# Chapter15_Exercises

June 5, 2017

## 0.1 Chapter 15 Exercises

## 0.2 Exercise 15.1

Write a definition for a class named Circle with attributes center and radius, where center is a Point object and radius is a number. Instantiate a Circle object that represents a circle with its center at (150, 100) and radius 75. Write a function named point_in_circle that takes a Circle and a Point and returns True if the Point lies in or on the boundary of the circle. Write a function named rect_in_circle that takes a Circle and a Rectangle and returns True if the Rectangle lies entirely in or on the boundary of the circle. Write a function named rect_circle_overlap that takes a Circle and a Rectangle and returns True if any of the corners of the Rectangle fall inside the circle. Or as a more challenging version, return True if any part of the Rectangle falls inside the circle.

In [4]:
```
import math

class Point:
    """Represents a point in 2-D space.

    attributes: x, y
    """

class Circle:
    """Represents a circle.

    Attributes: center, radius
    """
class Rectangle:
    """Represents a rectangle.

    attributes: width, height, corner.
    """

def distance(point1, point2):
    return math.sqrt((point2.x - point1.x)**2 + (point2.y - point1.x)**2)

def point_in_circle(circle, point):
    if distance(circle.center, point) <= circle.radius:
        return True
```

```
    else:
        return False

def rect_in_circle(rect, circle):
    point = copy.copy(rect.corner)
    print_point(p)
    if not point_in_circle(p, circle):
        return False

    p.x += rect.width
    print_point(p)
    if not point_in_circle(p, circle):
        return False

    p.y -= rect.height
    print_point(p)
    if not point_in_circle(p, circle):
        return False

    p.x -= rect.width
    print_point(p)
    if not point_in_circle(p, circle):
        return False

    return True

def rect_circle_overlap(circle, rectangle):
    point = copy.copy(rect.corner)
    print_point(p)
    if point_in_circle(p, circle):
        return True

    p.x += rect.width
    print_point(p)
    if point_in_circle(p, circle):
        return True

    p.y -= rect.height
    print_point(p)
    if point_in_circle(p, circle):
        return True

    p.x -= rect.width
    print_point(p)
    if point_in_circle(p, circle):
        return True

    return False
```

```
if __name__ == '__main__':
    circle = Circle()
    circle.center = Point()
    circle.center.x = 150
    circle.center.y = 100
    circle.radius = 75
```