Assignment 1 report

Amit Binu

binua

A Code for CircleADT.py

```
## @file CircleADT.py

## @title CircleADT
# @author Amit Binu
# @date 28/1/2017



from math import *
## @brief Creates a circle by taking its center x-coordinate, y-coordinate and radius.
# @details This class also has 2 methods that will modify the cordinates and radius
# and 4 methods that returns something about the circle.
class CircleT(object):

    ## @brief Constructor for CircleT
    # @details The Constructor for this class accepts 3 parameters for the circle's
    # coordinates and its radius.
    # @param x is for the xcoordinate of Circle's center
    # @param y is for the ycoordinate of Circle's center
    # @param r is for the radius of the Circle
    def __init__(self,x,y,r):
        self.x = x
        self.y = y
        self.r = r

    ## @brief Returns the x-coordinate of the Circle's center
    # @return an integer value of the Circle's x-coordinate
    def xcoord(self):
        return self.x

    ## @brief Returns the y-coordinate of the Circle's center
    # @return an integer value of the Circle's y-coordinate
    def ycoord(self):
        return self.y

    ## @brief Returns the radius of the Circle
    # @return an integer value of the Circle's raidus
    def radius(self):
        return self.r

    ## @brief Returns the area of the Circle.
    # @return an integer value of the Circle's Circle
    def area(self):
        return pi * (self.r)**2

    ## @brief Returns the circumference of the Circle.
    # @return an integer value of the Circle's circumference.
    def circumference(self):
        return 2 * pi * (self.r)

    ## @brief This function checks whether the Circle is inside a box whose coordinates
         are taken as pararmeters.
    # @details Assumes that the coordinates of the box are of the top left corner, no
         matter on what quadrant of the graph the box is.
    # @details It is also assumed that circle is inside the box if all the top, bottom,
         left and right edges of circle touches the box.
    # @param x0 The x-coordinate of the top left front facing box.
    # @param y0 The y-coordinate of the top top left fron facing box.
    # @param w The width of the box
    # @param h The height of the box
    # @return a boolean value True if the Circle is inside the box
    # @return a boolean value False if any part of the Circle is outside the box.
    def insidebox(self, x0, y0, w, h):
        if ((self.x-self.r) >= x0 and (self.x+self.r)<= (x0+w) and (self.y+self.r)<= y0
             and (self.y-self.r) >= y0-h):
            return True
        else:
            return False

    ## @brief This function checks whether 2 Circle intersected or not
    # @param c This is a Circle object which has its own coordinates and radius
    # @return a boolean value True, if the circles intersected
    # @return a boolean value False, if the circles did not intersect at all.
    def intersect(self, c):
        if (self.x - c.xcoord())**2 + (self.y - c.ycoord())**2 <= (self.r +
             c.radius())**2:
```

```
        return True
    else:
        return False

## @brief This function sets a new radius value for the Circle by multiplying it by
    a value that is taken as a parameter.
# @param k This is an integer value that will be multipied by the old radius to set
    the new radius.
def scale(self, k):
    self.r = self.r * k

## @brief This function sets new x-coordinate and y-coordinate by adding the old
    coordinates with the numbers that are taken as parameters.
# @param dx This is an integer value that is used to shift the circle's
    x-coordinate.
# @param dy This is an integer value that is used to shift tht circle's ycoordinate.
def translate(self, dx, dy):
    self.x = self.x + dx
    self.y = self.y + dy
```

B Code for Statistics.py

```
## @file Statistics.py

## @title CircleADT
# @author Amit Binu
# @date 28/1/2017
# @brief This program has functions that doesaverage, stadarad deviation on the radii
      of the circle objects in a list.
# @details This class has 3 functions in total. -average, stdDEV, rank

import CircleADT
import numpy as np

#instance_1 = CircleADT.CircleT(11, 12, 2)
#instance_2 = CircleADT.CircleT(10,10,9)
#instance_3 = CircleADT.CircleT(5,5,10)
#instance_4 = CircleADT.CircleT(2,3,9)
#instance_5 = CircleADT.CircleT(8,54,9)
#instance_6 = CircleADT.CircleT(89,20,9)

#list = [instance_1, instance_2, instance_3, instance_4, instance_5, instance_6]

##This is an empty list which will contain only radii when the appropriate function is
      called.
list1 = []

## @brief This functon takes a list of CircleADT objects and uses their radii to
      calculate an average of those radii.
# @param list is a list that contains CircleADT objects
# @return a floating number that represents the average of all radii which is
      calculated using numpy.
def average(list):


    for i in range (len(list)):
        list1.append(list[i].r)
    if (list1 == []):
        return "ERROR: The list should contain at least one item"

    return np.average(list1)

## @brief This function takes a list of CircleADT objects and uses their radii to
      calculate an average of those radii.
# @param list is a list that contains CircleADT objects
# @return a floating number that represents all radii's standard deviation, which is
      calculated using numpy.
def stdDev(list):

    for i in range(len(list)):
        list1.append(list[i].r)
    if (list1 == []):
        return "ERROR: The list should contain at least one item"
    return np.std(list1)


## @brief This function takes a list of CircleADT objects and uses their radii to
      produce another list with their ranks in descending order.
# @param list is a list that contains CircleADT objects
# @return a list with the ranking of each radii in descending order.
def rank(list):
    ## This list will contain radii after the rank function is implemented
    list1 =[]
    ## This list will contain radii in descending order after the rank list is
          implemented
    list2 = []
    ## This list will contain the ranking of radii in descending order after the rank
          function is implemented
    list3 = []
    for i in range(len(list)):
        list1.append(list[i].r)
        list2.append(list[i].r)

    for i in range(len(list1)):
        for j in range (i+1, len(list1)):
            if ( i != len(list1) - 1):
                if (list1[i] < list1[j]):
                    x = list1[i]
                    list1[i] = list1[j]
                    list1[j] = x
    # list1 now contains radii in descending order
    print list1
    print list2
    for i in range (len(list2)):
        list3.append(i)

    counter = 0

    for i in range(len(list1)):
        if (i < len(list1)-1):
            if (list1[i] == list1[i+1]):
                counter = counter + 1
                continue

        for j in range (len(list2)):
            if (list1[i] == list2[j]):
                if (counter == 0):
                    print i, j
                    list3[j] = i+1
                if (counter == 1):
                    print i, j
                    list3[j] = i
                if (counter > 1):
                    print i,j
                    print counter
                    list3[j] = i - counter + 1



    if (list1 == []):
        return "ERROR: The list should contain at least one item"

    return list3

#Testing

#print average(list)
#print stdDev(list)
#print rank(list)
```

C Code for testCircles.py

```
## @file testCircles.py


## @title Test Cases for CircleADT.py
# @author Amit Binu
# @date 28/1/2017



## @brief Case 1 for xcoord method in CircleADT:
# @details input: CircleADT.CircleT(5,4,3).xcoord()
# @details expected output: 5
# @details output: 5
# @details Test case Passed

## @brief Case 2 for ycoord method in CircleADT:
# @details input: print CircleADT.CircleT(5,4.0,3).ycoord()
# @details expected Output: 4.0
# @details Actual Output: 4
# @details Test Case Passed



## @brief Case 3 for radius method in CircleADT:
# @details input: print CircleADT.CircleT(5,4.0,3).radius()
# @details expected output: 3
# @details actual output: 3
# @details Test Case passed



## @brief Case 4 for area method in CircleADT
# @details input: print CircleADT.CircleT(5,4.0,3).area()
# @details expected Output: 28.2743338823
# @details Actual Ouput: 28.2743338823
# @details Test Case passed



## @brief Case 5 for circumference method in CircleADT
# @details input: print CircleADT.CircleT(5,4.0,3.0).area()
# @details expected Output: 18.8495559215
# @details actual output: 18.8495559215
# @details Test Case passed



## @brief Case 6 for insidebox method in CircleADT
# @details 1)
# @details input: CircleADT.CircleT(55, 44, 33).insidebox(20,78, 70, 70)
# @details Expected Output: True
# @details Actual Output: True
# @details Test Case Passed



## @brief 2nd Case for inside box
# @details 2)
# @details input: CircleADT.CircleT(80,20,0).insidebox(10,20,10,10)
# @details Expected Output: False
# @details Actual Output: False
# @details Test Case Passed



## @brief 3rd case for insidebox
# @details 3)
# @details input: CircleADT.CircleT(5, -5, 1).insidebox(0,0,20,20)
# @details Expected Output: True
# @details Actual Output: True
# @details Test Case passed


## @brief Case 7 for intersect method in CircleADT
# @details 1)
# @details input: CircleADT.CircleT(0,0,5).intersect(3,0,5)
# @details Expected Output: True
# @details Actual Output: True
# @details Test Case Passed

## @brief
# @details 2)
# @details input: CircleADT.CircleT(0,0,5).intersect(0,0,100)
# @details Expected Output: True
# @details Actual Output: True
# @details Test Case Passed

## @brief
# @details 3)
# @details input: CircleADT.CircleT(0,0,5).intersect(0,100,5)
# @details Expected Output: False
# @details Actual Output: False
# @details Test Case Passed


## @brief Case 8 for scale method in CircleADT
# @details input: h = CircleADT.CircleT(0,0,5); h.scale(8); h.radius()
# @details Expected Output: 40
# @details Actual Output: 40
# @details Test Case Passed


## @brief Case 9 for translate method in CircleADT
# @details input: h = CircleADT.CircleT(0,0,40); print h.xcoord(), h.ycoord()
# @details Expected Output: 5 , 6
# @details Actual Output: 5 , 6
# @details Test case Passed

## @brief Case 10 for average method in Statistics method
# @details input: Statistics.average([CircleADT.CircleT(11, 12,
    6),CircleADT.CircleT(10,10,5),CircleADT.CircleT(5,5,11),CircleADT.CircleT(2,3,9)])
# @details Expected Output: 7.75
# @details Actual Output: 7.75
# @details Test case passed



## @brief Case 11 for rank method in Statistics method
# @details input: Statistics.stdDev([CircleADT.CircleT(11, 12,
    6),CircleADT.CircleT(10,10,5),CircleADT.CircleT(5,5,11),CircleADT.CircleT(2,3,9)])
# @details Expected Output: 2.38484800354
# @details Actual Output: 2.38484800354
# @details Test case Passed



## @brief Case 12 for rank method in Statistics method
# @details input: Statistics.rank([CircleADT.CircleT(11, 12,
    6),CircleADT.CircleT(10,10,5),CircleADT.CircleT(5,5,11),CircleADT.CircleT(2,3,9)])
# @details Expected Output: [3, 4, 1, 2]
# @details Actual Output: [3, 4, 1, 2]
# @details Test Case Passed
```

D Makefile

```
PY = python
PYFLAGS =
DOC = doxygen
DOCFLAGS =
DOCCONFIG = ./test.py

SRC = ./src/testCircles.py

.PHONY: all prog doc clean

prog:
	$(PY) $(PYFLAGS) $(DOCCONFIG)
	cd latex && $(MAKE)

all: prog doc

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

E Partner's Code for CircleADT.py

```
## @file testCircles.py
#  @title testCircles
#  @author Kael Bosland
#  @date 1/26/2017

## @brief this class creates a CircleT object and has functions associated with items
class CircleT:

        ## @brief constructor that takes x and y coordinates and a radius variable
    def __init__(self, x, y, r):
        self.x = x
        self.y = y
        self.r = r

        ## @brief getter that returns the x coordinate of the CircleT
        #  @return returns the x coordinate of the CircleT
    def xcoord(self):
        return self.x

        ## @brief getter that returns the y coordinate of the CircleT
        #  @return returns the y coordinate of the CircleT
    def ycoord(self):
        return self.y

        ## @brief getter that returns the radius of the CircleT
        #  @return returns the radius of the CircleT
    def rcoord(self):
        return self.r


        ## @brief method to calculate the area of the CircleT
        #  @details imports the math function and uses the formula 2*pi*r^2 to
            calculate the area
        #  @param parameter 1 is a CircleT object
        #  @return returns the calculated area rounded to 2 decimal points
    @staticmethod
    def area(c):
        import math
        return round(c.rcoord()*c.rcoord()*math.pi, 2)

        ## @brief method that checks if a circle is inside a box
        #  @details checks to see if the 4 points of the circle are inside the boxes
            (left, right, top and bottom) coordinates
        #  @param parameter 1 is a CircleT object
        #  @param parameter 2 is the top-left x coordinate of the box
        #  @param parameter 3 is the top of the box (y coordinate)
        #  @param parameter 4 is the width of the box
        #  @param parameter 1 is the height of the box
        #  @return returns if the circle is inside the box or not (True or False)
    @staticmethod
    def insideBox(c, x0, y0, w, h):

                x = c.xcoord()
                y = c.ycoord()
                r = c.rcoord()

                if (y + r <= y0):
                        if (y - r >= y0 - h):
                                if (x + r <= x0 + w):
                                        if (x - r >= x0):
                                                return True

                return False

        ## @brief method to check if two circles intersect
        #  @details calculates the distance between the centre of the two circles and
            if the distance > than the radiuses, they will not intersect
        #  @param parameter 1 is the first CircleT object
        #  @param parameter 2 is the second CircleT object
        #  @return True if they intersect, False if they don't
    @staticmethod
    def intersect(c,c1):
        import math
        dy = math.pow((c1.ycoord() - c.ycoord()),2)
        dx = math.pow((c1.xcoord() - c.xcoord()),2)
        distance = math.sqrt(dy+dx)
        radiuses = c.rcoord() + c1.rcoord()

        if (distance < radiuses):
            return True
        else:
            return False

        ## @brief calculates the circumference of the CircleT
        #  @details imports math to use the pi function, use the formula (2*pi*r) to
            calculate the circumference
        #  @return returns the calculated circumference rounded to 2 decimal places
    @staticmethod
    def circumference (c):
            import math
            return round (2*math.pi*c.r, 2)

        ## @brief multiplies the radius of the given CircleT by keys
        #  @param parameter 1 is a CircleT object we want to scale
        #  @param parameter 2 is the scalar (k)
    @staticmethod
    def scale (c, k):
            c.r = c.r *k

        ## @brief translates a CircleT object by dx and dy
        #  @param parameter 1 is the CircleT object
        #  @param parameter 2 is dy (distance to move the y coordinate)
        #  @param parameter 3 is dx (distance to move the x coordinate)
    @staticmethod
    def translate (c, dy, dx):
            c.x = c.x + dx
            c.y = c.y + dy
```

F Partner's Code for Statistics.py

```
## @file Statistics.py
#  @title Statistics
#  @author Kael Bosland
#  @date 1/26/2017

from CircleADT import CircleT
import numpy as np

## @brief this class calculates the average, standard deviation and rank of an array of
     CircleT objects
class Statistics:

        ## @brief this function returns the average radius of a list of CircleT objects
        #  @param this function takes a list of CircleT objects as parameters
        #  @return this function returns the mean radius of the list, rounded to 2
             decimal points
        @staticmethod
        def average (circles):
                radi = np.array([])
                for x in range (0, len(circles)):
                        radi = np.append(radi, np.array([circles[x].rcoord()]))

                return round (np.mean(radi), 2)

        ## @brief this function returns the standard deviation of a list of CircleT
             objects (with respect to their radiuses)
        #  @param this function takes a list of CircleT objects as parameters
        #  @return this function returns the standard deviation of the list, rounded to
             2 decimal points
        @staticmethod
        def stdDev (circles):
                radi = np.array([])
                for x in range (0, len(circles)):
                        radi = np.append(radi, np.array([circles[x].rcoord()]))

                return round (np.std(radi), 2)

        ## @brief this function returns the a list with the rankings of the radiuses
             given in the CircleT array (param)
        #  @details using loops to calculate and sort the list, the function then
             returns a list with the corresponding indexes of the sorted radiuses
        #  @param this function takes a list of CircleT objects as parameters
        #  @return this function returns a list of the indexes where the largest
             radiuses occur (in descending order)
        @staticmethod
        def rank (circles):
                rank = [0]
                radiuses = []
                indexOfHi = 0
                for y in range (0, len(circles)):
                        radiuses = radiuses + [circles[y].rcoord()]

                hi = 0
                radiusCopy = radiuses[:]

                while (len(radiuses) > 0):
                        hi = 0
                        indexOfhi = 0
                        for x in range (0, len(radiuses)):
                                if (radiuses[x] > hi):
                                        hi = radiuses[x]
                                        indexOfHi = x
                                        rank[0] = radiuses[x]
                        rank = [0] + rank
                        radiuses.remove(radiuses[indexOfHi])

                rank = rank[1:]
                rank.reverse()
                positions = []
                for z in range (0, len(rank)):
                        positions = positions + [radiusCopy.index(rank[z]) + 1]

                return positions
```

G Results of testing my files

```
CircleADT.py

method:  xccord()
input:  CircleADT.CircleT(5,4,3).xcoord()
output:  5

method:  ycoord()
input:  print CircleADT.CircleT(5,4.0,3).ycoord()
output:  4

method:  radius()
input:  print CircleADT.CircleT(5,4.0,3).radius()
output:  3

method:  area()
input:  print CircleADT.CircleT(5,4.0,3).area()
Output:  28.2743338823

method:  radius()
input:  print CircleADT.CircleT(5,4.0,3.0).circumference()
output:  18.8495559215

method:  insidebox
input:  CircleADT.CircleT(80,20,0).insidebox(10,20,10,10)
Output:  False

method:  intersect
input:  CircleADT.CircleT(0,0,5).intersect(3,0,5)
Output:  True

method:  scale
input:  h = CircleADT.CircleT(0,0,5); h.scale(8); h.radius()
Output:  40

method:  translate
input:  h = CircleADT.CircleT(0,0,40) print h.xcoord(), h.ycoord()
Output:  5 6
```

```
   Statistics.py

method:  average
input:  Statistics.average([CircleADT.CircleT(11, 12, 6),CircleADT.CircleT(10,10,5),
CircleADT.CircleT(5,5,11),CircleADT.CircleT(2,3,9)])
Output:  7.75

method:  StdDev
input:  Statistics.stdDev([CircleADT.CircleT(11, 12, 6),CircleADT.CircleT(10,10,5),
CircleADT.CircleT(5,5,11), CircleADT.CircleT(2,3,9)])
output:  2.38484800354

method:  rank
input:  Statistics.rank([CircleADT.CircleT(11, 12, 6),CircleADT.CircleT(10,10,5),
CircleADT.CircleT(5,5,11), CircleADT.CircleT(2,3,9)])
Output:  [ 3, 4, 1, 2]
```

H Results of testing my files with the partner's files

```
my CircleADT.py and his Statistics.py
CircleADT.py will have the same results as above since it is not
changed

Statistics.py

method:  average
input:  Statistics.average([CircleADT.CircleT(11, 12, 6),CircleADT.CircleT(10,10,5),
CircleADT.CircleT(5,5,11), CircleADT.CircleT(2,3,9)])
Output:  error

method:  StdDev
input:  Statistics.stdDev([CircleADT.CircleT(11, 12, 6),CircleADT.CircleT(10,10,5)
,CircleADT.CircleT(5,5,11), CircleADT.CircleT(2,3,9)])
output:  error

method:  rank
input:  Statistics.rank([CircleADT.CircleT(11, 12, 6),CircleADT.CircleT(10,10,5)
,CircleADT.CircleT(5,5,11), CircleADT.CircleT(2,3,9)])
Output:  error
```

```
   my Statistics.py and his CircleADT.py

CircleADT.py

method:  xccord()
input:  CircleADT.CircleT(5,4,3).xcoord()
output:  5

method:  ycoord()
input:  print CircleADT.CircleT(5,4.0,3).ycoord()
output:  4

method:  radius()
input:  print CircleADT.CircleT(5,4.0,3).radius()
output:  error

method:  area()
input:  print CircleADT.CircleT(5,4.0,3).area()
Output:  error - his area takes a parameter

method:  radius()
input:  print CircleADT.CircleT(5,4.0,3.0).circumference()
output:  error - his circumference method takes a parameter

method:  insidebox
input:  CircleADT.CircleT(80,20,0).insidebox(10,20,10,10)
Output:  error - my insidebox method was named wrongly.

method:  intersect
input:  CircleADT.CircleT(0,0,5).intersect(3,0,5)
Output:  error - his method takes 2 arguements.

method:  scale
input:  h = CircleADT.CircleT(0,0,5); h.scale(8); h.radius()
Output:  error - his method takes 2 arguements

method:  translate
input:  h = CircleADT.CircleT(0,0,40); h.translate(5,6) print h.xcoord(),
h.ycoord()
```

```
Output:  error - his method takes 3 arguements

Statistics.py

method:  average
input:  Statistics.average([CircleADT.CircleT(11, 12, 6),CircleADT.CircleT(10,10,5),
CircleADT.CircleT(5,5,11), CircleADT.CircleT(2,3,9)])
Output:  7.75

method:  StdDev
input:  Statistics.stdDev([CircleADT.CircleT(11, 12, 6),CircleADT.CircleT(10,10,5)
,CircleADT.CircleT(5,5,11), CircleADT.CircleT(2,3,9)])
output:  2.38484800354

method:  rank
input:  Statistics.rank([CircleADT.CircleT(11, 12, 6),CircleADT.CircleT(10,10,5)
,CircleADT.CircleT(5,5,11), CircleADT.CircleT(2,3,9)])
Output:  [3, 4, 1, 2]
```

I Discussion on Results

- When my CircleADT and my partner’s Statistics was used, all the outputs were errors. This is because in my partner’s Statistics.py, a class called Statistics was made and this class contains all the functions. However, the rquirement was to only make functions. Also, wrong name was used for radius method in CircleADT.
- When my Statistics.py and my partner’s CircleADT.py were used to run to do the testing, some peculiar results were observed. First of all, only some methods in CircleADT ran properly without throwing any errors. However, some did not print the appropriate outputs and those methods were radius, area, circumference, insidebox, intersect, scale, translate.
- Area, Circumference, intersect, scale and translate methods did not run properly because in his CircleADT these methods had an extra paramter for taking CircleADT objects. It was not specified in the requirement to have circle objects as a parameter in any of these methods,so none of these methods in my CircleADT took a CircleADT object as a parameter.
- The insideBox method did not ran properly because it was spelt wrongly in my CircleADT file. I spelt it as insidebox instead of insideBox.
- The radius method did not work becasause in my partner’s CircleADT file, it was named as rcoord(). However in mine it was radius() because it is specified in the requirements to name it radius().
- Wrong name for the radius method caused to give only errors, when my CircleADT.py and my partner’s Statistics.py was used to test the methods in Statistics.py.

J Using Pi

The constant of Pi was used to calculate the area and circumference of the circle. This was achieved by importing the math library. Using this constant will not give a perfect answer, since it will not print all the decimals. This is because the value of Pi from the math library only has 10 decimals. However, the actual value has quadrillian decimals. Therefore, the result obtained from the area and circumference method is not absolutely perfect.