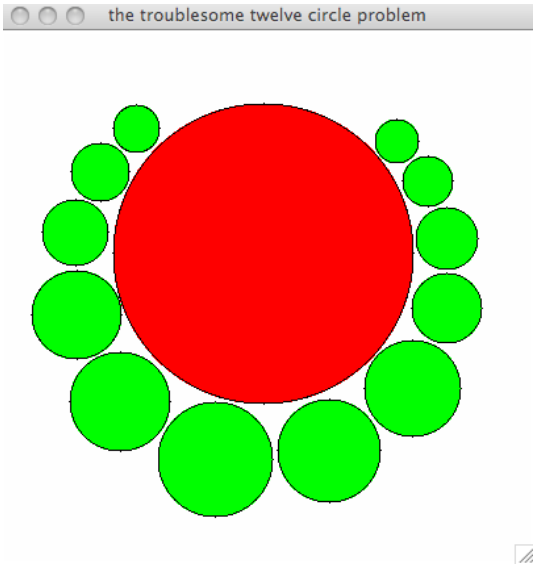
MCS 275 Project Six : solving the twubblesome twelve
due Monday 3 May at 1PM

The goal of project five is to develop a backtracking algorithm in Python to solve the puzzle "the twubblesome twelve." This puzzle is a circle packing problem: the problem is to pack twelve circles of different sizes onto a circular table so all circles fit entirely on the table and none of the circles overlap each other. Solving this problem is deemed hard.

The point of the project is not to provide a GUI to let the user solve the problem, but to have the computer find a solution by trial and error.

A simple script `twubble.py` defines the circles and sets the initial stage for the puzzle. You can download the script `twubble.py` from the class web site. The script `twubble.py` launches a canvas and displays the table and the initial position of the twelve circles around the table. The output of `twubble.py` is shown below:



From `http://demonstrations.wolfram.com/TheTroublesomeTwelveCircleProblem/` the following data (in Python notation) is taken:

```
radii = [ 0.5943, 0.7290, 0.8365, 1.1315, 1.2576, 1.4497,
          1.3067, 1.2151, 0.8788, 0.7941, 0.6383, 0.5495 ]
centers = [ ( -3.2208, -3.1614) , (-4.1287, -2.0767), (-4.7716, -0.5377),
            ( -4.733, 1.5501) , ( -3.6333, 3.7502) , (-1.2111, 5.2049),
            (1.6755, 5.0078), ( 3.7798, 3.4196), (4.6523, 1.3853),
            (4.6523, -0.3790), (4.1574, -1.8336), (3.3871, -2.8653) ]
table = [(0,0), 3.8149]
```

The list `radii` contains the radii of the twelve circles originally positioned at the points with coordinates in the list `centers`. The table has its origin at `(0,0)` and has radius `3.8149`. For visualization, all numbers are multiplied with the same factor.

Some important points:

1. You may work in pairs. Before starting on the project, both team members must send an email to `jan@math.uic.edu` stating the name of their team mate.
   All authors will receive the same score.
2. Developing a GUI to show how the computer moves the circles in its trial and error solution could be very nice and earn some extra credit. However, this GUI will slow down the progress of the backtracking algorithm and should remain optional, i.e.: the user should turn off the graphics.
3. The goal of the project is to provide a correct backtracking algorithm: the computer should enumerate all possible correct configurations so that, given enough time, it finds how to pack all circles onto the table. A configuration is correct if no two circles intersect each other and a circle is either entirely on or entirely off the table.
4. Start the project by writing code to check whether any given configuration of circles given by `(radii,centers,table)` is correct.
5. Provide meaningful documentation strings.
   Every function must have a documentation string. The documentation must describe every variable on input and what the function returns on output.
6. The first line of each of your Python scripts must be

   ```
   # MCS 275 Project Six by <Author(s)>
   ```

   where you replace the `<Author(s)>` by your name(s).
7. Handing in an incomplete but working program is better than handing in a program that crashes or does not run at all.
8. *Handing in late on the same day before 5PM (or even later) is NOT allowed.*
9. Email your solution to the project to `jan@math.uic.edu` before 1PM on Monday 3 May so the date of the email is proof of an on time submission. Bring also a printed version of your solution to the final exam.

If you have questions or difficulties with the project, feel free to come to my office for help.