Assignment 1

SFWR ENG 2AA4

Files due Jan 28 (11:59 pm), E-mail partner due Jan 29 (11:59 pm), Lab report due Feb 2 (11:59 pm)

The purpose of this software design exercise is to write a Python program that creates, uses, and tests a simple Abstract Data Type (ADT) that stores data on a circle. The Circle ADT will allow a program to create instances of the datatype `CircleT`. A circle ADT may be of interest in computer graphics or gaming applications. The program will consist of two modules and a test driver program.

All of your code (all files) should be documented using doxygen. All of your reports should be written using LaTeX. Your code should follow the given specification as closely as possible. In particular, you should not add public methods that are not specified and you should not change the number or order of parameters for methods. If you need private methods, please use the Python convention of naming the files with the double underscore (`__methodName__`).

Step 1

Write a first module that creates a circle ADT. It should consist of a Python code file named `CircleADT.py`. The module should define a class CircleT, which contains the following class methods that define the external interface:

- A constructor (`CircleT`) that takes three real numbers x, y and r as input and assigns them to private instance variables. The x and y values define the centre of the circle and r defines its radius.
- Three getters named `xcoord`, `ycoord` and `radius` that return the x and y coordinates of the centre of the circle and the radius of the circle, respectively.
- A method named `area` that returns the area of the circle.
- A method named `circumference` that returns the circumference of the circle.

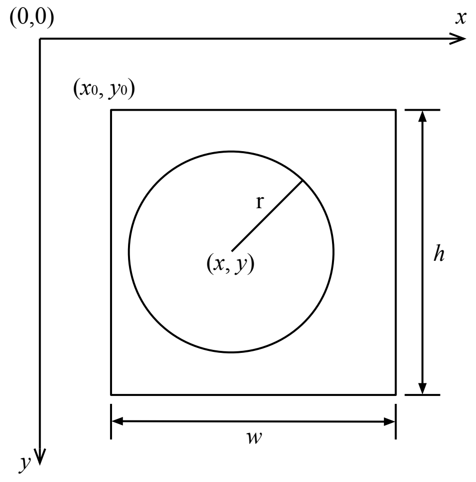

Figure 1: Determination of whether a circle is inside a box or not

- A method named `insideBox` that takes the following inputs: the x coordinate of the left side of a box (x_0), the y coordinate of the top of a box (y_0), the width (w) of the box and the height (h) of the box. The box, the circle and the associated coordinate system are shown in Figure 1. This method should return true if the circle is inside the box and false if it is not.

- A method named `intersect` that takes a second instance of `CircleT` c as input and returns true if the circles intersect and false otherwise. (Two circles intersect if they have any points in common. The interior of the circle is considered to be part of the circle.)

- A method named `scale` that takes a float k as an argument and changes the radius such that it is scaled by k.

- A method named `translate` that take two floats dx and dy as arguments and translates the centre of the circle by dx in the x direction and by dy in the y direction.

Step 2

Write a second module that uses the first module to calculate various statistics for a list of circles. It should consist of the Python file: `Statistics.py`. Some of the routines in this module should be implemented using the numpy, which is located at `http://www.numpy.org/`. The new module should consist of the following functions:

- A function named `average` that takes a list of instances of `CircleT` and returns the average radius of all of the circles in the list. This function should be implemented using `numpy`.
- A function named `stdDev` that takes a list of instances of `CircleT` and returns the standard deviation of the radii of all of the circles in the list. This function should be implemented using `numpy`.
- A function named `rank` that takes a list of instances of `CircleT` and returns a listed ranked by radius. A ranking list provides for each element in the list the position it would hold if the list were sorted in descending order of radius. The maximum entry in the list will have a rank of 1. For instance, the rank of radii [6.0, 5.0, 11.0, 9.0] would be [3, 4, 1, 2]. You are required to implement this function yourself, without using `numpy`. The efficiency of your implementation is not relevant, only the correctness. If you need to make assumptions to implement your algorithm, please state your assumptions as doxygen comments in the code.

Step 3

Write a third module that tests the first and second modules together. It should be a Python file named `testCircles.py`. Write a `Makefile` with a rule `test` that runs your `testCircles` source with the Python interpreter. Each procedure should have at least one test case. Record your rationale for test case selection and the results of using this module to test the procedures in the other two modules. The requirements for testing are deliberately vague; at this time, we are most interested in your ideas and intuition for how to build and execute your test suite.

Step 4

Add to your makefile a rule for `doc`. This rule should compile your documentation into an html and LaTeX version. Ideally, your documentation should be generated to the `A1` folder.

Step 5

Submit the files `CircleADT.py`, `Statistics.py`, `testCircles.py` and `Makefile` using git. Please use the names and locations for these files already given in your git project repo. You should also push your doxygen configuration file to the repo. You will have to add this file to the repo. Ideally, you should place it in the `A1` folder. You should NOT sumbit your generated documentation (html and latex folders). In general, files that can be regenerated are not put under version control.

After the deadline for submitting your solution has passed, please e-mail the `CircleADT.py` and `Statistics.py` files to your assigned partner. (Partner assignments will be posted.) Your partner will likewise e-mail you his or her files. This step must be completed by 11:59 pm of the deadline noted at the beginning of this assignment.

Step 6

After you have received your partner's files, replace your corresponding files with your partner's. Do not make any modifications to any of the code. Run your test module and record the results. Your evaluation for this step does not depend on the quality of your partner's code, but only on your discussion of the testing results.

Step 7

Write a report (using LaTeX) that includes the following:

1. Your name and macid.
2. Your `CircleADT.py`, `Statistics.py`, `testCircles.py` and `Makefile` files.
3. Your partner's `CircleADT.py` and `Statistics.py` files.
4. The results of testing your files.
5. The results of testing your files combined with your partner's files.
6. A discussion of the test results and what you learned doing the exercise. List any problems you found with (a) your program, (b) your partner's module, and (c) the specification of the modules.
7. A discussion of how you handled the value of π in your program and why you made this choice. Is π explicitly expanded in your formulae, or do you use a symbolic constant? If you use a constant, what is its scope?

Your commit (push) to the repository should include the file `report.tex` as given in your initial folder structure. You should also add the file `report.pdf` in the same folder. Although the pdf file is a generated file, we'll make an exception to the general rule of avoiding version control for generated files. The purpose of the exception is for the convenience of the TAs doing the grading.

Including code in your report is made easier by the `listings` package: https://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings.

You may also find it useful to link to code using the hyperref package: https://www.sharelatex.com/learn/Hyperlinks.

The final submission of your report, including your tex file, should be done using git by 11:59 pm on the assigned due date. If you notice problems in your original `*.py` files, you should discuss these problems, and what changes you would make to fix them, in your report. However, the code files submitted on the first deadline will be the ones that are graded.

Notes

1. Your git repo will be organizes with the following directories at the top level: `A1`, `A2`, `A3`, and `A4`.
2. Inside the `A1` folder you will start with initial stubs of the files and folders that you need to use. Please do not change the names or locations of any of these files or folders.
3. Please put your name and macid at the top of each of your source files.
4. Your program must work in the ITB labs on mills when compiled with its versions of Python (version 2), LaTeX, doxygen and make.
5. If your partner fails to provide you with a copy of his or her files by the deadline, please tell the instructor via e-mail as soon as possible.
6. If you do not send your files to your partner by the deadline, you will be assessed a **20% penalty** to your assignment grade.
7. Any changes to the assignment specification will be announced in class. It is your responsibility to be aware of these changes.