420-PRO-LCU Programming in Python - Assignment 3

Due: See Omnivox

Preliminary

In this assignment you will add **four** methods to an existing class that represents a circle, and write some code to test these methods.

To begin, extract the files `circle.py` and `point.py` from the zip file on Omnivox and put them into the same working folder. The zip also includes a Python file that is a mini-tutorial on the `circle` class. Be sure you understand this code before you begin coding this assignment.

Exercise 1

Add the following four methods to the `circle` class in `circle.py` as follows:

- A Boolean-valued method with the definition:

```
def intersects(self, other):
    '''returns True if this circle (denoted by self) and the other
    circle have at least one point in common, and False otherwise.'''
```

  Note that two circles intersect if and only if the distance between their centers is less than or equal to the sum of their radii. Remember that `distance()` is a method of the `point` class. For our purposes, we consider two circles "intersecting" if *any* part of their areas overlap.

- A Boolean-valued method with the definition:

```
def tangent(self, other):
    '''Returns True if the circle self is either internally or externally
    tangent to the circle other.'''
```

  This method should work by calling two other methods. Remember that circles can be either internally tangent or externally tangent. Write two different methods, `internally_tangent`, and `externally_tangent`, to test for each condition separately.

  - Two circles are externally tangent if and only if the distance between their centers equals the sum of the radii.
  - Two circles are internally tangent if and only if the distance between their centers equals the absolute value of the difference in their radii.

  Sketch these two situations out if you are having difficulty visualizing them.

  Remember that to be tangent, the circles must intersect. So you should start both of your tangent methods with a call to `intersects()` using an `if/else` construction:

```
def internally_tangent(self, other):
    if not self.intersects(other):
        return False # Can't possibly be tangent
    else:
        # Return True if the circles are internally tangent.
```

  Your `externally_tangent` method should have a similar structure to this example.

Once you have defined each of the specific (internal or external) tangent functions, use them in your combined `tangent()` function by using Boolean operators to combine their results.

Exercise 2

Create a separate module (call it a3_ex2.py) that imports the `circle` class from `circle.py`. This is where you will put the code to test *all four* of your new methods.

Write enough test cases to illustrate and test each possibility - disjointness, intersection with internal tangency, intersection with external tangency, or intersection without tangency. Test that each method returns the appropriate value in each case.

Be sure to test two different cases of intersection. Remember that any overlapping circles *should* be considered intersecting, so be sure to test the cases where you have two *completely* overlapping circles, as well as two partially overlapping circles.

1. Again, draw sketches on paper to help understand the geometry.
2. It may be a good idea to the use the `isclose()` function (*NOT* a method) from the `point` class. This is because the limited precision of floating-point values may sometimes cause equality checks to fail in unexpected ways (we discussed this in an early lecture).

Submitting your work

You must combine all three Python files (`point.py`, `circle.py`, and `a3_ex2.py`) into a single ZIP archive and submit it via Omnivox.

Things to remember for both exercises:

1. The identification sections.
2. To comment adequately.
3. Design useful, readable output.

In both exercises, the course corrector and myself may further test your solutions using our own data.