CS-521 Homework Assignment 3

Assignment Directions

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_3_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_3.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
- os
- re
- string
- sys

Programming Problems

The following includes concepts taught in Chapter 2

3.1: Write a Python program that does all the following steps:

a. Loops through the integers 2 – 130 (inclusive).
   - Use constants to set the beginning and ending of the range.

b. For each integer in the loop, count the number of integers that are:
   - odd numbers
   - even numbers
   - squares of integers
   - cubes of integers

c. When the loop is completed, print output as follows and using the Example below:
   - A title with the total range evaluated
   - For Odd and Even counts, print the totals and only the start and end of the range of the numbers in scope with three dots between them.
   - For Squares and Cubes, print the totals and a list of the numbers that meet the criteria.
   - Note that none of the numbers being printed should be hard coded.

Example of Output:

```
Checking numbers from 2 to 130
Odd (64): 3...129
Even (65): 2...130
Square (10): [4, 9, 16, 25, 36, 49, 64, 81, 100, 121]
Cube (4): [8, 27, 64, 125]
```

Notice that 64 is even, a square and a cube.

The following includes concepts taught in Chapter 4

3.2: Write a Python program does all the following steps:

a. Create a **docstring** constant with 3 sentence strings
b. Convert the docstring to a LIST with one element for each sentence in the docstring.
c. Using a loop calculate the number of:
- uppercase letters
- lowercase letters
- digits
- punctuation characters
    - **Spaces are NOT considered punctuation.**

d. Output the results from each loop in columns that are neatly formatted, centered, labeled, and underlined.
- Use Python 3's f-Strings or format() to solve.
- Follow the output format in the example below.
- The first column is the sentence number but starts counting at 1…. as humans do.

Hints:

- There is a nice function to split a string into a list based on a given delimiter
- Check out the library attribute string.punctuation for help in solving this problem.
- The centering format character is a ^

Example using first string from the tuple: The rain in #Spain in 2019, rained "mainly' on the plain.

Output Example (input 1 is in bold for illustration purposes):

```
#   # Upper   # Lower   # Digits  # Punct.
-   -------   --------  --------  --------
1       2         36        4         5
2       7         26        0         7
3      11         18       18         2
```

3.3: Write a Python program does all the following steps:

a. Prompt users, as shown in the example below, to enter a three-digit whole number such that the digits are in ascending order and without duplicates. Make sure there is a space between the ':' and where the user begins typing their input.

   Valid examples: 123 and 489
   Invalid examples: 133 and 174

b. Loops and re-prompts the user until a correct value is entered.
   - Checks whether the user entered exactly what is requested in step 'a'.
   - Otherwise prints only one of the 4 specific error messages before reprompting (see example)
     - The input may have more than one error but you stop at the first error you find.

c. Informs users when their number is accepted and exits program

Additional Rule:

- There can only be one input prompt in your code.

Example Run:

```
Please enter a 3-digit integer: 122
--> Error: Your number contains duplication.
Please enter a 3-digit integer: 1234
--> Error: You did not enter a 3-digit number.
Please enter a 3-digit integer: 1.23
--> Error: This is not an integer. Please re-enter.
Please enter a 3-digit integer: 376
--> Error: The digits are not in ascending order.
Please enter a 3-digit integer: 348
Number Accepted!
```

The following includes concepts taught in Chapter 6

3.4: Write a Python program does all the following steps:

a. Read a file located in the same directory as the program named: **cs521_3_4_input.txt**
    - Print a clear error message and end the program if the file is missing
    - The program must not crash because the file doesn't exist
    - The input file name must be a constant variable.
    - Do not include your test input file in your assignment zip container

b. Validate that the file contains a single sentence of 20 words.
    - Print a clear error message and end the program if the file has a different number of words.
    - Use a constant variable to set the number of words allowed in the file

c. Break up the sentence into four lines of five single spaced words
    - Use a constant variable to set the number of words per line

d. Write the lines to a new text file named: **cs521_3_4_output.txt**
    - The output file name must be a constant variable.
    - Automatically overwrite the output file if it already exists

e. On success, print "Success! Output written to: cs521_3_4_output.txt"
    - Use the output constant, do not hardcode the file name in the print.
    - This or an error message are all that are printed by this program.
    - Do not include this output file with your submission.

Notes:

- Do not hardcode your string slices
- You are not to include any text files from this problem in your zip container
- There are 4 constant variables in this program
- *Remember to properly close all files.*
- You will lose 0.5 pts for including a text file

CS-521 Homework Assignment 3

The following includes concepts taught in Chapter 14

3.5: This problem will require your program to read a file of student records.
Before you the coding part, manually create a text file named **cs521_3_5_input.txt**
Populate the file with rows of comma separated records in of the format:

- Name of Student
- Student ID
- GPA

For example (you can use your own names and data):

Jerry Seinfeld, 1, 3.7
Elaine Benes, 52, 2.8
George Costanza, 13, 3.9
Cosmo Kramer, 24, 3.4

Do NOT include your test input file in the zip container you submit for this assignment

Now write a python program that performs the following steps:

a. Open and read the file created above line by line
   - The text file must be stored in the same directory as your program
b. Parse the data in each row (records of one student) into a 3 element tuple
c. Store each **tuple** in a single **list**, making an array that will contain all student records
d. After the input file is completely processed, print the list of tuples
   - Include an appropriate description
   - The list object does not need to be formatted
   - There is no output file!

Example Output:

Student Records: [(Jerry Seinfeld, 1, 3.7), (Elaine Benes, 52, 2.8), <you fill in the rest>]

Note:

- *Remember to properly close all files.*
- You will lose 0.5 pts for including a text file

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 your response.