Dr.Erdem KAYA
ekaya@metu.edu.tr
Mehmet Taha ŞAHİN
staha@metu.edu.tr

Homework 02
METU CENG310 Fall 2021-2022
Data Structures and Algorithms with Python

Start Date: November 11th, 2021
Due Date: November 26nd, 2021

1 Operator Overloading

Write a `Vector` class that can handle various vector operations. You can (and should) start with the basic implementation shown in subsection 2.3.3 in the textbook.
Task-1: Copy the Code Fragment 2.4 from the book and make sure that the basic `Vector` class works. Simple test operations would be as listed:

```
In [47]: %run C:\                                    ecture-02\HW2\Vector.py

In [48]: Vector()
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-48-bcda4006949f> in <module>
----> 1 Vector()

TypeError: __init__() missing 1 required positional argument: 'd'

In [49]: Vector(5)
Out[49]: <__main__.Vector at 0x4da3760>

In [50]: v = Vector(5)

In [51]: len(v)
Out[51]: 5

In [52]: str(v)
Out[52]: '<0, 0, 0, 0, 0>'

In [53]: v
Out[53]: <__main__.Vector at 0x4307550>

In [54]: v[3] = 5

In [55]: v[3]
Out[55]: 5

In [56]: u = Vector(5)

In [57]: u[3] = 3

In [58]: str(u + v)
Out[58]: '<0, 0, 0, 8, 0>'

In [59]: u == v
Out[59]: False

In [60]: u != v
Out[60]: True
```

Enrich the `Vector` class such that it supports vector subtraction, negation, addition, multiplication, and dot product operations.
Task-2: Implement the `__sub__` method for the `Vector` class so that the expression `u-v` returns a new vector instance representing the difference between two vectors.
Task-3: Implement the `__neg__` method for the `Vector` class so that the expression `-v` returns a new vector instance whose coordinates are all the negated values of the respective coordinates of `v`.
Task-4: The `Vector` implementation in Section 2.3.3 supports a syntax such as `v = u + [5, 3, 10, -2, 1]`, in which the sum of a vector and list returns a new vector. However, the syntax `v = [5, 3, 10, -2, 1] + u` is illegal. Explain how the Vector class definition can be revised so that this syntax generates a new vector.
Task-5: Implement the `__mul__` method for the Vector class so that the expression `v * 3` returns a new vector with coordinates that are 3 times the respective coordinates of `v`.
Task-6: Implement the `__rmul__` method, to provide additional support for syntax `3 * v`.
Task-7: Enhance the `__mul__` method for the `Vector` class so that the expression `u * v` returns a scalar that represents the dot product of the vectors.
Task-8: Modify the constructor so that if a single integer is sent, it produces a vector of that dimension with all zeros (e.g., `Vector(3)` should produce `<0,0,0>` vector), but if a sequence of numbers is provided, it produces a vector with coordinates based on that sequence (e.g.,`Vector([1,2,3])` should produce `<1,2,3>` vector).

Example Test Statements (`$>` denotes the console prompt.):

```
$> u = Vector(3)
$> v = Vector(3)
$> u[0] = 1
$> v[1] = 1
$> u + v
```

```
$> u + [1,1,1]
$> [1,1,1] + u
$> u - v
$> u * 3
$> 3 * u
$> u * v
$> -u
```

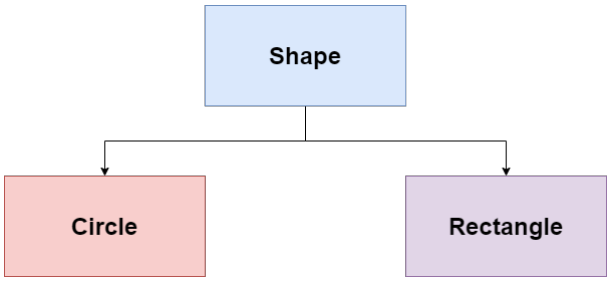
2 Class Inheritance

Introduction: For this assignment, you are asked to develop a Python program that creates shapes.

You are expected to create the `Point` class to represent a location on the coordinate system. `Point` class has two values; `x` and `y`. Point objects are initialized as follows:

```
p = Point(x, y);
```

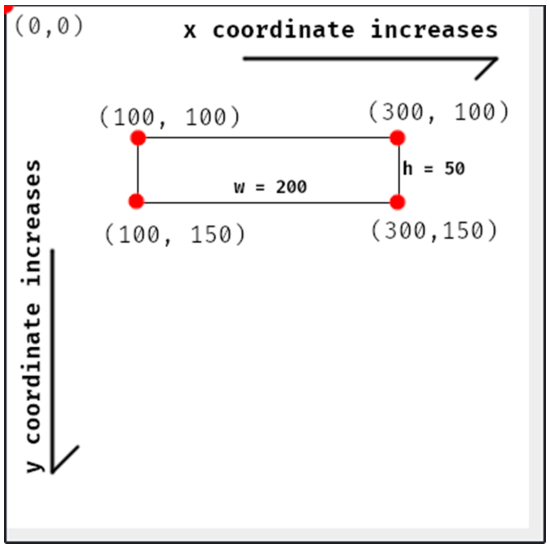
Shape Class: `Shape` class has a point variable named `leftTop`. `leftTop` point coordinates will be entered by the user through keyboard input. You can think the top left point as the anchor of the shape.
Shape class also has a points list as well. This list keeps the ordered list of vertices (in the top left, top right, bottom right and bottom left order) for a shape. For each shape, `calculatePoints()` function defines the points and adds them to the list. `calculateArea()` and `calculatePerimeter()` functions calculate the area and perimeter for the shape. For moving an object from one location to another with respect to the `leftTop` point, `move()` function needs to be implemented. Do not forget, you do not know how to fill these four functions in `Shape` class. Because you do not know the type of it. So, these functions need to be implemented as abstract in `Shape` class. You need to override these functions in `Circle` and `Rectangle` classes. You can see the class hierarchy in the figure below.



You can see the demo about the details of `__str__` implementation.

Calculating The Points: The following figure shows an example for a rectangle created with `leftTop` point (100,100), height of 50 and width of 200. The annotations were added later. Rectangle objects initialized as follows:

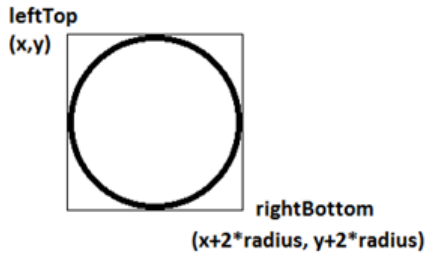
```
r = Rectangle(leftTop, height, width);
```



You need to keep the points of rectangular objects in the points List, described under `Shape` class. Again, the order should be: top left, top right, bottom right and bottom left.

For the `Circle` class, there are only two points that you need to keep track of; top left and bottom right as shown below. You can populate the points list with that order, using just 2 points. Circle objects initialized as follows:

```
r = Circle(leftTop, radius);
```



Scenario: The program will ask the user, type of shape in an infinite loop. The user has 3 different command options here. Command `r` for the selecting rectangle, `c` for the selecting circle, and `q` to exit the program.

If the command `r` is typed, the program would ask for the coordinates for the `leftTop` (`x` and `y`), `height` and `width`. Information on the rectangle will be printed afterward. Then it will ask for the new `leftTop` coordinate to move the rectangle. After the moving operation done by the program new information of the rectangle will be printed.

If the command `c` is typed, the program would ask for the coordinates for the `leftTop` (`x` and `y`) and `radius` . Information on the circle will be printed afterward. Then it will ask for the new `leftTop` coordinate to move the circle. After the moving operation done by the program new information on the circle will be printed.

The program will repeat this scenario again and again until the `q` command is typed.

Demo: Red ones are inputs and after the demo part, there is a further instructions part. Do not forget to look at that part before submitting your work.

```
Type of Shape (q for exit):  r

Coordinate(leftTop), height and width:  100 100 50 100
--Rectangle--
Height: 50
Width: 100
Left Top Point: (100,100)
Area: 5000.00
Perimeter: 300.00
Points: (100,100) (200,100) (200,150) (100,150)

Move object to the new coordinate(leftTop):  30 40
--Rectangle--
Height: 50
Width: 100
Left Top Point: (30,40)
Area: 5000.00
Perimeter: 300.00
Points: (30,40) (130,40) (130,90) (30,90)

Type of Shape (q for exit):  c

Coordinates(leftTop) and radius:  20 20 30
--Circle--
Radius: 30
Left Top Point: (20,20)
Area: 2827.43
Perimeter: 188.50
Points: (20,20) (80,80)

Move object to the new coordinate(leftTop):  100 150
--Circle--
Radius: 30
Left Top Point: (100,150)
Area: 2827.43
Perimeter: 188.50
Points: (100,150) (160,210)

Type of Shape (q for exit):  c

Coordinates(leftTop) and radius:  564 742 10
--Circle--
Radius: 10
Left Top Point: (564,742)
Area: 314.16
```

```
Perimeter: 62.83
Points: (564,742) (584,762)

Move object to the new coordinate(leftTop):  5 10
--Circle--
Radius: 10
Left Top Point: (5,10)
Area: 314.16
Perimeter: 62.83
Points: (5,10) (25,30)

Type of Shape (q for exit):  r

Coordinate(leftTop), height and width:  45 10 200 150
--Rectangle--
Height: 200
Width: 150
Left Top Point: (45,10)
Area: 30000.00
Perimeter: 700.00
Points: (45,10) (195,10) (195,210) (45,210)

Move object to the new coordinate(leftTop):  683 246
--Rectangle--
Height: 200
Width: 150
Left Top Point: (683,246)
Area: 30000.00
Perimeter: 700.00
Points: (683,246) (833,246) (833,446) (683,446)

Type of Shape (q for exit):  q
```

3 Further Instructions

Please hand in the results of your homework in text files with .py extension over ODTUClass by 11:59pm on due date. A Homework-02 page will be generated soon after the start date of this homework and there will be separate submission sections. One for **Operator Overloading** and the other for **Class Inheritance**. **Should you have any questions pertaining to homework tasks, please ask them in advance (not on the due date) to TA** for your own convenience. Whatever IDE you use, you have to make sure that your code works from a Python interpreter or iPython (e.g., `$> python.exe vector.py` or (after initiation of iPython (`$> ipython.exe`), `In [1] %run vector.py`).