Think Python

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.

```
import math


def main():
        # Create a Circle object with center at (150, 100) and radius 75
        circle = Circle(150, 100, 75)
        print(circle)

        # Create a Rect object at (150, 100) with width and height of 10
        rect = Rect(150, 100, 10, 10)
        print(rect)

        # Check if a point (100, 200) is inside the circle
        print(point_in_circle(Point(100, 200), circle))

        # Check if the rectangle is completely inside the circle
        print(rect_in_circle(rect, circle))

        # Check if any corner of the rectangle overlaps with the circle
        print(rect_circle_overlap(rect, circle))

        # Check if any part of the rectangle overlaps with the circle
        print(rect_part_circle_overlap(rect, circle))
```

```python
class Point():
        # Class representing a point in 2D space
        def __init__(self, x, y):
                self.x = x
                self.y = y

        def __str__(self):
                # String representation of the Point
                return f"Point at {self.x}x, {self.y}y"


class Circle():
        # Class representing a circle
        def __init__(self, x, y, r):
                self.center = Point(x, y)  # Circle center as a Point object
                self.r = r  # Circle radius

        def __str__(self):
                # String representation of the Circle
                return f"Circle created at {self.center.x}x, {self.center.y}y with a radius of {self.r}"


class Rect():
        # Class representing a rectangle
        def __init__(self, x, y, w, h):
                self.corner = Point(x, y)  # Top left corner as a Point object
                self.width = w  # Rectangle width
                self.height = h  # Rectangle height

        def __str__(self):
                # String representation of the Rect
                return f"Rectangle created at {self.corner.x}x, {self.corner.y}y of {self.width} width and {self.height} height"


def point_in_circle(point, circle):
        # Check if a point lies within or on the boundary of a circle
        dist = math.dist((point.x, point.y), (circle.center.x, circle.center.y))
        return dist <= circle.r


def rect_in_circle(rect, circle):
        # Check if all corners of the rectangle are within the circle
        corners = rect_corners(rect)
        return all(point_in_circle(corner, circle) for corner in corners)
```

```
def rect_circle_overlap(rect, circle):
        # Check if any corner of the rectangle is within the circle
        corners = rect_corners(rect)
        return any(point_in_circle(corner, circle) for corner in corners)


def rect_part_circle_overlap(rect, circle):
        # Check if any part of the rectangle overlaps with the circle
        # Determine the closest x-coordinate on the rectangle to the circle's center
        if circle.center.x < rect.corner.x:
                testX = rect.corner.x  # Left edge
        elif circle.center.x > rect.corner.x + rect.width:
                testX = rect.corner.x + rect.width  # Right edge
        else:
                testX = circle.center.x  # Inside rectangle width

        # Determine the closest y-coordinate on the rectangle to the circle's center
        if circle.center.y < rect.corner.y:
                testY = rect.corner.y  # Top edge
        elif circle.center.y > rect.corner.y + rect.height:
                testY = rect.corner.y + rect.height  # Bottom edge
        else:
                testY = circle.center.y  # Inside rectangle height

        # Check if the closest point on the rectangle boundary is within the circle's radius
        return math.dist((testX, testY), (circle.center.x, circle.center.y)) <= circle.r


def rect_corners(rect):
        # Return a list of corner points of the rectangle
        return [
                rect.corner,  # Top Left
                Point(rect.corner.x + rect.width, rect.corner.y),  # Top Right
                Point(rect.corner.x, rect.corner.y + rect.height),  # Bottom Left
                Point(rect.corner.x + rect.width, rect.corner.y + rect.height)  # Bottom Right
        ]


if __name__ == "__main__":
        main()
```

Exercise 15.1 Output.

```
Circle created at 150x, 100y with a radius of 75

Rectangle created at 150x, 100y of 10 width and 10 height

False

True

True

True
```

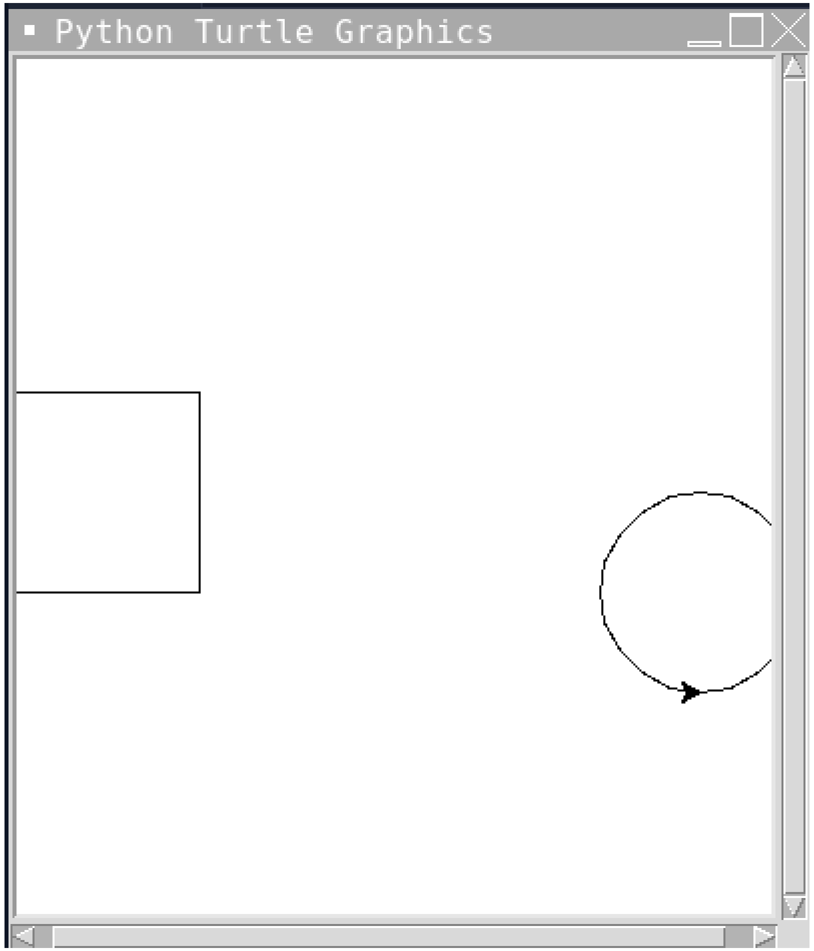
Exercise 15.2.

Write a function called draw_rect that takes a Turtle object and a Rectangle and uses the Turtle to draw the Rectangle. See Chapter 4 for examples using Turtle objects.

Write a function called draw_circle that takes a Turtle and a Circle and draws the Circle.

```python
import turtle

def main():
        # Create a turtle named bob
        bob = turtle.Turtle()

        # Create a rectangle with specified coordinates and dimensions
        myRect = Rect(x=-200, y=50, height=100, width=100)
        myRect.draw(bob)

        # Create a circle with specified coordinates and radius
        myCircle = Circle(x=150, y=-50, radius=50)
        myCircle.draw(bob)

        turtle.mainloop()


class Rect():
        def __init__(self, x, y, width, height):
                # Initialize the rectangle's position and dimensions
                self.x = x
                self.y = y
                self.width = width
                self.height = height

        def draw(self, t):
                # Move to the starting position
                t.penup()
                t.setpos(self.x, self.y)
                t.pendown()

                # Draw the rectangle shape
                for _ in range(2):
                        t.forward(self.width)
                        t.right(90)
                        t.forward(self.height)
                        t.right(90)


class Circle():
        def __init__(self, x, y, radius):
                # Initialize the circle's position and radius
                self.x = x
                self.y = y
                self.radius = radius

        def draw(self, t):
                # Move to the starting position
                t.penup()
                t.setpos(self.x, self.y - self.radius)
                t.pendown()

                # Draw the circle shape
                t.circle(self.radius)


if __name__ == "__main__":
        main()
```

Exercise 15.2 Output.


Python Turtle Graphics