Severance Chapter 14 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.
- Your code should be factored such that there is a function in your program for each part of the problem.
- Each function should contain code relating to the same thing – it should have *high cohesion*.
- Functions should know as little as possible about the workings of other functions – they should have *low coupling*.
- 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).

Assignment Overview

The goal of this assignment is to create a custom Python class to hold data values. In a previous assignment (Zelle 3e Chapter 11), I gave you a Python module named *my_countries.py*. This module contained the custom Python class that I created named *Country.* The *Country* class held data values related to countries, their population, and their area. In that assignment, you used instances of the *Country* class to hold each of your data records. You stored these instances of the *Country* class in a Python `list` in order to store and sort the data records. Finally, you created reports by reading the sorted list of instances and formatting the output.

In this assignment, you will be doing similar activities. The difference is that **you will be creating a new and different custom Python class**. During the tutorial for this assignment, we will begin by reviewing the *Country* class from our previous assignment. In Part 0 of the tutorial, I will introduce a Cheat Sheet for creating custom classes and use that Cheat Sheet to help you identify the components of a custom Python class. You will be able to use the Cheat Sheet for the remainder of this assignment and you will want keep a copy for it your future use. It should be very helpful when working on the Final Project for our course.

The following is a summary of the steps that you will follow to complete this assignment:

- Review the Cheat Sheet for the assignment. Do your best to use it to identify the parts of the Country class. Both of these items are provided with the starter files for this assignment.

- Exercise 1: Use the Cheat Sheet to help you create the basic version of a new custom class named *LandMammal*. Instances of this class will be used to hold data records regarding the world's heaviest land mammals later in this assignment. In exercise 1, you will only be creating a basic version of the *LandMammal* class.

- Exercise 2: Using the basic version of the *LandMammal* class, you will be creating instances of the class to hold data records. You will sort a list of these records in two different orders to create two reports regarding land mammals. You will do this reporting using the basic version of the *LandMammal* Class.

- Exercise 3: Using the Cheat Sheet and the basic version of the *LandMammal* class as starters, you will create an evolved version of the *LandMammal* class. The additional features added in this step will be implicit getter and setter code that use the Python `@property` decorator. The additional code will allow this evolved version of the class to prevent the creation of instances of the *LandMammal* class that have invalid instance variable values.

- Exercise 4: In this exercise, you modify the code from Exercise 2 to create a new version of the reporting program. Here, you will demonstrate that you can add error checking features to the reporting program simply by substituting the evolved version of the *LandMammal* class for the basic version that we used in Exercise 2. It is just that simple.

Please see detailed instructions for these four exercises on the following pages.

Exercise 1
Create a Python module file named *my_basic_land_mammals.py*. This module should contain the following:

1. A class named *LandMammal* that contains the basic implementation of a class that holds data facts regarding the world's largest land mammals.

2. A *main()* function that contains unit test cases for the *LandMammal* class.

This basic version of the *LandMammal* class should NOT contain any getter or setter features. When doing this part of the assignment, follow the approach that I demonstrated in the Part 1 tutorial video.

The LandMammal class should implement the following instance variables:

- name (string)
- minimum mass in pounds (int)
- maximum mass in pounds (int)

Your class should provide implementations of the following standard Python features:

- Constructor implemented with __init__()
- str() method implemented with __str__()
- repr() method implemented with __repr()

You will also need to provide a method to calculate and return the following attribute:

- calculate_variability_of_mass_in_pounds() returns an int.

In the main() function, create test cases for the following:

- the constructor
- the __str__() method
- the __repr__() method
- the calculate_variability_of_mass_in_pounds() method

When creating the custom class, rely on the Cheat Sheet and the steps that I have demonstrated in the Part 1 tutorial video for guidance. Remember that the code from the tutorial video has been provided in the starter files. So, **you do NOT need to type in the code from the tutorial video**.

When this program is run directly (rather than having been imported), the console session should contain the unit testing output and should look like this:

```
Unit testing output follows...

Test Case #1: Test constructor
Passed

Test Case #2: Test str method
Passed

Test Case #3: Test repr method
Passed

Test Case #4: Test calculate_variability_of_mass_in_pounds
Passed
```

Please continue to Exercise 2 on the next page…

Exercise 2

Create a program named *create_land_mammal_mass_reports_using_basic_class.* The program will follow the same pattern as the program that I demonstrate in Part 2 of the tutorial video. Instead of coding this program from scratch, you may find it easier to copy the tutorial program from the starter files and change that code to meet the requirements below.

Your program will create a list of *LandMammal* instances to hold the data facts from the input file. Since this program uses the basic version of the *LandMammal* class, there will be no checking done for invalid input values.

Follow the code that I demonstrate in Part 2 of the tutorial to sort the list of records in two different orders to create reports as follows:

- BY LAND MAMMAL NAME
- BY DESCENDING VARIABILITY OF MASS IN POUNDS

Here is a hint for coding the proper format string to print the detail lines:

- `print('{0:<20}{1: > 15,}{2: > 15,}{3: > 15,}'.format(`

When your report is finished, test it using both the clean and dirty versions of the input file:

- land_mammals.txt
- dirty_land_mammals.txt

Note that because this report program uses the basic version of the class, the dirty input file will lead to the creation of a dirty report (see below).

When testing, please check that the unit test output from the *my_basic_land_mammals.py* module is NOT printed in the test output for this program.

When this program is run, the console session should look like this:

```
Please enter input file name: land_mammals.txt

                        BY LAND MAMMAL NAME

Land Mammal             Minimum Mass   Maximum Mass Variability of
Name                       in Pounds      in Pounds Mass in Pounds
African elephant              10,000         24,000         14,000
American bison                   700          2,200          1,500
Asian elephant                 8,000         17,640          9,640
Black rhinoceros               1,500          4,000          2,500
Cape buffalo                   1,100          2,200          1,100
Gaur                           1,000          3,000          2,000
Giraffe                        1,544          4,255          2,711
Hippopotamus                   2,500          8,820          6,320
Water buffalo                    660          2,200          1,540
White rhinoceros               3,000          9,920          6,920

            BY DESCENDING VARIABILITY OF MASS IN POUNDS

Land Mammal             Minimum Mass   Maximum Mass Variability of
Name                       in Pounds      in Pounds Mass in Pounds
African elephant              10,000         24,000         14,000
Asian elephant                 8,000         17,640          9,640
White rhinoceros               3,000          9,920          6,920
Hippopotamus                   2,500          8,820          6,320
Giraffe                        1,544          4,255          2,711
Black rhinoceros               1,500          4,000          2,500
Gaur                           1,000          3,000          2,000
Water buffalo                    660          2,200          1,540
American bison                   700          2,200          1,500
Cape buffalo                   1,100          2,200          1,100

Please enter input file name: dirty_land_mammals.txt

                        BY LAND MAMMAL NAME

Land Mammal             Minimum Mass   Maximum Mass Variability of
Name                       in Pounds      in Pounds Mass in Pounds
                                 700          2,200          1,500
African elephant              -10,000         24,000         34,000
Asian elephant                 8,000         17,640          9,640
Black rhinoceros               1,500              0         -1,500
Cape buffalo                   1,100          2,200          1,100
Gaur                           1,000          3,000          2,000
Giraffe                        1,544          4,255          2,711
```

Hippopotamus	2,500	8,820	6,320
Water buffalo	660	2,200	1,540
White rhinoceros	9,920	3,000	-6,920

BY DESCENDING VARIABILITY OF MASS IN POUNDS

Land Mammal Name	Minimum Mass in Pounds	Maximum Mass in Pounds	Variability of Mass in Pounds
African elephant	-10,000	24,000	34,000
Asian elephant	8,000	17,640	9,640
Hippopotamus	2,500	8,820	6,320
Giraffe	1,544	4,255	2,711
Gaur	1,000	3,000	2,000
Water buffalo	660	2,200	1,540
	700	2,200	1,500
Cape buffalo	1,100	2,200	1,100
Black rhinoceros	1,500	0	-1,500
White rhinoceros	9,920	3,000	-6,920

Exercise 3
Create a Python module file named *my_evolved_land_mammals.py* by copying the module created in Exercise 1 and making changes. This module should contain the following:

1. A class named *LandMammal* that contains the evolved implementation of a class that holds data facts regarding the world's largest land mammals.
2. A *main()* function that contains unit test cases for the *LandMammal* class.

This evolved version of the *LandMammal* class SHOULD INCLUDE implicit getter and setter features implemented using the Python *@property* decorator. When doing this part of the assignment, follow the approach that I demonstrated in the Part 3 tutorial video. It may also be helpful to consult the Cheat Sheet while doing this exercise.

The implicit setter code should check for the following input file data errors and raise exceptions when they are found:

- Name is set to empty string.
- Minimum mass in pounds < 1.
- Maximum mass in pounds < 1.
- Maximum mass in pounds is less than minimum mass in pounds.

When writing the setter code and the related unit tests, follow the method that I demonstrate in Part 3 of the Tutorial video.

When this program is run directly (rather than having been imported), the console session should contain the unit testing output and should look like this:

```
Unit testing output follows...

Test Case #1: Test constructor
Passed

Test Case #2: Test str method
Passed

Test Case #3: Test repr method
Passed

Test Case #4: Test calculate_variability_of_mass_in_pounds
Passed

Test Case #5: Test passing empty string to name setter
Passed

Test Case #6: Test passing zero to minimum_mass_in_pounds setter
Passed
```

```
Test Case #7: Test passing zero to maximum_mass_in_pounds setter
Passed

Test Case #8: Test passing lesser maximum value than minimum
value to mass in pounds setters
Passed
```

Please continue to Exercise 4 on the next page…

Exercise 4

Create a program named *create_land_mammal_mass_reports_using_evolved_class* by copying the *create_land_mammal_mass_reports _using_basic_class* program created in exercise 2. When coding and testing this program, follow the approach that I take in the Part 4 tutorial video. The changes that you make to the original program should include:

- Change the import statement such that the *LandMammal* class is now imported from the *my_evolved_land_mammals.py* module that you created in Exercise 3.
- No further changes should be required!

When testing, please check that the unit test output from the *my_evolved _land_mammals.py* module is NOT printed in the test output for this program.

Note that because this report program uses the evolved version of the class, the dirty input file will lead to a runtime error rather than a report showing dirty values.

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

```
Please enter input file name: land_mammals.txt


                        BY LAND MAMMAL NAME

Land Mammal             Minimum Mass   Maximum Mass Variability of
Name                       in Pounds      in Pounds Mass in Pounds
African elephant              10,000         24,000         14,000
American bison                   700          2,200          1,500
Asian elephant                 8,000         17,640          9,640
Black rhinoceros               1,500          4,000          2,500
Cape buffalo                   1,100          2,200          1,100
Gaur                           1,000          3,000          2,000
Giraffe                        1,544          4,255          2,711
Hippopotamus                   2,500          8,820          6,320
Water buffalo                    660          2,200          1,540
White rhinoceros               3,000          9,920          6,920


            BY DESCENDING VARIABILITY OF MASS IN POUNDS

Land Mammal             Minimum Mass   Maximum Mass Variability of
Name                       in Pounds      in Pounds Mass in Pounds
African elephant              10,000         24,000         14,000
Asian elephant                 8,000         17,640          9,640
White rhinoceros               3,000          9,920          6,920
Hippopotamus                   2,500          8,820          6,320
Giraffe                        1,544          4,255          2,711
Black rhinoceros               1,500          4,000          2,500
```

```
Gaur                          1,000          3,000          2,000
Water buffalo                   660          2,200          1,540
American bison                  700          2,200          1,500
Cape buffalo                  1,100          2,200          1,100


Please enter input file name: dirty_land_mammals.txt
Traceback (most recent call last):
  File
"/Users/kevintrainor/Documents/____my_python_course_projects/trai
nor_kevin_exercises_severance_chapter_14/create_land_mammal_mass_
reports_using_evolved_class.py", line 57, in <module>
    main()
  File
"/Users/kevintrainor/Documents/____my_python_course_projects/trai
nor_kevin_exercises_severance_chapter_14/create_land_mammal_mass_
reports_using_evolved_class.py", line 10, in main
    mammals = get_mammals()
  File
"/Users/kevintrainor/Documents/____my_python_course_projects/trai
nor_kevin_exercises_severance_chapter_14/create_land_mammal_mass_
reports_using_evolved_class.py", line 27, in get_mammals
    LandMammal(name, int(minimum_mass_in_pounds_as_string),
int(maximum_mass_in_pounds_as_string))
  File
"/Users/kevintrainor/Documents/____my_python_course_projects/trai
nor_kevin_exercises_severance_chapter_14/my_evolved_land_mammals.
py", line 14, in __init__
    self.minimum_mass_in_pounds = int(minimum_mass_in_pounds)
  File
"/Users/kevintrainor/Documents/____my_python_course_projects/trai
nor_kevin_exercises_severance_chapter_14/my_evolved_land_mammals.
py", line 35, in minimum_mass_in_pounds
    raise AttributeError('The minimum_mass_in_pounds attribute
must be populated with an integer value > 0')
AttributeError: The minimum_mass_in_pounds attribute must be
populated with an integer value > 0
```

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_severance_chapter_14
```

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

```
trainor_kevin_exercises_severance_chapter_14
```

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_severance_chapter_14.zip
```

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

```
trainor_kevin_exercises_severance_chapter_14.zip
```

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

Last Revised
2021-03-25