Mr. Kramer – CS 1030 – Python Project #3

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. I am deliberately not giving step-by-step specifications as it is good practice for you to do that on your own. Before writing the code, do as the previous sentence says, particularly writing pseudocode. See “What and how to submit” near the end of this document for Python file naming conventions.

**1. Temperature Conversion Table – Loop Practice**

Write a program that displays a temperature conversion table for degrees Celsius and degrees Fahrenheit. (Find the Celsius to Fahrenheit conversion formula on the Internet.) The table should include rows for all temperatures between 0 and 100 degrees Celsius that are multiples of 10 degrees Celsius. Include a heading and a title. Here’s how the table might look:

Celsius to Fahrenheit

Conversion Table

Celsius Fahrenheit

0 32

10 50

and so on.

This problem is derived from *The Python Workbook*, by Ben Stephenson, page 30, exercise 63.

**2. Below and Above Average – List Practice**

Write a program that reads numbers from a user and places them into a list. Stop entering numbers when the user enters a 0. Don’t put the 0 in the list. Display the average of all the numbers in the list. Then in three displays one after the other, display all numbers that are below the average, that equal the average, and that are above the average. Include an appropriate title at the start of each of the three lists. Allow for the possibility that there are no numbers below the average and/or no numbers above the average. (What does that mean about the numbers in the list?)

This problem is derived from *The Python Workbook*, by Ben Stephenson, page 52, exercise 112.

**3. Canadian Postal Codes – Dictionary Practice**

In a Canadian postal code (zip code), the first, third and fifth characters are letters, while the second, fourth and sixth characters are digits. There is a space between the third and fourth characters for a total length of 7. A Canadian province can be determined from the first character of a postal code, as shown in the table below. No valid postal codes currently begin with D, F, I, O, Q, U, W, or Z. Create a dictionary from this table:

|  |  |
| --- | --- |
| Province | First character(s) |
| Newfoundland | A |
| Nova Scotia | B |
| Prince Edward Island | C |
| New Brunswick | E |
| Quebec | G, H and J |
| Ontario | K, L, M, N, and P |
| Manitoba | R |
| Saskatchewan | S |
| Alberta | T |
| British Columbia | V |
| Nunavut | X |
| Northwest Territories | X |
| Yukon | Y |

Note that the letter X can mean Nunavut or Northwest Territories. The second character indicates rural or urban addresses: 0 means a rural address, anything else an urban address.

Write a program that loops to read a postal code from a user. The program ends when the user presses just the <Enter> key in response to your prompt. If the user entered a postal code, validate it first. The string must be seven characters long (otherwise it is invalid) and cannot begin with the letters D, F, I, O, Q, U, W, or Z (which would make the entry invalid). If the first letter is valid, display the province associated with the letter and whether the address is urban or rural. For example, for T2N 1N4, your program should display that the address is an urban one in Alberta. For X0A 1B2, your program should display that the address is a rural one in Nunavut or Northwest Territories. Display a meaningful message if the postal code is invalid.

This problem is derived from *The Python Workbook*, by Ben Stephenson, page 64, exercise 132.

**What and how to submit**

You now have three Python files:

FirstnameLastname\_(Your section # here)\_03\_01.py

FirstnameLastname\_(Your section # here)\_03\_02.py

FirstnameLastname\_(Your section # here)\_03\_03.py

If your IDE supports it, turn on line numbering before printing. Print your programs **and their output** and staple them together in order (program, its output, program, its output, …). Write this information at the upper right of the first page: name, section and project number. You will lose points if you don’t follow this submission guide.

**How your programs are evaluated**

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