Mr. Kramer – CS 1030 – Python Project #2

Before starting to code your solutions, write pseudocode and/or draw flowcharts. Do not start coding until you have a deeper understanding of what the solutions should look like. Write pseudocode from which you can directly code your program.

**1. Compute a Grade Point Average**

The letter grades below are converted to points for the purpose of computing a grade point average (GPA). Note that this particular table does not include a D-. The name of your Python source code file will be FirstnameLastname\_S\_02\_01.py, where S = your single-digit section number.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Letter | Points |  | Letter | Points |
| A+ | 4.2 |  | C+ | 2.8 |
| A | 4.0 |  | C | 2.2 |
| A- | 3.9 |  | C- | 2.0 |
| B+ | 3.7 |  | D+ | 1.8 |
| B | 3.2 |  | D | 1.2 |
| B- | 3.0 |  | F | 0 |

Write a program that converts a list of letter grades entered by a user to a GPA for several GPA calculations. Also calculate the overall GPA for all grades entered.

1.1 Loop until the user enters ‘quit’ at which time proceed at step 1.6.

1.2 Do an inner loop until the user enters a blank line.

1.3 Prompt the user for the next grade. If the grade is invalid (e.g., E+, +A, G, HEY, or Q-), ignore this grade, inform the user and prompt again.

1.4 Here, the user entered a valid grade. Using a list or a series of if-elif-else statements, translate the grade to points and accumulate the total points and the count so you can calculate an overall average later. Prompt for the next grade.

1.5 If the user enters a blank line, calculate and print the GPA for the grades just entered. Your program should correctly handle the case where the user enters no grades. In that case, print “No GPA calculated” and resume at step 1.2.

1.6 Calculate and print the overall average of all GPAs that were processed.

The sum of all points processed

Overall average = ————————————————

The number of all grades processed

What information do you have to track to be able to produce the overall average?

This problem is derived from *The Python Workbook*, by Ben Stephenson, page 23, exercise 51 and page 31, exercise 66.

**2. Coin Flip Simulation.**

Flip a coin until three consecutive occurrences of the same side of the coin are tossed (three heads in a row, HHH, or three tails in a row, TTT). You will simulate the tossing of a coin and find the average number of flips, the minimum number of flips and the maximum number of flips in all simulations. The name of your Python code file will be FirstnameLastname\_S\_02\_02.py, where S = your single-digit section number.

Use Python’s randint() function to generate a 1 or 2 randomly representing H or T. You’ll need to import that function from the ‘random’ module with this statement before your code begins:

from random import randint

2.1 Ask the user how many simulations they want to run, say N. If the user presses the <Enter> key or enters 0 or a negative number, exit the program.

2.2 Loop N times.

2.3 Loop until you find three heads or three tails in a row, HHH or TTT, respectively.

2.4 Start flipping coins. Use randint(1,2) to generate a 1 or 2 for heads and tails respectively.

2.5 Count flips and check for HHH or TTT.

2.6 When you get HHH or TTT, print the flips in the current simulation and accumulate the number of flips, minimum number of flips to get three in a row and the highest number of flips to get three in a row. The flips should be displayed with Hs and Ts with a space between each flip, like H T T H H H

2.7 Go on to the next simulation, if there is one.

2.8 Print the average number of flips, the minimum number of flips and maximum number of flips.

2.9 Resume at step 2.1.

This problem is based on *The Python Workbook*, by Ben Stephenson, page 37, exercise 80.

**What and how to submit**

You now have two Python files:

FirstnameLastname\_S\_02\_01.py

FirstnameLastname\_S\_02\_02.py

where S = your section number. If your IDE supports it, turn on line numbering before printing. Print your programs **and their output** and staple them together in order. Write this information at the upper right of the first page: name, section and project #.

**How your programs are evaluated**

1. Do they work according to the specifications?
2. Are the input prompts correct?
3. Are the programs documented?
4. Do the programs follow Python and file naming standards?