CSE 231 **Fall 2015**

Computer Project #2

Assignment Overview

This assignment focuses on the design, implementation and testing of a Python program that analyzes distributions generated by a pseudo-random number generator in the `random` module (see below).

It is worth 30 points (3% of course grade) and must be completed no later than 11:59 PM on Monday, September 21.

Assignment Deliverable

The deliverable for this assignment is the following file:

`proj02.py` – the source code for your Python program

Be sure to use the specified file name and to submit it for grading via the **handin system** before the project deadline.

Assignment Background

Python's `random` module includes pseudo-random number generators for various distributions. One of them generates *Gaussian distributions*—a real-valued *normal* distribution often used in science to represent random events with a bell-shaped distribution. A Gaussian distribution is characterized by two values: a mean μ and a standard deviation σ. You can find more information about Gaussian distributions at:

https://en.wikipedia.org/wiki/Normal_distribution

Assignment Specifications

You will develop a Python program that analyzes the Gaussian distribution algorithm implemented by the `gauss` function in the `random` module.

1. The program will prompt the user to enter the following information:

   a. The desired mean μ (a `float`)
   b. The desired standard deviation σ (a positive `float`)
   c. The number of values *num* that are to be generated (a positive `int`)

   The program must prompt the user (and accept input) in the exact order shown above. It will not be tested with user inputs that do not meet these specifications.

2. The program will call function `gauss` with the user-supplied mean and standard deviation a total of *num* times.

3. For the *num* calls to function `gauss`, your program will compute:

    a. The range of the generated values.
    b. The actual mean (i.e., the average) of the generated values.
    c. The percentages of the generated values that are:
        -- more than two standard deviations above the desired mean
        -- above and within two standard deviations of the desired mean
        -- above and within one standard deviation of the desired mean
        -- at the desired mean (within 1.0e-6 of the desired mean)
        -- below and within one standard deviation of the desired mean
        -- below and within two standard deviations of the desired mean
        -- more than two standard deviations below the desired mean

4. Your program will generate output as follows:

    a. A brief description of what the program does.
    b. A synopsis of the inputs that it received. Floating point inputs will be rounded to two decimal places of accuracy when displayed.
    c. The results of the analyses, appropriately labeled and in the order listed above. Values indicating the range and average will be rounded to two decimal places of accuracy when displayed. Percentages will be rounded to whole numbers and displayed without any decimal point.

Assignment Notes

1. To clarify the project specifications, sample output is provided at the end of this document.

2. The coding standard for CSE 231 is posted on the course website:

    http://www.cse.msu.edu/~cse231/General/coding.standard.html

3. Items 1-7 of the Coding Standard will be enforced for this project.

4. Your program must prompt the user (and accept input) in the exact order given above.

5. All calculations will be done at full precision (i.e., without rounding).

6. Floating point values will be rounded to two decimal places of accuracy when displayed. As shown in the sample output, rounded values will be displayed with one or two digits after the decimal point. For example, a value that rounds to 2.30 will be displayed as 2.3; whereas a value that rounds to 2.32 will be displayed as 2.32.

The documentation about function `round` is available at:

https://docs.python.org/3.4/library/functions.html#round

7. The range of the generated values is defined by the minimum and maximum of the generated values—that is, if the minimum generated value is 4.33 and the maximum generated value is 54.04, then the values ranged from 4.33 to 54.04 (see the sample output).

8. The program named `example.py` in the project directory contains typical usages of some of the Python features you will use in this project. For instance, that program demonstrates the use of function `round` from the standard library and function `gauss` from the `random` module.

9. As shown in `example.py`, the following two lines must appear in your program:

```
import random
random.seed( 0 )
```

   The first statement allows your program to access the `random` module, including function `gauss`. The second statement initializes the random number generator so that the same sequence of values is generated every time you execute your program, which is useful when you develop and test your program (and when your TA grades it).

Suggested Procedure

- *Solve the problem using pencil and paper first.* You cannot write a program until you have figured out how to solve the problem. This first step can be done collaboratively with another student. However, once the discussion turns to Python specifics and the subsequent writing of Python statements, you must work on your own.

- Cycle through the following steps to incrementally develop your program:

  - Edit your program to add new capabilities (e.g. the first iteration might just gather the inputs and then call the `gauss` function the required number of times; the second iteration might add in calculation of the range; etc.)

  - Run the program and fix any errors. A good method to check for errors is to display the values generated when testing the program and run some tests that generate only a handful of values. That will allow you to easily check that you are getting the correct averages and percentages using a calculator. (You should comment out or delete the statements that you add for debugging purposes before you hand in the project for the last time.)

  - Use the **handin** system to submit the current version of your program.

- Be sure to use the **handin** system to submit the final version of your program.

- Be sure to log out when you leave the room, if you're working in a public lab.

The last version of your solution is the program which will be graded by your TA.

*You should use the **handin** system to back up your partial solutions, especially if you are working close to the project deadline. That is the easiest way to ensure that you won't lose significant portions of your work if your machine fails or there are other last-minute problems.*

*You would also be wise to save a copy of your completed program in your CSE file space (the H: drive on the lab machines) **before** the project deadline. If you write your program at home and turn it in from home, you should copy it to your CSE file space **before** the deadline.*

*In case of problems with electronic submission, an archived copy in the **handin** system or the CSE file space is the only acceptable evidence of completion.*

Sample Output

```
This program analyzes Python's Gaussian distribution algorithm.

Enter the desired mean: 50.0
Enter the desired standard deviation: 2.3
Enter the number of values to generate: 100

The requested mean:  50.0
The requested standard deviation:  2.3
The number of values generated:  100

The values ranged from 44.66 to 56.01
The actual mean was 49.81

The values distributed as follows:
  4 percent were higher than the mean by more than two standard deviations
  39 percent were higher than the mean and within two standard deviations
  27 percent were higher than the mean and within one standard deviation
  0 percent were at the mean
  39 percent were lower than the mean and within one standard deviation
  54 percent were lower than the mean and within two standard deviations
  3 percent were lower than the mean by more than two standard deviations
```