Zelle 3e Chapter 5 Coding Assignment

General Instructions

My expectations for your work on coding assignment exercises will grow as we progress through the course. In addition to applying any new programming techniques that have been covered in the current chapter, I will be expecting you to follow all of the good programming practices that we have adopted in the preceding weeks. Here is a quick summary of good practices that we have covered so far:

- Include a Python Docstring that describes the intent of the program.
- Place your highest-level code in a function named *main*.
- Include a final line of code in the program that executes the *main* function.
- Follow all PEP-8 Python coding style guidelines enforced by the PyCharm Editor. For example, place two blank lines between the code making up a function and the code surrounding that function.
- Choose names for your variables that are properly descriptive.
- Define CONSTANT_VALUES and use them in place of *magic numbers*.
- Always use f-strings for string interpolation and number formatting.
- When processing items from Python lists and tuples, unpack the values into variables with meaningful variable names to avoid using indexed expressions in your code.
- Close all files before the conclusion of the program.
- Remember that your program should behave reasonably when it is not given any input. This might be the result of the user pressing enter at a console prompt. Or, it might be the result of the user providing a an input file that is empty.
- Model your solution after the code that I demonstrate in the tutorial videos.
- Make sure that your test input/output matches the sample provided.
- Create a sub-directory named *data* within your PyCharm project to hold data files.
- Remember to submit all data files with your PyCharm project – including the files that were provided as starter files to this assignment.
- Remember to test your program thoroughly before submitting your work.

Exercise 1 (Regular)
Create a program named *numeric_character_analytics.py.* It should meet the following requirements:

1. Prompt the user to enter a line of text to be analyzed.

2. For each word in the line of text entered, print the quantity of numeric digits (0, 1, 2, 3, 4, 5, 6, 7, 8 9) included in the word.

3. Print the number of words that were analyzed.

The following is an example of expected input/output on your console from a typical test:

```
Please enter a line of text to be analyzed: I ate 5 hamburgers and
drank 6 beers at Studio54 on 54th street.

"I" contains 0 numeric characters.
"ate" contains 0 numeric characters.
"5" contains 1 numeric characters.
"hamburgers" contains 0 numeric characters.
"and" contains 0 numeric characters.
"drank" contains 0 numeric characters.
"6" contains 1 numeric characters.
"beers" contains 0 numeric characters.
"at" contains 0 numeric characters.
"Studio54" contains 2 numeric characters.
"on" contains 0 numeric characters.
"54th" contains 2 numeric characters.
"street." contains 0 numeric characters.

13 words were analyzed.
```

Your program should behave reasonably when no input is provided. The following is an example of expected input/output on your console from a test with empty input:

```
Please enter a line of text to be analyzed:


0 words were analyzed.
```

Exercise 2 (Regular)
Create a program named *batch_numeric_character_analytics.py.* This program will be a batch version of *numeric_character_analytics.py.*

Two files for this exercise have been provided as starter files to this assignment:

- *sentences_with_numbers.txt*
- *empty_file.txt*

These files should be placed in a sub-directory in your PyCharm project named *data*.

Your program should meet the following requirements:

1. Prompt the user for the name of the file to be analyzed.

2. For each text line in the input file:
    a. Print the contents of the line.
    b. Print each word and the number of numeric characters contained in the word.
    c. Print the total number of words analyzed for this line.

3. At the conclusion of the program, print the total number of text lines analyzed.

When this program is run with *sentences_with_numbers.txt,* you should expect the following input/output on your console:

```
Please enter the name of the file to be analyzed:
sentences_with_numbers.txt

Analyzing: There are 12 inches in 1 foot.
"There" contains 0 numeric characters.
"are" contains 0 numeric characters.
"12" contains 2 numeric characters.
"inches" contains 0 numeric characters.
"in" contains 0 numeric characters.
"1" contains 1 numeric characters.
"foot." contains 0 numeric characters.
7 words were analyzed.

Analyzing: The United States is comprised of 50 states.
"The" contains 0 numeric characters.
"United" contains 0 numeric characters.
"States" contains 0 numeric characters.
```

```
"is" contains 0 numeric characters.
"comprised" contains 0 numeric characters.
"of" contains 0 numeric characters.
"50" contains 2 numeric characters.
"states." contains 0 numeric characters.
8 words were analyzed.

Analyzing:
0 words were analyzed.

Analyzing: 1, 2, 3, 4. I am the giant that you are looking for.
"1," contains 1 numeric characters.
"2," contains 1 numeric characters.
"3," contains 1 numeric characters.
"4." contains 1 numeric characters.
"I" contains 0 numeric characters.
"am" contains 0 numeric characters.
"the" contains 0 numeric characters.
"giant" contains 0 numeric characters.
"that" contains 0 numeric characters.
"you" contains 0 numeric characters.
"are" contains 0 numeric characters.
"looking" contains 0 numeric characters.
"for." contains 0 numeric characters.
13 words were analyzed.

Analyzing: 50 men, and 75 horses, took 10 wagons over 1 big
mountain.
"50" contains 2 numeric characters.
"men," contains 0 numeric characters.
"and" contains 0 numeric characters.
"75" contains 2 numeric characters.
"horses," contains 0 numeric characters.
"took" contains 0 numeric characters.
"10" contains 2 numeric characters.
"wagons" contains 0 numeric characters.
"over" contains 0 numeric characters.
"1" contains 1 numeric characters.
"big" contains 0 numeric characters.
"mountain." contains 0 numeric characters.
12 words were analyzed.

5 lines were analyzed from the file: sentences_with_numbers.txt
```

Your program should behave reasonably when no input is provided. When this program is run with *empty_file.txt,* you should expect the following input/output on your console:

```
Please enter the name of the file to be analyzed: empty_file.txt

0 lines were analyzed from the file: empty_file.txt
```

Exercise 3 (Regular)
Create a program named *generate_usernames.py.* This will be a batch program that generates a file of username assignments from a file of names.

Two files for this exercise have been provided as starter files to this assignment:

- *employee_names.txt*
- *empty_file.txt*

These files should be placed in a sub-directory in your PyCharm project named *data*.

Your program should meet the following requirements:

1. Prompt the user for the name of the input file.

2. The name of the output file should be hard coded as *username_assignments.txt*.

3. For each text line in the input file:
    a. Extract the first name and last name.
    b. Create a username the consists of the first letter of the first name combined with the first 7 letters of the last name (all in lower case).
    c. Write a line to the output file with the following format:

```
FirstName LastName,username
```

4. At the conclusion of the program, print the total number of text lines processed from the input file.

When this program is run with *employee_names.txt,* you should expect the following input/output on your console:

```
Please enter the name of the file containing names:
employee_names.txt

125 lines were processed from input file: employee_names.txt

Created username assignments file as: username_assignments.txt
```

The first 3 records in username_assignments.txt should look like the following:

```
Elizabeth Wilson,ewilson
Victor Lewis,vlewis
Christopher Paterson,cpaterso
```

Your program should behave reasonably when no input is provided. When this program is run with *empty_file.txt,* you should expect the following input/output on your console:

```
Please enter the name of the file containing names: empty_file.txt

0 lines were processed from input file: empty_file.txt
```

The file username_assignments.txt should be empty.

Exercise 4 (Challenge)
Create a program named *flexible_adding_machine.py.* This program will read a flexibly formatted file of number entries and accumulate the sum of these entries.

Two files for this exercise have been provided as starter files to this assignment:

- *accumulator_data.txt*
- *empty_file.txt*

These files should be placed in a sub-directory in your PyCharm project named *data*.

Your program should meet the following requirements:

1. Prompt the user for the name of the input file.

2. For each line of the input file:

    a. Determine how many number entries are on this line.
    b. For each of the number entries (allow for float values):
        i. Print the entry.
        ii. Add that entry to the accumulated total.

3. At the conclusion of the program, print:

    a. The number of text lines processed.
    b. The number of entries processed.
    c. The accumulated total of the entries formatted to 3 decimal places.

When this program is run with *accumulator_data.txt,* you should expect the following input/output on your console:

```
Please enter the name of the input file: accumulator_data.txt

ENTRIES:
1.25
2.0
3.0
4.5
9.0
0.0
2.22
3.35
4.2
15.0
```

```
7 lines were read from file: accumulator_data.txt

10 entries were processed.

The accumulated total of the entries was 44.520
```

Your program should behave reasonably when no input is provided. When this program is run with *empty_file.txt,* you should expect the following input/output on your console:

```
Please enter the name of the input file: empty_file.txt

ENTRIES:

0 lines were read from file: empty_file.txt

0 entries were processed.

The accumulated total of the entries was 0.000
```

Tools
Use PyCharm to create and test all Python programs.

Submission Method
Follow the process that I demonstrated in the tutorial video on submitting your work. This involves:

- Locating the properly named directory associated with your project in the file system.
- Compressing that directory into a single .ZIP file using a utility program.
- Submitting the properly named zip file to the submission activity for this assignment.

File and Directory Naming
Please name your Python program files as instructed in each exercise. Please use the following naming scheme for naming your PyCharm project:

```
surname_givenname_exercises_zelle_3e_chapter_05
```

If this were my own project, I would name my PyCharm project as follows:

```
trainor_kevin_exercises_zelle_3e_chapter_05
```

Use a zip utility to create one zip file that contain the PyCharm project directory. The zip file should be named according to the following scheme:

```
surname_givenname_exercises_zelle_3e_chapter_05.zip
```

If this were my own project, I would name the zip file as follows:

```
trainor_kevin_exercises_zelle_3e_chapter_05.zip
```

Due By
Please submit this assignment by the date and time shown in the Weekly Schedule.

Last Revised
2022-05-22