|  |  |
| --- | --- |
| **NAME:** | **DUE:** |
| **Tomoki Koike** | **April 11th 2019** |

**Every Boiler Engineering Code – Entry Level Programming**

**Week 6 – Programming Exercises**

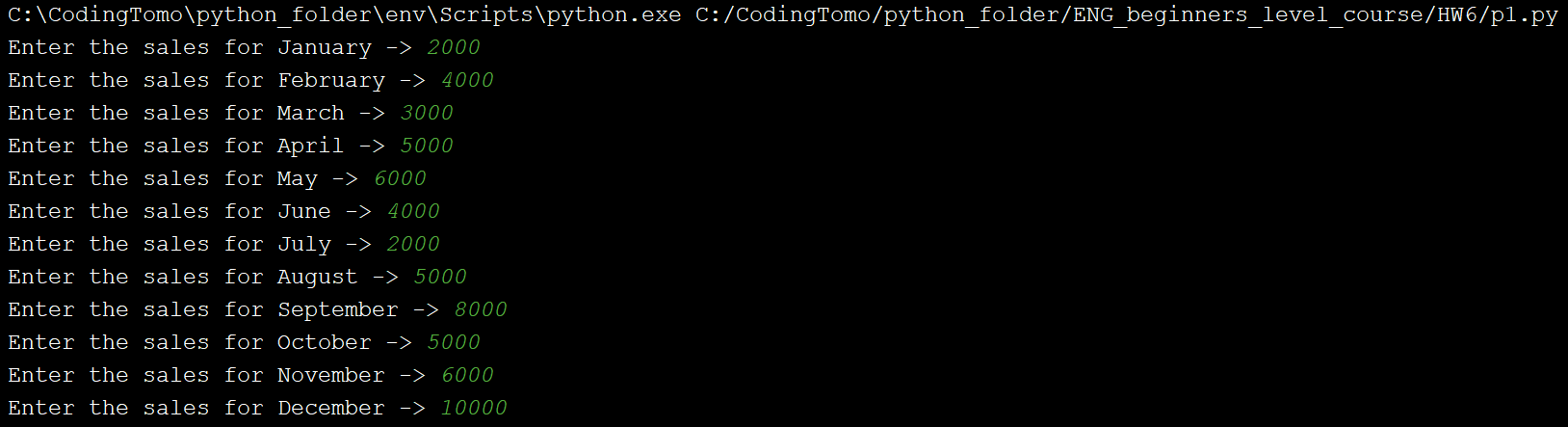
1. (**15 points**) Write a Python program, asking the user to input the sales of each month (1-12) of a year and storing all the sales in a list. Then using ***matplotlib*** to read the 12 sales values from the list and plot the sales value as a ***pie*** chart. When plotting, send the color of each slice, from January to December, by a tuple, (‘r’, ‘g’, ‘b’, ‘y’, ‘m’, ‘w’, ‘k’, ‘spring green’, ‘grey24’, ‘pink3’, ‘grey80’, ‘brown2’). Display the title of the pie chart and the label for each slice of the chart.

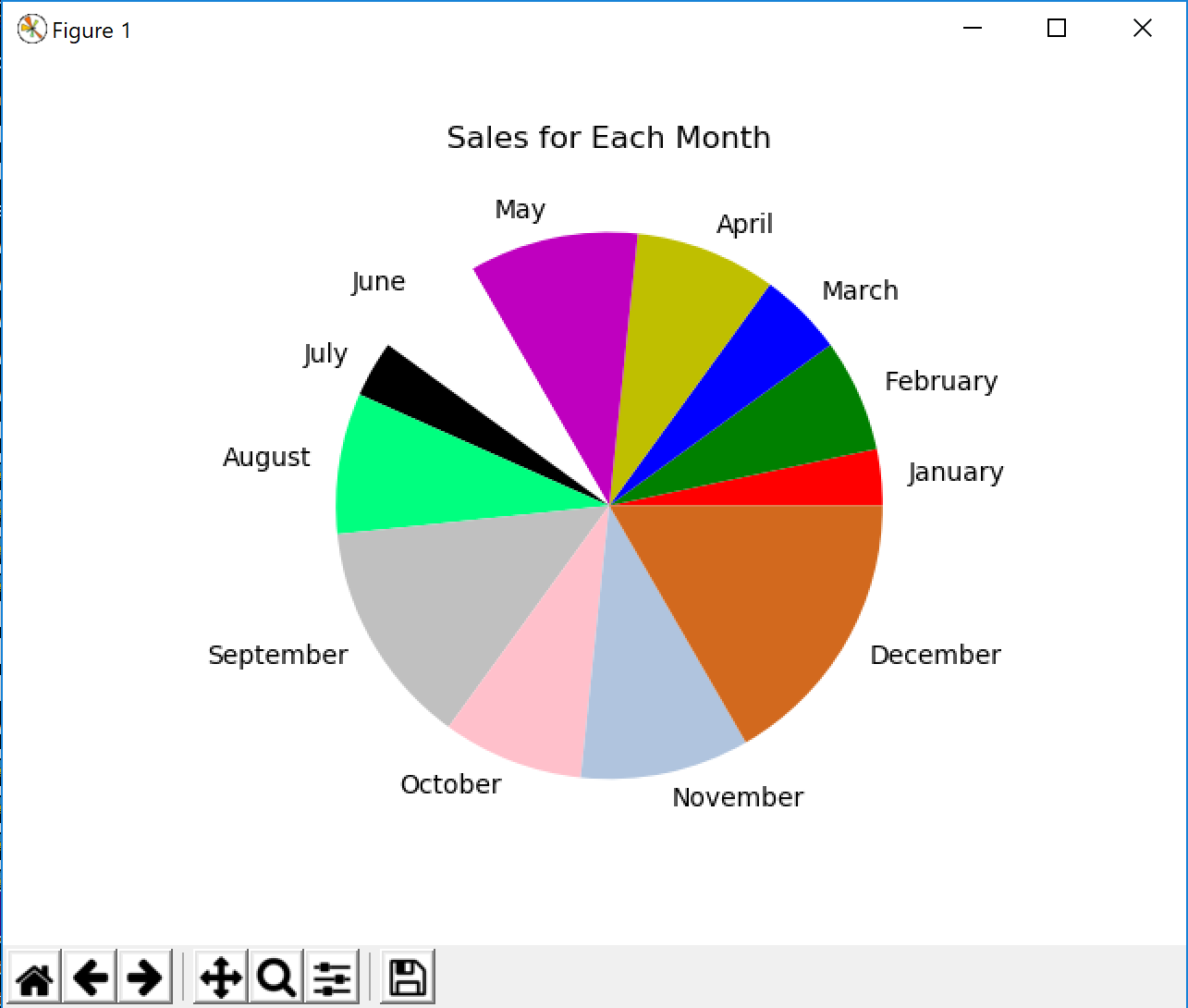
(**When you test your code, enter 2000, 4000, 3000, 5000, 6000, 4000, 2000, 5000, 8000, 5000, 6000, 10000 as the sales for the months.**)

**CODE:**

## PROBLEM #1  
# NAME: Tomoki Koike  
# DUE: 4/11/2019  
# DESCRIPTION: This program is designed to have the user input 12 values for the  
# sales of a certain product for 12 months. And then the program will plot a  
# pie chart for visual aid.  
# (with input validations)  
# STAND: Class of 2020  
##  
  
# Import Modules  
import calendar  
import matplotlib.pyplot as plt  
  
# Functions  
# Function for input validation  
def getValid(prompt):  
 while True:  
 try:  
 # trying input with without any conditions at first  
 this = float(input(prompt))  
 except ValueError:  
 # Prints the user to input again since the input was not valid  
 print('Sorry, could not understand. Please enter again.')  
 continue  
 # For when the input is a number  
 if this > **0**:  
 # Valid input  
 break  
 else:  
 # Invalid input  
 print('Error. Please enter a positive float value.')  
 continue  
 return this  
  
# Function to append values to a list with the inputs  
def create\_list(input\_data**,** subject\_list):  
 subject\_list.append(input\_data)  
 return subject\_list  
  
# Function to create the pie chart  
# There were no such colors as ‘grey24’, ‘pink3’, ‘grey80’, ‘brown2’ they kept on giving me errors  
def create\_pieChart(data\_list):  
 # Assigning the labels for the data  
 label\_months = calendar.month\_name[**1**::]  
 # Creating the pie chart  
 color\_tuple = ('r'**,** 'g'**,** 'b'**,** 'y'**,** 'm'**,** 'w'**,** 'k'**,** 'springgreen'**,** 'silver'**,** 'pink'**,** 'lightsteelblue'**,** 'chocolate')  
 plt.pie(data\_list**,** labels=label\_months**,** colors=color\_tuple)  
 # Showing the title of the chart  
 plt.title('Sales for Each Month')  
 # Display the chart  
 plt.show()  
 return  
  
# Main  
def main():  
 # Creating the list with the user input  
 # Preallocating the list  
 data\_list = []  
 for x in range(**12**):  
 create\_list(getValid('Enter the sales for {0} -> '.format(calendar.month\_name[x+**1**]))**,** data\_list)  
 # Create pie chart  
 create\_pieChart(data\_list)  
 return  
  
main()

**OUTPUT & CHART:**





1. (**15 points**) In the attachments, there is a file named ‘WorldSeriesWinner.txt’. This file contains a chronological list of the World Series’ winning teams from 1903 through 2009. The first line in the file is the name of the team that won in 1903, and the last line is the name of the team that won in 2009. (Note the World Series was not played in 1904 or 1994.There are entries in the file indicating this.)

Write a Python program that reads this file and creates a dictionary in which the keys are the names of the teams, and each key’s associated value is the number of times the team has won the World Series. The program should also create a dictionary in which the keys are the years, and each key’s associated value is the name of the team that won that year.

The program should prompt the user for a year in the range of 1903 through 2009. It should then display the name of the team that won the World Series that year, and the number of times that teams has won the World Series.

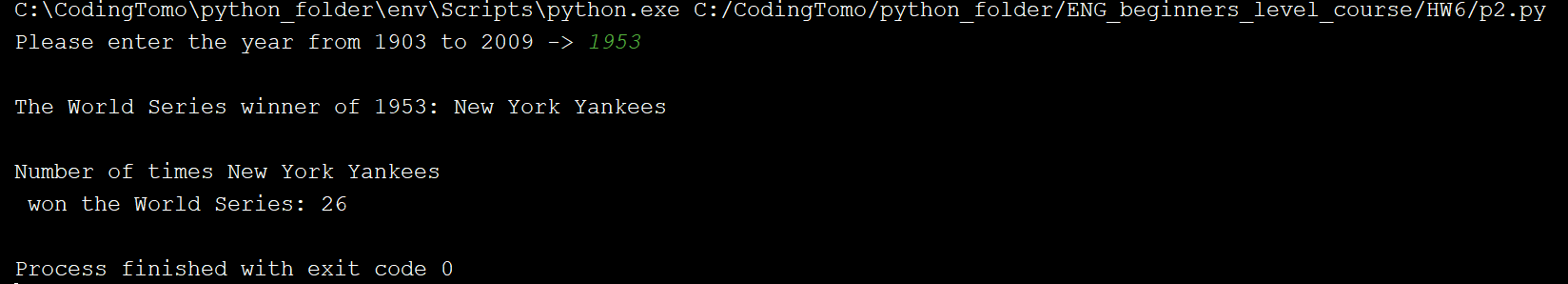
(**Use 1953, 1990, 2000, 2008 to test your program.** )

**CODE:**

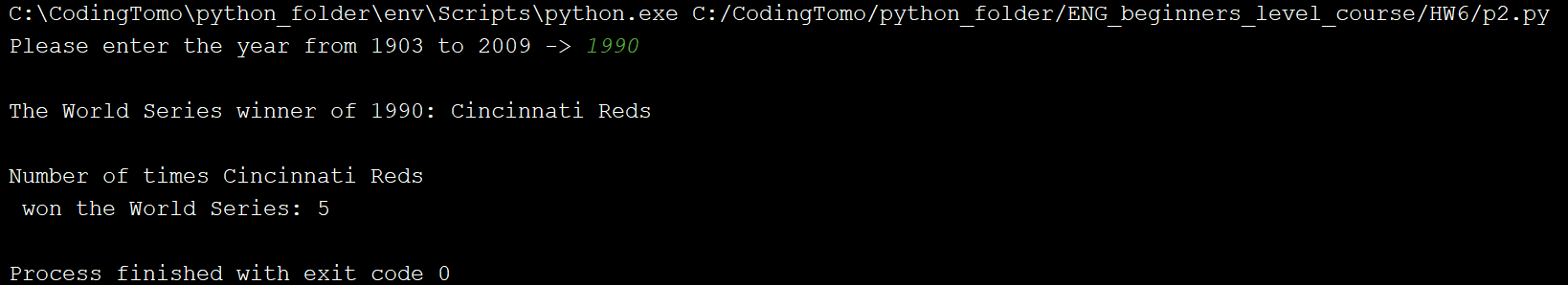
## PROBLEM #2  
# NAME: Tomoki Koike  
# DUE: 4/11/2019  
# DESCRIPTION: This program reads a file with the winners of the World Series  
# into a dictionary and when the user inputs a year from 1904 to 2009 the program will output the  
# winner of that year and the nunmber of times that the team won the World Series.  
# (with input validations)  
# STAND: Class of 2020  
##  
  
# Import Modules  
import matplotlib.pyplot as plt  
  
# Functions  
# Function for input validation  
def getValid(prompt):  
 while True:  
 try:  
 # trying input with without any conditions at first  
 this = int(input(prompt))  
 except ValueError:  
 # Prints the user to input again since the input was not valid  
 print('Sorry, could not understand. Please enter again.')  
 continue  
 # For when the input is a number  
 if this >= **1903** and this <= **2009**:  
 if this == **1904** or this == **1994**:  
 print('There was no world series in this year please select another year -> ')  
 continue  
 else:  
 # Valid input  
 break  
 else:  
 # Invalid input  
 print('Error. Please enter a year in between the period of 1903 ~ 2009 -> ')  
 continue  
 return this  
  
# Function to read a file  
def create\_dict1(file):  
 # Opening the file  
 specimen\_file = open(file**,**'r')  
 # Reading the first line of the file  
 line = specimen\_file.readline()  
 # Dictionary to store the team names  
 teamName = []  
 numberOfWins = []  
 # Looping to create the keys and values for the dict by reading the file  
 while line != '':  
 if teamName.count(line) == **0**:  
 # Append to the teamName list if it appears for the first time  
 teamName.append(line)  
 # Add one win for this team  
 numberOfWins.append(**1**)  
 else:  
 # Add the number of wins for this team  
 name\_idx = teamName.index(line)  
 numberOfWins[name\_idx] += **1** # Read the next line in the file  
 line = specimen\_file.readline()  
 # Close the open file  
 specimen\_file.close()  
 # Now create the dictionary  
 WorldSeriesWinner\_dict = dict(zip(teamName**,** numberOfWins))  
 return WorldSeriesWinner\_dict  
  
def create\_dict2(file):  
 # Opening the file  
 specimen\_file = open(file**,** 'r')  
 # Reading the file  
 line = specimen\_file.readline()  
 # List of year  
 years = []  
 starting\_year = **1903** winningTeam = []  
 # Loop  
 while line != '':  
 if starting\_year != **1904** and starting\_year != **1994**:  
 winningTeam.append(line)  
 years.append(starting\_year)  
 line = specimen\_file.readline()  
 starting\_year += **1** else:  
 starting\_year += **1** # Close the open file  
 specimen\_file.close()  
 # Create the dictionary  
 winnerTeam\_and\_year = dict(zip(years**,** winningTeam))  
 return winnerTeam\_and\_year  
  
# Function to create the output based on what year the user selects  
def output(dict1**,** dict2**,** user\_input):  
 # Printing out the winner of the year  
 team = dict2[user\_input]  
 print('The World Series winner of {0}: {1}'.format(user\_input**,** team))  
 # Printing out the number of times the team won the world series  
 print('Number of times {0} won the World Series: {1}'.format(team**,** dict1[team]))  
 return  
  
# Main  
def main():  
 # creating the dictionaries  
 WorldSeriesWinner\_dict = create\_dict1('WorldSeriesWinner.txt')  
 WinnerTeam\_and\_year = create\_dict2('WorldSeriesWinner.txt')  
  
 # User input of the year to sea the winner  
 user\_input = getValid('Please enter the year from 1903 to 2009 -> ')  
 # output  
 print()  
 output(WorldSeriesWinner\_dict**,** WinnerTeam\_and\_year**,** user\_input)  
 return   
  
main()

**SAMPLE EXECUTION:**

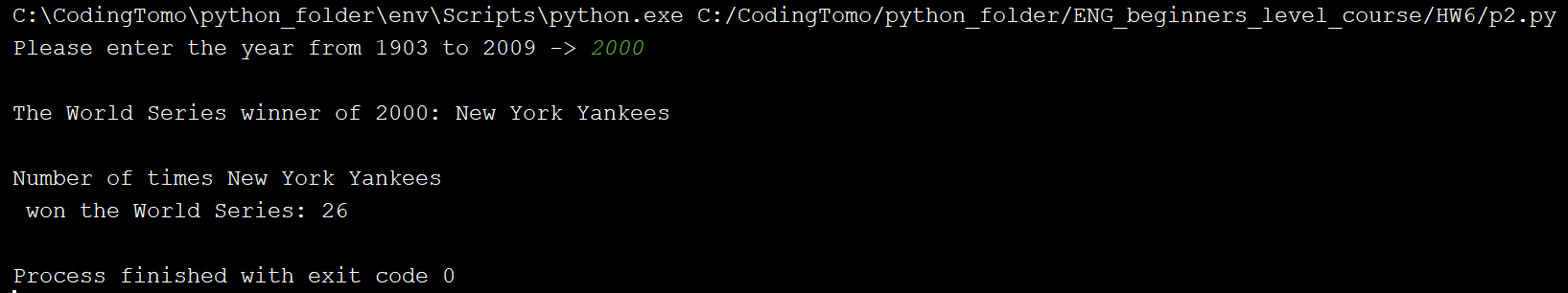
**--1953**



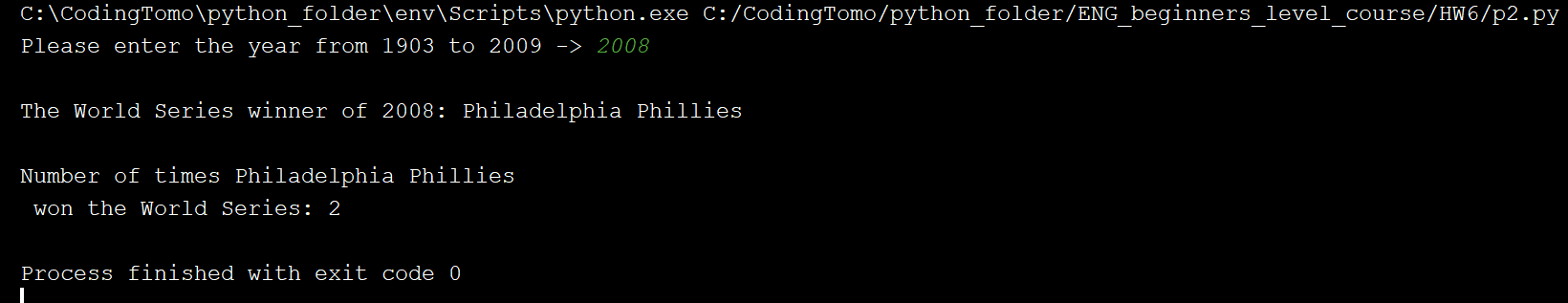
**--1990**



**--2000**



**--2008**

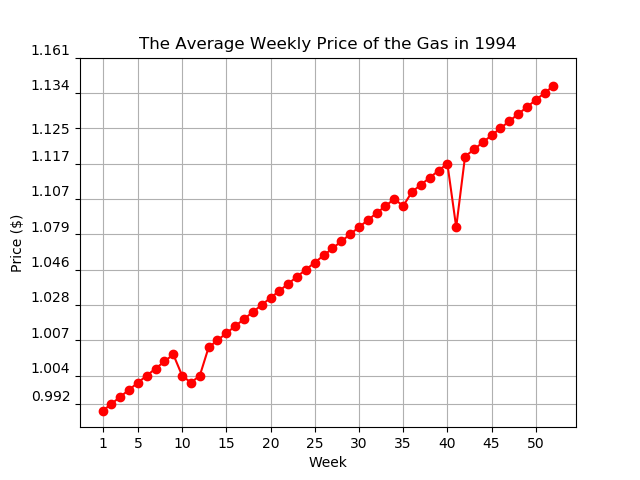


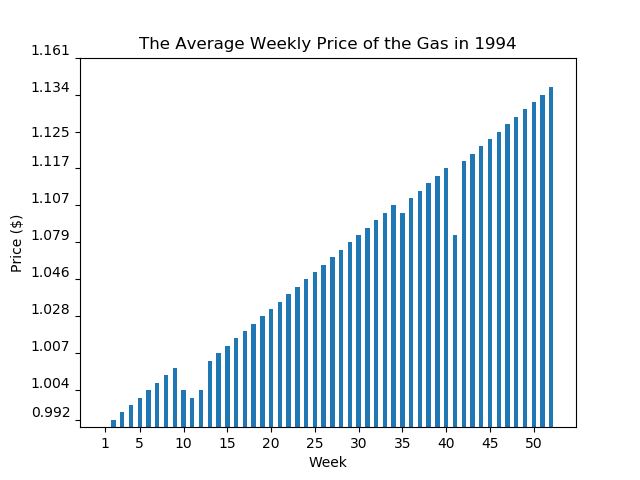
1. (**20 points**) In the attachments, you’ll find a text file named ‘1994\_weekly\_Gas\_Average.txt’. The file contains the average gas price for each week in the year 1994. (There are 52 lines in the file.) Using *matplotlib*, write two Python programs that reads the contents of the file then plots the data as a line graph and a bar chart. Be sure to display meaningful labels along the X and Y axes, as well as the tick marks.

**CODE:**

## PROBLEM #3  
# NAME: Tomoki Koike  
# DUE: 4/11/2019  
# DESCRIPTION: This program reads a file and creates a bar and line graph based on  
# the data from the text file.  
# STAND: Class of 2020  
##  
  
# Import modules  
import matplotlib.pyplot as plt  
  
# Functions  
# Function to read the data from the file  
def read\_file(file):  
 # Opening the flle  
 subject\_file = open(file**,** 'r')  
 # Reading the first line of the file  
 line = subject\_file.readline()  
 # Preallocate the list to store the data  
 data = []  
 # Loop to the read all the data from the file  
 while line != '':  
 # Appending the data read from the file to the list  
 data.append(line)  
 # Reading the next line  
 line = subject\_file.readline()  
 # Closing the file  
 subject\_file.close()  
 return data  
  
# Function to create an optimal tick list  
def create\_tick\_list(data**,** x\_tick):  
 tick\_list = []  
 counter = **0** sort\_data = sorted(data)  
 for x in x\_tick:  
 tick\_list.append(sort\_data[x-**1**])  
 return tick\_list  
  
def create\_lineGraph(data):  
 # Creating the x-values corresponding to the input data  
 x\_vals = range(**1,** len(data)+**1**)  
 x\_tick = [**1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50**]  
 x\_tick\_count = [**1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50**]  
 # The list for ticks  
 # most decimals  
 y\_tick\_list = create\_tick\_list(data**,** x\_tick)  
 y\_tick\_list\_count = x\_tick\_count  
 # Plotting  
 plt.figure(**1**)  
 plt.plot(x\_vals**,** data**,** 'ro-')  
 # Setting the title  
 plt.title('The Average Weekly Price of the Gas in 1994')  
 # Putting x labels  
 plt.xlabel('Week')  
 plt.ylabel('Price ($)')  
 # Putting the ticks on the graph  
 plt.xticks(x\_tick\_count**,** x\_tick)  
 plt.yticks(y\_tick\_list\_count**,** y\_tick\_list)  
 # Putting the grids on the graph  
 plt.grid(True)  
 # Saving the graph as  
 plt.savefig('gas\_price\_1994\_line.png')  
 # Showing the graph  
 plt.show()  
 return  
  
# Function to create a bar chart  
def create\_barChart(data):  
 # Creating the x-values corresponding to the input data  
 x\_vals = range(**1,** len(data) + **1**)  
 x\_tick = [**1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50**]  
 x\_tick\_count = [**1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50**]  
 # The list for ticks  
 # most decimals  
 y\_tick\_list = create\_tick\_list(data**,** x\_tick)  
 y\_tick\_list\_count = x\_tick\_count  
 # Plotting  
 bar\_width = **0.5** plt.figure(**1**)  
 plt.bar(x\_vals**,** data**,** bar\_width)  
 # Setting the title  
 plt.title('The Average Weekly Price of the Gas in 1994')  
 # Putting x labels  
 plt.xlabel('Week')  
 plt.ylabel('Price ($)')  
 # Putting the ticks on the graph  
 plt.xticks(x\_tick**,** x\_tick\_count)  
 plt.yticks(y\_tick\_list\_count**,** y\_tick\_list)  
 # Saving the chart as a file  
 plt.savefig('gas\_price\_1994\_bar.png')  
 # Showing the chart  
 plt.show()  
 return  
  
# Main Function  
def main():  
 # Reading the file  
 data = read\_file('1994\_Weekly\_Gas\_Averages.txt')  
 # Creating the line graph for the data  
 create\_lineGraph(data)  
 # Creating the bar chart for the data  
 create\_barChart(data)  
  
# Calling out the main function  
main()

**GRAPH:**





1. (**15 points**) Write a program that creates a dictionary containing the U.S. states as keys, and their capitals as values. (Use the Internet to get a list of the states and their capitals.) The program should then randomly quiz the user by displaying the name of a state and asking the user to enter that state’s capital. The program should keep a count of the number of correct and incorrect responses. (You need import random module in the program.)

**(When you run your program, try 10 states, and take screen shot of the results.)**

**CODE:**

## PROBLEM #4  
# NAME: Tomoki Koike  
# DUE: 4/11/2019  
# DESCRIPTION: This program will create a dictionary containing the  
# names of the US states as the keys and their capital cities as their  
# values. Then the program will enable the user to do a state and capital  
# name matching quiz.  
# STAND: Class of 2020  
##  
  
# Import modules  
import random as rand  
  
# Functions  
  
# Function for input validation  
def getValid(prompt):  
 while True:  
 try:  
 # trying input with without any conditions at first  
 this = input(prompt)  
 except ValueError:  
 # Prints the user to input again since the input was not valid  
 print('Sorry, could not understand. Please enter again.')  
 continue  
 # For when the input is a number  
 if (this.lstrip('-').replace('.'**,**''**,1**).isdigit()):  
 # For when the input is a number  
 if this.find('.') != -**1**:  
 # Ruling out float inputs as errors  
 print('Error. Please enter a positive INTEGER value.')  
 continue  
 elif int(this) > **0**:  
 # Valid input (a positive integer)  
 break  
 else:  
 # Invalid input  
 print('Error. Please enter a positive integer value.')  
 continue  
 else:  
 # For when the input is the file name (string)  
 if this.endswith('.txt'):  
 # Valid input for .txt extension for file name  
 break  
 else:  
 # Invalid file extension  
 print('Error. Please enter a valid file name with .txt extension.')  
 return this  
  
# Function to read the data from the file  
def read\_file(file):  
 # Opening the flle  
 subject\_file = open(file**,** 'r')  
 # Reading the first line of the file  
 line = subject\_file.readline()  
 # Preallocate the list to store the data  
 data = []  
 # Loop to the read all the data from the file  
 while line != '':  
 # Appending the data read from the file to the list  
 data.append(line.strip())  
 # Reading the next line  
 line = subject\_file.readline()  
 # Closing the file  
 subject\_file.close()  
 return data  
  
# Fucntion to create the dictionary  
def create\_dict(key\_list**,** value\_list):  
 return dict(zip(key\_list**,** value\_list))  
  
# Function to start the quiz  
def start\_quiz(user\_reply):  
 if user\_reply == 'y':  
 launcher = **1** else:  
 launcher = **0** return launcher  
  
def quiz\_main(launcher**,** US\_state\_capital\_dict**,** US\_state\_list):  
 if launcher == **1**:  
 rand.shuffle(US\_state\_list)  
 for x in range(len(US\_state\_capital\_dict)):  
 # Choosing a random key  
 chosen\_key = US\_state\_list[x]  
 # Asking the question  
 str1 = 'What is the capital city of '  
 str2 = chosen\_key  
 str3 = ' ? -> '  
 strAll = [str1**,** str2**,** str3]  
 user\_answer = input(''.join(strAll))  
 # Initialize a counter to keep track of the number of questions that the user answered correctly  
 right\_counter = **0** wrong\_counter = **0** # Evaluating the user's answer  
 if user\_answer == US\_state\_capital\_dict[chosen\_key]:  
 print('That is correct!')  
 right\_counter += **1** else:  
 print('Incorrect! The right answer is {0}!'.format(US\_state\_capital\_dict[chosen\_key]))  
 wrong\_counter += **1** # Displaying the result of the quiz  
 print('You got {0} out of 50 correct! You got a score of {1}!'.format(right\_counter**,** right\_counter/**50**\***100**))  
 else:  
 print('Too bad you''re not taking the quiz. Please come again.')  
 return  
  
# Main  
def main():  
 # Creating the dictionary with the US states and the state capitals  
 US\_states = read\_file(getValid('Enter the file name from where you would like to retrieve the name of the states -> '))  
 US\_capitals = read\_file(getValid('Enter the file name from where you would like to retireve the name of the state capitals -> '))  
 US\_state\_capital\_dict = create\_dict(US\_states**,** US\_capitals)  
 print()  
 # Starting the quiz  
 launcher = start\_quiz(input('Would you like to start the quiz? (y/n) -> '))  
 quiz\_main(launcher**,** US\_state\_capital\_dict**,** US\_states)  
  
main()

**SAMPLE EXECUTION:**

