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 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 *create_population_density_reports.* This program reads a data file regarding countries and their population density. Each line of the file contains 3 fields for 1 country:

- Country name
- Population
- Area in square miles

Two starter data files have been provided for your use:

- density_data.txt
- empty_file.txt

Please note that data fields in this file format are separated by a semicolon (;). The empty file is provided to test your program's resilience to being passed an empty input file.

A Python class named *Country* has been provided for your use in storing country data, sorting country data, and calculating population density. The Country class and related code are contained in the starter file:

- my_countries.py

Please note that the Country class has both instance variables and methods that provide data that you will need to create the report. The following are instance variables:

- country_name
- population

The following is a method:

- calculate_population_density_per_square_mile()

The program should create a Python list of Country instances, each instance containing the data for one country.

Having created the list of Country instances, this list should be sorted into country name order and a report should be printed with the title BY COUNTRY NAME.

When the first report is complete, the list of Country instances should be sorted into descending order of population density. Finally, a report should be printed with the title BY DESCENDING POPULATION DENSITY PER SQUARE MILE.

When coding and testing this program, follow the approach that I take in the Part 1 tutorial video. This includes using the techniques demonstrated for centering report titles, placing column headings, and inserting commas into number strings. Format strings for the report headings and column headings should be similar to those used in the tutorial example. Here is a hint to help you create a format string for the detail line in this report:

- **'{0:<15}{1: >15,.2f}{2: >15,.2f}{3: >15,.2f}'**.format(q, r, s, t)

Please remember to test your program with both of the test files provided:

- density_data.txt
- country_pop_data.txt

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

```
Please enter input file name: 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)



Please enter input file name: 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 (SQMI)	Density (/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 2

Create a program named *distribute_race_awards_with_dictionary.* This program performs a simple lookup based upon integer input provided by the user at the console. This program is different from a similar program created in a previous assignment in that the lookup is performed using a Python dictionary.

When coding and testing this program, follow the approach that I take in the Part 2 tutorial video.

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

```
Please enter the place in which the participant finished (1, 2,
3, ...): 1
Participant place: 1
Participant award: Blue Ribbon


Please enter the place in which the participant finished (1, 2,
3, ...): 6
Participant place: 6
Participant award: Purple Ribbon


Please enter the place in which the participant finished (1, 2,
3, ...): 7
Participant place: 7
Participant award: Participant Ribbon


Please enter the place in which the participant finished (1, 2,
3, ...): 0
Participant place: 0
Participant award: Input Error - Place must be greater than 0
```

Exercise 3

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

- This program should not include duplicate colors from a single try when accumulating counts for colors. Remember that a line of input data represents a single try.

Please refer to the Part 3 tutorial for an approach to removing duplicate entries from Python lists.

This program should process both test data files included in the starter files for the assignment and produce appropriate output. These files are named:

- empty_file.txt
- slot_values.txt

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

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


Please enter the name of the input file: slot_values.txt
Blue         2990
Green        2983
Orange       3024
Purple       3015
Red          2959
Yellow       3011
```

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_11

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

YourLastName_YourFirstName_exercises_zelle_3e_chapter_11.zip

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