Karl Michel Koerich

Lab #7

Computer Programming

Professor Vincent

Oct 30, 2017

**1 Class basics**

2. TypeError: 'module' object is not callable – This error appears since point is a module, but point is also the name of the class. To initialize an object from a Class, values for attributes have to be given.

4. What is actually printed is: 1 4 2 6 – These number are printed since they correspond to the values we initialized x and y of ‘a’ and ‘b’ with.

5. What is printed is: 5.0 5.0 0.0 – These are the distances from calculated by the method of the Class Point. On the last example, since the parameter given to the function distance was the object itself, the distance was 0.0. The distances appear in float value because of the calculations performed inside the distance method.

6. An error saying the name is not defined appears. This happens because the method distance is not global, it is specific for the class Point.

7. It prints: point(3, 2) point(4, 6)

8. It prints: point(3, 2) point(6, 6) point(6, 6) True – This happens because now ‘c’ and ‘b’ refer to the same space in memory, and if ‘c’ is altered, consequently ‘b’ changes also.

9. point(3, 2) point(6, 6) point(10, 6) False – This happens because now ‘c’ and ‘b’ refer to different spaces in memory, and if ‘c’ is altered ‘b’ does not change.

10. False True – This happens because ‘a.\_\_eq\_\_(b)’ works as ‘a==b’ and ‘c.\_\_eq\_\_(b)’ works as ‘c==b’

11. True True False – The False is printed since a is not an instance of the class int …/… False True True – The False is printed since a.x is an int, therefore is not an instance of the class point.

**2 Modifying the class**

1. The form in which a point is printed is different since we changed the \_\_repr\_\_ method.

2. Because that attribute belongs to a specific instance of the class. We could have multiple instances of a class, therefore, by using just x would make it impossible to know which instance the x is associated with.

3. We are basically just printing a message and returning True for whatever the points are, which is not what we expect == to do.

4. TypeError: unsupported operand type(s) for +: 'point' and 'point' – This error occurs because the class Point does not support the use of the + operator.

5. point(self.x + other.x, self.y + other.y)

6. POINT: 1, 2 POINT: 3, 4 POINT: 4, 6 – This is printed since now that we have implemented the + operand, when we do a + b we actually get a point that the x and y are the addition of the x and y of a and b.

**3 Basics of Inheritance**

2. <class '\_\_main\_\_.myList'> <class 'list'>…/… True – This is printed since the type of a is the class myList that we created above, and the type of b is just our regular built in class List. A and b have the same value, the same list in them, that why its True.

3. The same as would have happened if a belonged to class List. Since myList inherited the method and attributes from List, it behaves the same way as we would expect list to behave.

4. For a, the product is printed while for b, it is not. This happens because class List does not have the method product that we created for myList, but myList has all the method from List, it inherited them.

5. Because the operations that the methods ‘distance’ are doing are different.

(I DO NOT HAVE THE ENTIRE IDLE SESSION… I did several testing in between and it was too messy. Also, I closed and opened it a lot of times losing some part of the stuff I typed in… Thanks for understanding.)