Ky Kartchner
CS 1400-1
Assignment 6

Task 1: Rock, Paper, Scissors

1. Requirements Specification - The program must satisfy the following requirements:

- User is able to enter their move of Rock, Paper, or Scissors
- Program randomly generates a 0, 1, or 2, for the computer's move
- Program displays both moves, determines and displays winner (Rock beats Scissors; Scissors beats Paper; Paper beats Rock)
- User is able to play again
- Program has proper naming conventions and comments

2. System Analysis - The output is the winner, which is determine by the following rules: Rock beats Scissors; Scissors beats Paper; Paper beats Rock. If the player types "Y" when asked if they would like to play again, a new round begins, and the player is asked to enter their move.

Outputs: Winner of round
Inputs: (needed each round) Player's move
AI's move (generated randomly)
Player's answer to if they would like to play again

3. System Design

Step 1. Prompt user to enter their move (0-Rock, 1- Paper, 2-Scissors)

Step 2. Randomly generate computer's move

Step 3. Determine a winner or if the game is a tie

Step 4. Print the results

Console displays:

Enter your move (0-Rock, 1- Paper, 2-Scissors):
You chose <move>. AI chose <move>.
("You won! :)" or "You lost :(") or ("It's a tie!")
Would you like to play again? (Y/N):

4. Implementation

```
from random import randint
play_again = 'Y'


def name_of_move(move):
    if move == 0:
        return "Rock"
    elif move == 1:
```

```
        return "Paper"
    elif move == 2:
        return "Scissors"


print("Welcome to Rock-Paper-Scissors!")
while play_again == 'Y' or play_again == 'y':
    user_move = eval(input("\nEnter your move (0-Rock, 1-Paper, 2-Scissors): "))
    if user_move > 2 or user_move < 0:
        print("Invalid move. Enter only 0, 1, or 2")
    else:
        ai_move = randint(0, 2)

        if user_move == ai_move:
            round_result = "It's a tie!"
        elif (user_move - ai_move) % 3 == 1:
            round_result = "You won! :)"
        else:
            round_result = "You lost :("

        print("You chose " + name_of_move(user_move) + ". AI chose " +
name_of_move(ai_move) + ".\n" + round_result)
        play_again = input("\nWould you like to play again? (Y/N): ")
```

5. Testing

Test 1 – Manually set AI's move to always equal "Rock", then run the program and test output of entering "Rock" then "Paper" then "Scissors" for User's move.

AI Move	User Move	Expected Output (Winner)	Actual Output
0-Rock	0-Rock	Tie	Tie
0-Rock	1-Paper	User	User
0-Rock	2-Scissors	AI	AI

Problems: NONE. Output as expected.

Test 2 – Manually set AI's move to always equal "Paper", then run the program and test output of entering "Rock" then "Paper" then "Scissors" for User's move.

AI Move	User Move	Expected Output (Winner)	Actual Output
1-Paper	0-Rock	AI	AI
1-Paper	1-Paper	Tie	Tie
1-Paper	2-Scissors	User	User

Problems: NONE. Output as expected.

Test 2 – Manually set AI's move to always equal "Scissors", then run the program and test output of entering "Rock" then "Paper" then "Scissors" for User's move.

AI Move	User Move	Expected Output (Winner)	Actual Output
2-Scissors	0-Rock	User	User
2-Scissors	1-Paper	AI	AI
2-Scissors	2-Scissors	Tie	Tie

Problems: NONE. Output as expected.

Task 2 – Circle Comparisons

1. Requirements Specification - The program must satisfy the following requirements:

- User is prompted for input and can enter x-y coordinates and radii for two circles properly
- Program able to detect if one of the circles is inside the other, and if the circles overlap or not
- Program has proper naming conventions and comments

2. System Analysis – The output of the program is the relationship of the two circles. There are five possible outputs that can result from the input (center x-y coordinates and radii that are entered for the two circles). The output is determined by comparing the distance between the centers and the radii of the circles using the following formula and comparisons:

distance between centers = $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Output (Relationship of the two circles)	**Comparison of Inputs**
circle2 is inside circle1	distance between centers **<=**|radius1 – radius2| AND radius1 **>** radius2
circle1 is inside circle2	distance between centers **<=**|radius1 – radius2| AND radius1 **<** radius2
circle2 and circle1 have the same radius and center location	distance between centers **<=**|radius1 – radius2| AND radius1 **=** radius2
circle2 and circle1 overlap	distance between centers **<=** radius1 + radius2
circle2 and circle1 do *not* overlap	distance between centers **>** radius1 + radius2

Output: Relationship of the two circles

Input: circle1 attributes (x1, y1, radius1)
circle2 attributes (x2, y2, radius2)

3. System Design

Step 1. Prompt user to enter circle1 center coordinates and radius

Step 2. Prompt user to enter circle2 center coordinates and radius

Step 3. Calculate the distance between the centers and compare to radii

Step 4. Print relationship of circles

Console displays:

Enter circle1's center x-y coordinates and radius (x, y, radius):
Enter circle2's center x-y coordinates and radius (x, y, radius):

<Relationship of the two circles>

4. Implementation

```
import math

x1, y1, radius1 = eval(input("Enter circle1's center x-y coordinates and radius (x, y,
radius): "))
x2, y2, radius2 = eval(input("Enter circle2's center x-y coordinates and radius (x, y,
radius): "))

distance_between_centers = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
output = ""

if distance_between_centers <= abs(radius1 - radius2):
    if radius1 > radius2:
        output = "circle2 is inside circle1"
    elif radius1 < radius2:
        output = "circle1 is inside circle2"
    else:
        output = "circle1 and circle2 have the same radius and location"
elif distance_between_centers <= radius1 + radius2:
    output = "circle2 overlaps circle1"
else:
    output = "circle2 does not overlap circle1"

print("\n" + output)
```

5. Testing

Test 1 – Enter the following attributes for circle1 and circle2:

	x	**y**	**radius**
circle1:	0.5	5.1	13
circle2:	1	1.7	4.5
Expected Output:	circle2 is inside circle1		
Actual Output:	circle2 is inside circle1		

Problems: NONE. Output as expected.

Test 2 – Enter the following attributes for circle1 and circle2:

	x	**y**	**radius**
circle1:	1	1.7	4.5
circle2:	0.5	5.1	13
Expected Output:	circle1 is inside circle2		
Actual Output:	circle1 is inside circle2		

Problems: NONE. Output as expected.

Test 3 – Enter the following attributes for circle1 and circle2:

	x	**y**	**radius**
circle1:	4.4	5.7	5.5
circle2:	6.7	3.5	3
Expected Output:	circle2 overlaps circle1		
Actual Output:	circle2 overlaps circle1		

Problems: NONE. Output as expected.

Test 4 – Enter the following attributes for circle1 and circle2:

	x	**y**	**radius**
circle1:	4.4	5.5	1
circle2:	5.5	7.2	1
Expected Output:	circle2 does not overlap circle1		
Actual Output:	circle2 does not overlap circle1		

Problems: NONE. Output as expected.

Test 5 – Enter the following attributes for circle1 and circle2:

	x	**y**	**radius**
circle1:	5	6	10
circle2:	5	6	10
Expected Output:	circle2 and circle1 have the same radius and center location		
Actual Output:	circle2 and circle1 have the same radius and center location		

Problems: NONE. Output as expected.