Programming Assignment: Vectors

Background:

To make the lab simpler, I have split the lab up into parts. Complete each part of the lab. Do NOT edit the cases.py file. Edit the Vector.py ONLY!

Vector.py contains the Vector class that you need to edit
Cases.py contains the test cases that will run to see if your code is correct or not.
Upon running cases.py, you'll see which test cases you passed and failed.

Some helpful lines of code:

To make an empty list	`arr = []` `arr = list()`
To access something in a list using the index	`array = [-1, -2, -3, -4, -5]` `print(array[3]) #-4`

To add something onto the end of a list	`array = [1, 2, 3]`
	`array.append(4)`
	`# array is now [1, 2, 3, 4]`

To get the length of a list	`twoDarr = [1, 2, 3, 4, 5, 6]` `print(len(twoDArr)) # 6`

To loop through a list using a for loop	`arrloop = [1, 2, 3, 4, 5, 6]`
	`for index in range(len(arrloop)):`
	`  #do stuff`

To loop through a list using a for each loop	`arrloop = [1, 2, 3, 4, 5, 6]`
	`for element in arrloop:`
	`  #do stuff`

How to use operator overloading:

Let's say that you have the following:

```
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

p1 = Point(1, 2)
p2 = Point(2, 4)
print(p1 + p2)
```

You would get an error because Python is unable to decipher how "+" should work when it has the two Point objects as arguments.
We must overload the operator by implementing __add__() in the class.

To overload other operators:

Operator	Expression	Function
Addition	`c1 + c2`	`__add__(self, other)`
Subtraction	`c1 - c2`	`__sub__(self, other)`
Multiplication	`c1 * c2`	`__mul__(self, other)`

For more operator overloading, click here

See the example below:

```
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

# This would work with the case: self + other (in this order)
# This adds to the x/y attributes of self and other
# Returns a new Point
def __add__(self, other):
        x = self.x + other.x
        y = self.y + other.y
        return Point(x, y)

p1 = Point(1, 2)
p2 = Point(2, 4)
print(p1 + p2) # (3, 6)
```

Now you can see that the + function works without throwing an error

You will be using operator overloading to complete the vector subtraction and dot product functions of our Vector class.

Part 0 (looking at what is completed):

The following methods are already completed:

- Constructor
- get_ith_element
- __add__() – for vector addition
- __str__() = toString method

Please look through these and make sure that you understand these. You do not need to change these functions.

Part 1 (vector subtraction):

```
Function: __sub__()
inputs: self - vector1
        other - vector2
output: sum of two vectors or None
Post-Condition: the function returns the sum IF IT
IS POSSIBLE. If it's impossible, return None.
```

Example (Look at python.py for more cases):

Sample code	Command Window

`v0_0 = [1, 2, 3, 4, 5, 6]`
`v0_1 = [-1, -2, -3, -4, -5, -,6]`
`x = v0_0 + v0_1`
`print(x)`
	`[2, 4, 6, 8, 10, 12]`

`v1_0 = [1, 2, 3, 4, 5, 6]`
`v1_1 = [-1, -2, -3]`
`x = v1_0 + v1_1`
`print(x)`
	`None`

Part 2 (dot product):

Your next task is to complete the dot product function.

```
Function: __mul__()
inputs: self - vector1
        other - vector2
output: numerical value or None
Post-Condition: the function returns the dot product IF IT
IS POSSIBLE. If it's impossible, return None.
```

Example:

$[1,2,3]\cdot[-1,-2,-3]$
$=1(-1)+2(-2)+3(-3)$
$=-1-4-9=-14$

Example (Look at python.py for more cases):

Sample code	Command Window

`v0_0 = [1, 2, 3, 4, 5, 6]`
`v0_1 = [-1, -2, -3, -4, -5, -,6]`
`x = v0_0 * v0_1`
`print(x)`
	`-91`

`v1_0 = [1, 2, 3, 4, 5, 6]`
`v1_1 = [-1, -2, -3]`
`x = v1_0 * v1_1`
`print(x)`
	`None`

Testing:

There are 30 test cases. To test your code, please run the `cases.py` file, NOT the `vector.py` file. Please see sample below. (Yours will look a little difference because I had to change this terminal output to make it fit in one page. You will probably have to take more than one screenshot to capture everything)

```
[eunice@Eunices-MBP pa- vectors % python cases.py
VECTOR ADDITION!!!!
Test case 1: passed
========================================
Test case 2: passed
========================================
Test case 3: passed
========================================
Test case 4: passed
========================================
Test case 5: passed
========================================
Test case 6: passed
========================================
Test case 7: passed
========================================
Test case 8: passed
========================================
Test case 9: passed
========================================
Test case 10: passed
========================================
VECTOR SUBTRACTION!!!!
Test case 11: passed
========================================
Test case 12: passed
========================================
Test case 13: passed
========================================
Test case 14: passed
========================================
Test case 15: passed
========================================
Test case 16: passed
========================================
Test case 17: passed
========================================
Test case 18: passed
========================================
Test case 19: passed
========================================
Test case 20: passed
========================================
DOT PRODUCT!!!!
Test case 21: passed
========================================
Test case 22: passed
========================================
Test case 23: passed
========================================
Test case 24: passed
========================================
Test case 25: passed
========================================
Test case 26: passed
========================================
Test case 27: passed
========================================
Test case 28: passed
========================================
Test case 29: passed
========================================
Test case 30: passed
========================================
```

THIS IS FOR MAC
If you are using Windows, use
`py cases.py`

Submission:

Submit the screenshot(s) of your output AS A PDF. Also submit your source code (`vectors.py`) as a .txt file