**CIS 229 – Python Programming – Programming Assignment**

**Number File Generator**

Overview

In this assignment, the student will write a script that generated data files with random numbers based on the provided, constant specifications.

When completing this assignment, the student should demonstrate mastery of the following concepts:

* Pseudo-random Number Generation
* File I/O
* Exception Handling
* Arbitrary Sized Data Files

Assignment

In this program, you will demonstrate your knowledge of File IO and Exception Handling by writing a program that generates a file filled with an arbitrary number of pseudo-random integers and then reads the file to determine the number of elements written along with the average of the numbers. Begin your program with the following shell:

# Global Variables

FILE = "A06-DataFile.dat" # The name of the file this program writes to

LOWER\_BOUND = 0 # Lower boundary for random number value

UPPER\_BOUND = 10000 # Upper boundary for random number value

NUM\_ELEMENTS = 1000 # Prove program handles arbitrary file lengths

def main():

# generate the random number file

generateRandomNumberFile(NUM\_ELEMENTS)

# find the average value of all numbers in file

elementCount, averageValue = determineAverageValue(FILE)

# display results

displayCalculations(elementCount, averageValue)

def generateRandomNumberFile(NUM\_ELEMENTS):

""" IMPLIMENT ME """

def determineAverageValue(filename):

""" IMPLIMENT ME """

def displayCalculations(tally, average):

""" IMPLIMENT ME """

A number of global declarations have been given to get you started. The random numbers should be written to a file with the specified filename. This will be accomplished by a single call to the function generateRandomNumberFile(). This function will take a single argument, which contains the number of numbers to be written to the file. Inside generateRandomNumberFile() you should create a new file and fill it with random numbers on the interval [LOWER\_BOUND, UPPER\_BOUND]. Delimit the items in the data file with a newline (\n) character.

Once the file has been created, complete the implementation for determineAverageValue(). determineAverageValue(), take a single argument that contains the name of the file to be analyzed. The specified file is opened and the numbers inside are accumulated, counted, and have their average determined. The number of detected elements is returned as an integer representation in the first return value and the average value is returned as a floating-point representation as the second return value.

Since we are creating the data file, we know that it will contain a series of integers delimited by newline characters. However, in the event that a non-integer representation is found in the file, determineAverageValue() would throw an exception. When writing determineAverageValue(), make sure to include exception handling code for this exact event. If a non-integer representation is found, display a message on the screen and continue reading items (the program should not stop). Make sure not to count lines that contain non-numeric data. Test this functionality by commenting out the line of code that creates the random numbers and changing a numeric value in a pre-existing file to something that is not a number. Make sure the program handles this sort of situation gracefully.

After lines in the file have been counted, display the results of the average using displayCalculations(). Display calculations takes two parameters, the first being the tally of readable lines in the file, and the second being resulting average. Make your output look as close as possible to the sample below.

SAMPLE OUTPUT

There were 1000 elements found in the file.

The average value of the elements was determined to be 5146.584.

Assessment

This assignment will be assessed based on the provided grading rubric.