Work with External Libraries Using pip and Create Virtual Environments

Table of Contents

Introduction

In Python, external libraries are essential for extending functionality. The `pip` tool allows you to install these libraries efficiently. Virtual environments help isolate projects, ensuring that dependencies are managed separately for each project.

Problem Statement

Learn how to install external Python libraries using `pip`, create virtual environments, and manage dependencies for Python projects.

Prerequisites

Software Requirement

- **Python 3.13.0**
  Download Python

- **Code Editor**
  A text editor or IDE like **Visual Studio Code (VS Code)** is recommended.
  Download VS Code

Hardware Requirement

- **Processor**: Minimum dual-core processor.
- **RAM**: 4GB or more.
- **Storage**: 1GB of free space for Python and external libraries.

Implementation Steps

Why Do We Need Virtual Environments?

Virtual environments are essential for Python projects to:

1. **Isolate Project Dependencies**: Each project can have its own versions of libraries, preventing conflicts.
2. **Avoid Global Installation Issues**: Libraries installed globally might interfere with each other. Virtual environments keep them separate.
3. **Reproducibility**: Ensures that a project uses the exact same versions of libraries, making it easier to reproduce results.

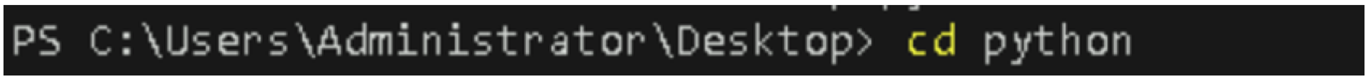
Without a virtual environment, you could face version conflicts between projects and difficulty in managing dependencies. For instance, two projects may require different versions of the same library. Virtual environments solve this problem by keeping project-specific dependencies isolated.

Creating a Virtual Environment

Step 1: Navigate to Your Project Directory

Open the terminal and navigate to your project folder using `cd` (change directory) commands:

```
cd path/to/your/project
```



```
PS C:\Users\Administrator\Desktop> cd python
```

Step 2: Create the Virtual Environment

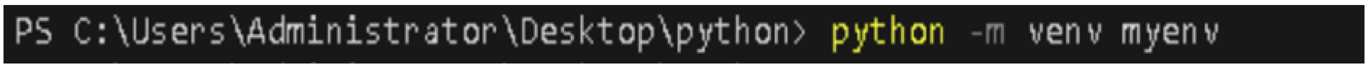
Once in your project directory, run the following command to create a virtual environment:

```
python -m venv venv_name
```

- `venv_name`: The name of your virtual environment (e.g., `myenv`).

For example:

```
python -m venv myenv
```

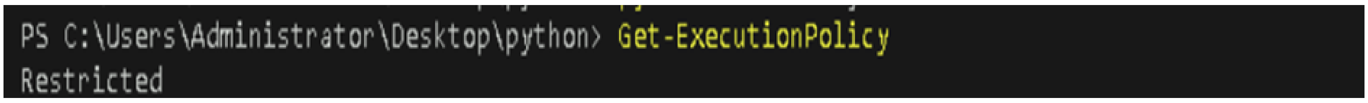


```
PS C:\Users\Administrator\Desktop\python> python -m venv myenv
```

Here is the step-by-step process to do this in VSCode:

1. Open PowerShell in VSCode:
   - Open your project in VSCode.
   - Open a new terminal in VSCode (View > Terminal or Ctrl+`).
   - Eurnse the terminal is set to PowerShell.
2. Check the current execution policy:

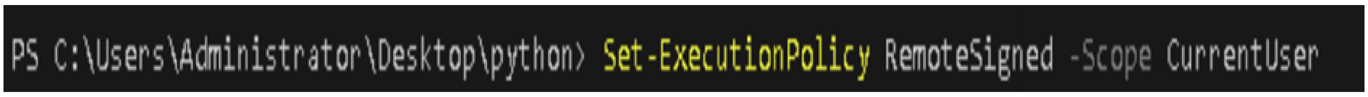
```
Get-ExecutionPolicy
```



```
PS C:\Users\Administrator\Desktop\python> Get-ExecutionPolicy
Restricted
```

3. Change the execution policy for the current user:

```
Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
```



```
PS C:\Users\Administrator\Desktop\python> Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
```

OR

```
Set-ExecutionPolicy Unrestricted -Scope CurrentUser
```

Step 3:Activate the Virtual Environment

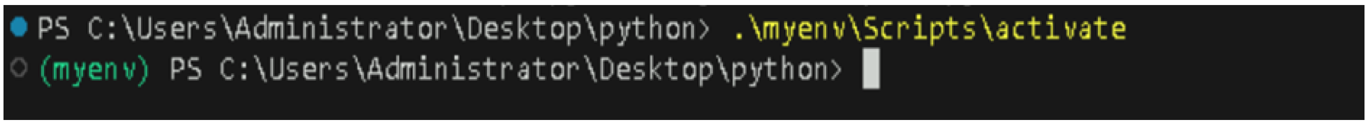
After creating the virtual environment, activate it:

- On **Windows**:

```
myenv\Scripts\activate
```

After activation, you'll notice that the command prompt changes to include the virtual environment name, like this:

```
(myenv) C:\path\to\your\project>
```

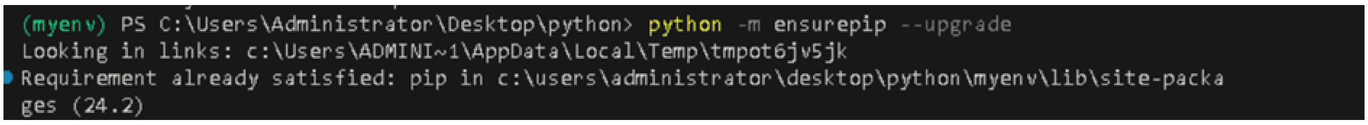


```
● PS C:\Users\Administrator\Desktop\python> .\myenv\Scripts\activate
○ (myenv) PS C:\Users\Administrator\Desktop\python> 
```

Installing pip

`pip` is the package manager for Python, and it comes pre-installed with Python 3.13.0 . You can check if `pip` is installed and up-to-date by running the following command inside your virtual environment:

```
python -m ensurepip --upgrade
```



```
(myenv) PS C:\Users\Administrator\Desktop\python> python -m ensurepip --upgrade
Looking in links: c:\Users\ADMINI~1\AppData\Local\Temp\tmpot6jv5jk
Requirement already satisfied: pip in c:\users\administrator\desktop\python\myenv\lib\site-packa
ges (24.2)
```

Now you're ready to install external libraries using `pip`!

Installing Libraries Inside the Virtual Environment

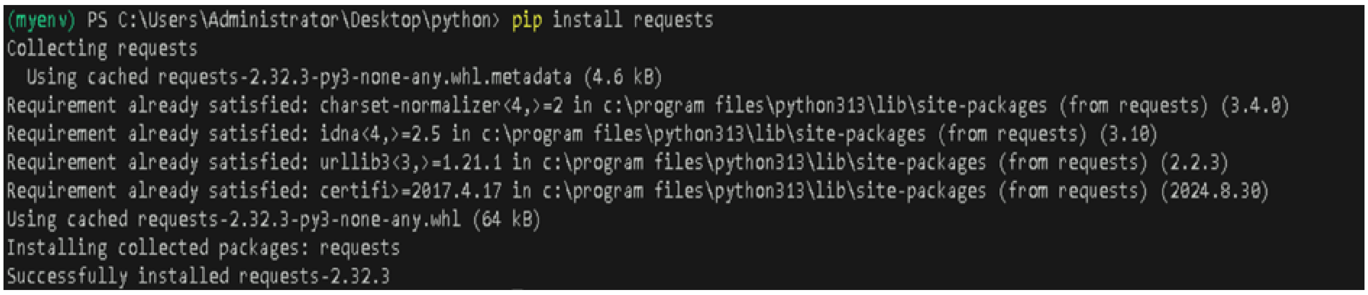
Once the virtual environment is activated, you can install libraries using `pip`. Let's install three common Python libraries:

- `requests` – For making HTTP requests.
- `matplotlib` – For plotting and data visualization.
- `pandas` – For data manipulation and analysis.

Install the Libraries

1. **Install `requests`:**

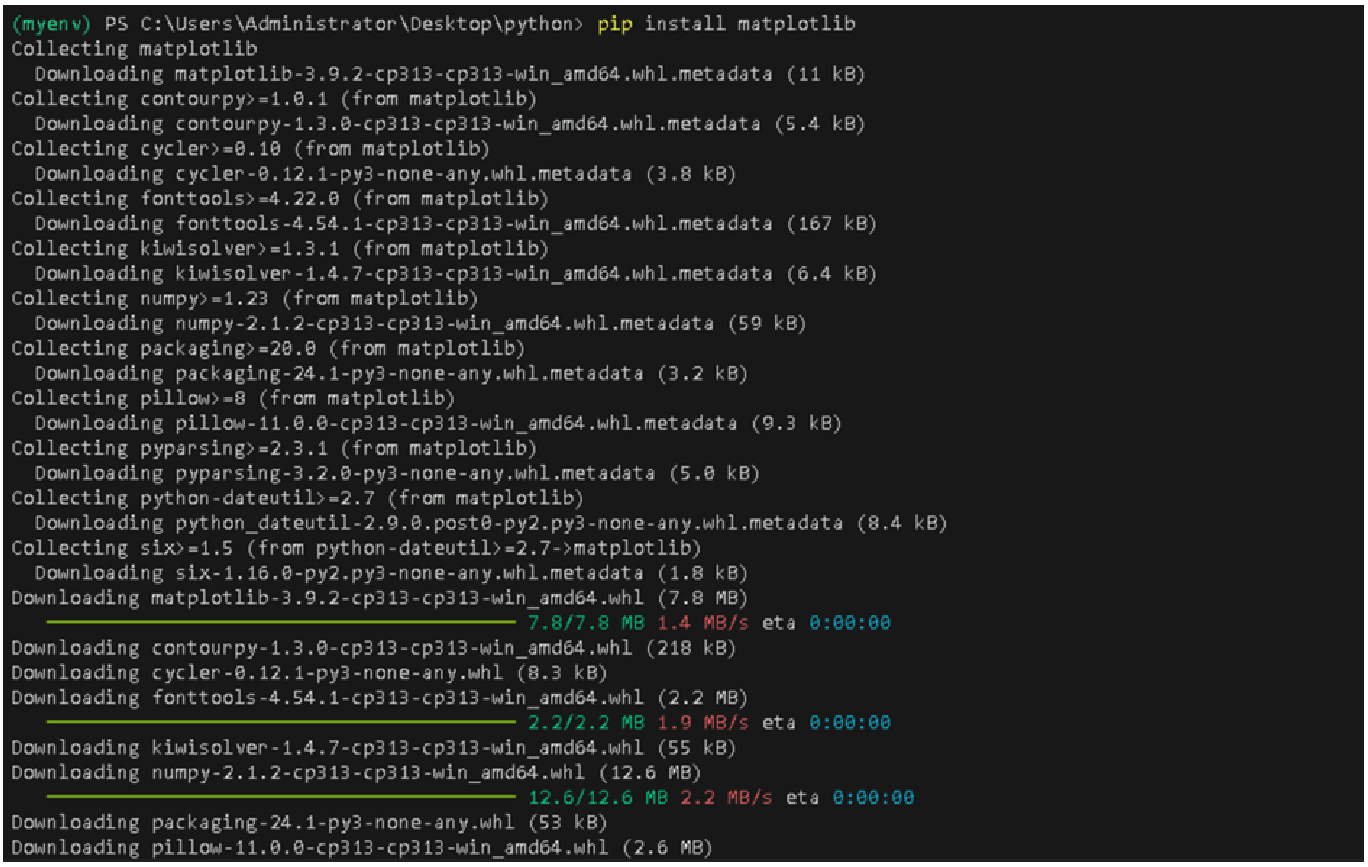
```
pip install requests
```



```
(myenv) PS C:\Users\Administrator\Desktop\python> pip install requests
Collecting requests
  Using cached requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\program files\python313\lib\site-packages (from requests) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in c:\program files\python313\lib\site-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\program files\python313\lib\site-packages (from requests) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\program files\python313\lib\site-packages (from requests) (2024.8.30)
Using cached requests-2.32.3-py3-none-any.whl (64 kB)
Installing collected packages: requests
Successfully installed requests-2.32.3
```

2. **Install `matplotlib`:**

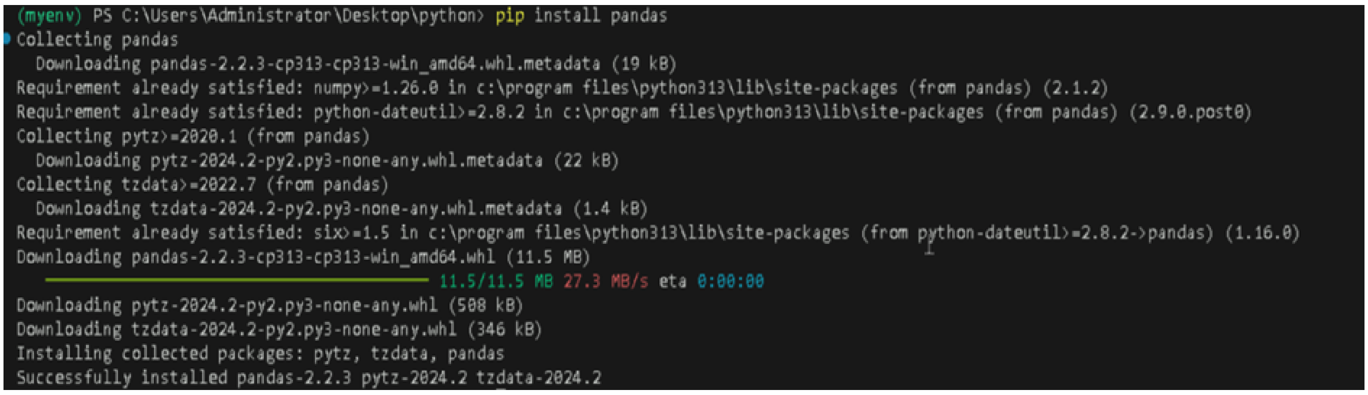
```
pip install matplotlib
```



```
(myenv) PS C:\Users\Administrator\Desktop\python> pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.9.2-cp313-cp313-win_amd64.whl.metadata (11 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.0-cp313-cp313-win_amd64.whl.metadata (5.4 kB)
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Downloading fonttools-4.54.1-cp313-cp313-win_amd64.whl.metadata (167 kB)
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Downloading kiwisolver-1.4.7-cp313-cp313-win_amd64.whl.metadata (6.4 kB)
Collecting numpy>=1.23 (from matplotlib)
  Downloading numpy-2.1.2-cp313-cp313-win_amd64.whl.metadata (59 kB)
Collecting packaging>=20.0 (from matplotlib)
  Downloading packaging-24.1-py3-none-any.whl.metadata (3.2 kB)
Collecting pillow>=8 (from matplotlib)
  Downloading pillow-11.0.0-cp313-cp313-win_amd64.whl.metadata (9.3 kB)
Collecting pyparsing>=2.3.1 (from matplotlib)
  Downloading pyparsing-3.2.0-py3-none-any.whl.metadata (5.0 kB)
Collecting python-dateutil>=2.7 (from matplotlib)
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
Collecting six>=1.5 (from python-dateutil>=2.7->matplotlib)
  Downloading six-1.16.0-py2.py3-none-any.whl.metadata (1.8 kB)
Downloading matplotlib-3.9.2-cp313-cp313-win_amd64.whl (7.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 1.4 MB/s eta 0:00:00
Downloading contourpy-1.3.0-cp313-cp313-win_amd64.whl (218 kB)
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.54.1-cp313-cp313-win_amd64.whl (2.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.2/2.2 MB 1.9 MB/s eta 0:00:00
Downloading kiwisolver-1.4.7-cp313-cp313-win_amd64.whl (55 kB)
Downloading numpy-2.1.2-cp313-cp313-win_amd64.whl (12.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.6/12.6 MB 2.2 MB/s eta 0:00:00
Downloading packaging-24.1-py3-none-any.whl (53 kB)
Downloading pillow-11.0.0-cp313-cp313-win_amd64.whl (2.6 MB)
```

3. **Install pandas:**

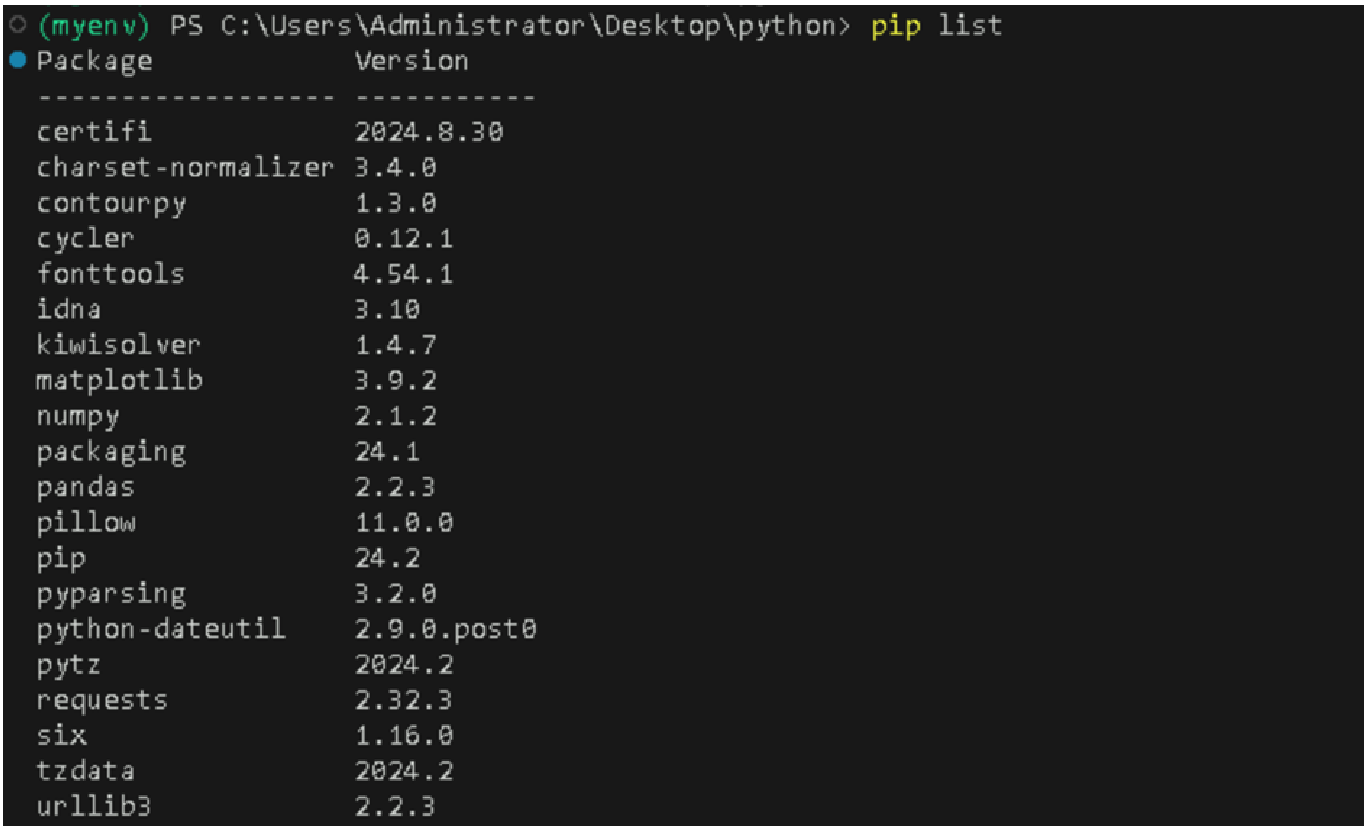
```
pip install pandas
```



```
(myenv) PS C:\Users\Administrator\Desktop\python> pip install pandas
Collecting pandas
  Downloading pandas-2.2.3-cp313-cp313-win_amd64.whl.metadata (19 kB)
Requirement already satisfied: numpy>=1.26.0 in c:\program files\python313\lib\site-packages (from pandas) (2.1.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\program files\python313\lib\site-packages (from pandas) (2.9.0.post0)
Collecting pytz>=2020.1 (from pandas)
  Downloading pytz-2024.2-py2.py3-none-any.whl.metadata (22 kB)
Collecting tzdata>=2022.7 (from pandas)
  Downloading tzdata-2024.2-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: six>=1.5 in c:\program files\python313\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Downloading pandas-2.2.3-cp313-cp313-win_amd64.whl (11.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.5/11.5 MB 27.3 MB/s eta 0:00:00
Downloading pytz-2024.2-py2.py3-none-any.whl (508 kB)
Downloading tzdata-2024.2-py2.py3-none-any.whl (346 kB)
Installing collected packages: pytz, tzdata, pandas
Successfully installed pandas-2.2.3 pytz-2024.2 tzdata-2024.2
```

Once installed, you can verify the libraries by listing them:

```
pip list
```



```
(myenv) PS C:\Users\Administrator\Desktop\python> pip list
Package             Version
------------------ -----------
certifi             2024.8.30
charset-normalizer  3.4.0
contourpy           1.3.0
cycler              0.12.1
fonttools           4.54.1
idna                3.10
kiwisolver          1.4.7
matplotlib          3.9.2
numpy               2.1.2
packaging           24.1
pandas              2.2.3
pillow              11.0.0
pip                 24.2
pyparsing           3.2.0
python-dateutil     2.9.0.post0
pytz                2024.2
requests            2.32.3
six                 1.16.0
tzdata              2024.2
urllib3             2.2.3
```

This command will display a list of installed libraries along with their versions.

Program 1: Using `requests` Library

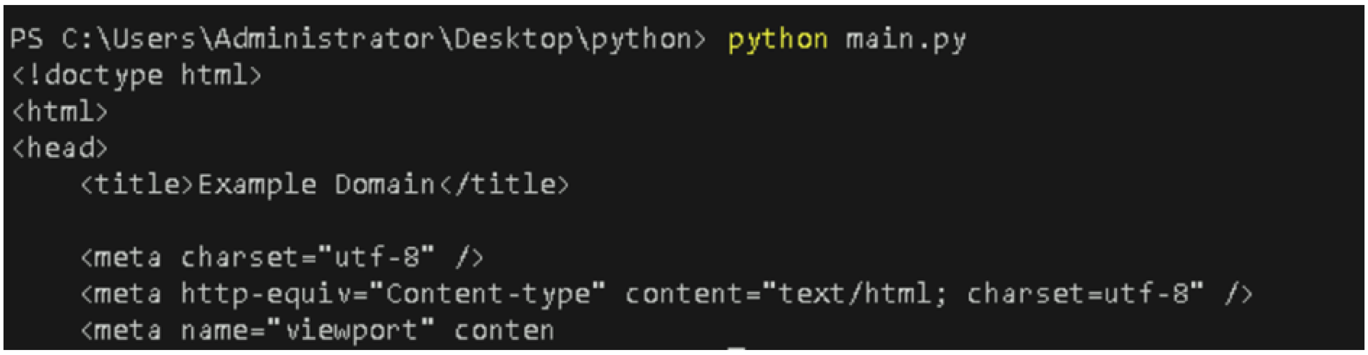
This program fetches the content of a webpage using the `requests` library.

```python
import requests

def fetch_webpage(url):
    """Fetch and return the content of a webpage."""
    response = requests.get(url)
    return response.text

url = "https://www.example.com"
content = fetch_webpage(url)
print(content[:200])  # Print first 200 characters of the webpage content
```

Output:



```
PS C:\Users\Administrator\Desktop\python> python main.py
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" conten
```

The program uses the `requests.get()` method to fetch the content from a URL.

Program 2: Using `matplotlib` Library

This program creates a simple plot using the `matplotlib` library.

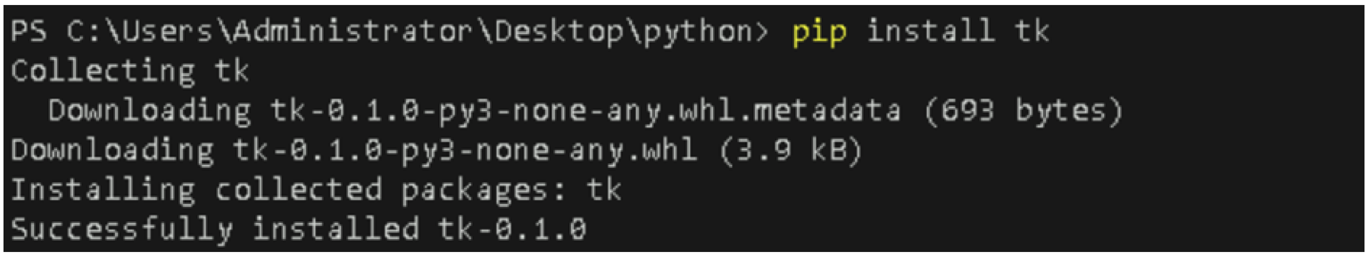
```python
import matplotlib.pyplot as plt

def plot_graph():
    """Plot a simple line graph."""
    x = [1, 2, 3, 4, 5]
    y = [2, 3, 5, 7, 11]
    plt.plot(x, y)
    plt.title('Simple Line Graph')
    plt.xlabel('X-axis')
    plt.ylabel('Y-axis')
    plt.show()

plot_graph()
```

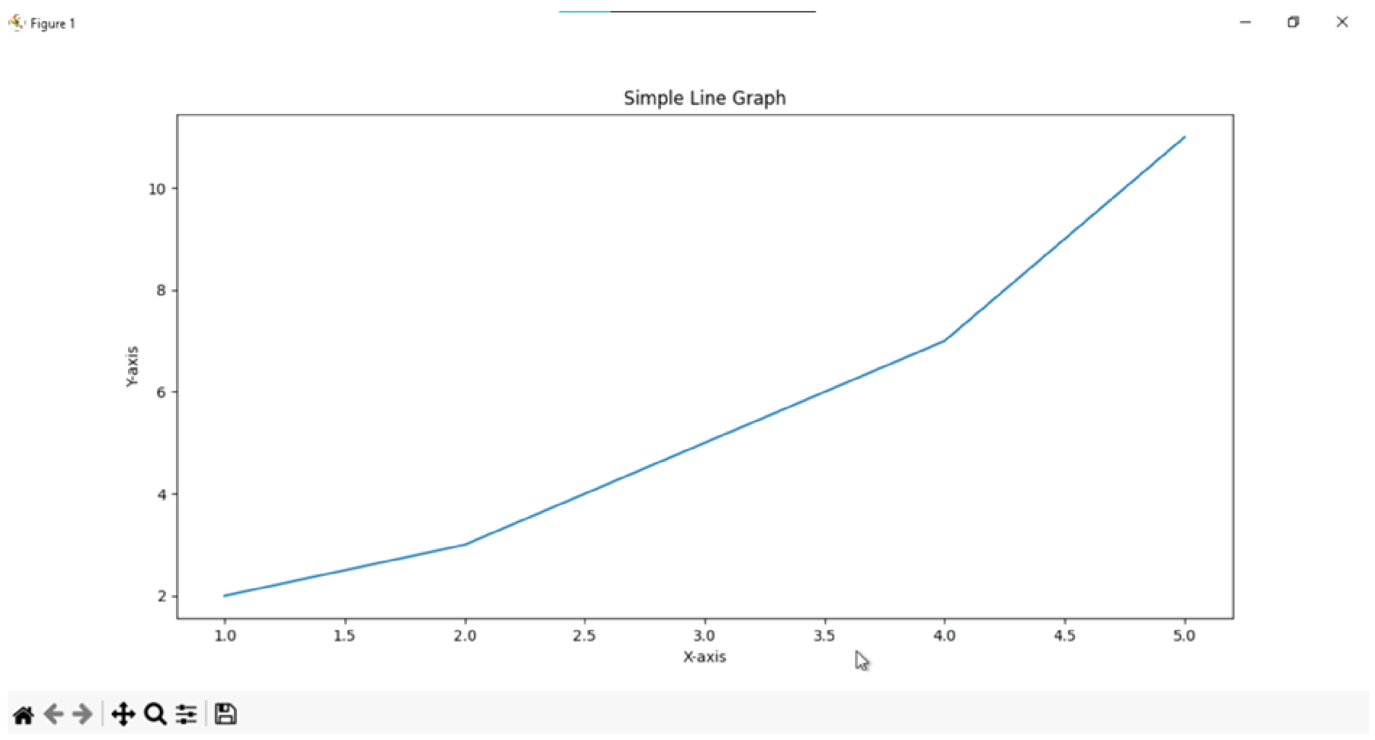
Note:

```
pip install tk
```



```
PS C:\Users\Administrator\Desktop\python> pip install tk
Collecting tk
  Downloading tk-0.1.0-py3-none-any.whl.metadata (693 bytes)
Downloading tk-0.1.0-py3-none-any.whl (3.9 kB)
Installing collected packages: tk
Successfully installed tk-0.1.0
```

Output:



A simple line graph will be displayed with the points plotted for x and y.

Program 3: Using pandas Library

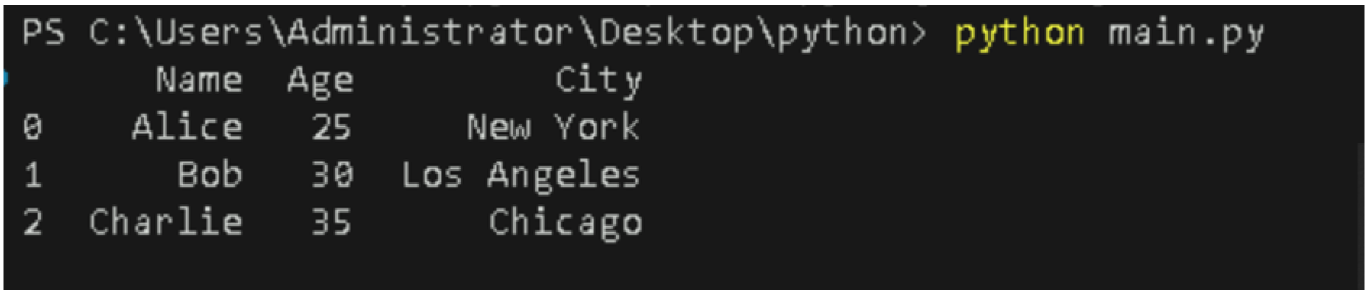
This program demonstrates how to create and manipulate a DataFrame using the pandas library.

```python
import pandas as pd

def create_dataframe():
    """Create and return a pandas DataFrame."""
    data = {
        'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35],
        'City': ['New York', 'Los Angeles', 'Chicago']
    }
    df = pd.DataFrame(data)
    return df

df = create_dataframe()
print(df)
```

Output:



```
PS C:\Users\Administrator\Desktop\python> python main.py
      Name  Age         City
0    Alice   25     New York
1      Bob   30  Los Angeles
2  Charlie   35      Chicago
```

The program creates a simple DataFrame using the `pandas` library and prints it in tabular format.

References

- Python pip Documentation
- Virtual Environments in Python
- Requests Library Documentation
- Matplotlib Documentation
- Pandas Documentation
