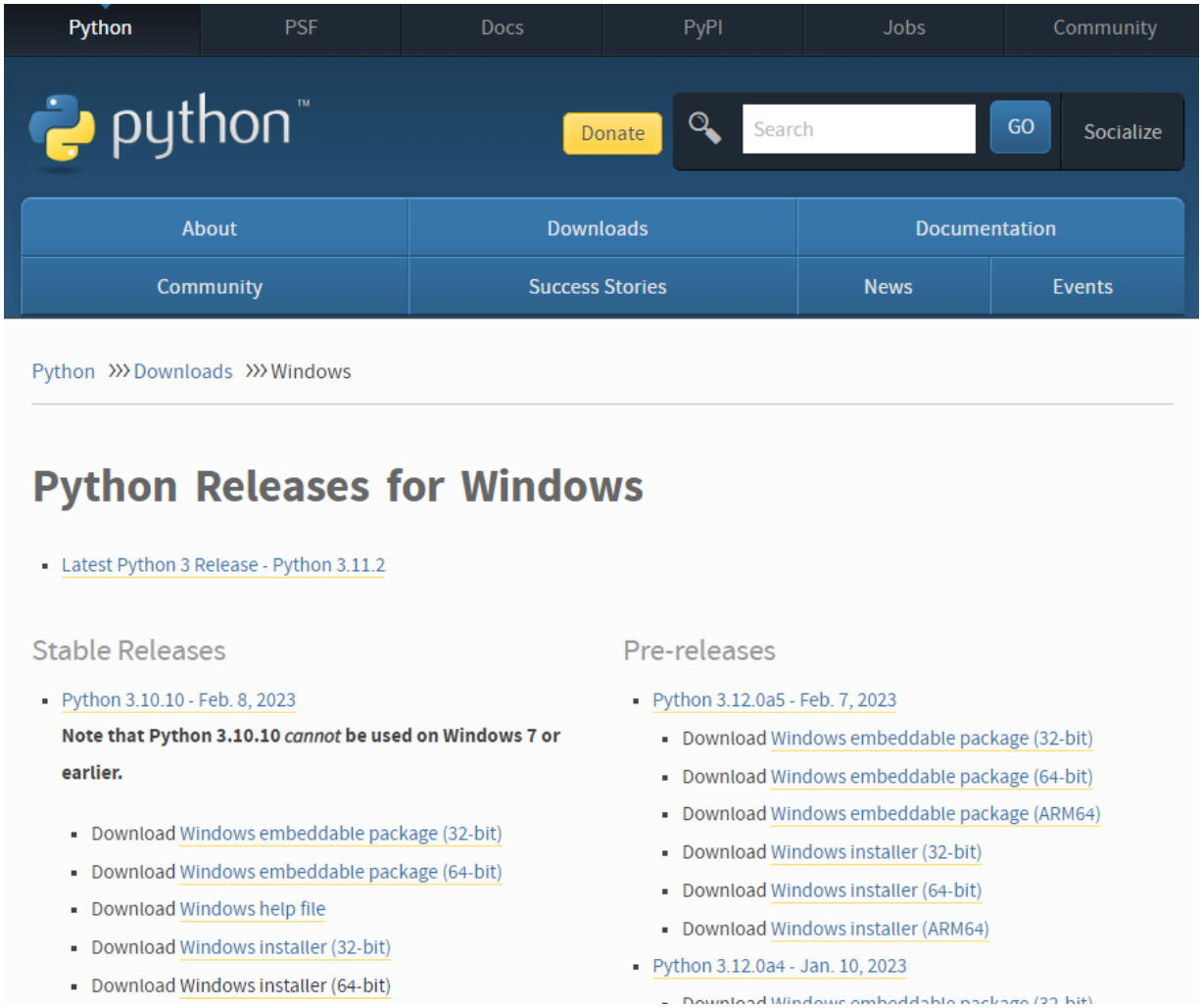
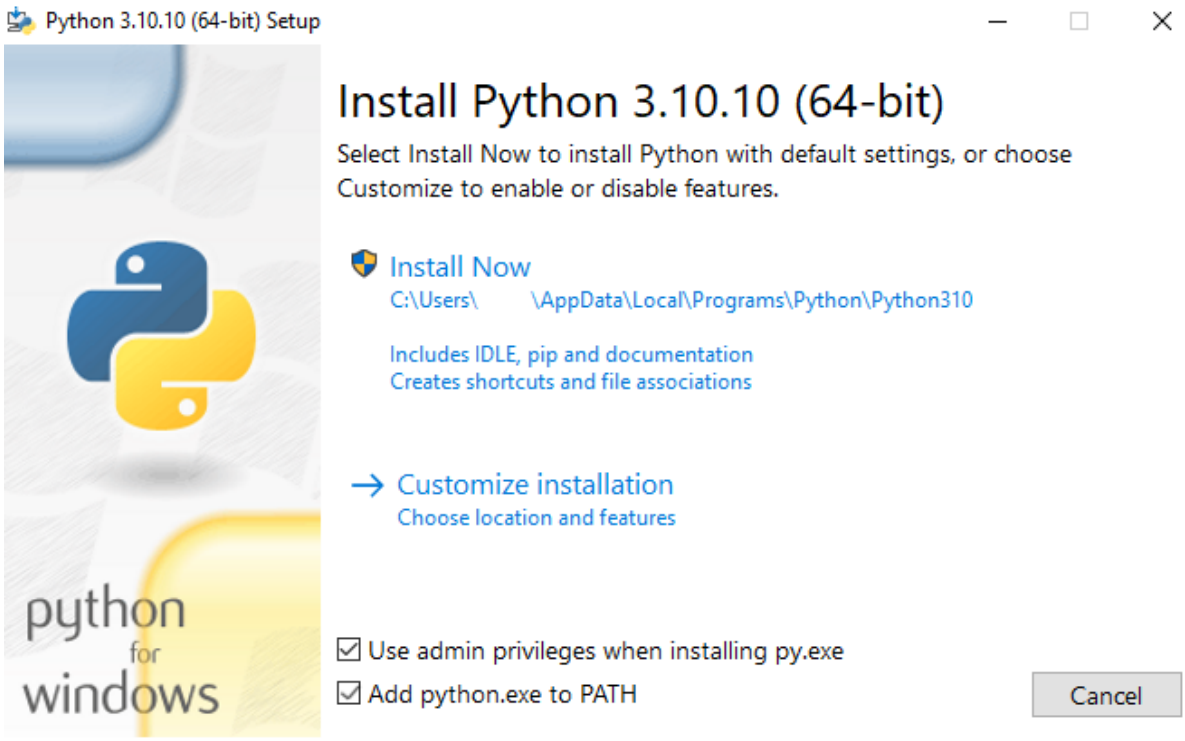
Step 1 — Downloading the Python Installer

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)**.

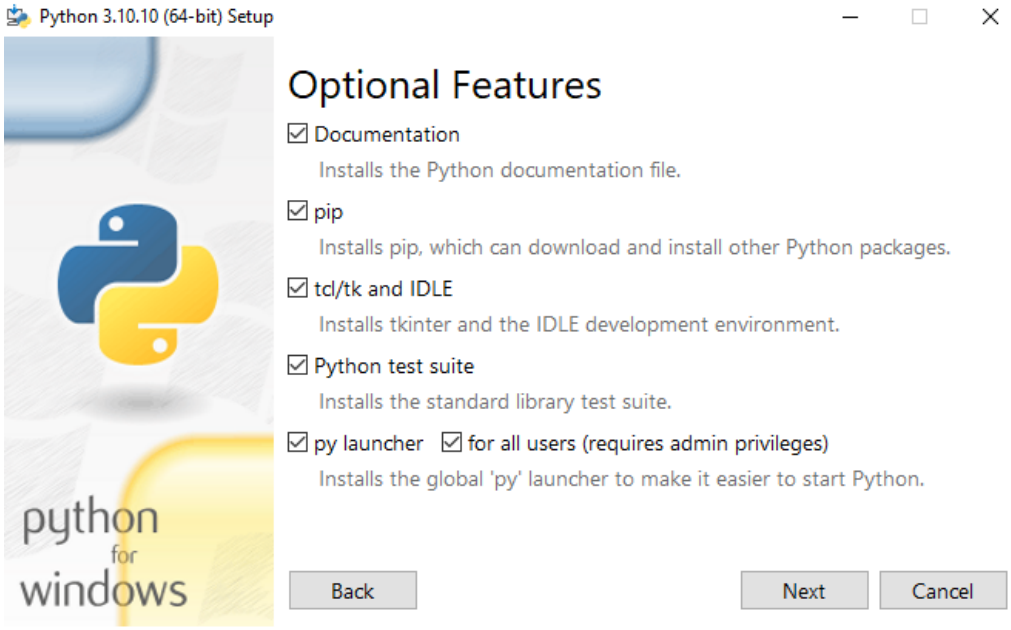


Step 2 — Running the Executable Installer

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.



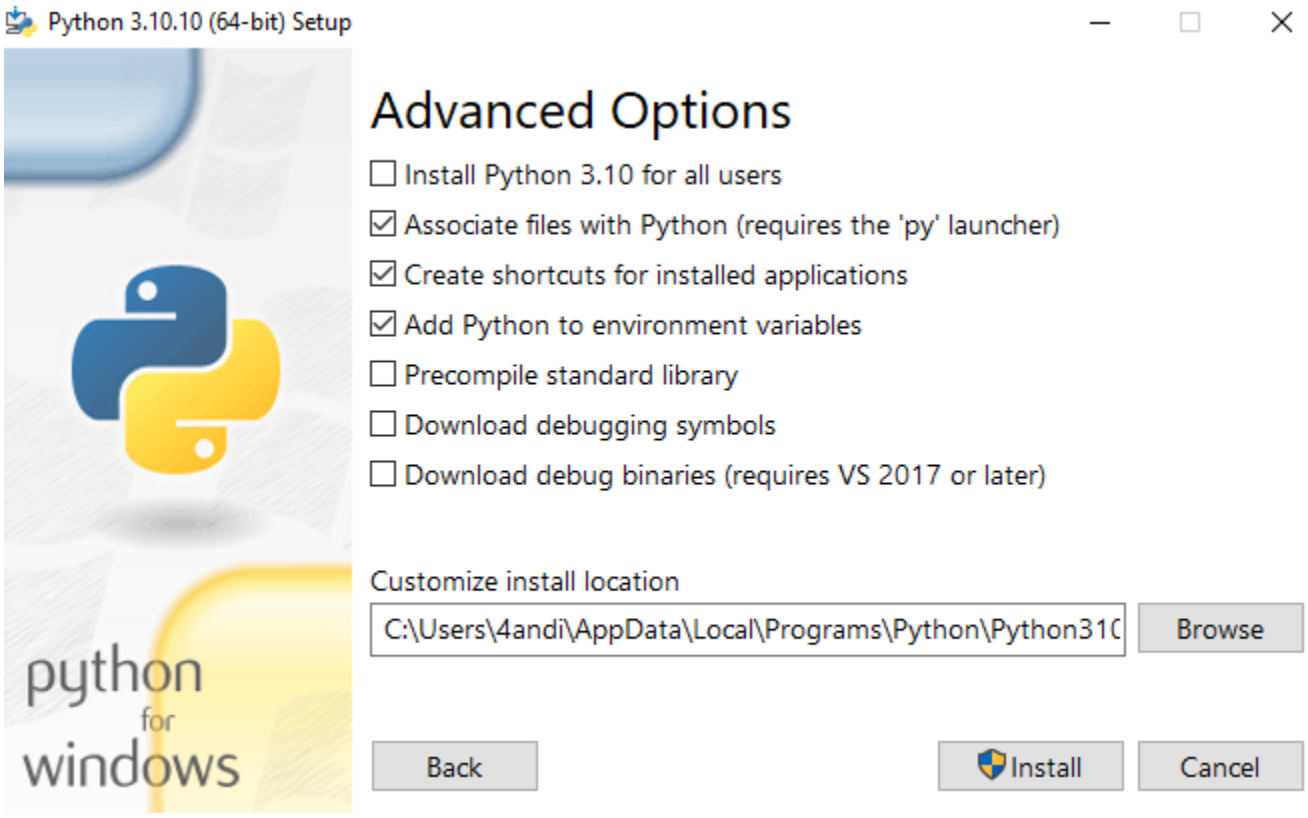
4. If you're just getting started with Python and you want to install it with default features as described in the dialog, then click **Install Now** and go to Step 4 - Verify the Python Installation. To install other optional and advanced features, click **Customize installation** and continue.
5. The **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.

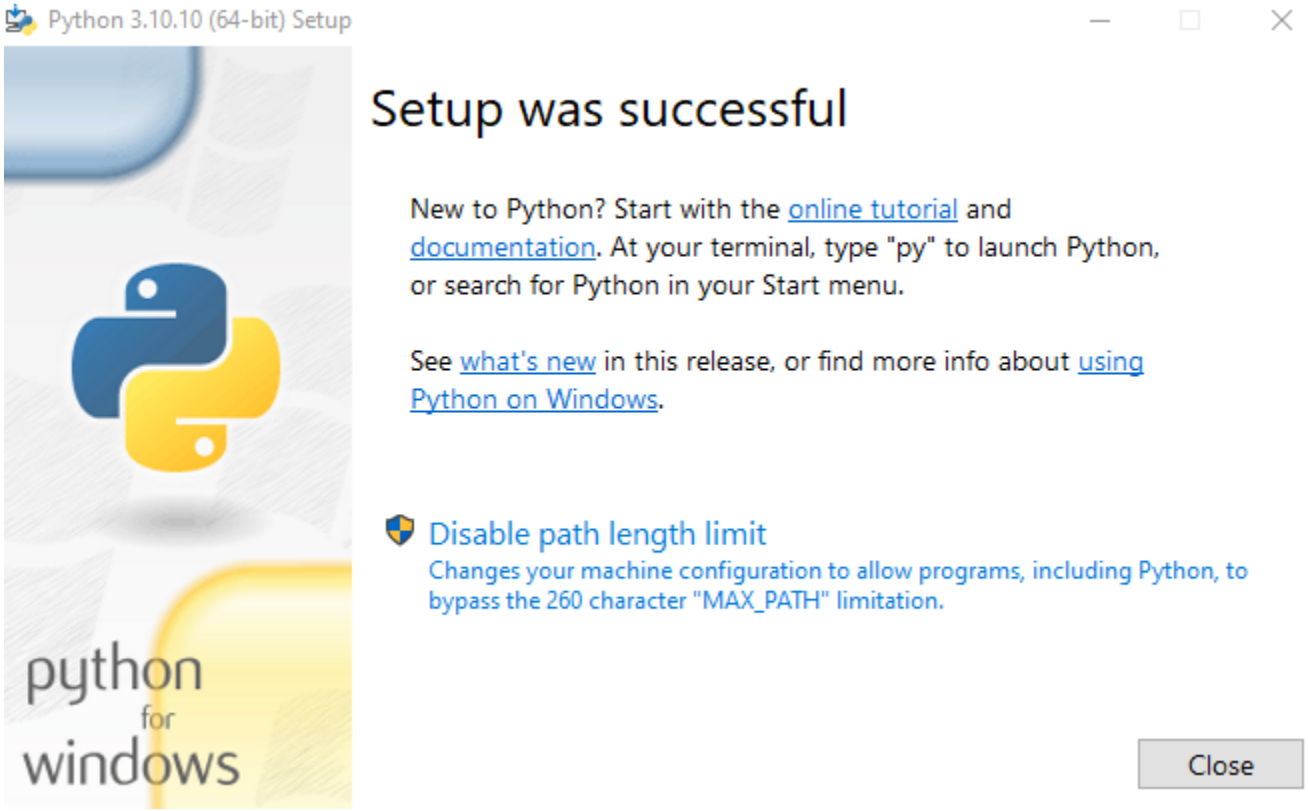


Select the options that suit your requirements:

- **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.

8. Click **Install** to start the installation.
9. 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.

1. Go to **Start** and enter advanced system settings in the search bar.
2. 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:

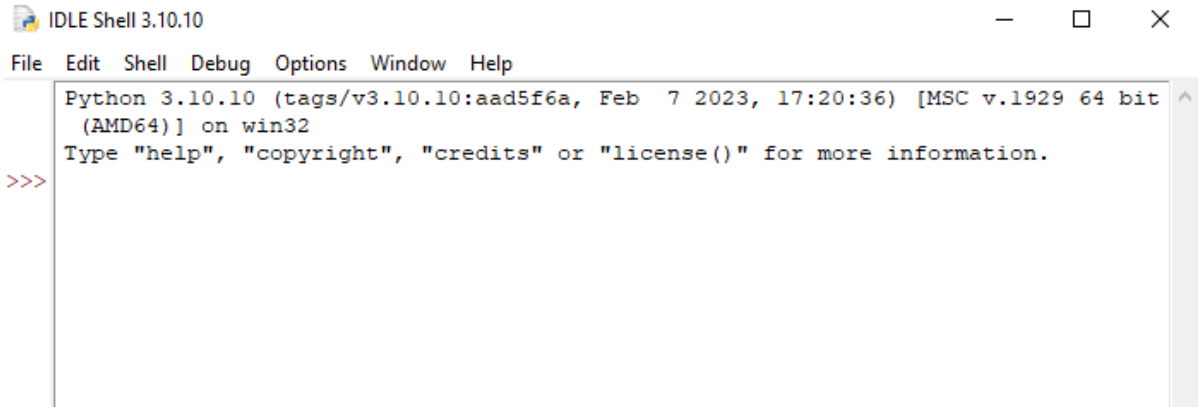
```
python --version
```

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)**.

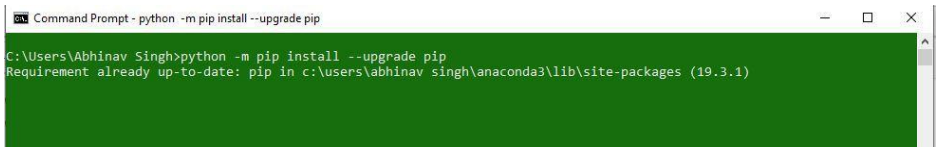


You can start coding in Python using IDLE or your preferred code editor.

1.b.) Installing Jupyter Notebook using pip:

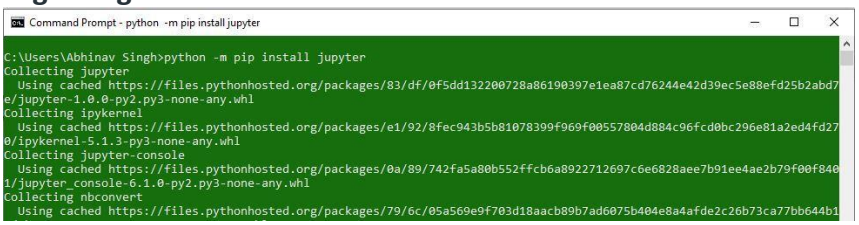
PIP is a package management system used to install and manage software packages/libraries written in Python. These files are stored in a large “on-line repository” termed as Python Package Index (PyPI).
pip uses PyPI as the default source for packages and their dependencies.
To install Jupyter using 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

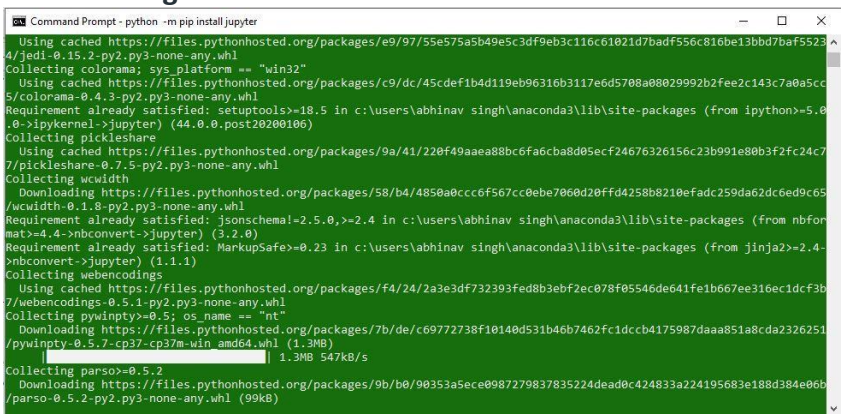


After updating the pip version, follow the instructions provided below to install Jupyter:

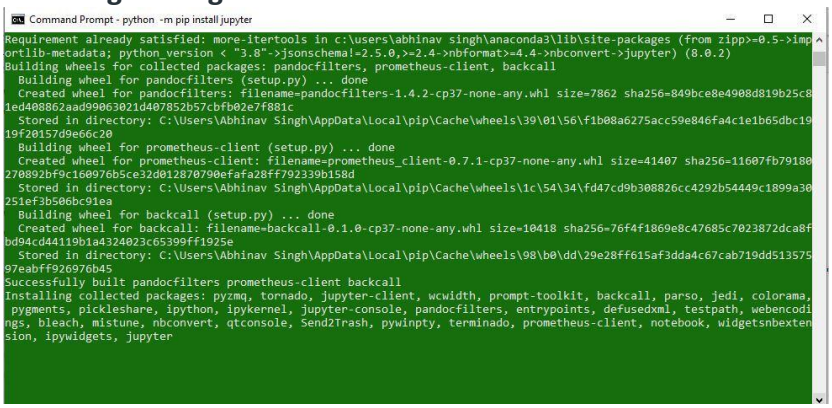
- **Command to install Jupyter:**
- python -m pip install jupyter
- **Beginning Installation:**



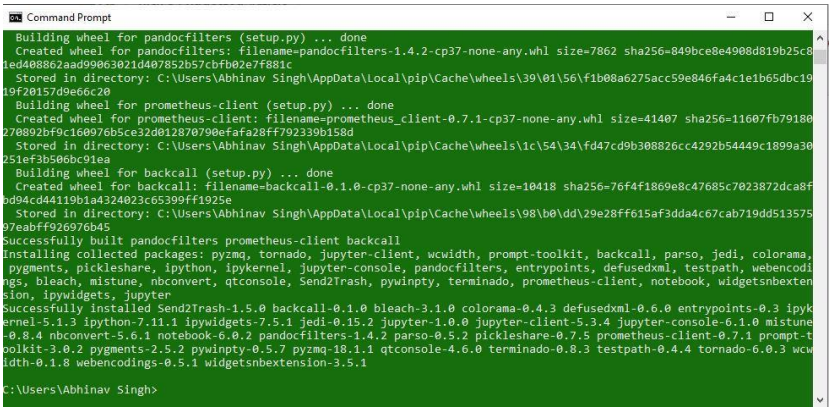
- **Downloading Files and Data:**



- **Installing Packages:**

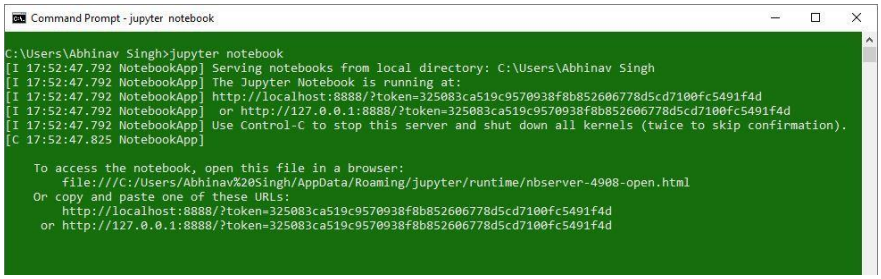


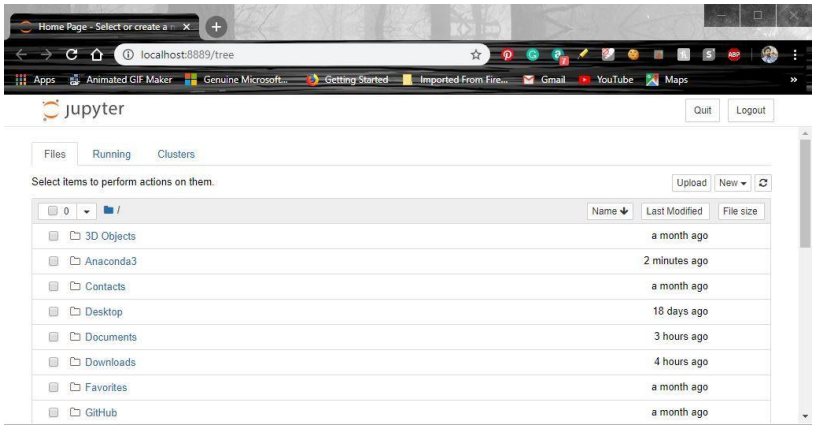
- **Finished Installation:**



Launching Jupyter:

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





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 than10 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]))
```