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 single-line comment with name of program file.
- Include a single-line comment 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.
- Close all files before the conclusion of the program.
- Model your solution after the code that I demonstrate in the tutorial videos.
- Remember to test your program thoroughly before submitting your work.

Exercise 1

Create a program named *numeric_character_analytics*. Prompt the user to enter one line of text at the console. Analyze the line of text word by word. For each word in the line of text, print the count of numeric characters (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) that occur in the word. At the conclusion of the program, print the total number of words analyzed. Use my tutorial example as a model for your code.

When this program is run, console sessions should look like this:

```
Please enter a line of text to be analyzed: Kevin is teaching 155
students in 6 classes this semester.
Kevin has 0 numeric characters.
is has 0 numeric characters.
teaching has 0 numeric characters.
155 has 3 numeric characters.
students has 0 numeric characters.
in has 0 numeric characters.
6 has 1 numeric characters.
classes has 0 numeric characters.
this has 0 numeric characters.
semester. has 0 numeric characters.
10 words were analyzed.
```

Exercise 2

Create a program named *batch_numeric_character_analytics.* This program will be a batch version of *numeric_character_analytics*. For the input file, use the file named *numeric_character_data.txt* that has been provided as a starter file for this exercise. Hard code the filename into your program.

For each text line in the input file:

1. Print the contents of the line
2. Print each word and the number of numeric characters contained in the word
3. Print the total number of words analyzed for this line

At the conclusion of the program, print the total number of text lines analyzed. Use my tutorial example as a model for your code.

When this program is run, console sessions should look like this:

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

Analyzing: The United States is comprised of 50 states.
The has 0 numeric characters.
United has 0 numeric characters.
States has 0 numeric characters.
is has 0 numeric characters.
comprised has 0 numeric characters.
of has 0 numeric characters.
50 has 2 numeric characters.
states. has 0 numeric characters.
8 words were analyzed.

Analyzing: 1, 2, 3, 4. I am the giant that you are looking for.
1, has 1 numeric characters.
2, has 1 numeric characters.
3, has 1 numeric characters.
4. has 1 numeric characters.
I has 0 numeric characters.
am has 0 numeric characters.
the has 0 numeric characters.
giant has 0 numeric characters.
that has 0 numeric characters.
you has 0 numeric characters.
are has 0 numeric characters.
```

```
looking has 0 numeric characters.
for. has 0 numeric characters.
13 words were analyzed.

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

4 lines were analyzed from the file: numeric_character_data.txt
```

Exercise 3

Create a program named *five_sixteenths_batch_accumulator*. This program will read an input file of text lines. Each text line will contain one or more float values separated by comma-space. Prompt the user to enter the name of the text file at the console. When testing, enter the name of the starter file named *accumulator_data.txt*.

For each line of text in the file:

1. Accumulate the sum of the float values.
2. Calculate a final amount that is five-sixteenths of the accumulated sum.
3. Print the contents of the text line.
4. Print the final amount without rounding (formatting).
5. Print the final amount rounded (formatted) to 3 decimal places.

Use my tutorial example as a model for your code.

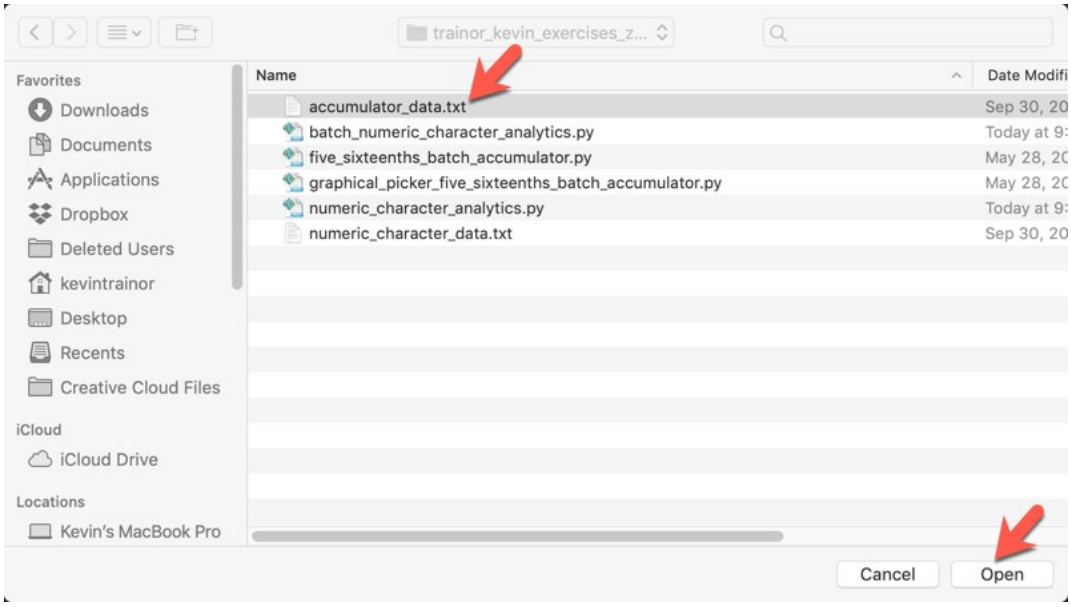
When this program is run, console sessions should look like this:

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

1.25, 2.00, 3.00, 4.50
Before rounding: 3.359375
After rounding: 3.359

9
Before rounding: 2.8125
After rounding: 2.812

0.00
Before rounding: 0.0
After rounding: 0.000

2.22, 3.35, 4.20
Before rounding: 3.0531249999999996
After rounding: 3.053

15.00
Before rounding: 4.6875
After rounding: 4.688
```

Exercise 4

Create a program named *graphical_picker_five_sixteenths_batch_accumulator*. Start with a copy of your code from the previous exercise. Modify the code to use the graphical file picker that is demonstrated in the end of Chapter 5. Use my tutorial example as a model for your code.

When this program is run, a graphical picker window should appear:



After having selected the appropriate input file, output from the console sessions should look like this:

```
1.25, 2.00, 3.00, 4.50
Before rounding: 3.359375
After rounding: 3.359

9
Before rounding: 2.8125
After rounding: 2.812

0.00
Before rounding: 0.0
After rounding: 0.000

2.22, 3.35, 4.20
Before rounding: 3.0531249999999996
After rounding: 3.053

15.00
Before rounding: 4.6875
After rounding: 4.688
```

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
2021-02-14