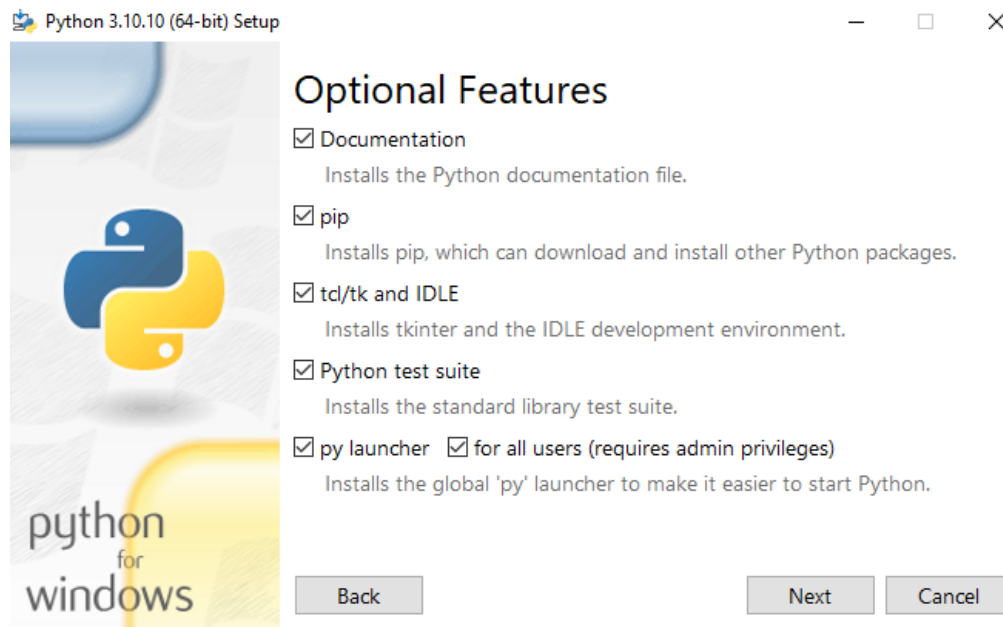


Step 2 — Running the Executable Installer

1. After the installer is downloaded, double-click the .exe file, for example python-3.10.10-amd64.exe, to run the Python installer.
2. Select the **Install launcher for all users** checkbox, which enables all users of the computer to access the Python launcher application.
3. Select the **Add python.exe to PATH** checkbox, which enables users to launch Python from the command line.

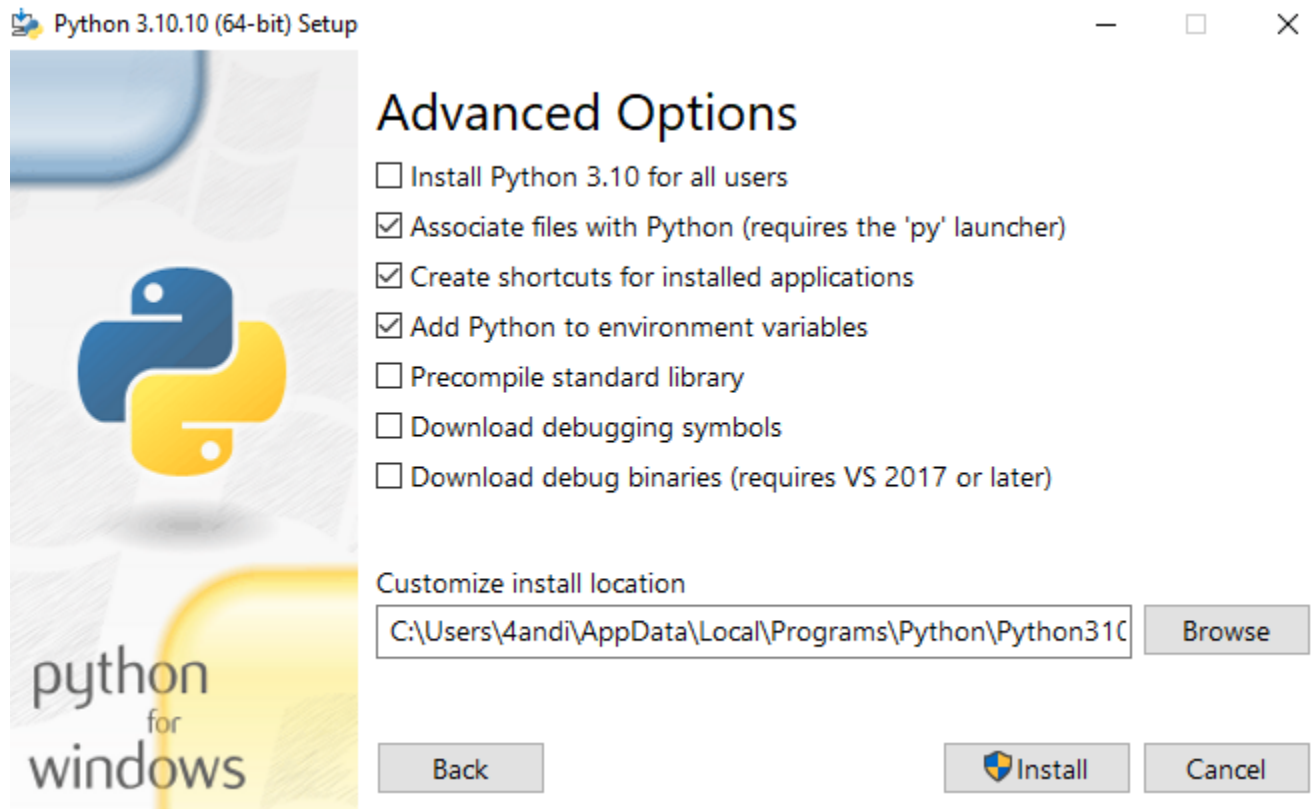


4. If you're just getting started with Python and you want to install it with default features as described in the dialog, then click **Install Now** and go to Step 4 - Verify the Python Installation. To install other optional and advanced features, click **Customize installation** and continue.
5. The **Optional Features** include common tools and resources for Python and you can install all of them, even if you don't plan to use them.



Select some or all of the following options:

- **Documentation:** recommended
 - **pip:** recommended if you want to install other Python packages, such as NumPy or pandas
 - **tcl/tk and IDLE:** recommended if you plan to use IDLE or follow tutorials that use it
 - **Python test suite:** recommended for testing and learning
 - **py launcher** and **for all users:** recommended to enable users to launch Python from the command line
6. Click **Next**.
 7. The **Advanced Options** dialog displays.

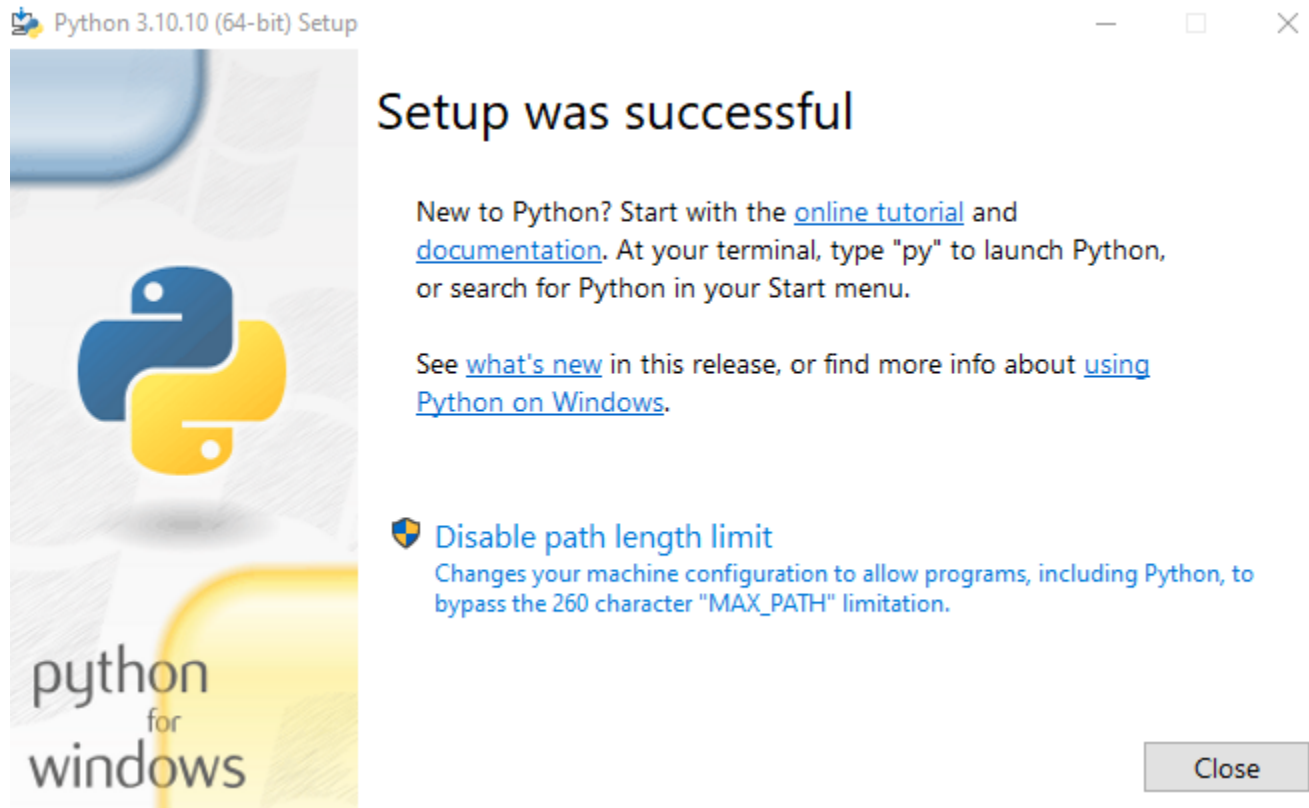


Select the options that suit your requirements:

- **Install for all users:** recommended if you're not the only user on this computer
- **Associate files with Python:** recommended, because this option associates all the Python file types with the launcher or editor
- **Create shortcuts for installed applications:** recommended to enable shortcuts for Python applications
- **Add Python to environment variables:** recommended to enable launching Python
- **Precompile standard library:** not required, it might down the installation
- **Download debugging symbols** and **Download debug binaries:** recommended only if you plan to create C or C++ extensions

Make note of the Python installation directory in case you need to reference it later.

- Click **Install** to start the installation.
- After the installation is complete, a **Setup was successful** message displays.



Step 3 — Adding Python to the Environment Variables (optional)

Skip this step if you selected **Add Python to environment variables** during installation.

If you want to access Python through the command line but you didn't add Python to your environment variables during installation, then you can still do it manually.

Before you start, locate the Python installation directory on your system. The following directories are examples of the default directory paths:

- C:\Program Files\Python310: if you selected **Install for all users** during installation, then the directory will be system wide
- C:\Users\Sammy\AppData\Local\Programs\Python\Python310: if you didn't select **Install for all users** during installation, then the directory will be in the Windows user path

Note that the folder name will be different if you installed a different version, but will still start with Python.

- Go to **Start** and enter advanced system settings in the search bar.
- Click **View advanced system settings**.

3. In the **System Properties** dialog, click the **Advanced** tab and then click **Environment Variables**.
4. Depending on your installation:
 - If you selected **Install for all users** during installation, select **Path** from the list of **System Variables** and click **Edit**.
 - If you didn't select **Install for all users** during installation, select **Path** from the list of **User Variables** and click **Edit**.
5. Click **New** and enter the Python directory path, then click **OK** until all the dialogs are closed.

Step 4 — Verify the Python Installation

You can verify whether the Python installation is successful either through the command line or through the Integrated Development Environment (IDLE) application, if you chose to install it.

Go to **Start** and enter cmd in the search bar. Click **Command Prompt**.

Enter the following command in the command prompt:

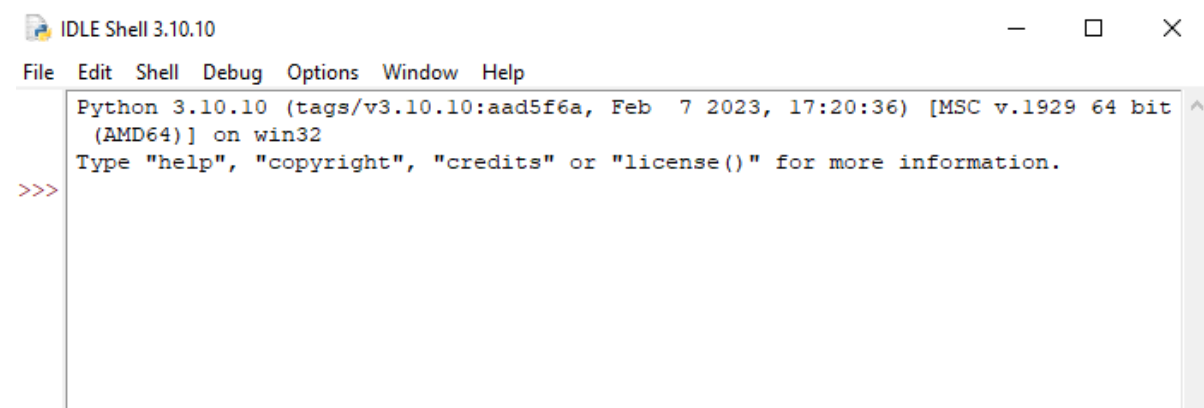
```
python --version
```

An example of the output is:

Output

```
Python 3.10.10
```

You can also check the version of Python by opening the IDLE application. Go to **Start** and enter python in the search bar and then click the IDLE app, for example **IDLE (Python 3.10 64-bit)**.



You can start coding in Python using IDLE or your preferred code editor.

1.b.) Installing Jupyter Notebook using pip:

PIP is a package management system used to install and manage software packages/libraries written in Python. These files are stored in a large “on-line repository” termed as Python Package Index (PyPI).

pip uses PyPI as the default source for packages and their dependencies.

To install Jupyter using pip, we need to first check if pip is updated in our system. Use the following command to update pip:

```
python -m pip install --upgrade pip
```

```

Command Prompt - python -m pip install --upgrade pip
C:\Users\Abhinav Singh>python -m pip install --upgrade pip
Requirement already up-to-date: pip in c:\users\abhinav singh\anaconda3\lib\site-packages (19.3.1)

```

After updating the pip version, follow the instructions provided below to install Jupyter:

- **Command to install Jupyter:**
- `python -m pip install jupyter`

Beginning Installation:

```

Command Prompt - python -m pip install jupyter
C:\Users\Abhinav Singh>python -m pip install jupyter
Collecting jupyter
  Using cached https://files.pythonhosted.org/packages/83/df/0f5dd132200728a86190397e1ea87cd76244e42d39ec5e88efd25b2abd7e/jupyter-1.0.0-py2.py3-none-any.whl
Collecting ipykernel
  Using cached https://files.pythonhosted.org/packages/e1/92/8fec943b5b81078399f96f00557804d884c96fcd0bc296e81a2ed4fd270/ipykernel-5.1.3-py3-none-any.whl
Collecting jupyter-console
  Using cached https://files.pythonhosted.org/packages/0a/89/742fa5a80b552ffcb6a8922712697c6e6828aee7b91ee4ae2b79f00f8401/jupyter_console-6.1.0-py2.py3-none-any.whl
Collecting nbconvert
  Using cached https://files.pythonhosted.org/packages/79/6c/05a569e9f703d18aacb09b7ad6075b404e8a4afde2c26b73ca77bb644b1

```

Downloading Files and Data:

```

Command Prompt - python -m pip install jupyter
  Using cached https://files.pythonhosted.org/packages/e9/97/55e575a5b49e5c3df9eb3c116c61021d7badf556c816be13bbd7ba55234/jedi-0.15.2-py2.py3-none-any.whl
Collecting colorama; sys_platform == "win32"
  Using cached https://files.pythonhosted.org/packages/c9/dc/45cdef1b4d119eb96316b3117e6d5708a08029992b2fee2c143c7a0a5cc5/colorama-0.4.3-py2.py3-none-any.whl
Requirement already satisfied: setuptools>=18.5 in c:\users\abhinav singh\anaconda3\lib\site-packages (from ipython>=5.0.0->ipykernel->jupyter) (44.0.0.post20200106)
Collecting pickleshare
  Using cached https://files.pythonhosted.org/packages/9a/41/220f49aaea88bc6fa6cba8d05ecf2467326156c23b991e80b3f2fc24c77/pickleshare-0.7.5-py2.py3-none-any.whl
Collecting wcwidth
  Downloading https://files.pythonhosted.org/packages/58/b4/4850a0ccc6f567cc0ebe7060d20ff4258b8210efadc259da62dc6ed9c65/wcwidth-0.1.8-py2.py3-none-any.whl
Requirement already satisfied: jsonschema>=2.5.0,>=2.4 in c:\users\abhinav singh\anaconda3\lib\site-packages (from nbformat>=4.4->nbconvert->jupyter) (3.2.0)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\abhinav singh\anaconda3\lib\site-packages (from Jinja2>=2.4->nbconvert->jupyter) (1.1.1)
Collecting webencodings
  Using cached https://files.pythonhosted.org/packages/f4/24/2a3e3df732393fed8b3ebf2ec078f05546de641fe1b667ee316ec1dcf3b7/webencodings-0.5.1-py2.py3-none-any.whl
Collecting pywinpty>=0.5; os_name == "nt"
  Downloading https://files.pythonhosted.org/packages/7b/de/c69772738f10140d531b46b7462f1dccb4175987daaa851a8cda2326251/pywinpty-0.5.7-cp37-cp37m-win_amd64.whl (1.3MB)
    | 1.3MB 547kB/s
Collecting parso>=0.5.2
  Downloading https://files.pythonhosted.org/packages/9b/b0/90353a5e0987279837835224dead0c424833a224195683e188d384e06b/parso-0.5.2-py2.py3-none-any.whl (99kB)

```


• Installing Packages:

```

Command Prompt - python -m pip install jupyter

Requirement already satisfied: more-iter-tools in c:\users\abhinav singh\anaconda3\lib\site-packages (from zipp>=0.5->importlib-metadata; python_version < "3.8")
Requirement already satisfied: jsonschema<2.5.0,>=2.4->nbformat>=4.4->nbconvert>jupyter (8.0.2)
Building wheels for collected packages: pandocfilters, prometheus-client, backcall
Building wheel for pandocfilters (setup.py) ... done
Created wheel for pandocfilters: filename=pandocfilters-1.4.2-cp37-none-any.whl size=7862 sha256=849bce8e4908d819b25c81ed408862aad99063021d407852b57cbfb02e7f881c
Stored in directory: C:\Users\Abhinav Singh\AppData\Local\pip\Cache\wheels\39\01\56\fb08a6275acc59e846fa4c1e1b65dbc19f20157d9e66c20
Building wheel for prometheus-client (setup.py) ... done
Created wheel for prometheus-client: filename=prometheus_client-0.7.1-cp37-none-any.whl size=41407 sha256=11607fb79180270892bf9c160976b5ce32d01287090efaf28ff792339b158d
Stored in directory: C:\Users\Abhinav Singh\AppData\Local\pip\Cache\wheels\1c\54\34\fd47cd9b308826cc4292b54449c1899a30251ef3b506bc91ea
Building wheel for backcall (setup.py) ... done
Created wheel for backcall: filename=backcall-0.1.0-cp37-none-any.whl size=10418 sha256=76f4f1869e8c47685c7023872dca8fb094cd44119b1a4324023c65399ff1925e
Stored in directory: C:\Users\Abhinav Singh\AppData\Local\pip\Cache\wheels\98\b0\dd\29e28ff615af3dda4c67cab719dd51357597eabff926976b45
Successfully built pandocfilters prometheus-client backcall
Installing collected packages: pyzmq, tornado, jupyter-client, wcwidth, prompt-toolkit, backcall, parso, jedi, colorama, pygments, pickleshare, ipython, ipykernel, jupyter-console, pandocfilters, entrypoints, defusedxml, testpath, webencodings, bleach, mistune, nbconvert, qtconsole, Send2Trash, pywinpty, terminado, prometheus-client, notebook, widgetsnbextension, ipywidgets, Jupyter

```

• Finished Installation:

```

Command Prompt

Building wheel for pandocfilters (setup.py) ... done
Created wheel for pandocfilters: filename=pandocfilters-1.4.2-cp37-none-any.whl size=7862 sha256=849bce8e4908d819b25c81ed408862aad99063021d407852b57cbfb02e7f881c
Stored in directory: C:\Users\Abhinav Singh\AppData\Local\pip\Cache\wheels\39\01\56\fb08a6275acc59e846fa4c1e1b65dbc19f20157d9e66c20
Building wheel for prometheus-client (setup.py) ... done
Created wheel for prometheus-client: filename=prometheus_client-0.7.1-cp37-none-any.whl size=41407 sha256=11607fb79180270892bf9c160976b5ce32d01287090efaf28ff792339b158d
Stored in directory: C:\Users\Abhinav Singh\AppData\Local\pip\Cache\wheels\1c\54\34\fd47cd9b308826cc4292b54449c1899a30251ef3b506bc91ea
Building wheel for backcall (setup.py) ... done
Created wheel for backcall: filename=backcall-0.1.0-cp37-none-any.whl size=10418 sha256=76f4f1869e8c47685c7023872dca8fb094cd44119b1a4324023c65399ff1925e
Stored in directory: C:\Users\Abhinav Singh\AppData\Local\pip\Cache\wheels\98\b0\dd\29e28ff615af3dda4c67cab719dd51357597eabff926976b45
Successfully built pandocfilters prometheus-client backcall
Installing collected packages: pyzmq, tornado, jupyter-client, wcwidth, prompt-toolkit, backcall, parso, jedi, colorama, pygments, pickleshare, ipython, ipykernel, jupyter-console, pandocfilters, entrypoints, defusedxml, testpath, webencodings, bleach, mistune, nbconvert, qtconsole, Send2Trash, pywinpty, terminado, prometheus-client, notebook, widgetsnbextension, ipywidgets, Jupyter
Successfully installed Send2Trash-1.5.0 backcall-0.1.0 bleach-3.1.0 colorama-0.4.3 defusedxml-0.6.0 entrypoints-0.3 ipykernel-5.1.3 ipython-7.11.1 ipywidgets-7.5.1 jedi-0.15.2 jupyter-1.0.0 jupyter-client-5.3.4 jupyter-console-6.1.0 mistune-0.8.4 nbconvert-5.6.1 notebook-6.0.2 pandocfilters-1.4.2 parso-0.5.2 pickleshare-0.7.5 prometheus-client-0.7.1 prompt-toolkit-3.0.2 pygments-2.5.2 pywinpty-0.5.7 pyzmq-18.1.1 qtconsole-4.6.0 terminado-0.8.3 testpath-0.4.4 tornado-6.0.3 wcwidth-0.1.8 webencodings-0.5.1 widgetsnbextension-3.5.1
C:\Users\Abhinav Singh>

```

Launching Jupyter:

Use the following command to launch Jupyter using command-line:
jupyter notebook

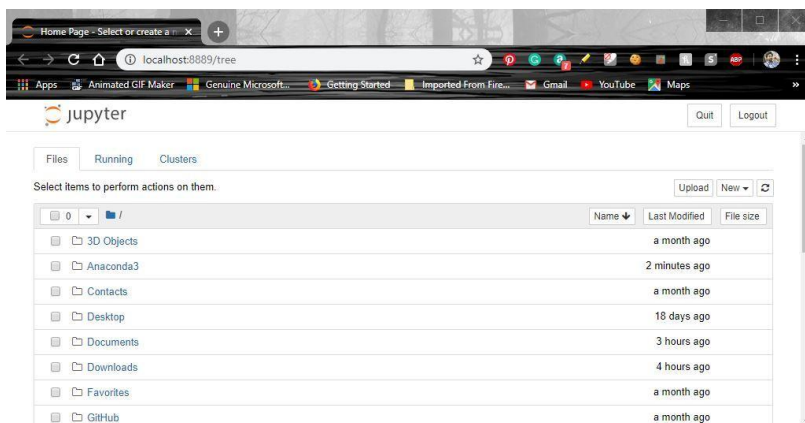
```

Command Prompt - jupyter notebook

C:\Users\Abhinav Singh>jupyter notebook
[I 17:52:47.792 NotebookApp] Serving notebooks from local directory: C:\Users\Abhinav Singh
[I 17:52:47.792 NotebookApp] The Jupyter Notebook is running at:
[I 17:52:47.792 NotebookApp] http://localhost:8888/?token=325083ca519c9570938f8b852606778d5cd7100fc5491f4d
[I 17:52:47.792 NotebookApp] or http://127.0.0.1:8888/?token=325083ca519c9570938f8b852606778d5cd7100fc5491f4d
[I 17:52:47.792 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 17:52:47.825 NotebookApp]

To access the notebook, open this file in a browser:
file:///C:/Users/Abhinav%20Singh/AppData/Roaming/jupyter/runtime/nbserver-4908-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=325083ca519c9570938f8b852606778d5cd7100fc5491f4d
or http://127.0.0.1:8888/?token=325083ca519c9570938f8b852606778d5cd7100fc5491f4d

```



1. Write a Numpy program to add a border filled with 0's around the existing array.

Sometimes we need to add a border around a NumPy matrix. Numpy provides a function known as '**numpy.pad()**' to construct the border. The below examples show how to construct a border of '**0**' around the **identity matrix**.

Syntax :

```
numpy.pad(array, pad_width, mode='constant', **kwargs)
```

Construct a border of 0s around 2D identity matrix

```
import numpy as np
```

```
# Creating a 2X2 Numpy matrix
```

```
array = np.ones((2, 2))
```

```
print("Original array")
```

```
print(array)
```

```
print("\n0 on the border and 1 inside the array")
```

```
# constructing border of 0 around 2D identity matrix using np.pad()
```

```
array = np.pad(array, pad_width=1, mode='constant', constant_values=0)
```

```
print(array)
```

Output:

we construct a border of 0s around the 2-D NumPy matrix.

```
Original array
```

```
[[1. 1.]
 [1. 1.]]
```

```
0 on the border and 1 inside the array
```

```
[[0. 0. 0. 0.]
 [0. 1. 1. 0.]
 [0. 1. 1. 0.]
 [0. 0. 0. 0.]]
```


2. Write a Numpy program to get the unique elements of an array.

finding unique elements from the array we are using **numpy.unique()** function of NumPy library.

Syntax: *np.unique(Array)*

Return: *Return the unique of an array.*

```
# import library

import numpy as np

# create 1d-array

arr = np.array([3, 3, 4, 5, 6, 5, 6, 4])

# find unique element

# from a array

rslt = np.unique(arr)

print(rslt)
```

Output:

```
[3 4 5 6]
```

3. Write a Numpy program to get the values and indices of the elements that are bigger than 10 in a given array.

```
import numpy as np

x = np.array([[0, 10, 20], [20, 30, 40]])

print("Original array: ")

print(x)

print("Values bigger than 10 =", x[x>10])

print("Their indices are ", np.nonzero(x > 10))
```

Output:

Original array:

[[0 10 20]

[20 30 40]]

Values bigger than 10 = [20 20 30 40]

Their indices are (array([0, 1, 1, 1]), array([2, 0, 1, 2]))