Zelle 3e Chapter 8 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.
- Output printed by the program (both prompts and results) should be polite and descriptive.
- Choose names for your variables that are properly descriptive.
- Choose names for your functions 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.
- Your code must pass all relevant test cases. Make sure that it passes tests at the boundaries created by *if, else*, and *elif* conditions in your program (boundary value tests).

Exercise 1
Create a program named *sentinel_loop_using_break*. This program should implement functionality that is the same as the Part 1 tutorial example, except for the following differences:

- This program should count even and odd integers.
- This program should report counts of even and odd integers.

Remember that even integers can be identified using the test condition:

value % 2 == 0

Remember to test your code for appropriate behavior when the user signals end of processing without providing any proper integer input.

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

```
Process a series of integers entered at the console.

Please enter an integer or press <Enter> to stop:
You did not provide any integers to process.


Process a series of integers entered at the console.

Please enter an integer or press <Enter> to stop: 1
Please enter an integer or press <Enter> to stop: 2
Please enter an integer or press <Enter> to stop: 3
Please enter an integer or press <Enter> to stop: 4
Please enter an integer or press <Enter> to stop: 5
Please enter an integer or press <Enter> to stop:
Number of evens is 2.
Number of odds is 3.
```

Exercise 2
Create a program named *sentinel_loop_without_break*. This program should implement functionality that is the same as the Part 2 tutorial example, except for the following differences:

- This program should count even and odd integers.
- This program should report counts of even and odd integers.

Remember that even integers can be identified using the test condition:

value % 2 == 0

Please note that this program may not use the **break** statement.

Remember to test your code for appropriate behavior when the user signals end of processing without providing any proper integer input.

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

```
Process a series of integers entered at the console.

Please enter an integer or press <Enter> to stop:
You did not provide any integers to process.


Process a series of integers entered at the console.

Please enter an integer or press <Enter> to stop: 1
Please enter an integer or press <Enter> to stop: 2
Please enter an integer or press <Enter> to stop: 3
Please enter an integer or press <Enter> to stop: 4
Please enter an integer or press <Enter> to stop: 5
Please enter an integer or press <Enter> to stop:
Number of evens is 2.
Number of odds is 3.
```

Exercise 3
Create a program named *process_file_one_integer_per_line*. This program should implement functionality that is the same as the Part 3 tutorial example, except for the following differences:

- This program should count even and odd integers.
- This program should report counts of even and odd integers.

Remember that even integers can be identified using the test condition:

value % 2 == 0

Please note that this program may not use the **break** statement.

Remember to test your code for appropriate behavior when the input file is empty.

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

```
Process a series of integers from a text file.

Please enter the name of the input file: empty_file.txt
You did not provide any integers to process.


Process a series of integers from a text file.

Please enter the name of the input file: single_file.txt
Number of evens is 5.
Number of odds is 5.
```

Exercise 4

Create a program named *identify_slot_machine_winner*. This program should implement functionality that is the same as the Part 4 tutorial example, except for the following differences:

- The winning condition is 3 Red and 2 Blue.

Please note that this program may not use the **break** statement.

Remember to test your code for appropriate behavior when the input file is empty.

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

```
Please enter the name of the input file: empty_file.txt
No winning slot values found.
0 lines read from empty_file.txt


Please enter the name of the input file: slot_values.txt
Winner found on line 3458.
Red Blue Red Red Blue
```

Tools
Use PyCharm to create and test both 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 project:

YourLastName_YourFirstName_exercises_zelle_3e_chapter_08

When you have compressed your project directory into a .ZIP file, it should have the following name structure:

YourLastName_YourFirstName_exercises_zelle_3e_chapter_08.zip

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