Lab 7
Object-Oriented Programming

Tasks:

1. (Car Class with Read-Only Properties) Write a class **Car** that contains the **model**, **year** (of manufacture), and **speed** of a car as **hidden attributes** (name these attributes with a leading underscore). Provide the necessary **read-only properties**. Write two methods to drive the car **faster** and **slower**. Test your Car class.

2. (Coin Class) Write a Coin Class to keep track of the number of 2 euro, 1 euro, 50 euro cent, 20 euro cent, and 10 euro cent coins in a wallet. Provide a **read-only property universal_str** that returns a string with the number of coins for each coin type. Write a second method that returns the total amount of money. Keep your **attributes hidden** and use the read-only properties. Create a coin object and test the methods.

3. (Modifying the Internal Data Representation of a Class) Section 10.4.2's Time class represents the time as three integer values. Modify the class to ***store the time as the total number of seconds since midnight***. Replace the _hour, _minute and _second attributes with one **_total_seconds** attribute. Modify the bodies of the hour, minute and second properties' methods to get and set _total_seconds. Re-execute Section 10.4's IPython session using the modified Time class to show that the updated class Time is interchangeable with the original one.

4. (Calculating Vector Lengths) A vector is an object that has a magnitude and a direction. A vector is represented as a line segment with an arrow where the arrow represents the direction and the length of the vector is the magnitude. A vector can also be a single point with x- and y-coordinates. The magnitude calculated from this type of vector equals the distance between the origin of the cartesian coordinate system and the point. Write a class Point that represents an (x-y) coordinate pair and provides x and y **read-write properties** as well as a method to calculate the distance between the point and the origin of the coordinate system. Include __init__ and __repr__ methods. Write a class Vector that has two Point objects as its attributes. Include __init__ and __repr__ methods and a method that can calculate the magnitude of the vector. Test your Point and your Vector class.

5. (Manipulating Dates and Times with Module datetime) The Python Standard Library's **datetime** module contains a datetime class for manipulating dates and times. The class provides various **overloaded operators**. Research class datetime's capabilities, then perform the following tasks:
    a) Get the current date and time and store it in variables.
    b) Repeat Part (a) and store the result in variable y.
    c) Display Each Datetime Object.
    d) Display each datetime object's data attributes individually.
    e) Use the comparison operators to compare the two datetime objects.
    f) Calculate the difference between y and x.

6. (Calculator Class) Write a class that implements a Calculator. The class should contain 2 numbers. Provide an __init__ method that takes the 2 numbers as arguments. Also, provide the following read-only properties:
    a) **sum** returns the sum of the 2 numbers
    b) **difference** returns the difference between the 2 numbers
    c) **product** returns the product between the 2 numbers
    d) **division** returns the quotient of number1 divided by number2
    These properties should not have corresponding data attributes; rather, they should use side in calculations that return the desired values. Create a Calculator object and display its properties.

7. (Player Class) Write a class called Player that can be used in a turn-based strategy game to create player objects. Create the following data attributes-the **name** (a string), **level** (an int), **strength** (an int), and **Health Points** (a Decimal). The class should have an __init__ method to initialize the data attributes. Provide a property for each data attribute. Use validation in the properties of the level and health attribute to ensure that they remain non-negative. Provide a **defence** method to defend from an attack. The intensity of the attack is calculated as follows—divide the strength of the attacker by a random num- ber between 1 and 3, subtract from this result the strength of our player multiplied with the level of our player. If the intensity is ≤ 0, the player loses, and the intensity is subtracted from the health points of the player.

8. (Class Fraction) The Python Standard Library module fractions provides a Fraction class that stores the numerator and denominator of a fraction, such as:
    2 / 4
    Research Fraction's capabilities, then demonstrate:
    a) AddingtwoFractions.
    b) Subtracting two Fractions.
    c) MultiplyingtwoFractions.
    d) Dividing two Fractions.
    e) Printing Fractions in the form a/b, where a is the numerator and b is the denominator.
    f) Converting Fractions to floating-point numbers with built-in function float.