



**MAPUA UNIVERSITY**

**SCHOOL OF ELECTRICAL, ELECTRONICS, AND COMPUTER ENGINEERING**

## **Lab 6: Web Scraping and Data Visualization**

CPE106L (Software Design Laboratory)

Hannah Mae Antaran

Darrel Tristan Virtusio

Kathleen Joy Tupas

Group: 01

Section: E01



# PreLab

---

## Readings, Insights and Reflection

**SysNucleus. (2018). WebHarvy Web Scraper. Retrieved August 06, 2020, from <https://www.webharvy.com/articles/what-is-web-scraping.html>.**

Web scraping, also known as Screen Scraping, Web Data Extraction or Web Harvesting, is a process of extracting large data from websites and saving it in computer or a database with formats such as CSV. Web scraping allows analyzing, retrieving, and using data in any way the user wants as it simplifies and speeds up obtaining data. For some websites, data can only be viewed and cannot be copied. With web scraping, these kinds of data are still obtained as it extracts data by just a click of a button and can be saved in one's computer. Web scraping has two methods where it can be done using software or writing code. Through web scraping software, data can be saved automatically in one's computer or can be run through Cloud via browsers. Examples of web scraping software are OutWit Hub and WaveHarby. For Cloud web scraping, an example is Mozenda. In writing a code for web scraping, custom data extractions are built to obtain data.

Every brand wants to have a clean and positive online sentiment to improve the chances that customers will choose to purchase their solution instead of their competitors. Web scraping can be used to monitor forums, reviews on e-commerce websites and social media channels for mentions of a brand name to better understand the current voice of the customers. This provides the opportunity to quickly identify and triage any negative comments to mitigate any damage to brand awareness or affinity.

**Data Visualization: What it is and why we use it. (2020). Retrieved August 06, 2020, from <https://www.microstrategy.com/us/resources/introductory-guides/data-visualization-what-it-is-and-why-we-use-it>**

Data visualization refers to techniques used to communicate insights from data through visual representation. Its main goal is to distill large datasets into visual graphics to allow for easy understanding of complex relationships within the data. One data visualization used in Python is Matplotlib. It can be used in multiple ways in Python, including Python scripts, the Python and iPython shells, Jupyter Notebooks. Matplotlib supports all the popular charts (lots, histograms, power spectra, bar charts, error charts, scatterplots, etc.) right out of the box. There are also extensions that can be used to create advanced visualizations like 3-Dimensional plots, etc. Another data visualization in Python is Seaborn. One useful feature of Seaborn is that it supports a plethora of advanced plots like categorical plotting (catplot), distribution plotting using kde (distplot), swarm plot, etc. right out of the box.

Companies who can gather and quickly act on their data will be more competitive in the marketplace because they can make informed decisions sooner than the competition. Speed is key, and data visualization aides in the understanding of vast quantities of data by applying visual representations to the data. This visualization layer typically sits on top of a data warehouse or data lake and allows users to discover and explore data in a self-service manner. Not only does this spur creativity, but it reduces the need for IT to allocate resources to continually build new models.

# InLab

---

- **Objectives**

1. To explore web scraping in Python.
2. To demonstrate basic web scraping using the BeautifulSoup library.
3. To understand data visualization with matplotlib library.
4. To visualize datasets using different kinds of graphs.

- **Tools Used**

1. Visual Studio Code
2. Anaconda Prompt
3. Spyder

- **Procedure**

## PART 1: Web Scraping

```
#
# To activate this environment, use
#
#     $ conda activate Lab6
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) C:\Users\dtvir>conda activate Lab6

(Lab6) C:\Users\dtvir>
```

**Figure 1.1** Anaconda virtual environment.

For this experiment, we used Anaconda Prompt to create a virtual environment named Lab6. We activated the environment using the command `conda activate Lab6` so that we can perform the experiment on the said virtual environment.

```
Anaconda Prompt (Anaconda3)

(Lab6) C:\Users\dtvir>pip install requests
Collecting requests
  Using cached requests-2.24.0-py2.py3-none-any.whl (61 kB)
Collecting idna<3,>=2.5
  Using cached idna-2.10-py2.py3-none-any.whl (58 kB)
Collecting chardet<4,>=3.0.2
  Using cached chardet-3.0.4-py2.py3-none-any.whl (133 kB)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\dtvir\anaconda3\en
) (2020.6.20)
Collecting urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1
  Using cached urllib3-1.25.10-py2.py3-none-any.whl (127 kB)
Installing collected packages: idna, chardet, urllib3, requests
Successfully installed chardet-3.0.4 idna-2.10 requests-2.24.0 urllib3-1.25.10

(Lab6) C:\Users\dtvir>pip install beautifulsoup4
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.9.1-py3-none-any.whl (115 kB)
  | 115 kB 312 kB/s
Collecting soupsieve>1.2
  Downloading soupsieve-2.0.1-py3-none-any.whl (32 kB)
Installing collected packages: soupsieve, beautifulsoup4
Successfully installed beautifulsoup4-4.9.1 soupsieve-2.0.1
```

**Figure 1.2** Installing required libraries.

Before proceeding to the web scraping experiment, we installed the necessary libraries first. These libraries are requests and BeautifulSoup. We used the command pip install to install these libraries.

```
WebScraping.py X
C: > Users > dtvir > Desktop > WebScraping.py > ...
1  from urllib.request import urlopen
2
3  html = urlopen("http://pythonscraping.com/pages/page1.html")
4  read = html.read()
5
6  print(read)
7
```

**Figure 1.3** Simple Web Scraping code.

We opened VS Code in the Anaconda Prompt and then we entered these lines of codes for web scraping. It gets the data from the website: <http://pythonscraping.com/pages/page1.html>.

```
Anaconda Prompt (Anaconda3)

(Lab6) C:\Users\dtvir\Desktop>python WebScraping.py
b'<html>\n<head>\n<title>A Useful Page</title>\n</head>\n<body>\n<h1>An Interesting Title</h1>\n<div>\nLorem ipsum dolor
sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad m
inim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in
reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non pro
ident, sunt in culpa qui officia deserunt mollit anim id est laborum.\n</div>\n</body>\n</html>\n'
```

**Figure 1.4** Running the Web Scraping program.

We navigated to the path where the program is saved and run it on the Anaconda Prompt. The data that can be found on the web page is printed out on the terminal.

```
WebScraping.py •
C: > Users > dtvir > Desktop > WebScraping.py > ...
1  from urllib.request import urlopen
2  from bs4 import BeautifulSoup
3  import re
4
5  html = urlopen("http://www.pythonscraping.com/pages/page3.html")
6  bsObj = BeautifulSoup(html)
7  images = bsObj.findAll(
8      "img", {"src": re.compile("\.\.\/img\/gifts\/img.*\.jpg")}
9  )
10 for image in images:
11     print(image["src"])
12
(Lab6) C:\Users\dtvir\Desktop>python WebScraping.py
WebScraping.py:6: GessedAtParserWarning: No parser was explicitly specified, so I'm using the best available HTML parse
r for this system ("html.parser"). This usually isn't a problem, but if you run this code on another system, or in a di
fferent virtual environment, it may use a different parser and behave differently.

The code that caused this warning is on line 6 of the file WebScraping.py. To get rid of this warning, pass the addition
al argument 'features="html.parser"' to the BeautifulSoup constructor.

bsObj = BeautifulSoup(html)
../img/gifts/img1.jpg
../img/gifts/img2.jpg
../img/gifts/img3.jpg
../img/gifts/img4.jpg
../img/gifts/img6.jpg
(Lab6) C:\Users\dtvir\Desktop>
```

**Figure 1.5** Using BeautifulSoup class.

In this web scraping code, we used BeautifulSoup to create an instance which can be used to acquire the img tag and src on a web page. The sample run of the program was provided below the code.



## PART 2: Data Visualization with Matplotlib

```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bi
Type "help", "copyright", "credits" or "license" for more information.
>>> import matplotlib as mlp
>>> import matplotlib.pyplot as plt
```

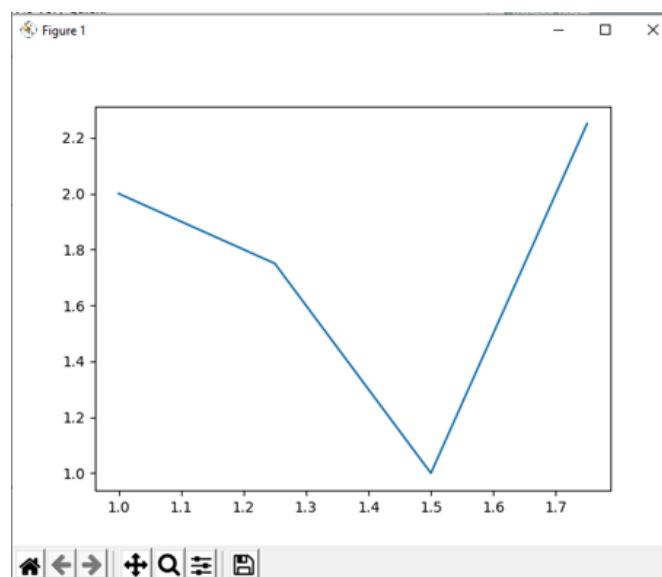
**Figure 2.1** Using matplotlib in an interactive shell.

Since matplotlib is already installed in my system, I can directly import it on my shell. If matplotlib is not yet installed, it can be installed using the command `pip install matplotlib`.

```
C:\Users\dtvir>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [M
Type "help", "copyright", "credits" or "license" for more in
>>> import matplotlib as mlp
>>> import matplotlib.pyplot as plt
>>>
>>> x = [1, 1.25, 1.50, 1.75]
>>> y = [2, 1.75, 1, 2.25]
>>>
>>> plt.plot(x, y)
[<matplotlib.lines.Line2D object at 0x0031CDF0>]
>>> plt.show
<function show at 0x0CA7EA48>
>>> plt.show()
```

**Figure 2.2** Sample commands in matplotlib.

In this part, we created an array of integers named `x` and `y`. We plotted them using `plt.plot()` command and displayed the graph using `plt.show()`. The sample graph is shown below.



**Figure 2.3** Sample run of the matplotlib commands.

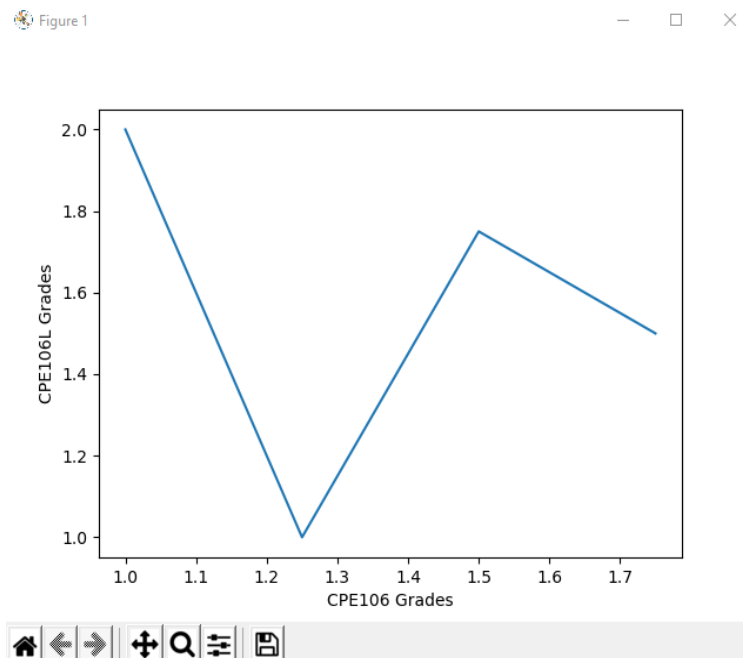
```

C:\Users\dtvir>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 b
Type "help", "copyright", "credits" or "license" for more information.
>>> import matplotlib as mlp
>>> import matplotlib.pyplot as plt
>>>
>>> x = [1, 1.25, 1.50, 1.75]
>>> y = [2, 1, 1.75, 1.50]
>>>
>>> plt.plot(x, y)
[<matplotlib.lines.Line2D object at 0x0069DDF0>]
>>> plt.xlabel('CPE106 Grades')
Text(0.5, 0, 'CPE106 Grades')
>>> plt.ylabel('CPE106L Grades')
Text(0, 0.5, 'CPE106L Grades')
>>> plt.show()

```

**Figure 2.4** Sample commands in matplotlib.

In this part, we introduced a new command which is the `plt.xlabel()` and `plt.ylabel()`. It basically adds label on the x-axis and the y-axis of the graph. The sample run of the code is displayed below.



**Figure 2.4** Sample run of the script.

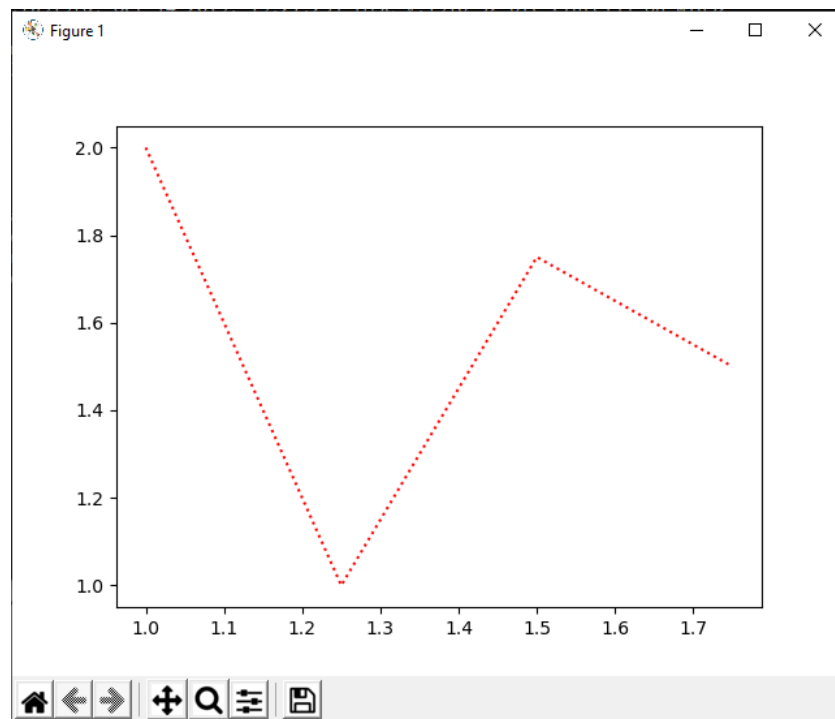
After running the created API, we obtain the CSV file of the top posts on the subreddit and we successfully used the Reddit API to scrape the data.



```
>>> plt.plot(x, y, ':r')  
[<matplotlib.lines.Line2D object at 0x110F3CD0>]  
>>> plt.show()
```

**Figure 2.5** Sample commands in matplotlib.

We used the same commands in Figure 4 and added a new parameter in the `plt.plot()` command. The third parameter controls how the graph will look like. There are several options and it can be found in the matplotlib documentation. For this example, we used `'r'` as the third parameter and it means dotted lines and in color red. The sample run is displayed below.



**Figure 2.6** Sample run of the script.

After running the created API, we obtain the CSV file of the top posts on the subreddit and we successfully used the Reddit API to scrape the data.

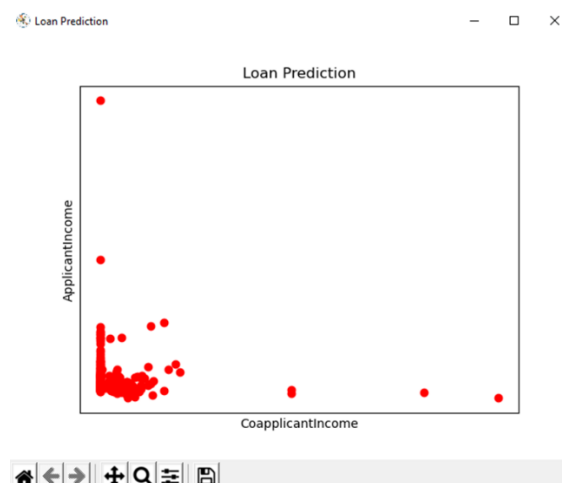
```

visual.py x
C:\Users> dtvir > Desktop > visual.py > ...
1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5
6  # Load CSV and columns
7  df = pd.read_csv("train_data.csv")
8
9  Y = df['ApplicantIncome']
10 X = df['CoapplicantIncome']
11
12
13 # Split the data into training/testing sets
14 X_train = X[:-250]
15 X_test = X[-250:]
16
17 # Split the targets into training/testing sets
18 Y_train = Y[:-250]
19 Y_test = Y[-250:]
20
21
22 # Plot outputs
23 fig = plt.gcf()
24 fig.canvas.set_window_title("Loan Prediction")
25 plt.scatter(X_test, Y_test, color='red')
26 plt.title('Loan Prediction')
27 plt.xlabel('CoapplicantIncome')
28 plt.ylabel('ApplicantIncome')
29 plt.xticks(())
30 plt.yticks(())
31
32 plt.show()

```

**Figure 2.7** Sample python script using matplotlib.

This is an example of python script that utilizes matplotlib. It uses several other libraries such as pandas and numpy. We used pandas to read into the dataset named train\_data.csv. In this script, we will use matplotlib to graph the applicant income and the co-applicant income in the dataset. It sets the window title to "Loan Prediction". It also utilizes scatter plot and set the color to red. The label of the x-axis is the co-applicant income and for the y-axis, the label is applicant income. The sample run of the code is displayed below.

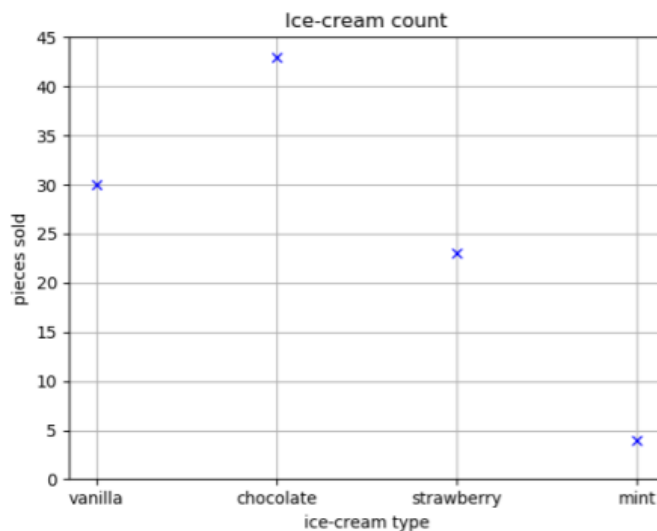


**Figure 2.8** Sample run of the python script.

# Postlab

## Practice Problem 1

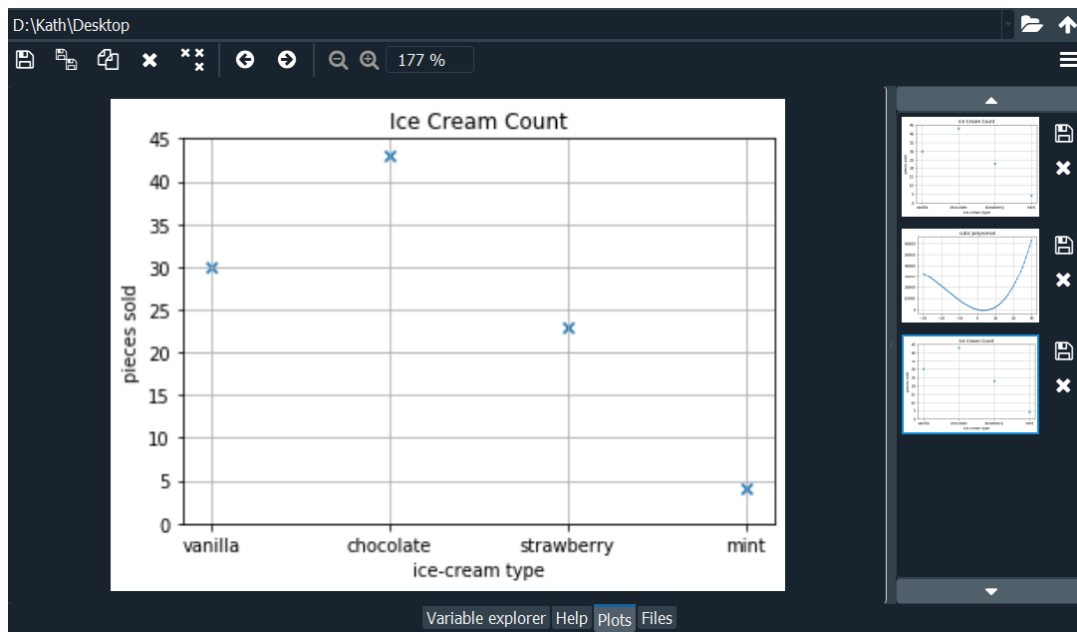
Produce the following picture:



```
untitled0.py* X
1  import matplotlib.pyplot as plt
2
3  x_axis = ["vanilla", "chocolate", "strawberry", "mint"]
4  y_axis = [30, 43, 23, 4]
5
6  plt.scatter(x_axis, y_axis, marker = "x")
7  plt.ylim(0,45)
8  plt.grid(True)
9  plt.xlabel("ice-cream type")
10 plt.ylabel("pieces sold")
11 plt.title("Ice Cream Count")
12 plt.show()
```

**Figure 3.1** Source code for problem 1.

This figure shows the source code for problem 1. As observed from the codes above, matplotlib.pyplot was used to create the following graph in the problem. The x-axis represents the ice cream flavors, vanilla, chocolate, strawberry, and mint while the y-axis represents the pieces sold for each flavor of the ice cream.



**Figure 3.2** Output for problem 1.

This figure shows the output for the source code done and as observed, the group was able to accomplish the goal of providing the same graph in problem 1.

## Practice Problem 2

Plot the cubic function  $x^3 + 53x^2 - 400x + 25$  from  $x = -30$  to  $x = 30$ , in green. Include the grid and mark each value of the function for an integer argument with a point marker. Annotate the graph as 'cubic polynomial'

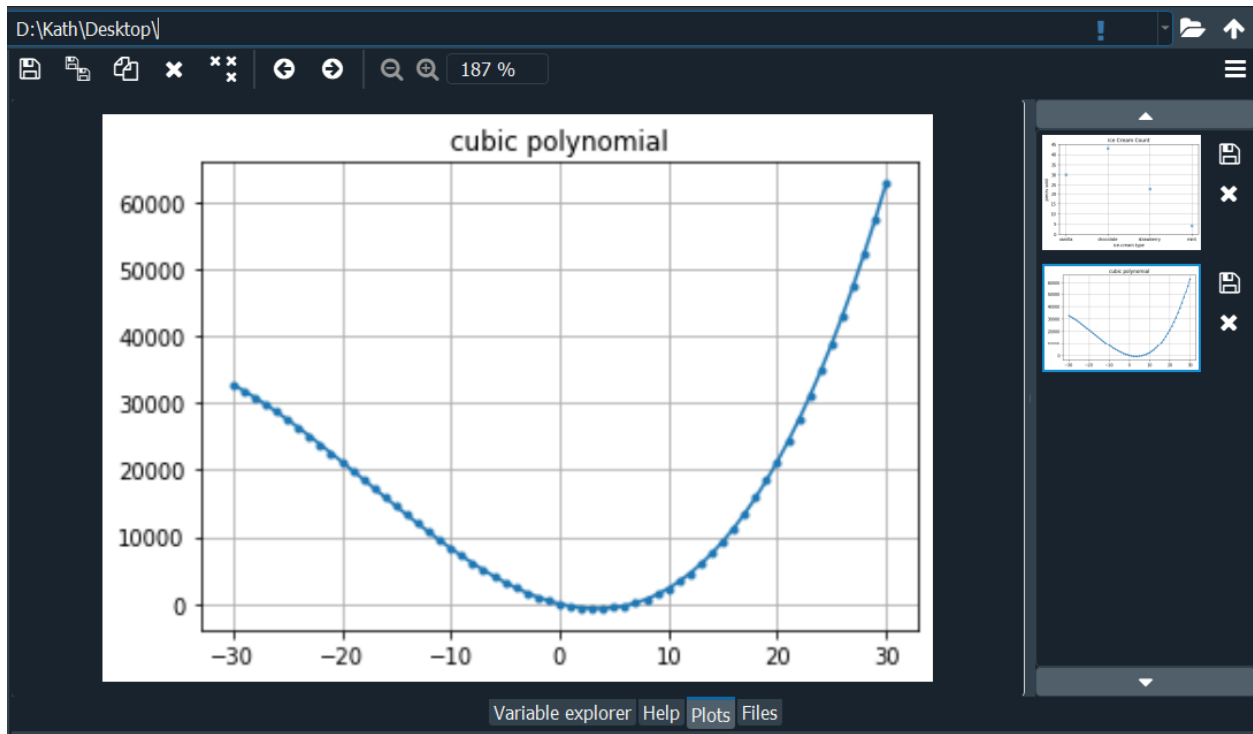
```

1  import matplotlib.pyplot as plt
2
3  def cubic(x):
4      return (x**3) + (53*(x**2)) - (400*x) + 25
5
6  x = []
7  y = []
8
9  for i in range(-30, 31):
10     x.append(i)
11     y.append(cubic(i))
12
13  plt.plot(x, y, marker = ".")
14  plt.grid(True)
15  plt.title("cubic polynomial")
16  plt.show()

```

**Figure 3.3** Source code for problem 2.

This figure shows the source code for problem 2. The def cubic module returns the value the points for the function to create the graph.



**Figure 3.4** Output for problem 2.

This figure shows the output for the source code done and as observed, the group was able to accomplish the goal of providing the graph for the given function in problem 2.

## Practice Problems 3-5

The data files for these problems were not provided, hence, the group was not able to solve it.

### OneDrive:

Group 1: <https://bit.ly/3gDtyiX>

### Github:

Darrel Virtusio: <https://bit.ly/2C6iWu9>

Hannah Antaran: <https://bit.ly/3hYjP7b>

Kathleen Tupas: <https://bit.ly/2XE3CMS>