CS 2ME3 Assignment 1 Report

Name: Polly yao MacId: yaos5

February 2, 2017

A Code for CircleADT.py

```
## @file CircleADT.py
#  @title CircleADT
#  @author Polly Yao
#  @date 1/28/2017

## @brief This class represents a circle
#  @details This class represents a circle as x and y values define the centre of the circle
#   and r defines the radius of the circle

import math


## @brief Constructor for CircleT
#  @param x and y as the coordinate for the centre of the circle
#  @param r as the radius of the circle
class CircleT:
    def __init__(self, x, y, r):
        self.x = x
        self.y = y
        self.r = r


## @brief This function receives the x coordinate from the constructor
#  @return x coordinate of the circle
    def xcoord(self):
        return (self.x)


## @brief This function receives the y coordinate from the constructor
#  @return y coordinate of the circle
    def ycoord(self):
        return (self.y)


## @brief This function receives the radius from the constructor
#  @return radius of the circle
    def radius(self):
        return (self.r)


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


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

```
## @brief This function checks if a circle is inside a box.
#  @details This function takes in the (bx, by, w, h) coordinate of a rectangle box
#  @param bx is the x coordinate of the left side of a box, by is the y coordinate of the top of the
    box
#  @param w is the width of the box and h is the height of the box
#  @return True if the circle is inside the box, False if it is not
    def insideBox(self, bx, by, w, h):

        rightx = bx + w
        topy = by + h
        leftc = self.x - self.r
        rightc = self.x + self.r
        bottomc = self.y - self.r
        topc = self.y + self.r

        if (leftc > bx and rightc < rightx) and (bottomc > by and topc < topy):
            return True

        else:
            return False



## @brief This function checks if two circles intersect
#  @details This function takes in another circle c
#  @details The two  circles intersect if the sum of two radius is greater and equal to the sum of
    differences between x and y coordinate of two circles
#  @param c is the second instance of CircleT
#  @return True if two circles intersect, False if they do not
    def intersect(self, c):
        distance = math.sqrt((c.y - self.y)**2 +(c.x - self.x)**2)
        radii = c.r + self.r

        if (radii >= distance):
            return True

        else:
            return False


## @brief This function change the radius by a factor k
#  @param k is the a float that stretch or shorten the radius
    def scale(self, k):
        assert type(k) == float
        self.r = self.r * k


## @brief This function move the circle up around
#  @param dx moves the x coordinate up or down
#  @param dy moves the y coordinate left or right
    def translate(self, dx, dy):
        self.x += dx
        self.y += dy
```

B Code for Statistics.py

```
## @file Statistics.py
#  @title Statistics
#  @author Polly Yao
#  @date 1/28/2017

## @brief This module has three methods(average, stdDev, rank) for CircleADT.py
#  @details This module takes in a list of instances of circles

import CircleADT
import numpy as np


## @brief This method calculate the average of radius of a list circle instances
#  @param thelist is a list of instance of circles
#  @return the average of all the radius
def average(thelist):
    ans = []

    for element in thelist:
        ans.append(element[2])

    avg = np.average(ans)
    return (avg)


## @brief This method calculate the standard deviation of radius of a list circle instances
#  @param thelist is a list of instance of circles
#  @return the standard deviation of all the radius
def stdDev(thelist):
    ans = []
    for element in thelist:
        i = ans.append(element[2])

    stdD = np.std(ans)
    return (stdD)


## @brief This method rank a list of instances CircleT by radius
#  @param thelist is a list of instance of circles
#  @assumption if there are element duplicates in the list, the function will remove all the
    duplicates and leave only one copy of that element
#  @return the list ranked by radius
def rank(thelist):
    array = []
    arrai = []
    arraa = []
    for element in thelist:
        array.append(element[2])
        arrai.append(element[2])


    arrayR = []
    sawA = set()
    for element in array:
        if element not in sawA:
            arrayR.append(element)
            sawA.add(element)


    arraiR = []
    sawB = set()
    for element in arrai:
        if element not in sawB:
            arraiR.append(element)
            sawB.add(element)


    for i in range(len(arrayR)):
        for j in range(i+1, len(arrayR)):
            if arrayR[j] > arrayR[i]:
                arrayR[j], arrayR[i] = arrayR[i], arrayR[j]


    for element in arrayR:
        if (element in arraiR):
            c = arraiR.index(element)
```

```
        arraa.append(c+1)

return arraa
```

C Code for testCircles.py

```
## @file testCircles.py
#  @title test
#  @author Polly Yao
#  @date 1/28/2017

## @brief This module includes all the test cases for CircleADT.py and Statistics.py


import CircleADT
import Statistics


## @brief create the Circle a
a = CircleADT.CircleT(3, 4, 1)


## @brief output the x coordinate of the circle
print(a.xcoord())


## @brief output the y coordinate of the circle
print(a.ycoord())


## @brief output the radius of the circle
print(a.radius())


## @brief output the area of the circle
print(a.area())


## @brief output the circumference of the circle
print(a.circumference())


## @brief check if the circle is inside the box(1, 1, 5, 6)
#  @detail output True
print(a.insideBox(1, 1, 5, 6))


## @brief check if the circle is inside the box(2, 1, 5, 6)
#  @detail output False
print(a.insideBox(2, 1, 5, 6))


## @brief create the Circle b
b = CircleADT.CircleT(6, 3, 3)


## @brief Test if Circle a and Circle b intersect
## @detail output True
print(a.intersect(b))


## @brief create the Circle c
c = CircleADT.CircleT(6, 2, 1)


## @brief Test if Circle a and Circle c intersect
## @detail output False
print(a.intersect(c))


## @brief create the Circle d
d = CircleADT.CircleT(5, 3, 1.24)


## @brief Test if circle a and cirlce d intersect
#  @detail Circle d intersect with circle at one point(x == 4)
#  @detail output True
print(a.intersect(d))


## brief scale radius by a factor 4
a.scale(4.0)
```

```
## brief output the radius after scale
print(a.radius())


## brief move x and y coordinate of the circle by (8, 6)
a.translate(8, 6)


## brief output the x and y coordinate after translate
print(a.xcoord(), a.ycoord())


## @brief create the Circle e
e = CircleADT.CircleT(2, 3, 2)


## @brief create the Circle f
f = CircleADT.CircleT(4, 4, 1)


## @brief create the Circle g
g = CircleADT.CircleT(6, 5, 4)


## @brief create an array with circle e, f, g
array = [[e.x,e.y,e.r], [f.x,f.y,f.r], [g.x, g.y, g.r]]


## brief output the average radius
print(Statistics.average(array))


## brief output the standard deviation
print(Statistics.stdDev(array))


## brief output a ranked list of radius
print(Statistics.rank(array))


## @brief create the Circle h
h = CircleADT.CircleT(3, 5, 2)

## @brief create an array with circle e, f, g, h, contain duplicates(e.r, h.r)
arr = [[e.x,e.y,e.r], [f.x,f.y,f.r], [g.x, g.y, g.r], [h.x, h.y, h.r]]


## @brief output a ranked list of radius with duplicates(two 2)
#  @detail The rank function will remove one of the 2
print(Statistics.rank(arr))
```

D Makefile

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

SRC = src/testCircles.py

.PHONY: all prog doc clean

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

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

all: test doc

clean:
	rm -rf html
	rm -rf latex
	rm -rf CircleADT.pyc
	rm -rf Statistics.pyc
```

E Partner's CircleADT Code

```
## @file CircleADT.py
#  @title CircleADT
#  @author Alan Yin
#  @date 1/26/2017

from math import *


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

class CircleT(object):

    ## @brief Constructor for Circle
    #  @details Constructor accepts two parameters for the x-coordinate of center of circle,
    #  y-coordinate of center of circle and radius.
    #  @param x for x-coordinate of center of circle.
    #  @param y for y-coordinate of center of circle.
    #  @param r for radius of circle.

    def __init__(self, x, y, r):
        self.x = x
        self.y = y
        self.r = r

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

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

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

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

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

    ## @brief This function determines if the circle is inside a given box.
    #  @param x0 x-coordinate of top left corner of the box.
    #  @param y0 y-coordinate of top left corner of the box.
    #  @param w width of the box.
    #  @param h height of the box.
    #  @return boolean that indicates if the circle is inside the box.
    def insideBox(self, x0, y0, w, h):
        result = True
        if self.x-self.r > x0 or self.x+self.r < w or self.y-self.r > y0 or self.y+self.r < h:
            result = True
        else:
            result = False
        return result

    ## @brief This function determines if two circles intersect.
    #  @param others Another CircleT object.
    #  @return boolean that indicates if the two circles intersect.
    def intersect(self, others):
        result = True
        if (self.x-others.x)**2 + (self.y-others.y)**2 >= (self.r+others.r)**2 or (self.x-others.x)**2
            + (self.y-others.y)**2 <= (self.r-others.r)**2:
            result = False
        else:
            result = True
        return result
```

```
## @brief Scale the radius of circle by a given number.
#  @param k the number that is used to scale the circle.
def scale(self, k):
    self.r = self.r * k

## @brief Translate the center of circle by dx in x direction and dy in y direction.
#  @param dx The amount to translate in x direction.
#  @param dy The amount to translate
def translate(self, dx, dy):
    self.x = self.x + dx
    self.y = self.y + dy
```

F Partner's Statistics Code

```
## @file Statistics.py
#  @title Statistics
#  @author Alan Yin
#  @date 1/26/2017



from CircleADT import CircleT
import numpy as np


## @brief This function takes a list of instances of CircleT and calculates the average radius.
#  @return the average radius of all of the circles
def average(a,b,c,d):
    return np.average([a.r,b.r,c.r,d.r])

## @brief This function takes a list of instances of CircleT and calculates the
#  standard deviation of radii.
#  @return the standard deviation of the radii of all of the circles
def stdDev(a,b,c,d):
    return np.std([a.r,b.r,c.r,d.r])


## @brief This function takes a list of instances of CircleT and rank them by radius.
#  @return a listed ranked by radius.
def rank(a,b,c,d):
    mylist = [a.r,b.r,c.r,d.r]
    print(mylist)
    mylist1 = sorted(mylist,reverse = True)
    print(mylist1)
    a1 = mylist1.index(a.r) + 1
    b1 = mylist1.index(b.r) + 1
    c1 = mylist1.index(c.r) + 1
    d1 = mylist1.index(d.r) + 1
    return [a1,b1,c1,d1]
```

Testing results of my files and my partner's files

The results of testing my CircleADT file and Statistics file came out successfully. Each method is tested and passed by at least one test case. The rank method is specifically tested with a test case which include duplicates, and this test case has also successfully passed through. After finished testing my partner's file, several problemes arise. First, the result came out different on the insideBox method between my partner and I. Second, my test cases does not work on my partner's Statistics module, every method in his Statistis module assume to take in four instances of CircleT seperately which means his methods need to take in four arguments, while all my methods in Statistics file only takes in one argument which is a list of instances of CircleT. Other than all the problems addressed previously, all other methods in his CircleADT.py works out perfectly, they have passes through all my test cases and the result were exactly the same as mine.

Discussion on the test results

As stated previously, my result for insideBox method is different from the result of my partner's. His insideBox implemetation assumes a circle is inside a box if the circle's border is less than or equal to the border of the box, and my insideBox method assumes a circle is inside a box if the circle borader is only less than the border of the box. I consider both of our implementation for this method is appropriate, however we should all state our assumption in the doxygen comments. Second, the conflicts occurs in the input arguments of Statistics module. My partner's module need to take in four instances of CircleT seperately and mine takes in a list of instances of CircleT, however his avaerage and standard deviation function works out perfectly after I changed my test cases to four seperate instance of CircleT. The result of my partner's rank function is still incorrect even after i changed the input into four seperate instance of CircleT; as I looked at his code in details, for every element in the given input list, the function finds its corresponding index in the sorted list. However my implementation is exactly the opposite, for every element in the sorted list, my function finds its corresponding index in the given input list, and this is also what the instruction ask us to do. Finally, my partner did not take care of the situation where there is a tie in the list. From doing this exercise, I learned that documentation is very important for programs, especially when we have an assumption, we should always state it in the doxygen comments.

Discussion on how to handle the value of pi

I have used the pi value from the standard math library of python. I have made this chocie because i think the pi value from the library will be more accurate instead of rounding the value myself. If we want to reuse this function in another procedure or class, the accurate result from previous step will greatly reduce the precision error in the final result. Using libraries(like numpy) has for sure improve the reliablilty since they are standardized, and their result will be very accurate,therefore it will be very reliable for other class or function to use it. The libraries could also make the program very productive since the algorithm implemented by ourselves might not be as efficiencent and quick as the one in the library. From this exersice, I have observed the differences of specifications or documentation would acutally cause differences in implementation. For examle, the specification in my Statistics.py is different from what my partner has in his Statistics.py since he takes in a different set of parameters.