|  |  |
| --- | --- |
| **NAME:** | **DUE DATE:** |
| **Tomoki Koike** | **4 April 2019** |

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

**Week 5 – Programming Exercises**

1. (**20 points** **Random number File Writer and Reader**)
2. Write a Python program that writes a series of random numbers to a file. Each random number should be in the range of 1 through 500. The application should let the user specify how many random numbers the file will hold.
3. Write another program that reads the random numbers from the file you create from a), displays the numbers, then display the following data:
   * The total of the numbers
   * The number of random numbers read from the file

**>>HEADER**

## PROBLEM #1

# NAME: Tomoki Koike

# DUE: 4/4/2019

# DESCRIPTION: This program is separated into 2 parts. In the first part, the user

# will open a new file and add arbitrary amount of random numbers in the range of 1

# to 500. Then, in the second part, there will be a program which will read the file

# that was created in the previous part of this program and calculate the sum of the

# random numbers and also count the total amount of numbers in that file.

# (with input validations)

# STAND: Class of 2020

##

**>>FUNCTIONS:**

# 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)

revStr = this[::-1]

if revStr[0:4] == '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

# Functions for Part (a)

import random as rand

# Function to create and write a file with random numbers from 1 to 500

def create\_randomNum\_file(totNum, fileName):

# Convert string to int

totNum = int(totNum)

# Holding the file name

file = fileName

# Openning a file or if the file does not exist creates a new one

newFile = open(fileName, 'w')

# Loop to write random numbers in the interval of [1, 500]

for x in range(totNum):

# Feeding a new seed

rand.seed(x)

# Generating a random float number

value = rand.random()

# Scaling the number to be in the interval of [1, 500]

scaledValue = 1 + (value \* (500 - 1))

# Writing the scaled value onto the file as a string (not forget to indent the line)

newFile.write('{}\n'.format(scaledValue))

# Closing the file to prevent future errors

newFile.close()

return file

# Functions for part (b)

# Function to read the file that has been created in the previous part and calculating the sum of the numbers

def sum\_file\_data(fileName):

# Opening the file in reading mode

subject\_file = open(fileName, 'r')

# Read the first line of the file

lineNum = subject\_file.readline()

# List to store all the numbers read from the file

numbers = []

# Loop to see if it is the end of the file or not

while lineNum != '':

# Convert number to float

num = float(lineNum)

# Appending the number to the number list

numbers.append(num)

# Printing the number

print(num)

# Reading the next line

lineNum = subject\_file.readline()

# Closing the file

subject\_file.close()

# Printing the sum of the numbers in the file

print('The total number in the file is {0:f}'.format(sum(numbers)))

print('The number of random numbers read from the file is {0:d}'.format(len(numbers)))

return

**>>MAIN:**

# MAIN

# Part (a)

# Creating a new file with random numbers inside

file = create\_randomNum\_file(getValid('Enter a number of random numbers to put into your file -> '), getValid('Enter the file name -> '))

# Part (b)

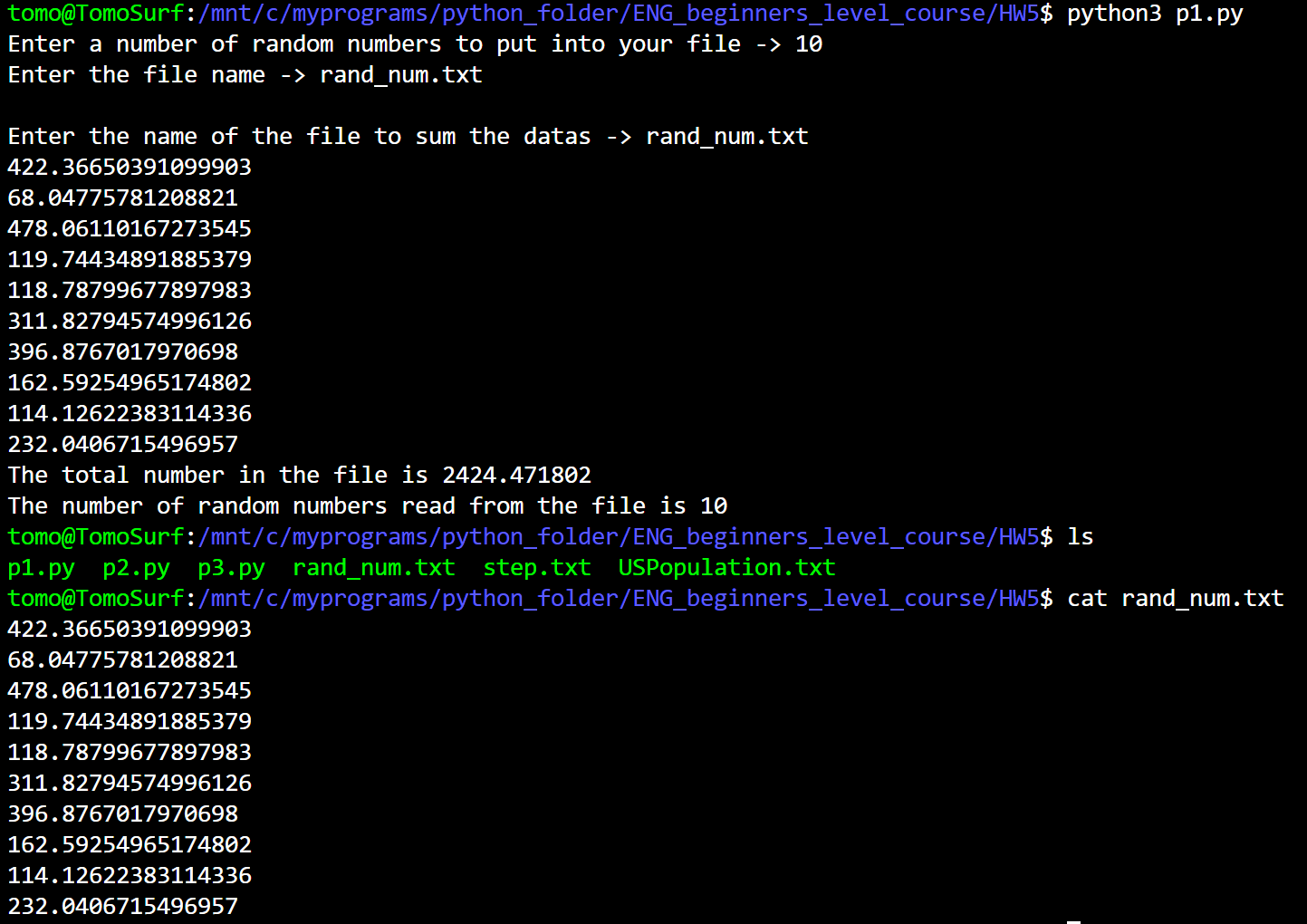
# Summing the numbers in the file you have created with the previous command

print()

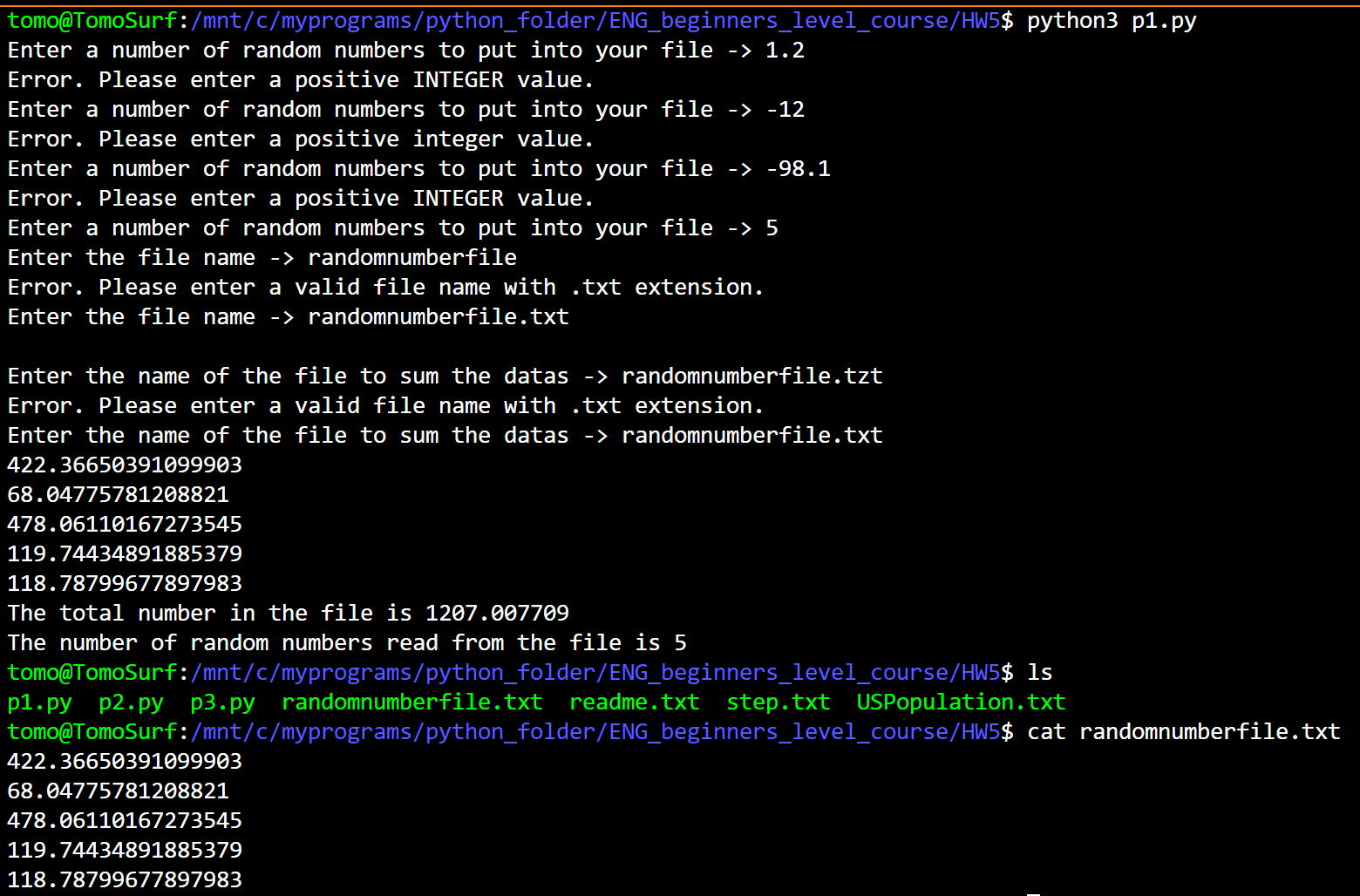
sum\_file\_data(getValid('Enter the name of the file to sum the datas -> '))

**>>SAMPLE EXECUTION:**

**Valid Input Case**



**Invalid Input Case**



1. (**15 points** **Average Steps Taken**) A Personal Fitness Tracker is a wearable device that tracks your physical activity, calories burned, heart rate, sleeping patterns, and so on. One common physical activity that most of these devices track is the number of steps you take each day.

You’ll find a file named steps.txt in the attachment. The steps.txt file contains the number of steps a person has taken each day for a year. There are 365 lines in the file, and each line contains the number of steps taken during a day. (The first line is the number of steps taken on January 1st, the second line is the number of steps taken on January 2nd, and so forth.) Write a Python program that read the file, then displays the average number of steps taken for each month. (The data is from a year that was not a leap year, so February has 28 days.)

**>>HEADER:**

## PROBLEM #2

# NAME: Tomoki Koike

# DUE: 4/4/2019

# DESCRIPTION: This program will read a file containing the number of steps

# made by a certain individual for one year. The program will output the average

# steps for each month

# (with input validations)

# STAND: Class of 2020

##

**>>MODULES**

# Modules

# Importing calendar module

import calendar

# Importing mean from statistics module

from statistics import mean

**>>FUNCTION:**

# 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)

revStr = this[::-1]

if revStr[0:4] == '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

# This function reads the file into one list with all the step per day data

def read\_file(fileName):

# Open the file to read

file = open(fileName, 'r')

# Read the first line of the file

read\_line = file.readline()

# Preallocating list to hold all step values

steps\_list = []

# Loop to read all the values

while read\_line != '':

# Append the read string number into the list

steps\_list.append(int(read\_line))

# Reading the next line

read\_line = file.readline()

# Closing the open file

file.close()

return steps\_list

# This function generates the average steps for each month

def avg\_month\_step(steps\_list):

# List with the number of days in the months in order from Jan to Dec

monthDays = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

# Initialize Counter

counter = 1

for x in monthDays:

sliced\_list = steps\_list[0:x+1]

# Calculate mean for the month

step\_month\_mean = mean(sliced\_list)

print('{0:10}: {1:6>d}'.format(calendar.month\_name[counter], int(step\_month\_mean)))

# Deleting unnecesary indecies from the list with steps

del steps\_list[0:x+1]

# Increment counter

counter += 1

return

**>>MAIN:**

# Main

# Reading the data text file with steps

list\_of\_steps = read\_file(getValid('Type the name of the file you would like to read including the file extension -> '))

# Calculating and output the average for each month

print()

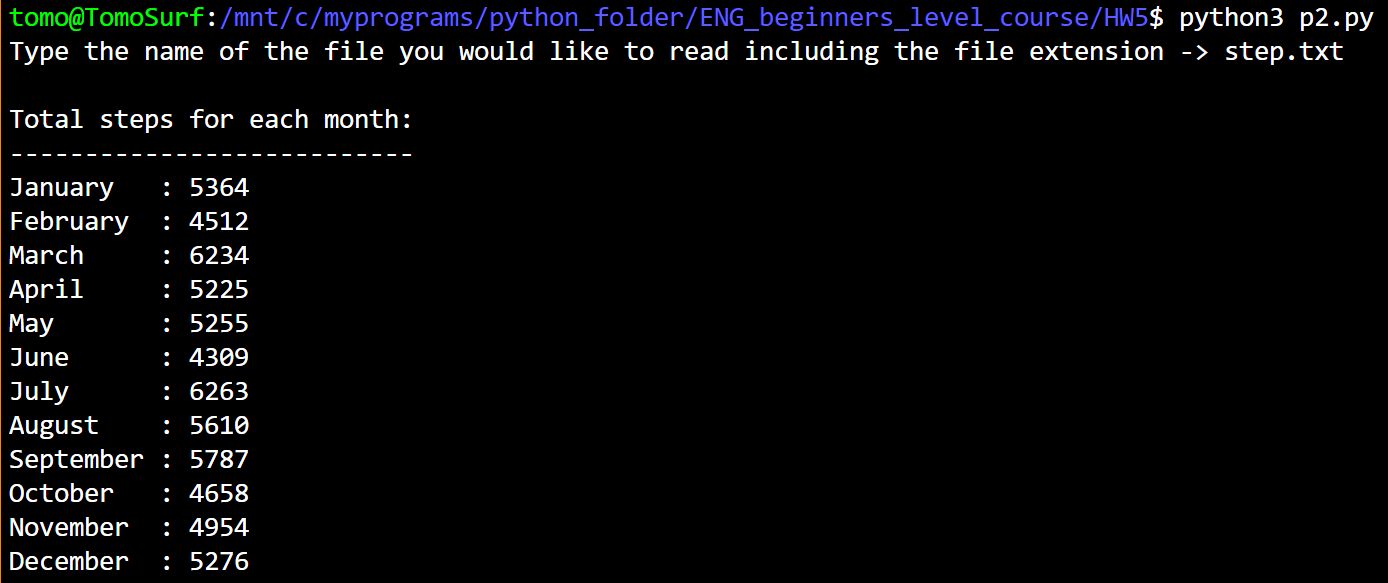
print('Total steps for each month:')

print('---------------------------')

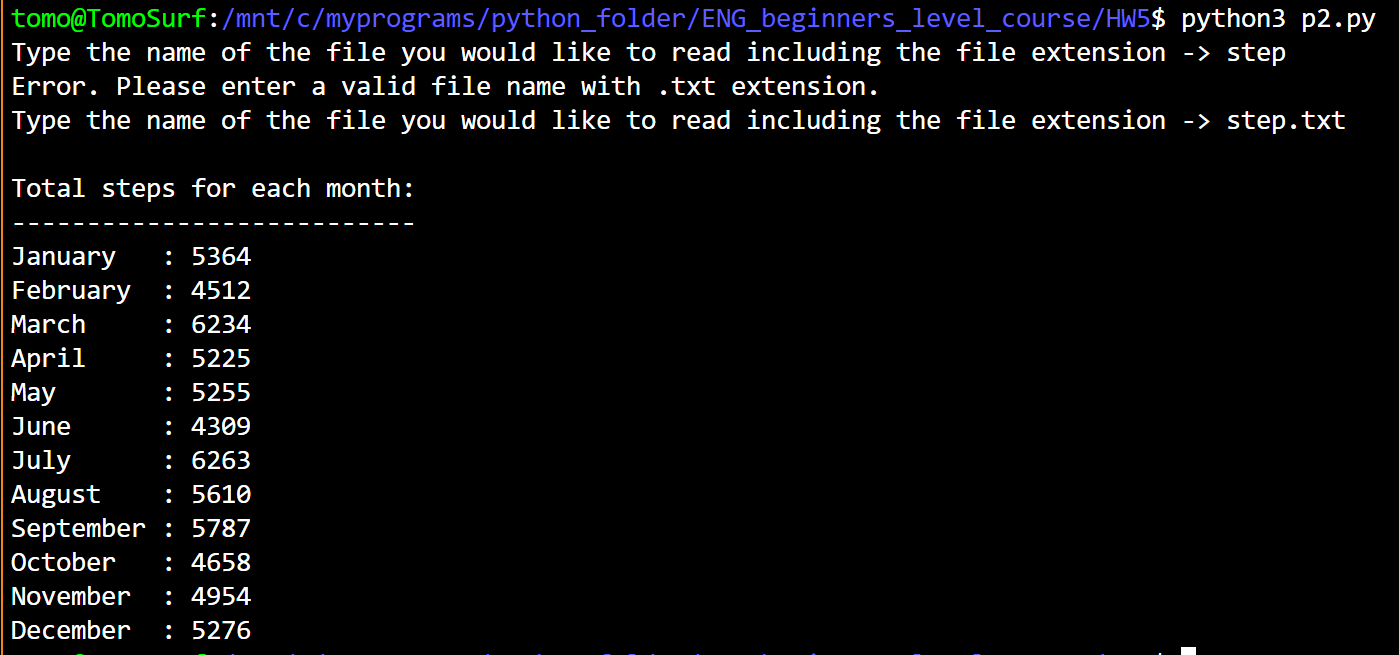
avg\_month\_step(list\_of\_steps)

**>>SAMPLE EXECUTION:**

**Valid Input**



**Invalid Input**



1. **(15 points, Rainfall Statistics)** Design a program that lets the user enter the total rainfall for each 12 months into a list. The program should calculate the display the total rainfall for the year, the average monthly rainfall, the months with the highest and lowest amounts. Assume that February has 28 days.

**(Test your program by inputting rainfalls for each month, from January to December accordingly: 2.8, 3.2, 4.5, 4.8, 6.9, 7.8, 12.0, 12.2, 5.5, 5.8, 3.0, 2.9)**

**>>CODE:**

## PROBLEM #1

# NAME: Tomoki Koike

# DUE: 4/4/2019

# DESCRIPTION: This program will allow the user to input the total

# rainfall for each 12 months into a list, and then will compute the

# total rainfall of the year, monthly average rainfall, and the maximum

# and minimum rainfall.

# (with input validations)

# STAND: Class of 2020

##

# Modules

# Importing calendar module

import calendar

# Importing mean from statistics module

from statistics import mean

# 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

# Main

# Inputing the rainfall data for 12 months

# Preallocating a list

rainfall\_list = []

for x in range(12):

rainfall\_list.append(getValid('Enter the total rainfall for {0} -> '.format(calendar.month\_name[x+1])))

# Computing the sum, average, max, and min values

total\_rainfall = sum(rainfall\_list)

avg\_rainfall = mean(rainfall\_list)

max\_rainfall = max(rainfall\_list)

max\_index = rainfall\_list.index(max\_rainfall)

min\_rainfall = min(rainfall\_list)

min\_index = rainfall\_list.index(min\_rainfall)

# Printing out the results

print()

print('The yearly total rainfall: {0:.2f}'.format(total\_rainfall))

print('The monthly average rainfall of this year: {0:.2f}'.format(avg\_rainfall))

print('The maximum rainfall of this year: {0:.2f} in {1}'.format(max\_rainfall, calendar.month\_name[max\_index+1]))

print('The minimum rainfall of this year: {0:.2f} in {1}'.format(min\_rainfall, calendar.month\_name[min\_index+1]))

**>>SAMPLE EXECUTION:**



1. **(15 points, Population Data)** The attached file ‘USPopulation.txt’ contains the midyear population of the United States, in thousands, during the years 1950 through 1990. The first line in the file contains the population for 1950, the second line contains the population for 1951, and so forth.

Write a program that reads the file’s contents into a list. The program should display the following data:

* The average annual change in population during the time period
* The year with the greatest increase in population during the time period
* The year with the smallest increase in population during the time period.

**>>CODE:**

## PROBLEM #1  
# NAME: Tomoki Koike  
# DUE: 4/4/2019  
# DESCRIPTION: This program is designed to read the file USPopulation.txt and compile the  
# data into a list, then outputs such as average, max, and min will be computed by manipulating  
# the list to show annual change in population from the data.  
# STAND: Class of 2020  
##  
  
# Modules  
from statistics import mean  
  
# 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)  
 revStr = this[::-**1**]  
 if revStr[**0**:**4**] == '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 file data into a list  
def read\_data\_into\_list(file):  
 # Opening file in reading mode  
 newfile = open(file**,** 'r')  
 # Read the first line of the file  
 line = newfile.readline()  
 # Preallocate a list to hold all the data in the file  
 data\_list = []  
 while line != '':  
 # Convert the read data as an integer  
 data = int(line)  
 # Appending the data into the list  
 data\_list.append(data)  
 # Reading the next line  
 line = newfile.readline()  
 # Closing the file  
 newfile.close()  
 return data\_list  
  
# Function to manipulate the list into a list of annual change of population  
def annual\_pop\_change(pop\_list):  
 # Preallocating a new list  
 pop\_change\_list = []  
 # Preallocating the difference and temporary holder variable  
 diff = **0** temp = **0** for x in range(len(pop\_list)):  
 # Calculating the difference  
 if x >= **1**:  
 diff = pop\_list[x] - temp  
 # Holding the value in memory  
 temp = pop\_list[x]  
 # Appending the difference value in the list  
 pop\_change\_list.append(diff)  
 return pop\_change\_list  
  
# Function to calculate the average population change  
def avg\_cal(subject\_list):  
 return mean(subject\_list)  
  
# Main Function  
def main():  
 # Reading the file  
 poplist = read\_data\_into\_list(getValid('Enter the name of the data file that you need -> '))  
 # Manipulating the list  
 pop\_change\_list = annual\_pop\_change(poplist)  
 # Calculating the average  
 avgPopChange = avg\_cal(pop\_change\_list)  
 # Calculating the max and min values  
 max\_pop\_change = max(pop\_change\_list)  
 min\_pop\_change = min(pop\_change\_list)  
 # Printing out the results  
 print()  
 print('The average annual population change: {0:.2f}'.format(avgPopChange))  
 print('The maximum annual population change in the period: {0}'.format(max\_pop\_change))  
 print('Tne minimum annual population change in the period: {0}'.format(min\_pop\_change))  
 return  
  
# Calling out the main function  
main()

**>>SAMPLE EXECUTION:**

