Zelle 3e Chapter 7 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).

No Tutorial Videos and Examples for This Chapter

Note that I have not provided tutorial videos and examples for this chapter. I did this because I used an extensive set of custom program examples during my lecture. I have provided a copy of a PyCharm project containing all of those examples in the Weekly Schedule. In the exercise descriptions below, I will identify the sample program that most resembles the program that I am asking you to create.

Exercise 1

A foot race has been held for a large group of children. In keeping with modern thinking regarding children's competitions, every participant will receive an award. The following table indicates which award the participant should receive based up the place number in which they finished.

Place	Award Earned
1	Blue Ribbon
2	Red Ribbon
3	White Ribbon
4	Gold Ribbon
5	Green Ribbon
6	Purple Ribbon
>6	Participant Ribbon

Create a program named *distribute_race_awards.* This program will be used by race organizers when distributing the race awards. Each time the program is run, it will prompt the user for an integer representing the place in which the child finished. Then, it will print a description of the award that the child has earned.

Design your program such that the code that looks up the award earned is in a separate function named *determine_race_award*. For this program, you can trust the user to enter an integer. Nevertheless, you need to check for inputs that are proper integers but do not represent proper finishing places. In this case, the function should return an error message instead of the description of the race award.

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, ...): 100
Participant place: 100
Participant award: Participant Ribbon

Please enter the place in which the participant finished (1, 2,
3, ...): -4096
Participant place: -4096
Participant award: Input Error - Place must be greater than 0.
```

When creating this program, you may use the sample program *decisions_25_lookup_in_function* as a model.

Exercise 2

A state motor vehicle agency needs to calculate annual registration fees for vehicles registered in the state. Fees are based upon the vehicle type (car or truck) and the vehicle weight.

Create a python program named *calculate_registration_fees*.

Within that program, create a Python function that will be used in the vehicle registration system. The function should be named *determine_annual_registration_fee*. It should accept two input parameters: vehicle type and weight. It should return the annual_fee as a float value. The annual fee will be based upon the following table:

Vehicle Type	Weight	Annual Fee
Car	< 3000	125.00
Car	>= 3000	200.00
Truck	< 4000	250.00
Truck	>= 4000	350.00

When creating your program, employ a rudimentary automated testing approach by placing your test cases in the *main* function. For each test case that passes, the program should print the Boolean value *True*. For each test case that fails, the program should print the Boolean value *False*.

Since the *determine_annual_registration_fee* function will eventually be part of a larger program and system, you can assume that the input values provided to the function have already been checked for type and validity. Your code only needs to calculate a proper annual fee based upon these inputs. Please be sure to create sufficient test cases to fully test your new function. Pay particular attention to the need for boundary value testing.

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

```
True
True
True
True
True
True
True
True
```

When creating this program, you may use the sample program *decisions_35_nested_in_function* as a model.

Exercise 3

Create a program named *robust_integer_prompt_and print_loop*. This program will be based upon the sample program *decisions_40_try*. Feel free to copy the sample program to provide a starting point for your code.

Your program should be different from the sample program in the following respects:

1. Instead of only prompting the user for 1 integer, your program should use a for/in loop to prompt the user for an integer 10 times.
2. Your program does not need to use the *finally* option of the try/except control structure. So, you may eliminate that portion of the code.
3. Your program should print polite messages to the user at the start of the program so that the user knows how many integers they will be prompted for and what kind of output to expect.

When this program is run, console sessions should look like this:
(Windows 10 users: Please see note below)

```
This program prompts you for 10 integers...
It prints proper inputs...
And it complains about improper inputs.

Please enter an integer: 1
You have entered the integer 1
Please enter an integer: bad input
That was not an integer and you know it!
Please enter an integer: 3
You have entered the integer 3
Please enter an integer: 4
You have entered the integer 4
Please enter an integer: 5
You have entered the integer 5
Please enter an integer:
I detected a keyboard interrupt. Now that was just mean!!!!!
Please enter an integer: 7
You have entered the integer 7
Please enter an integer: 8
You have entered the integer 8
Please enter an integer: 9
You have entered the integer 9
Please enter an integer: 10
You have entered the integer 10
Thanks for playing.
```

Windows 10 Users: Due to known bugs in the combination of products that we are using, you can expect different behavior when simulating a keyboard interrupt in PyCharm during testing. When you simulate a keyboard interrupt, the program will terminate and you will receive the following message:

```
Process finished with exit code -1
```

Someday, these bugs will be resolved and the expected behavior will be the same when testing this code on both Windows 10 and macOS. Until then, we will need to tolerate this difference.

PLEASE NOTE
This assignment has 4 exercises.
Don't forget to do exercise 4.
SEE BELOW!!

Exercise 4

Create a program named *find_highest*. This program will be based upon the sample program *decisions_50_find_lowest*. Feel free to copy the sample program to provide a starting point for your code.

Your program should be different from the sample program in the following respects:

1. It should find the highest rather than the lowest value.

When conducting your tests, make sure that your program works properly when the *input_list* is empty.

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

```
There are 11 entries in the input list.
The highest value is 4096.

The input list is empty.
```

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_07
```

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

```
trainor_kevin_exercises_zelle_3e_chapter_07
```

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_07.zip
```

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

```
trainor_kevin_exercises_zelle_3e_chapter_07.zip
```

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

Last Revised
2021-02-14