Course: Computing for Engineers - ENGG 233

ENGG233 Final Project Milestone 2 - Calgary Weather Analyzer

Instructor: Steve Liang

Group Name: 404Error_TeamNamenotFound

Students: Nishi Solanki (email: nishi.solanki1@ucalgary.ca)

Manraj Singh (email: manraj.singh@ucalgary.ca)

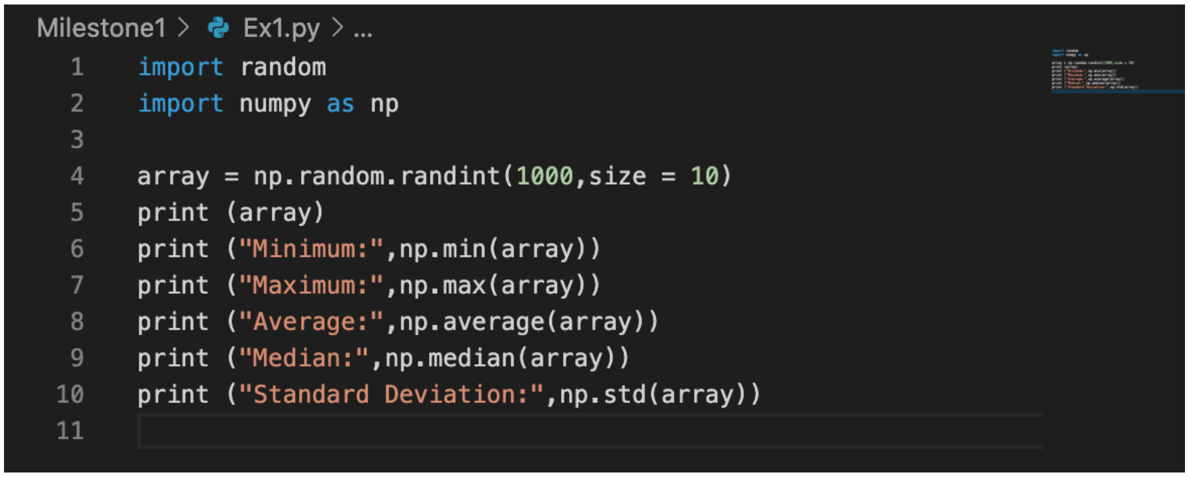
Lecture Section: L02

TA Group: VIOLET

Date submitted: December 2nd, 2020

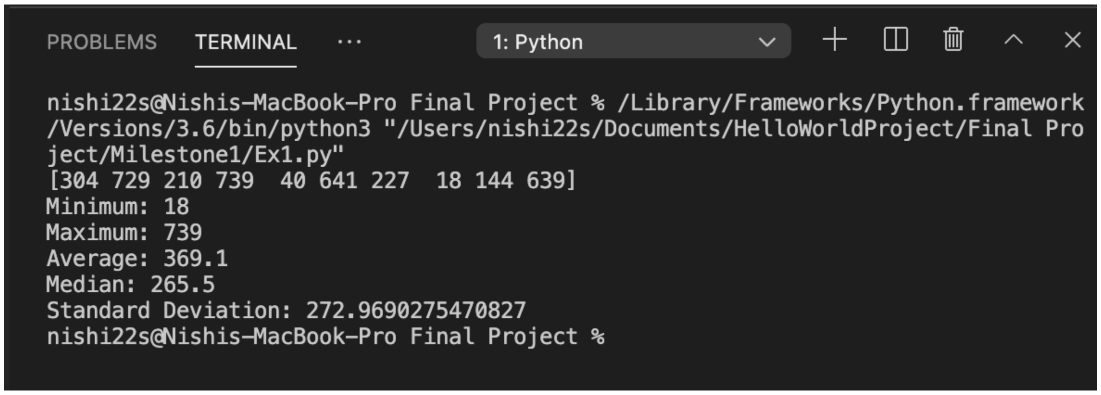
Numerical Analysis by NumPy: Assignment and Work:

Here is the code sample:

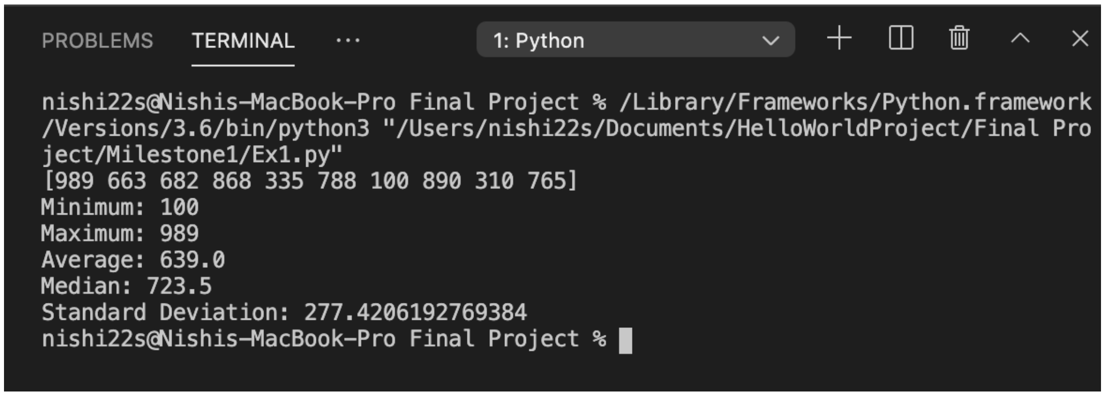


```python
import random
import numpy as np

array = np.random.randint(1000,size = 10)
print (array)
print ("Minimum:",np.min(array))
print ("Maximum:",np.max(array))
print ("Average:",np.average(array))
print ("Median:",np.median(array))
print ("Standard Deviation:",np.std(array))

```

Here are a couple samples of running the code:



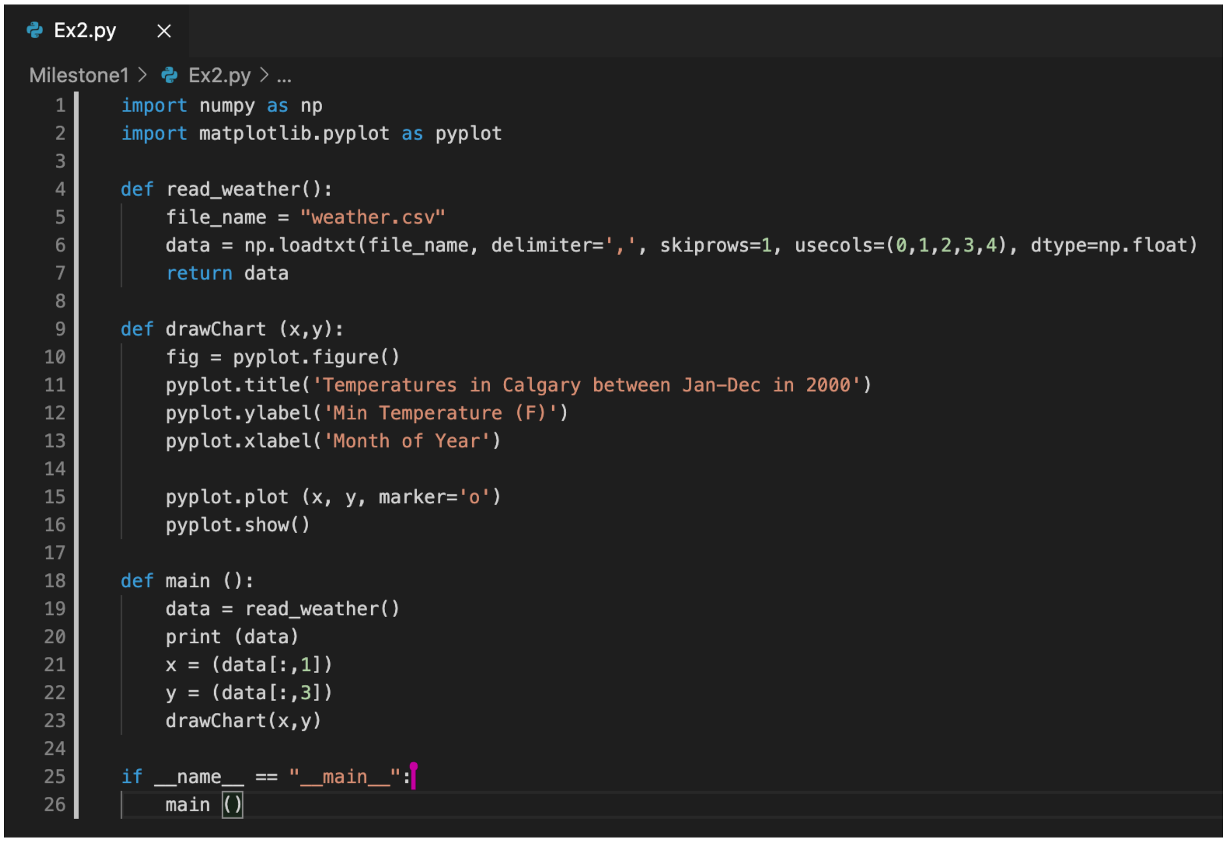
```
nishi22s@Nishis-MacBook-Pro Final Project % /Library/Frameworks/Python.framework
/Versions/3.6/bin/python3 "/Users/nishi22s/Documents/HelloWorldProject/Final Pro
ject/Milestone1/Ex1.py"
[304 729 210 739  40 641 227  18 144 639]
Minimum: 18
Maximum: 739
Average: 369.1
Median: 265.5
Standard Deviation: 272.9690275470827
nishi22s@Nishis-MacBook-Pro Final Project %
```



```
nishi22s@Nishis-MacBook-Pro Final Project % /Library/Frameworks/Python.framework
/Versions/3.6/bin/python3 "/Users/nishi22s/Documents/HelloWorldProject/Final Pro
ject/Milestone1/Ex1.py"
[989 663 682 868 335 788 100 890 310 765]
Minimum: 100
Maximum: 989
Average: 639.0
Median: 723.5
Standard Deviation: 277.4206192769384
nishi22s@Nishis-MacBook-Pro Final Project %
```

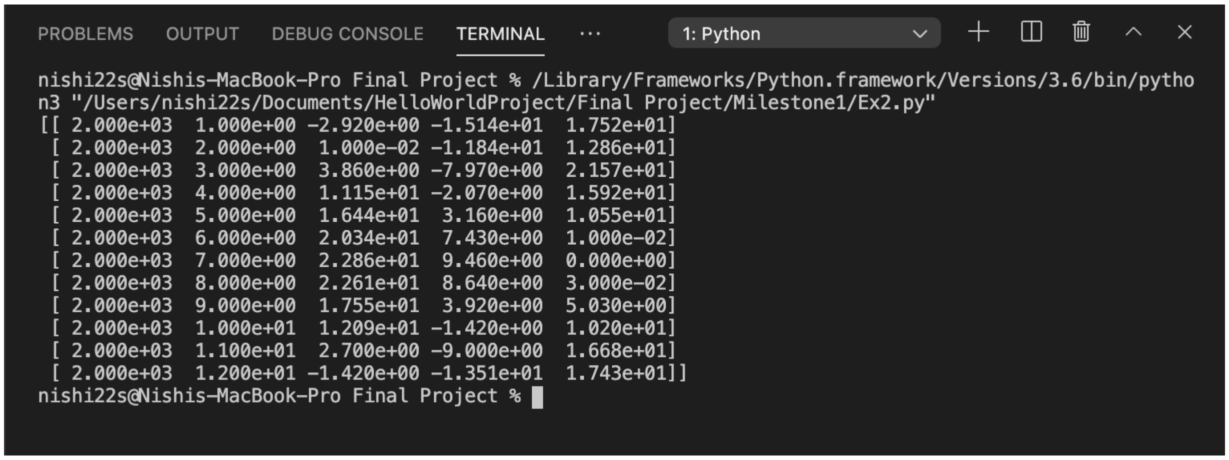
Reading CSV file by NumPy and Visualization: Assignment and Work:

Here is the code sample:



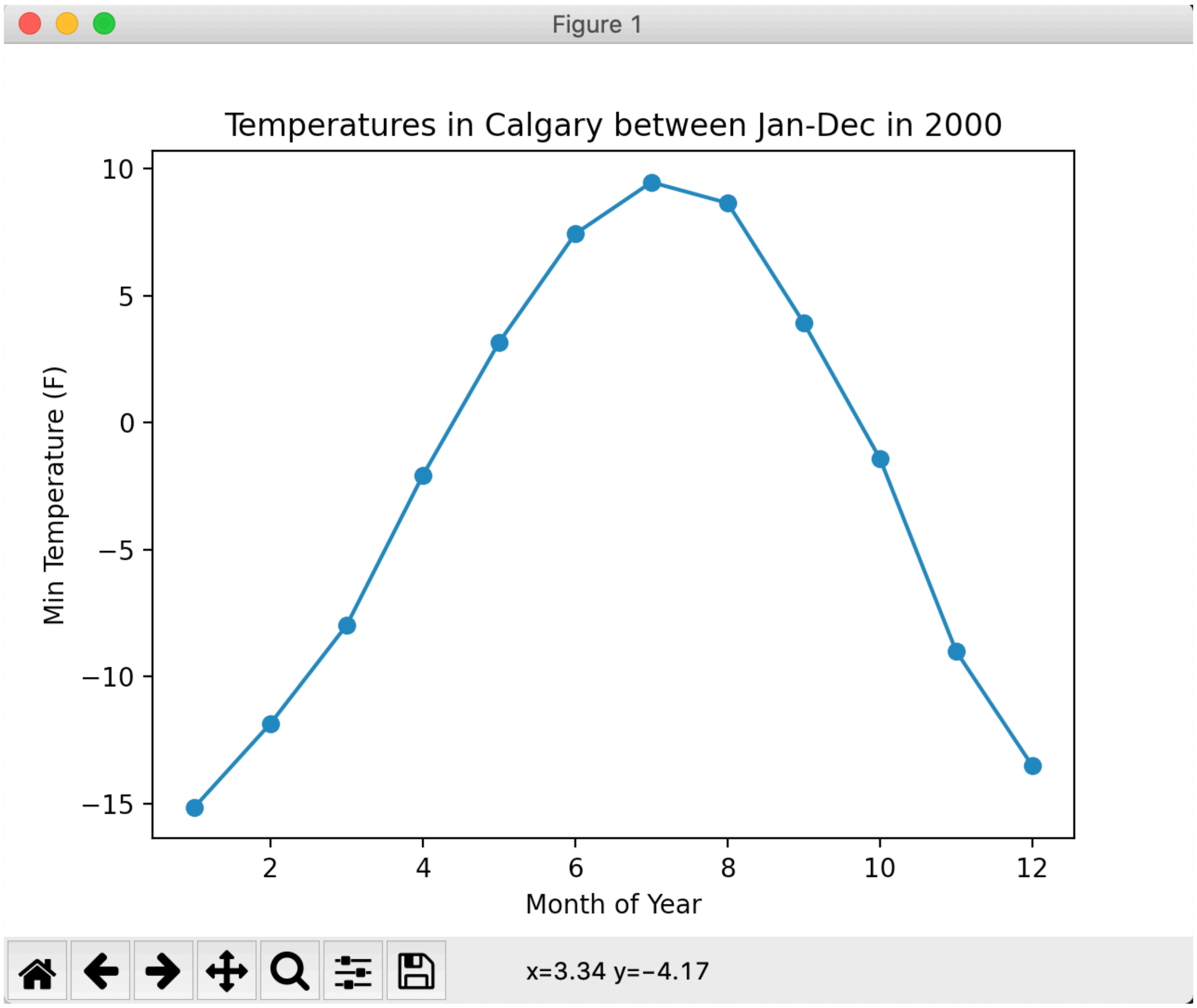
```python
import numpy as np
import matplotlib.pyplot as pyplot

def read_weather():
    file_name = "weather.csv"
    data = np.loadtxt(file_name, delimiter=',', skiprows=1, usecols=(0,1,2,3,4), dtype=np.float)
    return data

def drawChart (x,y):
    fig = pyplot.figure()
    pyplot.title('Temperatures in Calgary between Jan-Dec in 2000')
    pyplot.ylabel('Min Temperature (F)')
    pyplot.xlabel('Month of Year')

    pyplot.plot (x, y, marker='o')
    pyplot.show()

def main ():
    data = read_weather()
    print (data)
    x = (data[:,1])
    y = (data[:,3])
    drawChart(x,y)

if __name__ == "__main__":
    main ()
```

Terminal for printing the CSV data:



```
nishi22s@Nishis-MacBook-Pro Final Project % /Library/Frameworks/Python.framework/Versions/3.6/bin/pytho
n3 "/Users/nishi22s/Documents/HelloWorldProject/Final Project/Milestone1/Ex2.py"
[[ 2.000e+03  1.000e+00 -2.920e+00 -1.514e+01  1.752e+01]
 [ 2.000e+03  2.000e+00  1.000e-02 -1.184e+01  1.286e+01]
 [ 2.000e+03  3.000e+00  3.860e+00 -7.970e+00  2.157e+01]
 [ 2.000e+03  4.000e+00  1.115e+01 -2.070e+00  1.592e+01]
 [ 2.000e+03  5.000e+00  1.644e+01  3.160e+00  1.055e+01]
 [ 2.000e+03  6.000e+00  2.034e+01  7.430e+00  1.000e-02]
 [ 2.000e+03  7.000e+00  2.286e+01  9.460e+00  0.000e+00]
 [ 2.000e+03  8.000e+00  2.261e+01  8.640e+00  3.000e-02]
 [ 2.000e+03  9.000e+00  1.755e+01  3.920e+00  5.030e+00]
 [ 2.000e+03  1.000e+01  1.209e+01 -1.420e+00  1.020e+01]
 [ 2.000e+03  1.100e+01  2.700e+00 -9.000e+00  1.668e+01]
 [ 2.000e+03  1.200e+01 -1.420e+00 -1.351e+01  1.743e+01]]
nishi22s@Nishis-MacBook-Pro Final Project % 
```

Line Chart – Figure (3)



The next page marks the beginning of the Milestone Project itself, it is broken down into 10 sections as required.

Visualizing Weather Data: Assignment and Work:

Note: All the class python files are handed separately in a compressed folder. These screenshots show the working code and how the implementation goes through the program.

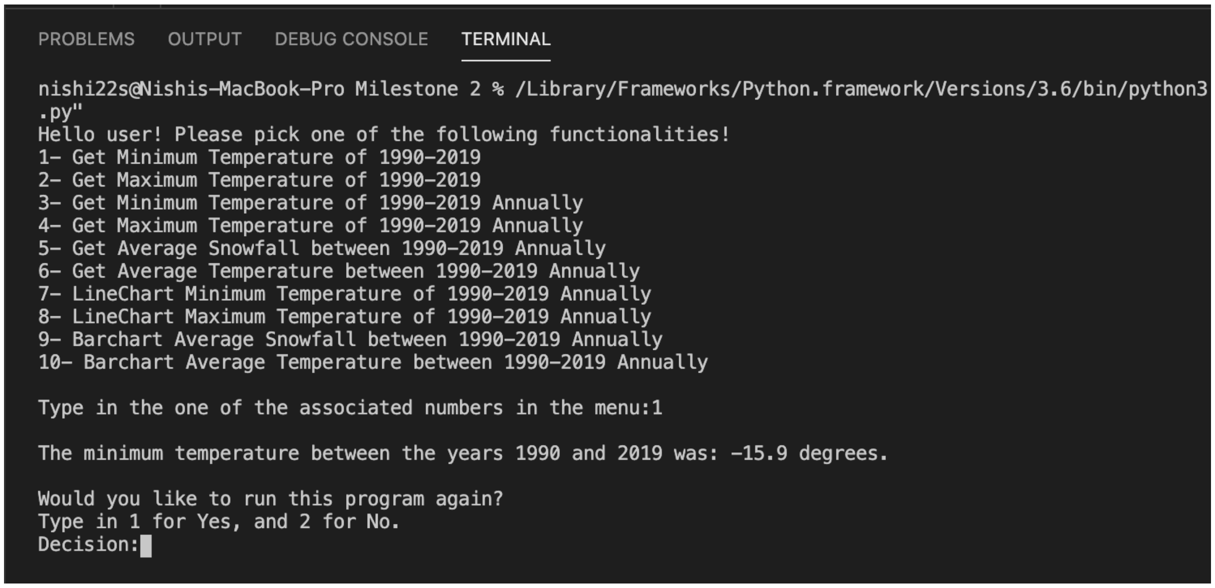
At first, the main function provides for a directory to present the user the options for Milestone #1 by calling upon the menu bar function. Once the user inputs an answer in the terminal, the program will run the code to analyze the weather data given in the csv 'Calgary Weather' file. In order to run the remainder of the program and classes, the main function also proves useful by acting as a constructor for the different class types and exchanging data to the instance variables. Now, the classes and their dependent functions – also known as methods – can be called via the main function.

The first step of the code is to create a file path from the main function (*correction from Milestone 1 feedback)* and load in the file from documents to the code itself, and this is done using the *FileIO* class. This class is responsible for reading data from the file CalgaryWeather.csv using the constraints enlisted by the delimiter, skiprows, and usecols.

The second step of the code is to read the year and month columns from the stored CSV file as independent lists by declaring them as instance variables inside *Class Date*. The third step of the code is to read the min temperature, max temperature, and snowfall columns from the stored CSV file as independent lists by declaring them as instance variables inside *Class Temperature Data*. These two classes – Date, and Temperature – are called as needed by the *Weather Analyzer* class below for the output of desired data type.

The fourth step of the code runs two while loops which allow for the directory's options to be played out. The first while loop accounts for any mistakes or wrong input on behalf of the user. It prompts the user again for an input if it doesn't fall within the number range indicated (1-10). The second loop is responsible for a series of nested if and else statements when the input does fall within the number range indicated. It runs a series of unique commands for each individual input, which will be explained below.]

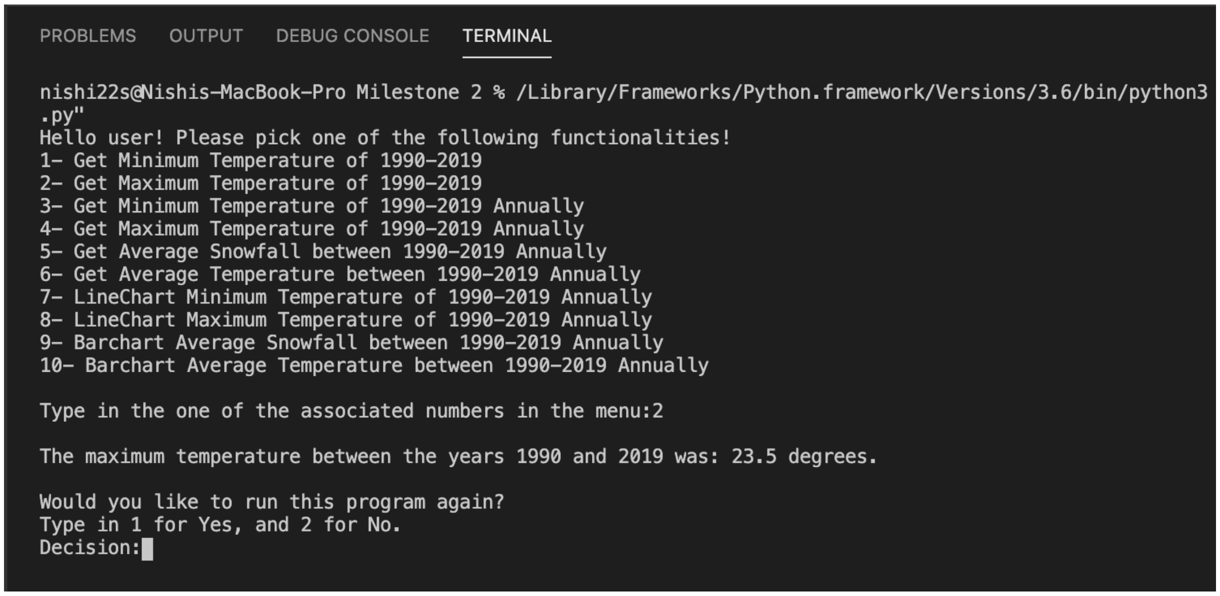
<u>Option #1</u> – Get Minimum Temperature between the years 1990 – 2019.



```
nishi22s@Nishis-MacBook-Pro Milestone 2 % /Library/Frameworks/Python.framework/Versions/3.6/bin/python3
.py"
Hello user! Please pick one of the following functionalities!
1- Get Minimum Temperature of 1990-2019
2- Get Maximum Temperature of 1990-2019
3- Get Minimum Temperature of 1990-2019 Annually
4- Get Maximum Temperature of 1990-2019 Annually
5- Get Average Snowfall between 1990-2019 Annually
6- Get Average Temperature between 1990-2019 Annually
7- LineChart Minimum Temperature of 1990-2019 Annually
8- LineChart Maximum Temperature of 1990-2019 Annually
9- Barchart Average Snowfall between 1990-2019 Annually
10- Barchart Average Temperature between 1990-2019 Annually

Type in the one of the associated numbers in the menu:1

The minimum temperature between the years 1990 and 2019 was: -15.9 degrees.

Would you like to run this program again?
Type in 1 for Yes, and 2 for No.
Decision:
```

According to the provided Calgary Weather Data, the minimum temperature between years 1990 and 2019 was –15.9 degrees Celsius as shown in the terminal of our program. The program achieves this result by calling on the method *getMinTemp* using its class *Weather Analyzer*. This method takes the array of min temperature stored in class *Temperature Data* and using pre-defined function of np.min from NumPy and finds the lowest number present in the list. Then, it prints the results out to the terminal. Back to the main function, it ends the program by using a break, and the user is asked if they want to retry the program which restarts the whole main function which houses our program. The preceding explanation achieves this by using an if loop *(correction from Milestone 1 program).*

<u>Option #2</u> – Get Maximum Temperature between the years 1990 – 2019.



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
nishi22s@Nishis-MacBook-Pro Milestone 2 % /Library/Frameworks/Python.framework/Versions/3.6/bin/python3
.py"
Hello user! Please pick one of the following functionalities!
1- Get Minimum Temperature of 1990-2019
2- Get Maximum Temperature of 1990-2019
3- Get Minimum Temperature of 1990-2019 Annually
4- Get Maximum Temperature of 1990-2019 Annually
5- Get Average Snowfall between 1990-2019 Annually
6- Get Average Temperature between 1990-2019 Annually
7- LineChart Minimum Temperature of 1990-2019 Annually
8- LineChart Maximum Temperature of 1990-2019 Annually
9- Barchart Average Snowfall between 1990-2019 Annually
10- Barchart Average Temperature between 1990-2019 Annually

Type in the one of the associated numbers in the menu:2

The maximum temperature between the years 1990 and 2019 was: 23.5 degrees.

Would you like to run this program again?
Type in 1 for Yes, and 2 for No.
Decision:
```

According to the provided Calgary Weather Data, the maximum temperature between years 1990 and 2019 was 23.5 degrees Celsius as shown in the terminal of our program. The program achieves this result by calling on the method *getMaxTemp* using its class *Weather Analyzer*. This method takes the array of max temperature stored in class *Temperature Data* and using pre-defined function of np.max from NumPy and finds the highest number present in the list. Then, it prints the results out to the terminal. Back to the main function, it ends the program by using a break, and the user is asked if they want to retry the program which restarts the whole main function which houses our program. The preceding explanation achieves this by using an if loop *(correction from Milestone 1 program)*.

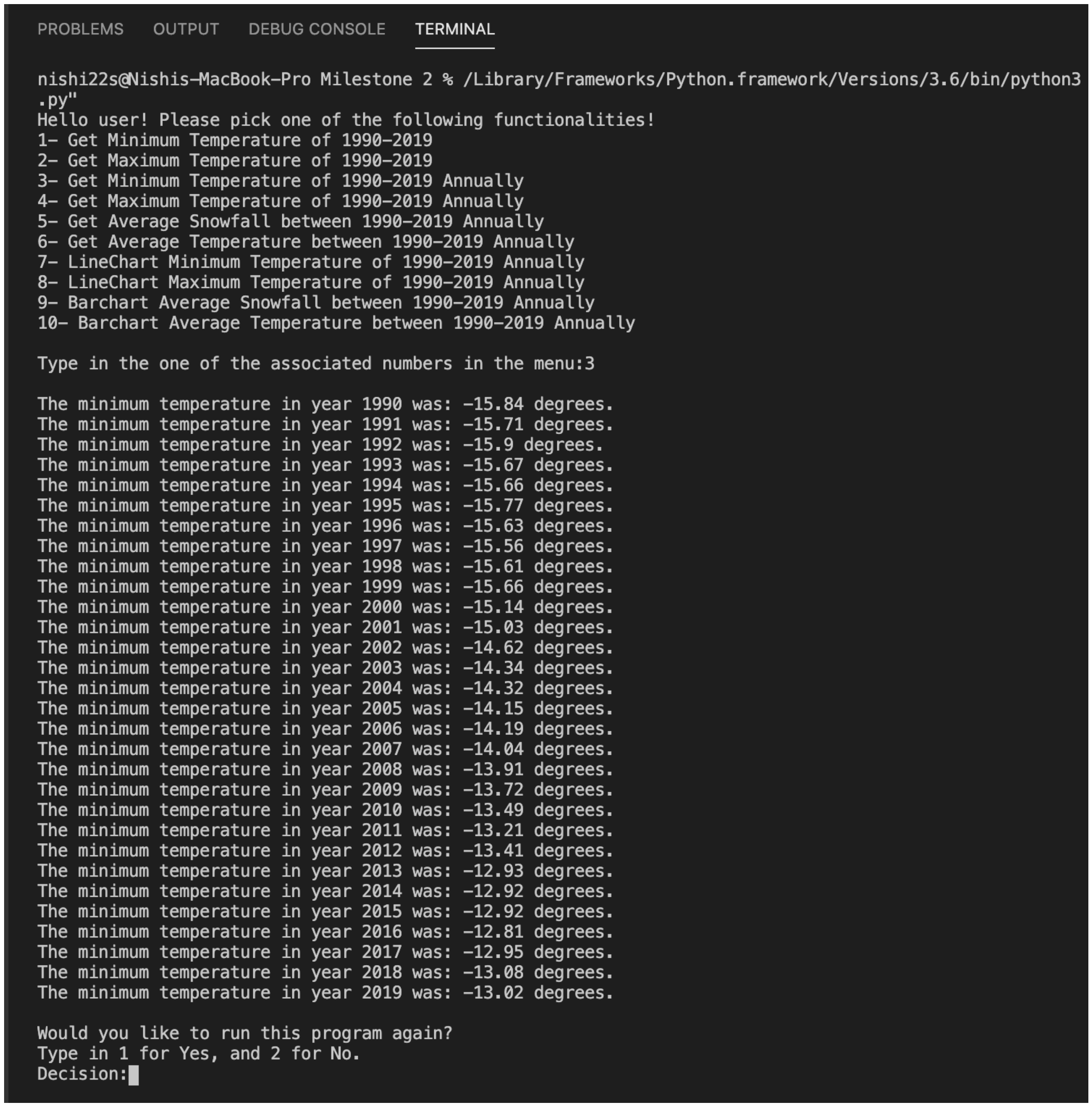
<u>Option #3</u> – Get the annual Minimum Temperature between the years 1990 – 2019.

This function calculates the annual minimum temperature for all 12 months in each year (excluding the year 2019 which only consists of 11 months as stated in the CSV file) then displays this information by presenting a 2D list which includes each year and its corresponding minimum temperature. This feature was implemented by calling on the *getMinTempAnnually* method using its class *WeatherAnalyzer*. This method takes the array of min temperature stored in class *Temperature Data* and the array of years stored in class *Date* and runs a while loop to accomplish the sorting and calculation of temperature values. The while loop runs through the number of rows in the CSV file in accordance with the specific arrays we've called on. For every 12 rows through years 1990-2018, and 11 rows for year 2019, a nested if loop allows the minimum temperature to be determined using the pre-defined function np.min from NumPy. This value gets stored repeatedly, alongside its year, in a 2D array through appending and clearing the list to start again for the next 12 values (rows). Ultimately, you end up with the following results which display information in the terminal as print statements:

The output shows the minimum temperature occurring every year in ascending order, and you can see a general trend of temperature increasing down the years. The user can pick any year in between years 1990 and 2019, and they can see the corresponding minimum temperature from the year – after considering all the minimum temperatures in each month of the year. This data can be compared amongst the years to prove which year was the coldest. In this data set, 1992 was the coldest year, reaching temperatures of – 15.9 degrees.

For example: The minimum temperature in year 2018 was –13.08 degrees.

Back to the main function, it ends the program by using a break, and the user is asked if they want to retry the program which restarts the whole main function which houses our program. The preceding explanation achieves this by using an if loop *(correction from Milestone 1 program)*.



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
nishi22s@Nishis-MacBook-Pro Milestone 2 % /Library/Frameworks/Python.framework/Versions/3.6/bin/python3
.py"
Hello user! Please pick one of the following functionalities!
1- Get Minimum Temperature of 1990-2019
2- Get Maximum Temperature of 1990-2019
3- Get Minimum Temperature of 1990-2019 Annually
4- Get Maximum Temperature of 1990-2019 Annually
5- Get Average Snowfall between 1990-2019 Annually
6- Get Average Temperature between 1990-2019 Annually
7- LineChart Minimum Temperature of 1990-2019 Annually
8- LineChart Maximum Temperature of 1990-2019 Annually
9- Barchart Average Snowfall between 1990-2019 Annually
10- Barchart Average Temperature between 1990-2019 Annually

Type in the one of the associated numbers in the menu:3

The minimum temperature in year 1990 was: -15.84 degrees.
The minimum temperature in year 1991 was: -15.71 degrees.
The minimum temperature in year 1992 was: -15.9 degrees.
The minimum temperature in year 1993 was: -15.67 degrees.
The minimum temperature in year 1994 was: -15.66 degrees.
The minimum temperature in year 1995 was: -15.77 degrees.
The minimum temperature in year 1996 was: -15.63 degrees.
The minimum temperature in year 1997 was: -15.56 degrees.
The minimum temperature in year 1998 was: -15.61 degrees.
The minimum temperature in year 1999 was: -15.66 degrees.
The minimum temperature in year 2000 was: -15.14 degrees.
The minimum temperature in year 2001 was: -15.03 degrees.
The minimum temperature in year 2002 was: -14.62 degrees.
The minimum temperature in year 2003 was: -14.34 degrees.
The minimum temperature in year 2004 was: -14.32 degrees.
The minimum temperature in year 2005 was: -14.15 degrees.
The minimum temperature in year 2006 was: -14.19 degrees.
The minimum temperature in year 2007 was: -14.04 degrees.
The minimum temperature in year 2008 was: -13.91 degrees.
The minimum temperature in year 2009 was: -13.72 degrees.
The minimum temperature in year 2010 was: -13.49 degrees.
The minimum temperature in year 2011 was: -13.21 degrees.
The minimum temperature in year 2012 was: -13.41 degrees.
The minimum temperature in year 2013 was: -12.93 degrees.
The minimum temperature in year 2014 was: -12.92 degrees.
The minimum temperature in year 2015 was: -12.92 degrees.
The minimum temperature in year 2016 was: -12.81 degrees.
The minimum temperature in year 2017 was: -12.95 degrees.
The minimum temperature in year 2018 was: -13.08 degrees.
The minimum temperature in year 2019 was: -13.02 degrees.

Would you like to run this program again?
Type in 1 for Yes, and 2 for No.
Decision:
```

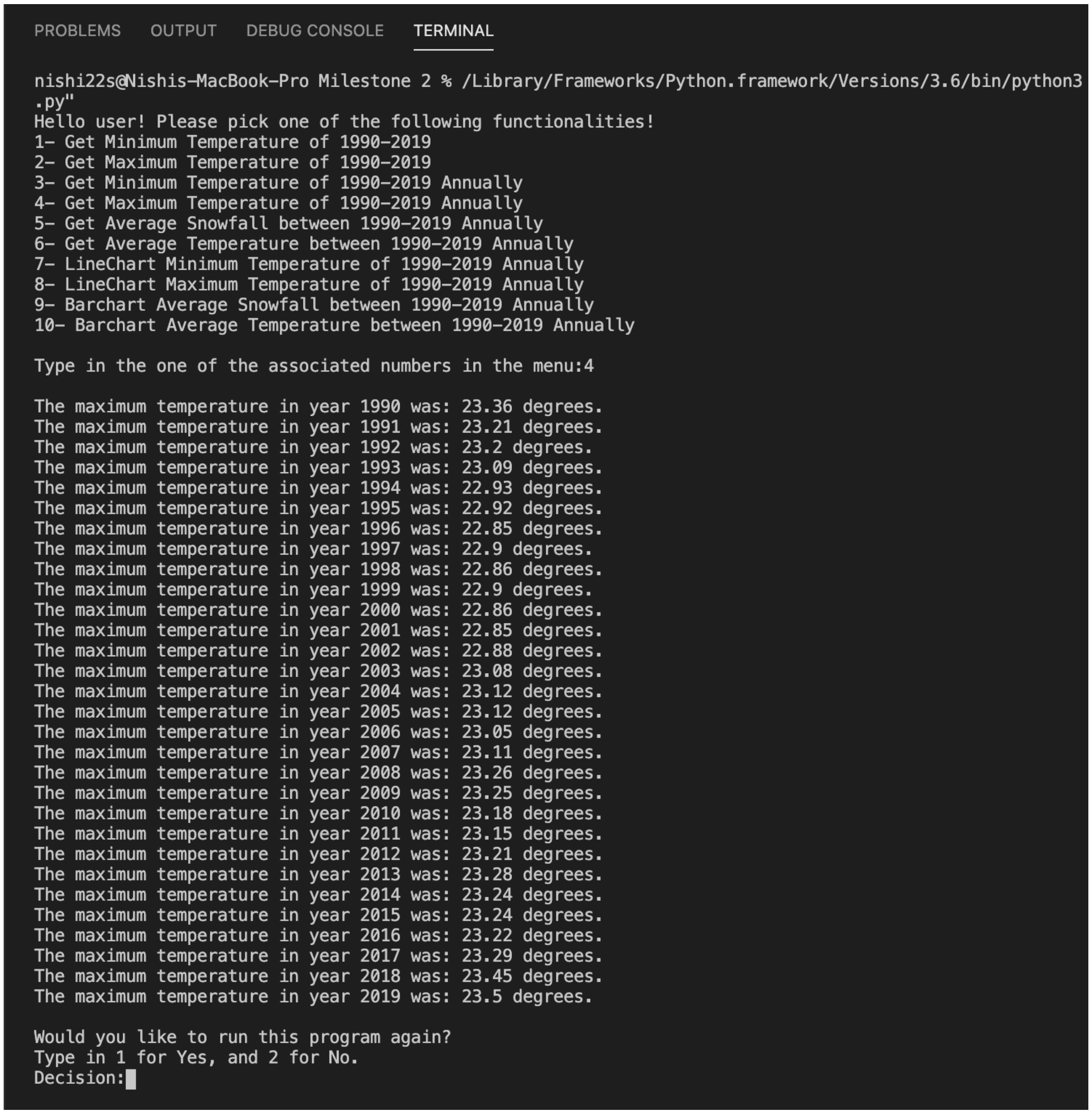
<u>Option #4</u> - Get the annual Maximum Temperature between the years 1990 – 2019.

This function calculates the annual maximum temperature for all 12 months in each year (excluding the year 2019 which only consists of 11 months as stated in the CSV file) then displays this information by presenting a 2D list which consists of each year and its corresponding maximum temperature. This feature was implemented by calling on the *getMaxTempAnnually* method using its class *WeatherAnalyzer*. This method takes the array of max temperature stored in class *Temperature Data* and the array of years stored in class *Date* and runs a while loop to accomplish the sorting and calculation of temperature values. The while loop runs through the number of rows in the CSV file in accordance with the specific arrays we've called on. For every 12 rows through years 1990-2018, and 11 rows for year 2019, a nested if loop allows the maximum temperature to be determined using the pre-defined function np.max from NumPy. This value gets stored repeatedly, alongside its year, in a 2D array through appending and clearing the list to start again for the next 12 values (rows). Ultimately, you end up with the following results which display information in the terminal as print statements:

The output shows the maximum temperature occurring every year in ascending order, and you can see a general trend of temperature increasing down the years. The user can pick any year in between years 1990 and 2019, and they can see the corresponding maximum temperature from the year – after considering all the maximum temperatures in each month of the year. This data can be compared amongst the years to prove which year was the hottest. In this data set, 2019 was the hottest year, reaching temperatures of 23.5 degrees.

For example: The maximum temperature in year 2002 was 22.88 degrees.

Back to the main function, it ends the program by using a break, and the user is asked if they want to retry the program which restarts the whole main function which houses our program. The preceding explanation achieves this by using an if loop *(correction from Milestone 1 program)*.



```
nishi22s@Nishis-MacBook-Pro Milestone 2 % /Library/Frameworks/Python.framework/Versions/3.6/bin/python3
.py"
Hello user! Please pick one of the following functionalities!
1- Get Minimum Temperature of 1990-2019
2- Get Maximum Temperature of 1990-2019
3- Get Minimum Temperature of 1990-2019 Annually
4- Get Maximum Temperature of 1990-2019 Annually
5- Get Average Snowfall between 1990-2019 Annually
6- Get Average Temperature between 1990-2019 Annually
7- LineChart Minimum Temperature of 1990-2019 Annually
8- LineChart Maximum Temperature of 1990-2019 Annually
9- Barchart Average Snowfall between 1990-2019 Annually
10- Barchart Average Temperature between 1990-2019 Annually

Type in the one of the associated numbers in the menu:4

The maximum temperature in year 1990 was: 23.36 degrees.
The maximum temperature in year 1991 was: 23.21 degrees.
The maximum temperature in year 1992 was: 23.2 degrees.
The maximum temperature in year 1993 was: 23.09 degrees.
The maximum temperature in year 1994 was: 22.93 degrees.
The maximum temperature in year 1995 was: 22.92 degrees.
The maximum temperature in year 1996 was: 22.85 degrees.
The maximum temperature in year 1997 was: 22.9 degrees.
The maximum temperature in year 1998 was: 22.86 degrees.
The maximum temperature in year 1999 was: 22.9 degrees.
The maximum temperature in year 2000 was: 22.86 degrees.
The maximum temperature in year 2001 was: 22.85 degrees.
The maximum temperature in year 2002 was: 22.88 degrees.
The maximum temperature in year 2003 was: 23.08 degrees.
The maximum temperature in year 2004 was: 23.12 degrees.
The maximum temperature in year 2005 was: 23.12 degrees.
The maximum temperature in year 2006 was: 23.05 degrees.
The maximum temperature in year 2007 was: 23.11 degrees.
The maximum temperature in year 2008 was: 23.26 degrees.
The maximum temperature in year 2009 was: 23.25 degrees.
The maximum temperature in year 2010 was: 23.18 degrees.
The maximum temperature in year 2011 was: 23.15 degrees.
The maximum temperature in year 2012 was: 23.21 degrees.
The maximum temperature in year 2013 was: 23.28 degrees.
The maximum temperature in year 2014 was: 23.24 degrees.
The maximum temperature in year 2015 was: 23.24 degrees.
The maximum temperature in year 2016 was: 23.22 degrees.
The maximum temperature in year 2017 was: 23.29 degrees.
The maximum temperature in year 2018 was: 23.45 degrees.
The maximum temperature in year 2019 was: 23.5 degrees.

Would you like to run this program again?
Type in 1 for Yes, and 2 for No.
Decision:
```

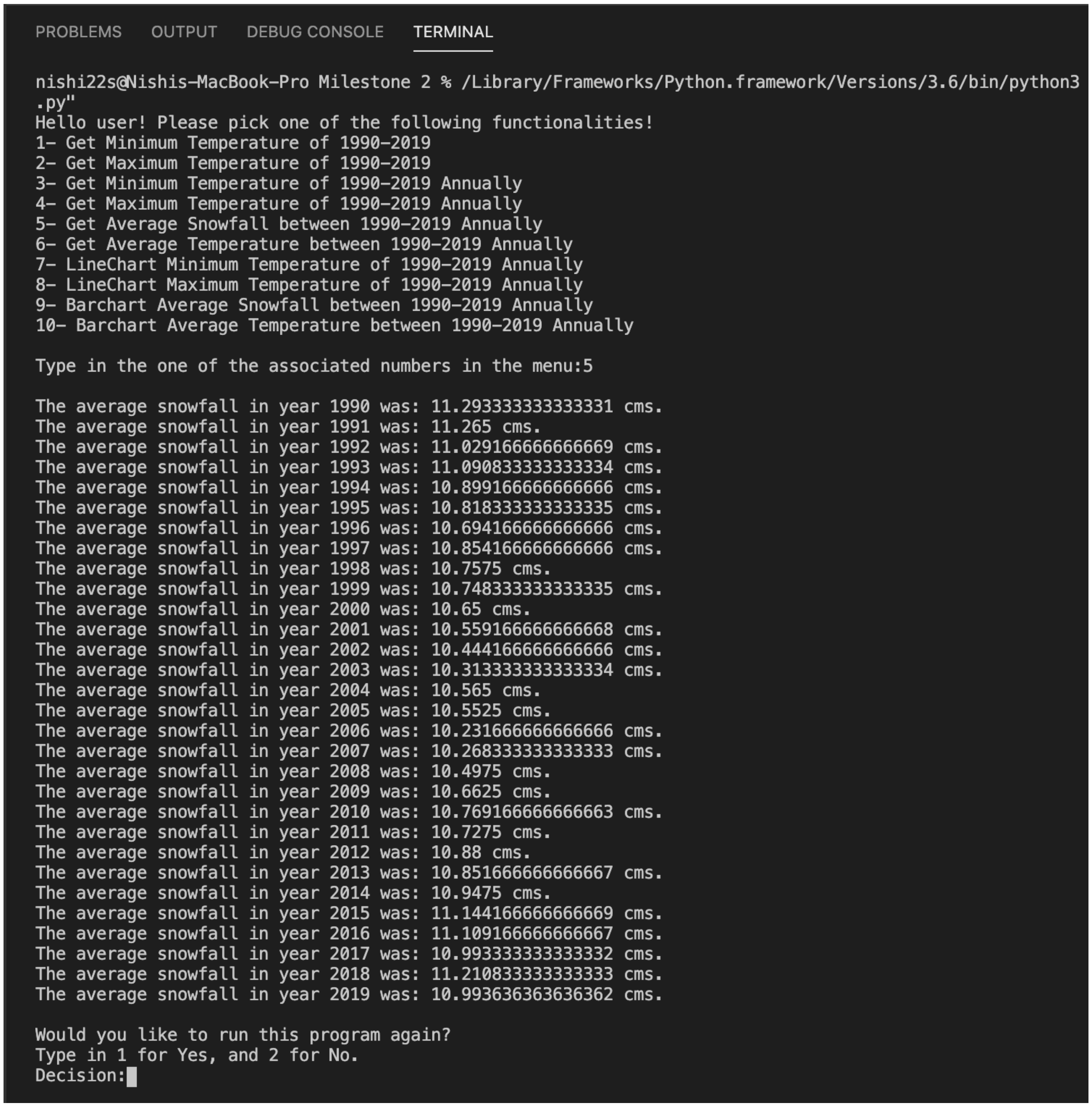
<u>Option #5</u> – Get the annual average Snowfall between the years 1990 – 2019.

This function calculates the annual average snowfall for all 12 months in each year (excluding the year 2019 which only consists of 11 months as stated in the CSV file) then displays this information by presenting a 2D list which consists of each year and its corresponding average snowfall. This feature was implemented by calling on the *getAvgSnowFallAnnually* method using its class *WeatherAnalyzer*. This method takes the array of snowfall data stored in class *Temperature Data* and the array of years stored in class *Date* and runs a while loop to accomplish the sorting and calculation of snowfall values. The while loop runs through the number of rows in the CSV file in accordance with the specific arrays we've called on. For every 12 rows through years 1990-2018, and 11 rows for year 2019, a nested if loop allows the average snowfall to be determined using the pre-defined function np.average from NumPy. This value gets stored repeatedly, alongside its year, in a 2D array through appending and clearing the list to start again for the next 12 values (rows). Ultimately, you end up with the following results which display information in the terminal as print statements:

The output shows the average snowfall occurring every year in ascending order, and you can see how there is no general trend in the data fluctuation – there are slight changes yearly, but it remains constant. The user can pick any year in between years 1990 and 2019, and they can see the corresponding average snowfall from the year – after considering all the average snowfall in each month of the year. This data can be compared amongst the years to prove which year had the most or least snowfall. In this data set, 2006 had the least snowfall of 10.23 centimeters, while 1990 has the most snowfall of 11.29 centimeters.

For example: The average snowfall in year 2012 was 10.88 centimeters.

Back to the main function, it ends the program by using a break, and the user is asked if they want to retry the program which restarts the whole main function which houses our program. The preceding explanation achieves this by using an if loop *(correction from Milestone 1 program)*.


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
nishi22s@Nishis-MacBook-Pro Milestone 2 % /Library/Frameworks/Python.framework/Versions/3.6/bin/python3
.py"
Hello user! Please pick one of the following functionalities!
1- Get Minimum Temperature of 1990-2019
2- Get Maximum Temperature of 1990-2019
3- Get Minimum Temperature of 1990-2019 Annually
4- Get Maximum Temperature of 1990-2019 Annually
5- Get Average Snowfall between 1990-2019 Annually
6- Get Average Temperature between 1990-2019 Annually
7- LineChart Minimum Temperature of 1990-2019 Annually
8- LineChart Maximum Temperature of 1990-2019 Annually
9- Barchart Average Snowfall between 1990-2019 Annually
10- Barchart Average Temperature between 1990-2019 Annually

Type in the one of the associated numbers in the menu:5

The average snowfall in year 1990 was: 11.293333333333331 cms.
The average snowfall in year 1991 was: 11.265 cms.
The average snowfall in year 1992 was: 11.029166666666669 cms.
The average snowfall in year 1993 was: 11.090833333333334 cms.
The average snowfall in year 1994 was: 10.899166666666666 cms.
The average snowfall in year 1995 was: 10.818333333333335 cms.
The average snowfall in year 1996 was: 10.694166666666666 cms.
The average snowfall in year 1997 was: 10.854166666666666 cms.
The average snowfall in year 1998 was: 10.7575 cms.
The average snowfall in year 1999 was: 10.748333333333335 cms.
The average snowfall in year 2000 was: 10.65 cms.
The average snowfall in year 2001 was: 10.559166666666668 cms.
The average snowfall in year 2002 was: 10.444166666666666 cms.
The average snowfall in year 2003 was: 10.313333333333334 cms.
The average snowfall in year 2004 was: 10.565 cms.
The average snowfall in year 2005 was: 10.5525 cms.
The average snowfall in year 2006 was: 10.231666666666666 cms.
The average snowfall in year 2007 was: 10.268333333333333 cms.
The average snowfall in year 2008 was: 10.4975 cms.
The average snowfall in year 2009 was: 10.6625 cms.
The average snowfall in year 2010 was: 10.769166666666663 cms.
The average snowfall in year 2011 was: 10.7275 cms.
The average snowfall in year 2012 was: 10.88 cms.
The average snowfall in year 2013 was: 10.851666666666667 cms.
The average snowfall in year 2014 was: 10.9475 cms.
The average snowfall in year 2015 was: 11.144166666666669 cms.
The average snowfall in year 2016 was: 11.109166666666667 cms.
The average snowfall in year 2017 was: 10.993333333333332 cms.
The average snowfall in year 2018 was: 11.210833333333333 cms.
The average snowfall in year 2019 was: 10.993636363636362 cms.

Would you like to run this program again?
Type in 1 for Yes, and 2 for No.
Decision:
```

Visualizing Weather Data: Assignment and Work:

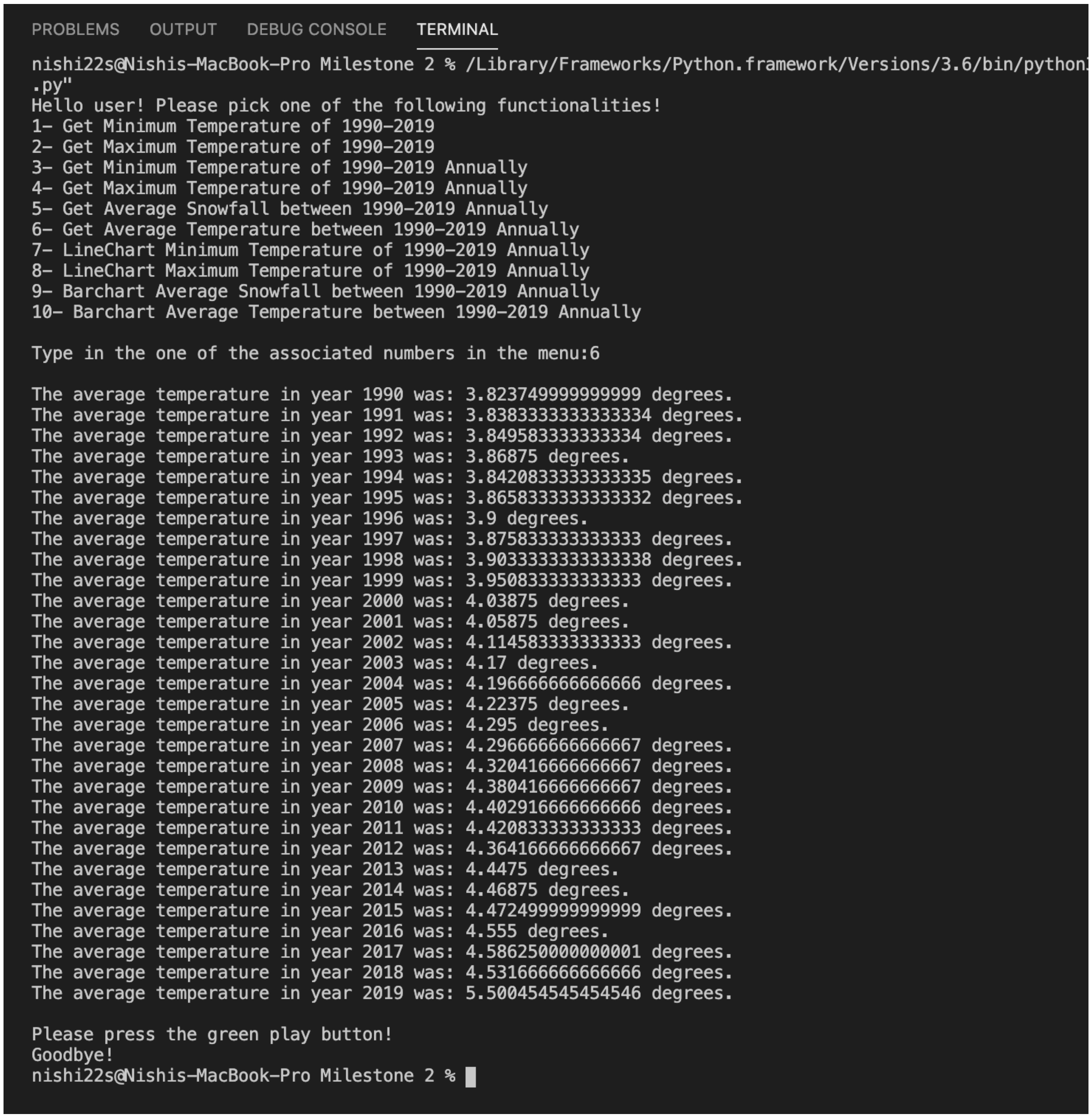
<u>Option #6</u> – Get the annual average Temperature between the years 1990 – 2019.

This function calculates the annual average temperature for all 12 months in each year (excluding the year 2019 which only consists of 11 months as stated in the CSV file) then displays this information by presenting a 2D list which consists of each year and its corresponding average temperature. This feature was implemented by calling on the *getAvgTemperatureAnnually* method using its class *WeatherAnalyzer*. This method takes the array of min temperature and the array of max temperature stored in class *Temperature Data* and the array of years stored in class *Date* and runs a while loop to accomplish the sorting and calculation of temperature values. The while loop runs through the number of rows in the CSV file in accordance with the specific arrays we've called on. For every 12 rows through years 1990-2018, and 11 rows for year 2019, a nested if loop allows the average temperature to be determined using the pre-defined function np.average from NumPy. Essentially, we would be finding the mean of all the maximum and minimum temperatures in the specific year. This value gets stored repeatedly, alongside its year, in a 2D array through appending and clearing the list to start again for the next 12 values (rows). Ultimately, you end up with the following results which display information in the terminal as print statements:

The output shows the average temperature occurring every year in ascending order, and you can see a general trend of temperature increasing down the years. The user can pick any year in between years 1990 and 2019, and they can see the corresponding average temperature from the year – after considering all the average temperature in each month of the year. This data can be compared amongst the years to prove which year was the hottest or coldest. In this data set, 1990 was the coldest year, with an average temperature of 3.82 degrees, while 2019 was the hottest year, with an average temperature of 5.50 degrees.

For example: The average temperature in year 2009 was 4.3804 (rounded to thousandth place) degrees.

Back to the main function, it ends the program by using a break, and the user is asked if they want to retry the program which restarts the whole main function which houses our program. The preceding explanation achieves this by using an if loop *(correction from Milestone 1 program)*.



```
nishi22s@Nishis-MacBook-Pro Milestone 2 % /Library/Frameworks/Python.framework/Versions/3.6/bin/python
.py"
Hello user! Please pick one of the following functionalities!
1- Get Minimum Temperature of 1990-2019
2- Get Maximum Temperature of 1990-2019
3- Get Minimum Temperature of 1990-2019 Annually
4- Get Maximum Temperature of 1990-2019 Annually
5- Get Average Snowfall between 1990-2019 Annually
6- Get Average Temperature between 1990-2019 Annually
7- LineChart Minimum Temperature of 1990-2019 Annually
8- LineChart Maximum Temperature of 1990-2019 Annually
9- Barchart Average Snowfall between 1990-2019 Annually
10- Barchart Average Temperature between 1990-2019 Annually

Type in the one of the associated numbers in the menu:6

The average temperature in year 1990 was: 3.823749999999999 degrees.
The average temperature in year 1991 was: 3.8383333333333334 degrees.
The average temperature in year 1992 was: 3.849583333333334 degrees.
The average temperature in year 1993 was: 3.86875 degrees.
The average temperature in year 1994 was: 3.8420833333333335 degrees.
The average temperature in year 1995 was: 3.8658333333333332 degrees.
The average temperature in year 1996 was: 3.9 degrees.
The average temperature in year 1997 was: 3.875833333333333 degrees.
The average temperature in year 1998 was: 3.9033333333333338 degrees.
The average temperature in year 1999 was: 3.950833333333333 degrees.
The average temperature in year 2000 was: 4.03875 degrees.
The average temperature in year 2001 was: 4.05875 degrees.
The average temperature in year 2002 was: 4.114583333333333 degrees.
The average temperature in year 2003 was: 4.17 degrees.
The average temperature in year 2004 was: 4.196666666666666 degrees.
The average temperature in year 2005 was: 4.22375 degrees.
The average temperature in year 2006 was: 4.295 degrees.
The average temperature in year 2007 was: 4.296666666666667 degrees.
The average temperature in year 2008 was: 4.320416666666667 degrees.
The average temperature in year 2009 was: 4.380416666666667 degrees.
The average temperature in year 2010 was: 4.402916666666666 degrees.
The average temperature in year 2011 was: 4.420833333333333 degrees.
The average temperature in year 2012 was: 4.364166666666667 degrees.
The average temperature in year 2013 was: 4.4475 degrees.
The average temperature in year 2014 was: 4.46875 degrees.
The average temperature in year 2015 was: 4.472499999999999 degrees.
The average temperature in year 2016 was: 4.555 degrees.
The average temperature in year 2017 was: 4.586250000000001 degrees.
The average temperature in year 2018 was: 4.531666666666666 degrees.
The average temperature in year 2019 was: 5.500454545454546 degrees.

Please press the green play button!
Goodbye!
nishi22s@Nishis-MacBook-Pro Milestone 2 % 
```

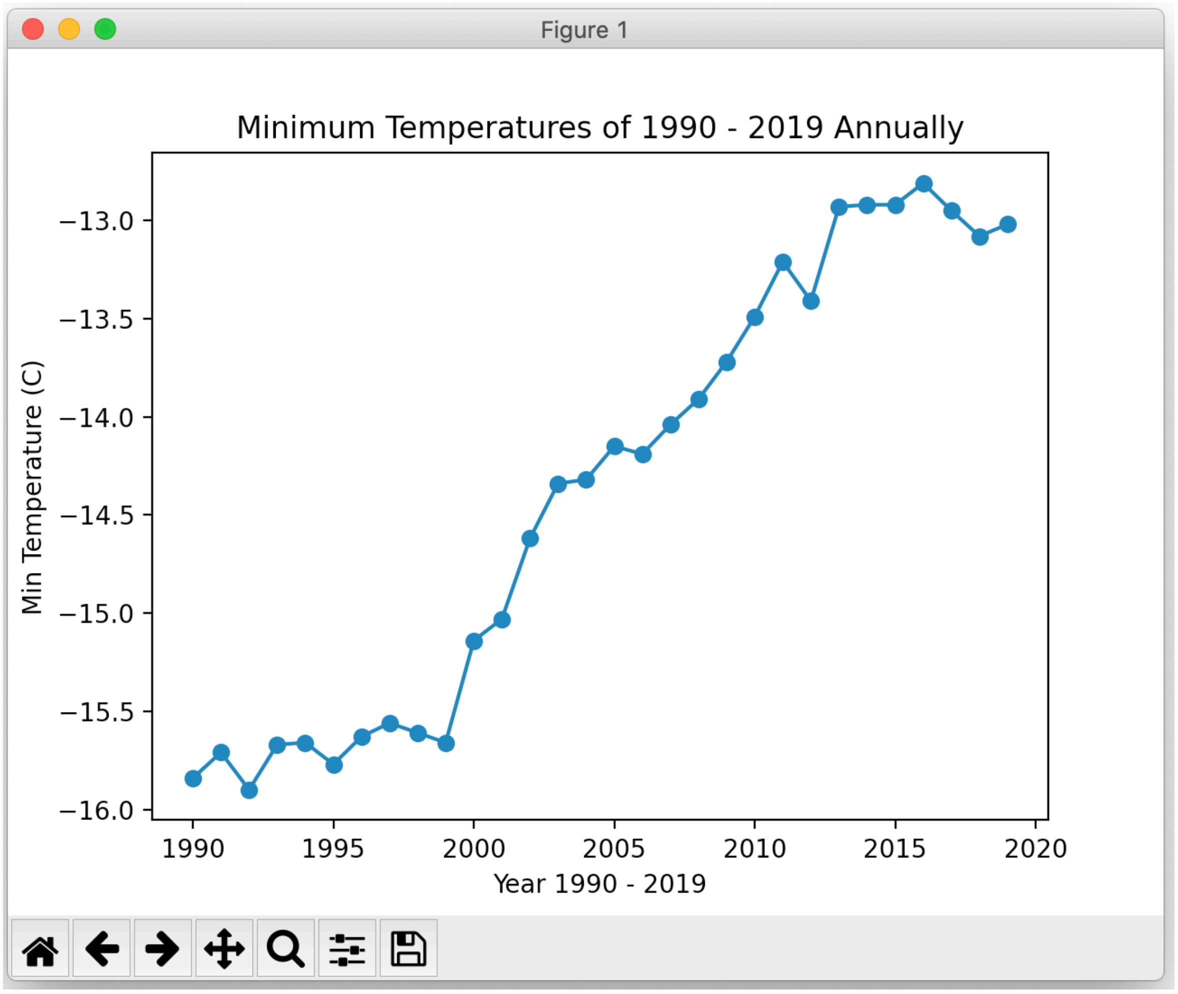
<u>Option #7</u> – Get the line chart of annual minimum Temperature between the years 1990 – 2019.

This function graphs the data calculated by *getMinTempAnnually*, upon receiving the 2D array previously created in the specific method. The main function calls on the *getMinTempAnnually* again using its class *Weather Analyzer* and stores the 2D list as a variable array in its nested elif statement. Next, this 2D array is broken down into its two element arrays by referring to its indices - [0] for the year data and [1] for the temperature data. This is possible because of how it was stored originally. The [0] index-array represents the x-axis data, while the [1] index-array represents the y-axis data. These axes are given their corresponding names alongside the title and stored as string variables in the same elif statement. Then, this data is referred to the class *Chart*, and is used for the components of the graph itself, allowing for the following results:

The output shows the line graph of minimum temperatures occurring every year in ascending order, and you can see a general trend of temperature increasing as years progress visually, instead of reading data from the terminal. The user can pick any year in between years 1990 and 2019 (on the graph), and they can see the corresponding minimum temperature from the selected year (calculated after considering all the minimum temperatures in each month of the year). Matplotlib allows for the cursor to be moved along the graph, for better reading of the temperature data.

For example: The highest minimum temperature was in year 2016 which was –12.81 degrees.

Back to the main function, it ends the program by using a break, and the user is asked if they want to retry the program which restarts the whole main function which houses our program. The preceding explanation achieves this by using an if loop *(correction from Milestone 1 program)*.



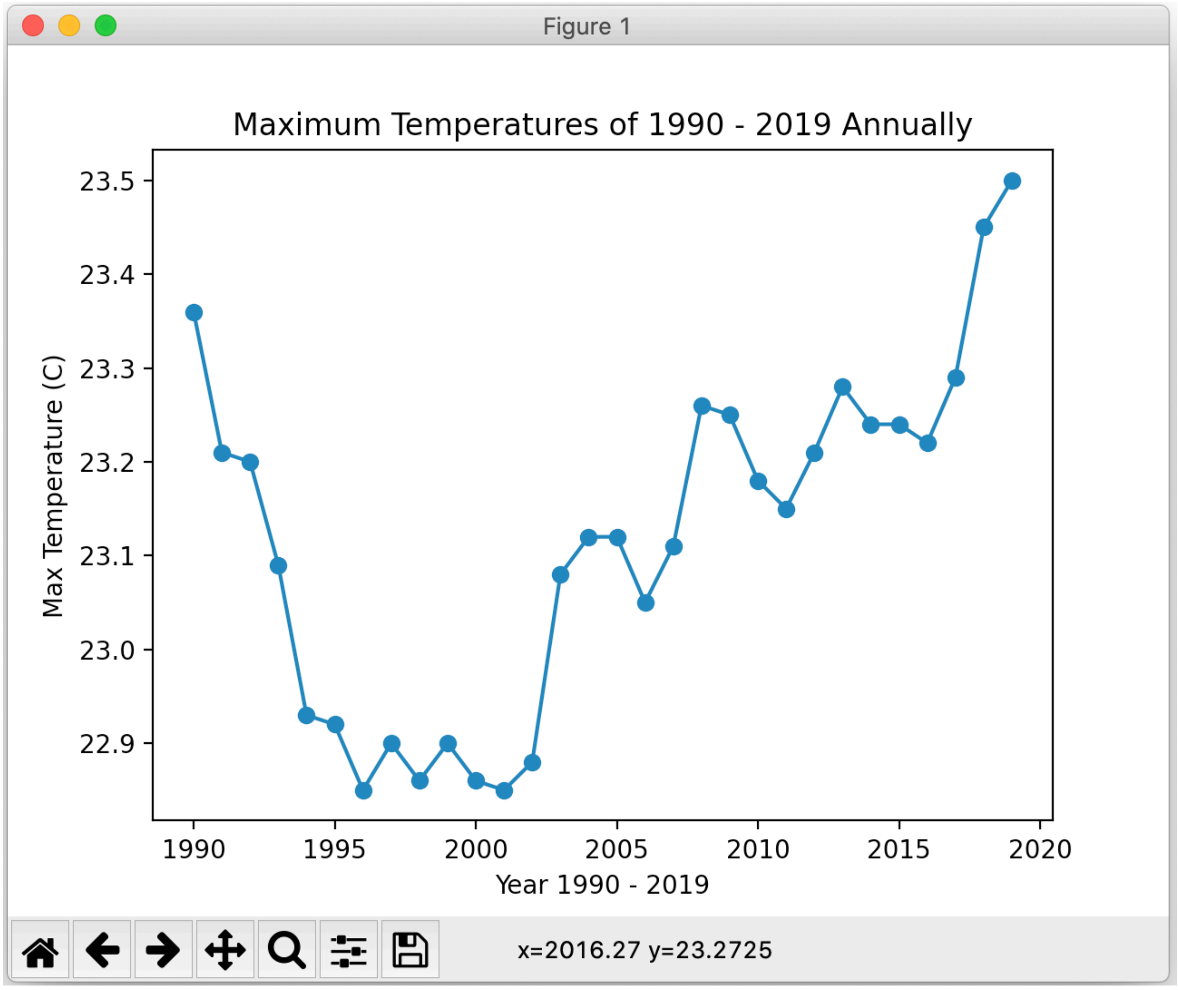
<u>Option #8</u> – Get the line chart of annual maximum Temperature between the years 1990 – 2019.

This function graphs the data calculated by *getMaxTempAnnually*, upon receiving the 2D array previously created in the specific method. The main function calls on the *getMaxTempAnnually* again using its class *Weather Analyzer* and stores the 2D list as a variable array in its nested elif statement. Next, this 2D array is broken down into its two element arrays by referring to its indices - [0] for the year data and [1] for the temperature data. This is possible because of how it was stored originally. The [0] index-array represents the x-axis data, while the [1] index-array represents the y-axis data. These axes are given their corresponding names alongside the title and stored as string variables in the same elif statement. Then, this data is referred to the class *Chart*, and is used for the components of the graph itself, allowing for the following results:

The output shows the line graph of maximum temperatures occurring every year in ascending order, and you can see a general trend of temperature increasing progressively after the lowest tick mark in year 2001. The user can pick any year in between years 1990 and 2019 (on the graph), and they can see the corresponding maximum temperature from the selected year (calculated after considering all the maximum temperatures in each month of the year). Matplotlib allows for the cursor to be moved along the graph, for better reading of the temperature data.

For example: The lowest maximum temperature was in year 2001 which was 22.85 degrees. The highest peak in maximum temperature was in year 2019 which was 23.5 degrees.

Back to the main function, it ends the program by using a break, and the user is asked if they want to retry the program which restarts the whole main function which houses our program. The preceding explanation achieves this by using an if loop *(correction from Milestone 1 program)*.



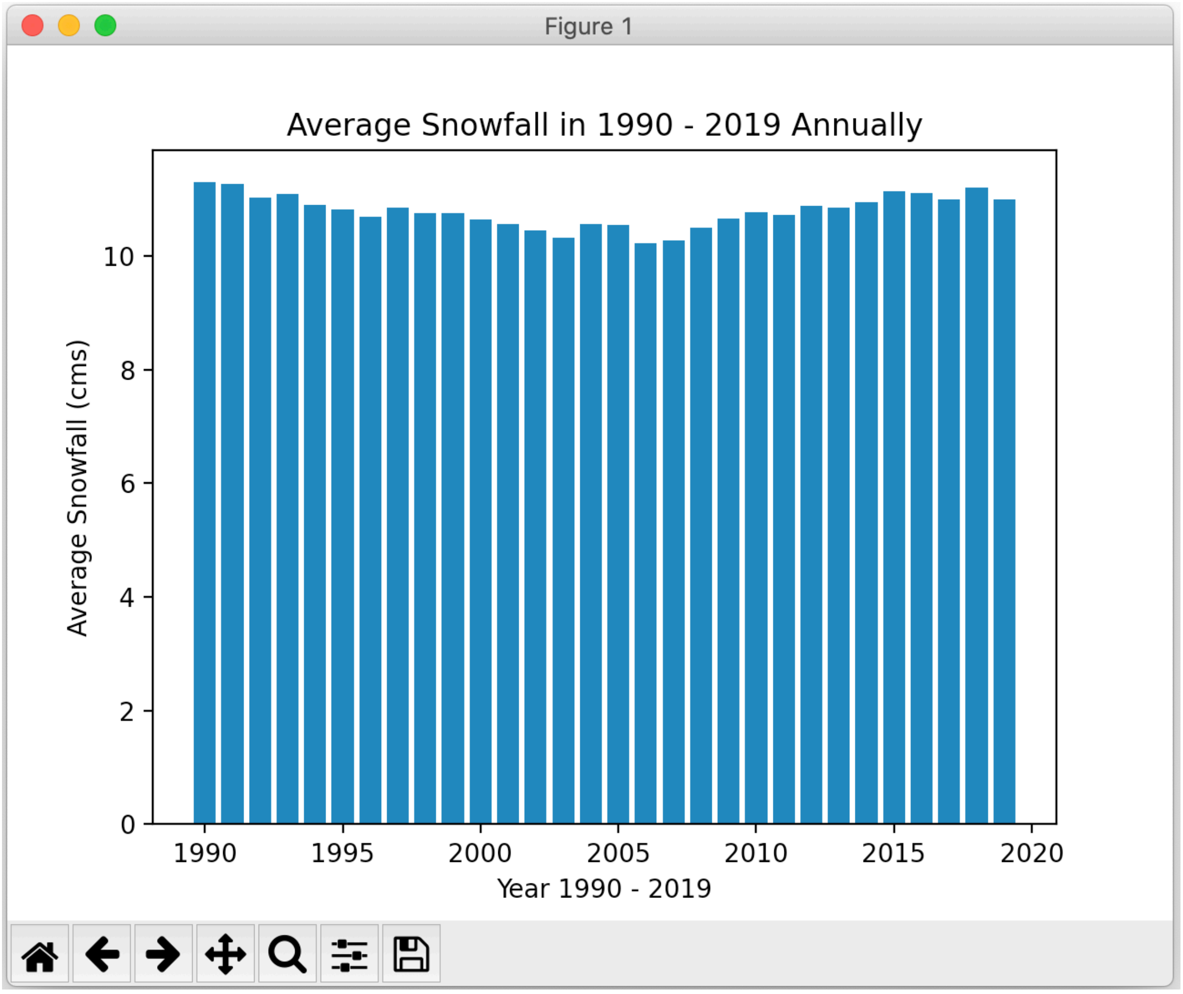
<u>Option #9</u> – Get the bar chart of annual average Snowfall between the years 1990 – 2019.

This function graphs the data calculated by *getAvgSnowFallAnnually*, upon receiving the 2D array previously created in the specific method. The main function calls on the *getAvgSnowFallAnnually* again using its class *Weather Analyzer* and stores the 2D list as a variable array in its nested elif statement. Next, this 2D array is broken down into its two element arrays by referring to its indices - [0] for the year data and [1] for the snowfall data. This is possible because of how it was stored originally. The [0] index-array represents the x-axis data, while the [1] index-array represents the y-axis data. These axes are given their corresponding names alongside the title and stored as string variables in the same elif statement. Then, this data is referred to the class *Chart*, and is used for the components of the graph itself, allowing for the following results:

The output shows the bar graph of average snowfall occurring every year in ascending order, and you can see how there is no general trend in the data fluctuation – there are slight changes yearly, but it remains consistent. The user can pick any year in between years 1990 and 2019 (on the graph), and they can see the corresponding average snowfall from the selected year (calculated after considering all the average snowfall in each month of the year). Matplotlib allows for the cursor to be moved along the graph, for better reading of the snowfall data.

For example: The average snowfall in year 1995 was 10.8183 (rounded to the thousandth decimal place) centimeters. The highest peak of snowfall was in year 1990 and the second highest was in year 2018, by a difference of 0.08 centimeters.

Back to the main function, it ends the program by using a break, and the user is asked if they want to retry the program which restarts the whole main function which houses our program. The preceding explanation achieves this by using an if loop *(correction from Milestone 1 program)*.



<u>Option #10</u> – Get the bar chart of annual Temperature between the years 1990 – 2019.

This function graphs the data calculated by *getAvgTemperatureAnnually*, upon receiving the 2D array previously created in the specific method. The main function calls on the *getAvgTemperatureAnnually* again using its class *Weather Analyzer* and stores the 2D list as a variable array in its nested elif statement. Next, this 2D array is broken down into its two element arrays by referring to its indices - [0] for the year data and [1] for the temperature data. This is possible because of how it was stored originally. The [0] index-array represents the x-axis data, while the [1] index-array represents the y-axis data. These axes are given their corresponding names alongside the title and stored as string variables in the same elif statement. Then, this data is referred to the class *Chart*, and is used for the components of the graph itself, allowing for the following results:

The output shows the bar graph of average temperature occurring every year in ascending order, and you can see a general trend of temperature increasing over the time period of 30 years. There is also a boost of higher temperature in the final year of 2019. The user can pick any year in between years 1990 and 2019 (on the graph), and they can see the corresponding average temperature from the selected year (calculated after considering all the average temperature in each month of the year). Matplotlib allows for the cursor to be moved along the graph, for better reading of the temperature data.

For example: The average temperature in year 2019 was 5.500 (rounded to the thousandth decimal place) degrees. This also happens to be the warmest year in this time period. The temperature in Calgary has heated up an average difference of 1.68 degrees between the warmest and coldest year.

Back to the main function, it ends the program by using a break, and the user is asked if they want to retry the program which restarts the whole main function which houses our program. The preceding explanation achieves this by using an if loop *(correction from Milestone 1 program)*.

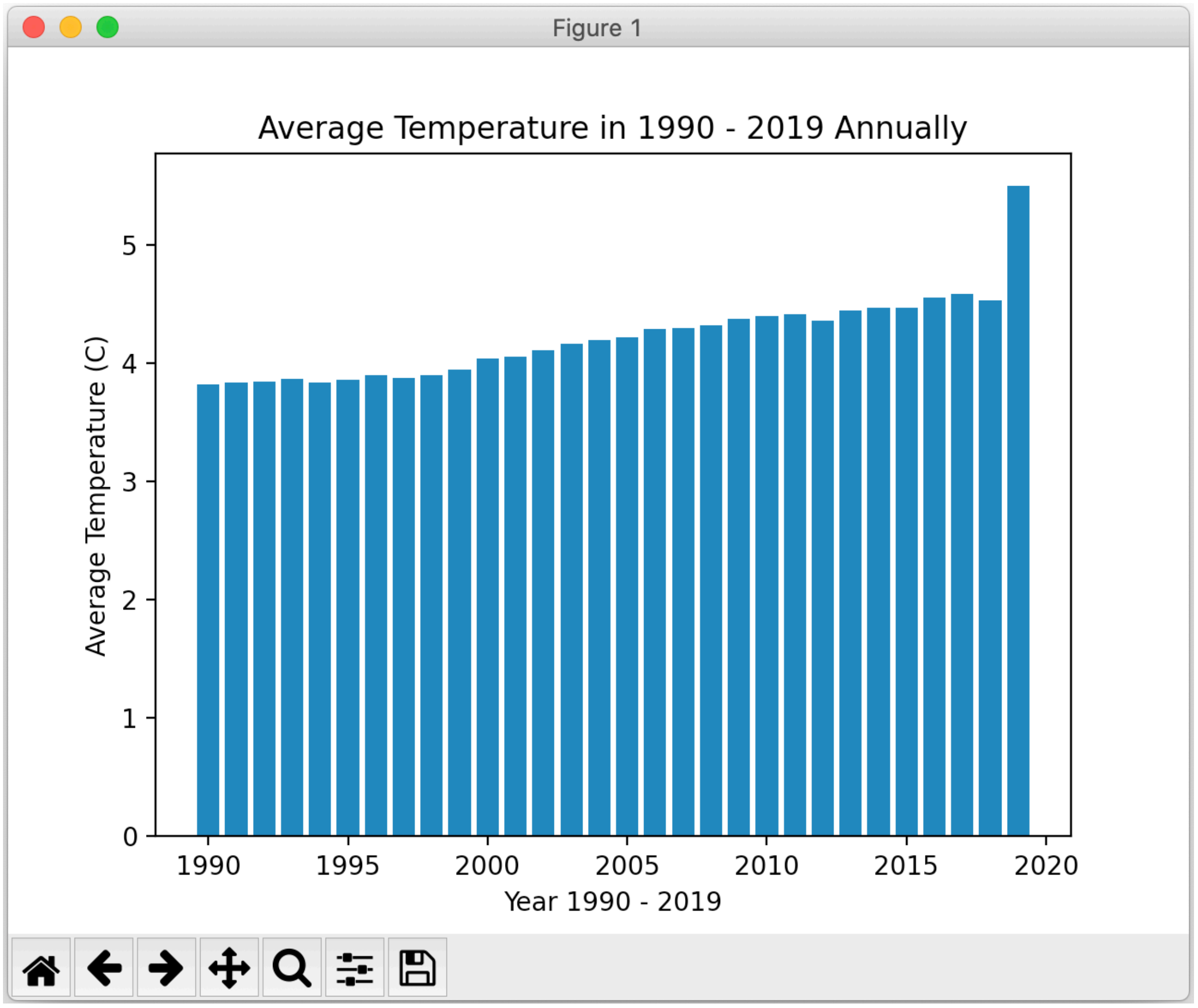


Figure 1
Average Temperature in 1990 - 2019 Annually
Average Temperature (C)
Year 1990 - 2019