SFWR ENG 2AA4: Assignment 1 Solution

Meet Patel, patelm16

January 31, 2017

This exercise uses Python to write a program which creates, uses, and tests an ADT that stores circles. The program consists of the following files: `CircleADT.py`, `Statistics.py`, `testCircles.py`, and `Makefile` as shown in the Appendices. My files will be shown from Appendices C- F, and my partner's files will be in Appendices G- H.

Contents

1 Testing Results

1.1 Results of Testing My Files

Function	Result	Additional Comments
xcoordTest	pass	got correct x-coordinate
ycoordTest	pass	got correct y-coordinate
radiusTest	pass	got correct radius
areaTest	pass	got correct area of circle
circumferenceTest	pass	got correct circumference of circle
insideBoxTest	pass	cases: circle inside box, outside box, and edges touching box
intersectTest	pass	cases: one circle inside another, two same circles, no common points
scaleTest	pass	radius value was correctly scaled
translateTest	pass	x and y coords tested correctly for positive and negative numbers
averageTest	pass	got correct area of list of circles
stdDevTest	pass	got correct standard deviation of list of circles
rankTest	pass	correctly worked for unique radii and repeated radii in list

Table 1: Results of testing my files

My CircleADT.py code can be found in Appendix C.
My Statistics.py code can be found in Appendix D.
My testCircles.py code can be found in Appendix E.
My Makefile code can be found in Appendix F.

1.2 Testing the Makefile

To check that my makefile was working correctly, I had to make sure that calling make test correctly ran my testCircles module and calling make doc correctly created the HTML and Latex folders for my entire project. I was able to use my knowledge from 2XA3 and the example on the course's gitlab page to get my makefile to work properly.

1.3 Results of Testing Partner's Files

My partner's CircleADT.py code can be found in Appendix G.
My partner's Statistics.py code can be found in Appendix H.

Function	Result	Additional Comments
xcoordTest	pass	got correct x-coordinate
ycoordTest	pass	got correct y-coordinate
radiusTest	pass	got correct radius
areaTest	pass	got correct area of circle
circumferenceTest	pass	got correct circumference of circle
insideBoxTest	pass	cases: circle inside box, outside box, and edges touching box
intersectTest	pass	cases: one circle inside another, two same circles, no common points, one common point
scaleTest	pass	radius value was correctly scaled
translateTest	pass	x and y coords tested correctly for positive and negative numbers
averageTest	pass	got correct area of list of circles
stdDevTest	pass	got correct standard deviation of list of circles
rankTest	pass	correctly worked for unique radii and repeated radii in list

Table 2: Results of testing partner's files

2 Discussion

2.1 Test Results

When testing my circleADT and Statistics modules, I had to make sure that the edge cases were convered as well as the expected cases. Testing for the expected cases was simple as I was able to get my functions other than rank and intersect to work successfully by the first test. For many functions, I had more than one test case to check if the fucntion worked with various kinds of inputs. For the function intersect, I assumed that the two circles intersect if they had a common point and had to make sure I tested if the two circles are the same, one circle inside another, and the two circles not touching at all. I forgot to check for one point touching only because I assumed it would work the same as a circle inside another. I realized that in my mathematical statement I should have used $<=$ rather than just $<$when comparing distances of the two circles. For the rank function, I had to check that it worked when radii were repeated and after playing around with my code, I was able to successfully test for it as well. As for the insideBox function, I had to test for the circle inside the box, outside the box, and touching the edges of the box. I had successfully tested for this as I was able to correctly implement the design easily from the diagram shown in the project document. At the end, when I was successful with all my test cases, I saw that it was a lot of trial and error to get things to work correctly

especially with the rank and intersect function. Also, for the hard-coded values, I had to first run the program to see what numbers python gives and also check that it matches to the real calculated value before I used it in my test case. I feel that other than the one-point touching test for the intersect method, I covered all other possible cases which I thought were important for all the functions in circleADT and Statistics.

My partner, Shalmi Patel's circleADT and Statistics modules worked perfectly with my test cases. The modules covered for various inputs and she also made sure that her intersect method worked for one common point. She used the absolute function to good effect in more than one case to cover for negative scaling values, and for negative coordinates. Looking at her implementation, it seems that although most methods were simple and hence done similarily to mine, the intersect method was done slightly differently than mine. This shows that the same functions could be implemented in different ways.

2.2 What I Learnt

One of the biggest things I learnt from doing this assignment was that there are various ways to achieve the same functionality while creating a program. Due to the ambiguity of the specifications, I figured that people are implementing their ADT in various ways. I also learnt that using a package like Numpy helps to improve reliability and productivity as it was very quick to use. Since it was a specification, it was also standardized throughout everyone's programs which ultimately should have yielded the same values. This assignment also taught to code slightly better in the Python language as the last time I coded in it was last year and using Java since then has been slightly different. Although using Doxygen seemed to be tedious at first, I found that commenting in it was very standardized and helpful with files it ends up creating. This assignment also taught me to better test my programs as I ended up making a couple mistakes prior to testing and debugging and testing helped me find out where those mistakes were at. Learning about and of Latex was very important to me as I actually find it much quicker to use than Microsoft Word. Lastly, this assignment also me improve my time management and problem solving skills as I thought the testing of the program would be quick and easy; it ended up being the harder part of the assignment as it took a long time to think and figure out the defects of my program. The functions intersect, insideBox, and rank were intersecting as they took time and thought to complete. All in all, this assignment actually made me feel like I am in software engineering and not just general engineering anymore.

2.3 Problems Found with my Program

My program did not have many flaws after its completion. However, one thing I missed out on testing for which I realized after was testing for the intersection of one point only between two circles. In my function, I should have used <= instead of just <. Other than this, I found the program to work just fine with the assumptions I made. While creating the program, the major problem I encountered was getting rank to work the way I wanted it to work; which was for it skip a number if the previous number was repeated. In other words, I wanted the rank of [2,5,11,5,6] to be [5,3,1,3,2]. Other than these, I did not find any problems with my program.

2.4 Problems Found with Partner's Program

My partner's program worked with all my test cases and also the test for one common point for the intersect function. The program seemed to be very nicely organized and efficient, but I felt that her comments did not state her assumptions for rank, insideBox, and intersect. For insideBox, she did not explicitly state the assumption of whether the edges of the circle touching the edges of the box count as inside or outside the box. Also for rank, she did not include her assumption of she wants her output to be as it had vague requirements. For the intersect function, she did not indicate the assumption of < or <=. Other than that, I found the program's implementation to be correct with all the functions.

2.5 Problems Found with the Specification of Modules

The specifications of the modules for this assignment were very vague and hence, we had to actually think about how to do certain things instead of being told how to do it. For example, the rank function was open to interpretation in terms of how our output is. Although it was a relatively simple ADT to create and test, it was still open-ended to the extent where we had to think of our own implementation. The testing part of the assignment was most vague as we had to think of our own cases to test and also figure out how we are testing out methods.

2.6 Handling Pi and Rationale

For the value of pi, I imported python's math library and used math.pi. Math.pi in python is a constant and I used it because it is a standardized value which can be reused to the same amount of precision everytime. If I were to write out the actual numbers, it would simply be more difficult to reuse it with the same amount of precision. Using the actual constant also allows for comparisions to be made easier because you know that

math.pi will generate a number to the same precision everytime. The scope of the math.pi constant is global as the library was imported outside the class and therefore, its value is accessible throughout the program.

C Code for CircleADT.py

```
## @file CircleADT.py
#  @title CircleADT
#  @author Meet Patel
#  @date 1/28/2017

import math

## @brief This class represents a circle
#  @details This class includes functions that can be performed on the circle
#  object. It has values x, y, and r where x and y represent the coordinates of
#  the centre point of the circle and r represents the radius of the circle.
class CircleT:

## @brief Constructor for CircleADT
#  @details Constuctor accepts three parameters for x-coordinate, y-coordinate, and radius.
#  @param x is the x-coordinate of the centre point of the circle
#  @param y is the y-coordinate of the centre point of the circle
#  @param r is the radius of the circle
    def __init__(self, x, y, r):
        self.x = x
        self.y = y
        self.r = r

## @brief Returns x-coordinate of the centre point of the circle
#  @return x-coordinate of the centre point of the circle
    def xcoord(self):
        return self.x

## @brief Returns y-coordinate of the centre point of the circle
#  @return y-coordinate of the centre point of the circle
    def ycoord(self):
        return self.y

## @brief Returns the radius of the circle
#  @return radius of the circle
    def radius(self):
        return self.r

## @brief This function calculates the area of the circle
#  @return Area of the circle
    def area(self):
        return math.pi * self.r**2

## @brief This function calculates the circumference of the circle
#  @return Circumference of the circle
    def circumference(self):
        return 2*math.pi*self.r


## @brief This function checks if the circle is inside a box
#  @details Using inputs 'xo', 'yo', 'w', and 'h', a box is created
#  @details Assumed that a circle's edges touching the edges of the box are considered inside the box
#  @param xo is the x coordinate of the left side of the box
#  @param yo is the y coordinate of the top of the box
#  @param w is the width of the box
#  @param h is the height of the box
#  @return Boolean value of true or false of whether the circle is inside the box
    def insideBox(self, xo, yo, w, h):
        if (self.r + self.x <= xo + w and self.x - self.r >= xo):
            if (self.y + self.r <= yo + h and self.y - self.r >= yo):
                return True
        return False

## @brief This function checks if two circles are intersecting
#  @details Two circles are intersecting if they share at least one point.
#  @details Another circle object is used to compare the common point(s) with
#  @param circle2 is the other circle object which the first circle object is comparing with
#  @return Boolean value of true or false of whether the two circles intersect
    def intersect(self, circle2):
        if (math.sqrt((self.x - circle2.x)**2 + (self.y - circle2.y)**2) < self.r + circle2.r):
            return True
        return False

## @brief This function scales the radius of a circle
#  @param k is the scaling factor of the radius
    def scale(self, k):
```

```
        self.r = k * self.r

## @brief This function translates the centre point (x and y coordinates) of a circle object
#  @param dx is the translation value of the centre point of the circle in the x direction
#  @param dy is the translation value of the centre point of the circle in the y direction
    def translate(self, dx, dy):
        self.x = self.x + dx
        self.y = self.y + dy
```

This code can also be found at the following link: Link

D Code for Statistics.py

```
## @file Statistics.py
#  @title Statistics
#  @author Meet Patel
#  @date 1/28/2017

import numpy
import CircleADT

## @brief Calculates the average radius from a given list of circle objects
#  @details Uses numpy library to calculate the average.
#  @param List containing circles created by CircleT class.
#  @return Average radius of the list of circles.
def average(circles):
    radiiList = [ i.radius() for i in circles]
    return numpy.average(radiiList)

## @brief Calculates the standard deviation of the radii from a given list of circle objects
#  @details Uses numpy library to calculate the standard deviation.
#  @param List containing circles created by CircleT class.
#  @return Standard deviation of the radii of the list of circle objects.
def stdDev(circles):
    radiiList = [i.radius() for i in circles]
    return numpy.std(radiiList)

## @brief Creates a list of the ranking of the radii from a given list of circle objects in descending
    order.
#  @details Assumed that the ranking when two or more elements are the same
#  is produced by that rank appearing the same number of times and the next number being skipped
#  ie; rank of radii [9,4,6,2,4] would be [1,3,2,5,3] (notice there is nothing ranked as 4)
#  @param List containing circles created by CircleT class.
#  @return List ranked by descending order of radii of list of circle objects
def rank(circles):

    radiiList = []

    for i in circles:
        radiiList.append(i.radius())

    sortedList = sorted(radiiList, reverse=True)
    rankedList = []

    for i in radiiList:
        rankedList.append(sortedList.index(i)+1)

    return rankedList
```

This code can also be found at the following link: Link

E Code for testCircles.py

```
## @file testCircles.py
#  @title TestCircles
#  @author Meet Patel
#  @date 1/28/2017

from CircleADT import *
from Statistics import *
import numpy

C1 = CircleT(3.0, 5.0, 5.0)
C2 = CircleT(13.0, 8.0, 9.0)
C3 = CircleT(7.0, 2.0, 4.0)
C4 = CircleT(-6.0, -8.0, 3.0)
C5 = CircleT(13.0, 8.0, 9.0)
C6 = CircleT(4.0, 5.0, 2.0)
C7 = CircleT(8.0, 4.5, 2.5)
C8 = CircleT(3.5, 2.5, 5.0)

## @brief Tests the x-coordinate getter
def xcoordTest():
    if (C1.xcoord() == 3.0):
        print "The xcoord method works correctly."
    else:
        print "The xcoord method does not work correctly."

## @brief Tests the y-coordinate getter
def ycoordTest():
    if (C2.ycoord() == 8.0):
        print "The ycoord method works correctly."
    else:
        print "The ycoord method does not work correctly."

## @brief Tests the radius value getter
def radiusTest():
    if (C3.radius() == 4.0):
        print "The radius method works correctly."
    else:
        print "The radius method does not work correctly."

## @brief Tests the area function from class CircleADT
def areaTest():
    if (C1.area() == 78.53981633974483):
        print "The area method works correctly."
    else:
        print "The area method does not work correctly."

## @brief Tests the circumference function from class CircleADT
def circumferenceTest():
    if (C1.circumference() == 31.41592653589793):
        print "The circumference method works correctly."
    else:
        print "The circumference method does not work correctly."

## @brief Tests the insideBox function from class CircleADT
#  @details Tests different possible cases of the function which includes:
#  the box inside the circle, the box outside the circle, and the box edges aligning with the circle's
    edges
def insideBoxTest():
    if (C6.insideBox(1,1,14,14)):
        print "The inside box method works when the circle is inside the box."
    else:
        print "The inside box method does not work correctly when the circle is inside the box."

    if (not C6.insideBox(4,6,5,2)):
        print "The inside box method works when the circle is not inside the box."
    else:
        print "The inside box method does not work correctly when the circle is not inside the box."

    if (C6.insideBox(2,3,4,4)):
        print "The inside box method works when the circle's edges are touching the box's edges."
    else:
        print "The inside box method does not work correctly when the circle's edges are touching the
            box's edges."

## @brief Tests the intersect function from class CircleADT
#  @details Tests different possible cases of the function which includes:
```

```
#  the circles on top of each other, one circle inside another, two circles with no common points
def intersectTest():
    if (C2.intersect(C1)):
        print "The intersect method works when one circle is inside another."
    else:
        print "The intersect method does not work when one circle is inside another."

    if (not C2.intersect(C4)):
        print "The intersect method works when the two circles do not have any points in common."
    else:
        print "The intersect method does not work correctly when the two circles do not have any
            points in common."

    if (C2.intersect(C5)):
        print "The intersect method works when two circles are lined up on top of each other."
    else:
        print "The intersect method does not work when two circles are lined up on top of each other."

## @brief Tests the scale function from the class CircleADT
#  @details Uses an appropriate 'k' value to test if the outputted radius is correct
def scaleTest():
    C7.scale(3.0)
    if (C7.radius() == 7.5):
        print "The scale method works correctly."
    else:
        print "The scale method does not work correctly."

## @brief Tests the translate function from the class CircleADT
#  @details Uses an appropriate 'dx' and 'dy' values to test if the
#  outputted x and y coordinates are correctly translated
def translateTest():
    C7.translate(3.0, 5.0)
    if (C7.xcoord() == 11.0 and C7.ycoord() == 9.5):
        print "The translate method works correctly with positive inputs."
    else:
        print "The translate method does not work correctly with positive inputs."

    C8.translate(-4.0, -6.0)
    if (C8.xcoord() == -0.5 and C8.ycoord() == -3.5):
        print "The translate method works correctly with negative inputs."
    else:
        print "The translate method does not work correctly with negative inputs."

## @brief Tests the average function from Statistics
#  @details Takes in a list of circles to verify that
#  outputted average value is correct
def averageTest():
    if (average([C1, C2, C3, C4]) == 5.25):
        print "The average method works correctly."
    else:
        print "The average method does not work correctly."

## @brief Tests the stdDev function from Statistics
#  @details Takes in a list of circles to verify that
#  outputted standard deviation value is correct
def stdDevTest():
    if (stdDev([C1, C2, C3, C4]) == 2.2776083947860748):
        print "The stdDev method works correctly."
    else:
        print "The stdDev method does not work correctly."

## @brief Tests the rank function from Statistics
#  @details Takes in a list of circles to verify that the outputted
#  list is correctly ranked. Tests a list of circles with all
#  unique radii and one with a repeated radius
def rankTest():
    if (rank([C1,C2,C3,C4,C6]) == [2,1,3,4,5]):
        print "The rank method works correctly with 5 different radii in the list."
    else:
        print "The rank method does not work correctly with all different radii in the list."

    if (rank([C1,C2,C3,C4,C5]) == [3,1,4,5,1]):
        print "The rank method works correctly with a repeated radius in the list."
    else:
        print "The rank method does not work correctly with a repeated radius in the list."

xcoordTest()
ycoordTest()
radiusTest()
areaTest()
```

```
circumferenceTest()
insideBoxTest()
intersectTest()
scaleTest()
translateTest()
averageTest()
stdDevTest()
rankTest()
```

This code can also be found at the following link: Link

F Code for Makefile

```
PY = python
PYFLAGS =
DOC = doxygen
DOCFLAGS =
DOCCONFIG = circleDoxygen

SRC= ./src/testCircles.py

.FORCE: test doc clean

test:
	$(PY) $(PYFLAGS) $(SRC)

doc:
	$(DOC) $(DOCFLAGS) $(DOCCONFIG)
	cd latex && make

clean:
	rm -rf html
	rm -rf latex
```

This code can also be found at the following link: Link

G Partner's CircleADT.py Code

```
import math

## @file CircleADT.py
#  @author Shalmi patel
#  @date 1/25/2017

## @brief This class represents a Circle.
#  @details This class represents a Circle as (x,y,r) coordinate representing
#  the middle of the circle and the r repersenting as the radius.

class CircleT:

    ## @brief Constructor for CircleADT
    #  @details Constructor accepts 3 parameters for the coordinates of the circle and the radius.
    #  @param x value of the x coordinate
    #  @param y value of the y coordinate
    #  @param r value of the radius
    def __init__(self, x, y, r):
        self.x = x
        self.y = y
        self.r = r

    ## @brief Returns the x-coord of the circle
    #  @return the x-coord
    def xcoord(self):
        return self.x

    ## @brief Returns the y-coord of the circle
    #  @return the x-coord
    def ycoord(self):
        return self.y

    ## @brief Returns the radius of the circle
    #  @return the radius
    def radius(self):
        return self.r

    ## @brief Returns the area of the circle
    #  @return the area of the circle
    def area(self):
        area = math.pi*(self.r**2)
        return area

    ## @brief Returns the circumference of the circle
    #  @return the circumference of the circle
    def circumference(self):
        circumference = 2*math.pi*self.r
        return circumference

    ## @brief This function checks if the circle is inside the box that is created with this function.
    #  @param x0  the x-coord of the box
    #  @param y0  the y-coord of the box
    #  @param w  the width of the box
    #  @param h  the height of the box
    #  @return boolean true if the circle is inside the box
    def insideBox(self, x0, y0, w, h):
        if (((self.x + self.r) <= (x0 + w)) and ((self.y - self.r) >= y0) and ((self.y + self.r) <=
            (y0 + h)) and ((self.x - self.r) >= x0)):
            return True
        else:
            return False

    ## @brief This function checks if another circle is intersecting with the current circle.
    #  @param c object that is a circle with (x,y,r)
    #  @return boolean true if circle intersects current circle
    def intersect(self, c):
        distance = self.r +c.radius()
        if (abs(c.xcoord() - self.x) <= distance) and (abs(c.ycoord() -self.y) <=distance):
            return True
        else:
            return False

    ## @brief This function scales the radius
    #  @param k the factor to scale by
    def scale(self, k ):
        if(k == 0):
```

```
            self.r = self.r
        else:
            self.r = abs(k)*self.r

    ## @brief This function translate factor for x, y
    #  @param dy translating factor for y
    #  @param dx translating factor for x
    def translate(self, dx, dy):
        self.y = dy + self.y
        self.x = dx + self.x
```

H Partner's Statistics.py Code

```
import numpy
## @file Statistics.py
#  @author Shalmi patel
#  @date 1/25/2017

## @brief This class takes an list of radii and finds the average, stdDev, rank
#  @details This class uses the library numpy that gets the average, and stdDev of an array


## @brief Returns the average of the radii of the circles in the list provided.
#  @param Circle A list of CircleT objects with param of x y r
#  @return the average of the radii
def average(Circle):
    myarray = []

    for i in Circle:
        myarray.append(i.radius())

    return numpy.average(myarray)

## @brief Returns the standard deviation of the radii of the circles in the list provided.
#  @param Circle A list of CircleT objects with param of x y r
#  @return standard deviation of the radii
def stdDev(Circle):
    myarray= []

    for i in Circle:
        myarray.append(i.radius())

    return numpy.std(myarray)

## @brief Returns the rank of the radii from largest to smallest.
#  @param Circle A list of CircleT objects with param of x y r
#  @return the ranked array
def rank(Circle):
    radii= []
    finalrank= []

    for i in Circle:
        radii.append(i.radius())

    array = sorted(radii, reverse = True)

    for i in radii:
        arrayindex = array.index(i) + 1
        finalrank.append(arrayindex)

    return finalrank
```