CS-521 Homework Assignment 4

Complete the 5 python coding problems below. Each is worth 10 points for a total of 50 points (100%).

Individual programs must be named with your BU email prefix (the part before @bu.edu) and the problem number. If your email is alex@bu.edu, then your first program in this assignment would be called: alex_hw_4_1.py

The programs must all be combined into a single zip file named with your email prefix and the assignment number. alex@bu.edu would name their submission alex_4.zip.

Style Requirements

For all assignments, follow the guidelines in the PEP8 Standards and Best Practices that have been shared to date, along with course specific requirements. Remember these to avoid minor deductions. In bold are new style requirements since the previous module.:

- **Name programs and zip container correctly**
    - **Among other things, names for regular programs and zip file must be all lower case.**
- Include a program docstring
- Stay under 80 characters on all code and comment lines
- Ask for input() with descriptive prompts telling users what is expected
    - **Especially remember to tell user what delimiter you're expecting when using split()**
- **Make sure to validate user input for data type, value range and other problem requirements**
- Print output that clearly explains what is being printed (where necessary)
    - In other words, don't just print a '5' unless it's clear what that 5 represents.
- **Import statements should be immediately after the program docstring**
- **Make sure to include # line comments (just a few in each program, don't go crazy)**
- **Use snake_case (lower case plus underscores) for variables**
- **CONSTANT variables must be all upper case and immediately following any import statements**

Do NOT use functions for this assignment.

Modules Allowed

You may only import the following modules into your programs and use their methods and attributes, unless first receiving special (and unlikely) permission from your facilitator:

- math
- operator
- os
- re
- string
- sys

CS-521 Homework Assignment 4

Programming Problems

The following includes concepts taught in Chapter 7

4.1 Write a python program to solve the following:

a. Create a constant list with the integers from 55 to 5, but only every 10th number (see example).
   - Set this up using the python range function
b. Generate a new list with same number of elements as the original list such that each integer in the new list is the sum of its nearest neighbors and itself from the original list.
   - The integers at the beginning and end of the list only have one nearest neighbor
c. Print both lists with descriptions. Your code should be able to work with an integer list of any size.

Example of Output:
```
Input List: [55, 45, 35, <etc>]
Result List: [100, 135, <etc>]
```

The following includes concepts taught in Chapter 9

4.2: Write the following python program.

a. Assign a sentence of at least 15 characters into a string constant
b. Perform analysis that counts the number of each letters and saves the result into a dictionary
   - The letters are the keys, and the counts are the values
   - Find the letter(s) that appears most frequently by evaluating the dictionary counts
c. Write print statements with appropriate descriptions for the items calculated in step 'b'.
   - Format print using the following examples.
   - Need to handle the cases of one most frequent letter vs. multiple letters.

Example Output #1
```
The string being analyzed is: "WAS IT A RAT I SAW"
1. Dictionary of letter counts: {'W': 2, 'A': 4, 'S': 2, 'I': 2, 'T': 2, 'R': 1}
2. Most frequent letter "A" appears 4 times.
```

Example Output #2
```
The string being analyzed is: "WWWAS IT A RAT I SAW"
1. Dictionary of letter counts: {'W': 2, 'A': 4, 'S': 2, 'I': 2, 'T': 2, 'R': 1}
2. Most frequent letters ['A', 'W'] appear 4 times.
```

CS-521 Homework Assignment 4

4.3 Write a python program that merges two lists into a dictionary as described below

a. Create 2 constant lists. One with first names and another of last names.
b. Validate that the last name list is the same size or larger than the first name list
   - If not, exit with an error message.
c. Use zip function to create a dictionary with the last names as keys and the first names as values
   - If there are more last names than first names, set the missing first names to None (not a string)
d. Print the input lists and generated dictionary with appropriate descriptions.

Example of Output:

```
First Names: ['Jane', 'John']
Last Names: ['Doe', 'Deer', 'Black']
Name Dictionary: {'Doe': 'Jane', 'Deer': 'John', 'Black': None}
```

4.4: Using my_dict = {a':15, 'c':18, 'b':20}, write a program to:

a. calculate and print all the keys as a list.
b. calculate and print all the values as formatted comma separated data (not as a list).
c. print all the keys and values pairs as formatted data (not as a dictionary).
d. sort in ascending order by **key** and print a list of tuples for all the keys and values pairs
e. sort in ascending order by **value** and print as formatted data all the keys and values pairs

Output must look as follows (obviously, without any data being hard-coded):

```
a. Keys: ['a', 'c', 'b']
b. Values: 15, 18, 20
c. Key value pairs: a: 15, c: 18, b: 20
d. Key value pairs ordered by key: [('a', 15), ('b', 20), ('c', 18)]
e. Key value pairs ordered by value: a: 15, c: 18, b: 20
```

CS-521 Homework Assignment 4

4.5: Create a program that:

a. creates a constant dictionary with keys for the characters '1234567890-.' and values that represent these characters as words.
   - For the decimal point, use 'point' as the value
   - For the negative sign, use 'negative' as the value

b. inside of an infinite loop
   - prompts a user for a number with an appropriate description
   - validates that a number was entered
     - ✓ if valid - break out of the loop
     - ✗ otherwise print error message and stay in loop to reprompt

c. convert the validated number to words using the dictionary created in step 'a'.

d. print the converted number words with an appropriate description

Notes:
- The program must only have one input function and work for any size number.
- If the user enters commas, tell them to try again without the commas.

Example Output #1
```
Enter a number: 123
As Text: One Two Three
```

Example Output #2
```
Enter a number: -123
As Text: Negative One Two Three
```

Example Output #3
```
Enter a number: 1234.76
As Text: One Two Three Four Point Seven Six
```

Example Output #4 – invalid Input
```
Enter a number: 1,000
Please try again without entering commas.
Enter a number: 1 thousand
"1 thousand" is not a valid number. Please try again
Enter a number: 1000.00
As Text: One Zero Zero Zero Point Zero Zero
```

Where to submit?

Click Assignments in the Navigation Area and then click on the title of the assignment to enter the submission area and upload the zip file with your response.