Zelle 3e Chapter 11 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.
- All functions that are not *main()* should have descriptive, action-oriented names.
- All functions should be of reasonable size.
- All functions should have high *cohesion*, and low *coupling*.
- 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).
- Use of the *break* statement is allowed but not encouraged.
- Use of the *continue* statement is forbidden.

Exercise 1 (Required)
Create a program named *distribute_race_ribbons_with_dictionary.py*. It should modeled after the program that I demonstrated in the tutorial *(lookup_region_name_with_dictionary.py*). Your program should be different in the following respects:

1. Your program will prompt the user for the place number in which the runner finished and it will respond with the name of the ribbon to be awarded.

The following table indicates which ribbon the participant should receive based up the place number in which they finished.

Place	Ribbon
1	Blue
2	Red
3	Orange
4	Gold
5	Green
6	Purple
>6	White

If the user enters a place number that is less than 1, then the program should display an error message in place of a ribbon name.

When running a test where the user provides no input, you should expect the following input/output on your console:

```
Please enter place finished (1, 2, 3...):

Thanks for playing.
```

When running a test where the user provides bad integer input, you should expect the following input/output on your console:

```
Please enter place finished (1, 2, 3...): hi mom
An integer value was expected. You entered hi mom
Please enter place finished (1, 2, 3...): 7
Ribbon Awarded: White

Please enter place finished (1, 2, 3...):

Thanks for playing.
```

When running a test where a user provides an invalid place number, you should expect the following input/output on your console:

```
Please enter place finished (1, 2, 3...): 0
Ribbon Awarded: ERROR - Place must be greater than zero.

Please enter place finished (1, 2, 3...):

Thanks for playing.
```

When running a test with more typical input, you should expect the following input/output on your console:

```
Please enter place finished (1, 2, 3...): 1
Ribbon Awarded: Blue

Please enter place finished (1, 2, 3...): 2
Ribbon Awarded: Red

Please enter place finished (1, 2, 3...): 3
Ribbon Awarded: Orange

Please enter place finished (1, 2, 3...): 6
Ribbon Awarded: Purple

Please enter place finished (1, 2, 3...): 7
Ribbon Awarded: White

Please enter place finished (1, 2, 3...): 10
Ribbon Awarded: White

Please enter place finished (1, 2, 3...): 22
Ribbon Awarded: White

Please enter place finished (1, 2, 3...):

Thanks for playing.
```

Exercise 2 (Required)
Create a program named *analyze_expense_records_2.py*. It should modeled after the program that I demonstrated in the tutorial *analyze_expense_records.py*). Your program should be different in the following respects:

1. It should target expense records with a year of 2021.

2. It should target expense records with a category of 'Utilities'.

3. In addition to providing the mean value, it should also provide the median value.

Mean and median values should be calculated using the Python statistics package: https://docs.python.org/3/library/statistics.html .

The following starter files are provided for your use in this exercise:

- `empty_file.txt`
- `test_expense_2021.csv`
- `expense_records.csv`

When running a test with the empty input file, you should expect the following input/output on your console:

```
Please enter the input filename: empty_file.txt

SUMMARY OF UTILITIES EXPENSES FOR 2021

Month          Amount
    1            0.00
    2            0.00
    3            0.00
    4            0.00
    5            0.00
    6            0.00
    7            0.00
    8            0.00
    9            0.00
   10            0.00
   11            0.00
   12            0.00

The mean value was 0.00
The median value was 0.00

0 lines were read from the input file.
0 qualifying expense records were found.
```

When running a test with the test input file, you should expect the following input/output on your console:

```
Please enter the input filename: test_expense_2021.csv

SUMMARY OF UTILITIES EXPENSES FOR 2021

Month          Amount
    1          333.33
    2          333.33
    3          333.33
    4          333.33
    5          333.33
    6          333.33
    7          333.33
    8          333.33
    9          333.33
   10          333.33
   11          333.33
   12          333.33

The mean value was 333.33
The median value was 333.33

73 lines were read from the input file.
24 qualifying expense records were found.
```

When running a test where the production input file, you should expect the following input/output on your console:

```
Please enter the input filename: expense_records.csv

SUMMARY OF UTILITIES EXPENSES FOR 2021

Month          Amount
    1       58,986.73
    2       71,418.23
    3       67,361.43
    4       64,864.79
    5       69,780.78
    6       67,408.51
    7       56,411.29
    8       64,604.41
    9       86,856.16
   10       49,318.79
   11       63,268.65
   12       73,163.29

The mean value was 66,120.26
The median value was 66,113.11

20,001 lines were read from the input file.
755 qualifying expense records were found.
```

Exercise 3 (Required)
Create a program named *create_population_density_reports.py*. It should modeled after the program that I demonstrated in the tutorial (*create_state_area_reports.py*). Your program should be different in the following respects:

1. It should create reports about population density.

2. It should use the provided Country class as the data holder class.

3. The data to be analyzed are in a file named desity_data.txt. Each line of the file contains three fields (separated by semicolons):

    a. Country name
    b. Population
    c. Area in square miles

The following starter files are provided for your use in this exercise:

- `empty_file.txt`
- `density_data.txt`
- `my_countries.py`
- `is430_unit_test_helpers.py`

When running a test with the empty input file, you should expect the following input/output on your console:

```
Please enter the input filename: empty_file.txt


                          BY COUNTRY NAME

Country                   Population          Area        Density
                                            (SQMI)        (/SQMI)


        BY DESCENDING POPULATION DENSITY PER SQUARE MILE

Country                   Population          Area        Density
                                            (SQMI)        (/SQMI)
```

When running a test with the production input file, you should expect the following input/output on your console:

```
Please enter the input filename: density_data.txt


                        BY COUNTRY NAME

Country                   Population          Area        Density
                                           (SQMI)        (/SQMI)
Bangladesh               166,132,772         55,598          2,988
Belgium                   11,454,906         11,787            972
Burundi                   10,681,186         10,740            995
Dominican Republic        10,266,149         18,485            555
Germany                   82,979,100        137,903            602
Haiti                     11,112,945         10,450          1,063
India                  1,344,098,517      1,269,211          1,059
Israel                     8,997,000          8,522          1,056
Japan                    126,320,000        145,925            866
Netherlands               17,301,708         16,033          1,079
Nigeria                  195,875,237        356,669            549
North Korea               25,610,672         47,399            540
Pakistan                 203,841,217        310,403            657
Philippines              107,275,680        115,831            926
Rwanda                    12,001,136         10,169          1,180
South Korea               51,635,256         38,691          1,335
Sri Lanka                 21,670,000         25,332            855
Taiwan                    23,590,744         13,976          1,688
United Kingdom            66,040,229         93,788            704
Vietnam                   94,660,000        127,882            740


        BY DESCENDING POPULATION DENSITY PER SQUARE MILE

Country                   Population          Area        Density
                                           (SQMI)        (/SQMI)
Bangladesh               166,132,772         55,598          2,988
Taiwan                    23,590,744         13,976          1,688
South Korea               51,635,256         38,691          1,335
Rwanda                    12,001,136         10,169          1,180
Netherlands               17,301,708         16,033          1,079
Haiti                     11,112,945         10,450          1,063
India                  1,344,098,517      1,269,211          1,059
Israel                     8,997,000          8,522          1,056
Burundi                   10,681,186         10,740            995
Belgium                   11,454,906         11,787            972
Philippines              107,275,680        115,831            926
Japan                    126,320,000        145,925            866
```

Sri Lanka	21,670,000	25,332	855
Vietnam	94,660,000	127,882	740
United Kingdom	66,040,229	93,788	704
Pakistan	203,841,217	310,403	657
Germany	82,979,100	137,903	602
Dominican Republic	10,266,149	18,485	555
Nigeria	195,875,237	356,669	549
North Korea	25,610,672	47,399	540

Exercise 4 (Required)
Create a program named *analyze_slot_machine_tries_ignoring_duplicates.py*. It should modeled after the program that I demonstrated in the tutorial (*analyze_slot_machine_tries.py*). Your program should be different in the following respects:

1. It should ignore duplicate color values that occur on the same line of the input file. So, if the line holds the values: "Red Blue Red Red Blue", then the program should process this line as though it holds the values: "Red Blue".

Please refer to the tutorial for an approach to removing duplicate entries from Python lists.

The following starter files are provided for your use in this exercise:

- `empty_file.txt`
- `slot_values.txt`

When running a test with the empty input file, you should expect the following input/output on your console:

```
Please enter the input filename: empty_file.txt

COLOR       COUNT
```

When running a test with the production input file, you should expect the following input/output on your console:

```
Please enter the input filename: slot_values.txt

COLOR       COUNT
Blue        2,990
Green       2,983
Orange      3,024
Purple      3,015
Red         2,959
Yellow      3,011
```

Exercise 5 (Optional Challenge Exercise)
Create a program named *create_population_density_reports_with_lambda.py*. This program should be modeled after your solution to Exercise 3 (*create_population_density_reports.py*). This program should be different in the following respects:

1. Instead of using conventional Python functions to specify the sort keys, this program should use Python lambdas.
   See https://realpython.com/python-lambda/ .

The testing for this program should be the same as the program in Exercise 3. Please refer to those instructions for expected output.

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_11
```

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

```
trainor_kevin_exercises_zelle_3e_chapter_11
```

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_11.zip
```

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

```
trainor_kevin_exercises_zelle_3e_chapter_11.zip
```

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

Last Revised
2022-03-9