CS-521 Homework Assignment 3

Assignment Directions

The specific instructions for these programming exercises are adapted from problems shown in the textbook. Make sure to follow the specific instructions in THIS document.

Each of the 6 program assignments is worth 10 points for a total of 60 points (100%).

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. The following will be minor deductions:

- Improper naming of programs or zip container
    - Among other things, names for regular programs and zip file must be all lower case.
- Missing program docstring
- Inadequate # line comments (just a few in each program, don't go crazy)
- Going significantly over 80 characters for code/comment lines
- Not using snake_case (lower case plus underscores) for variables and function names
    - CONSTANT variables must be all upper case
    - Class Objects, which are taught in module 6, must be CamelCase
- Asking for input() without descriptive prompts telling the user what is expected.
    - Especially not mentioning the delimiter when input will be split()
- Printing output that is not clearly explained (where necessary)

For all assignments, follow the guidelines in the PEP8 Standards for this Class document in the Student Docs section. Your programs and zip files must be named correctly, contain a docstring heading and include the other style requirements taught to this point. Be sure that your user prompts and print outputs are descriptive.

The specific instructions for these problems are adapted from problems shown in the textbook. Make sure to follow the specific instructions in THIS document.

Chapter 2 Exercise

3.2.1: Write a Python program that counts the number of odd numbers, even numbers, squares of an integer and cubes of an integer from 2 to 130 (inclusive). For example, 9 is both odd and a square, 8 is even and a cube.
Use constants to set the beginning and ending of the range.
For output, print a title with the total range.
For Odd and Even, print the totals and the range of the numbers in scope.
For Squares and Cubes, print the totals and a list of the numbers that meet the criteria
Nothing 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]
```

Chapter 4 Exercises

3.4.2: Set a constant with an odd length string.
Confirm in code that the string is of an odd length. Otherwise, print a relevant message for the user and end the program.
For a string of odd length, print each of the following in double quotes:

- Print the entire string and its length.
- Print the middle character.
- Print the string up to but not including the middle character.
- Print the string from immediately following the middle character to the end.

Example of Output:

```
My 27-character string is: "A man a plan a canal Panama"
The middle character is: "a"
The 1st half of string is: "A man a plan "
The 2nd half of string is: " canal Panama"
```

Do not hard-code the locations in the string – figure them out in your program.

3.4.3: Write a program that prompts the user for a sentence and calculates the number of uppercase letters, lowercase letters, digits, and punctuation. Output the results neatly formatted, centered and labeled in columns. Use Python 3's f-Strings or format() to solve. Check out the library attribute string.punctuation for help in solving this problem. Spaces are NOT considered punctuation.

Example of Expected Output:

```
# Upper   # Lower   # Digits  # Punct.
--------  --------  --------  --------
   2         36        4         5
```

3.4.4: Write a program that prompts the user to enter a three-digit whole number such that the digits are in ascending order and without duplicates.

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

The program loops and re-prompts the user until a correct value is entered. Make sure to check whether the user entered the correct data type.

Example Run:

```
Please enter a 3-digit integer: 122
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!
```

Chapter 6 Exercise

3.6.5: Manually create a text file with a single sentence of 20 words.

Write a program that reads the file and writes the words to a new text file so that there are four lines of five single spaced words.

- Print an error message and stop if the sentence in the file has other than 20 words.
- Test the input file for existence and not crash if the file does not exist.
- Overwrite the output file
- Use constant variables for the number of words allowed and words per line
- Do not hardcode your string slices

Remember to properly close the files.

Chapter 14 Exercise

3.14.6: Create a text file containing student records by line and each record is of the format:

- Name of Student
- Student ID
- GPA

For example (you can use your own data):

```
Tyrion Lannister, 1, 3.7
Daenerys Targaryen, 52, 2.8
Jon Snow, 13, 3.9
Sansa Stark, 24, 3.4
```

Write a program to read the file line by line and store all the records in lists or tuples.
Hint: you need to create a list of lists or list of tuples.
Afterwards, print the array that you created.

Include the file best practices described in exercise #5.

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.