ROCK PAPER SCISSOR GAME REPORT

=> Introduction

Rock paper scissor is a very popular game and regularly played physically. A player who decides to play rock will beat another player who has chosen scissors ("rock crushes scissors" or "breaks scissors" or sometimes "blunts scissors"), but will lose to one who has played paper ("paper covers rock"); a play of paper will lose to a play of scissors ("scissors cuts paper"). The game developed using Python programming language. The program is designed using original rules of hand game. Game development with Python has many advantages like simplicity and code vulnerability. Python is a very good choice for programming and prototyping. All the resultant output can be displayed instantly and many libraries, module, frameworks and documentation that make Python language easier and learn faster.

This document is to design and develop Rock, Paper and Scissor game using Python Language and study development skill.

- The program of game is to choose random option which is **Rock, Paper** and **Scissor** from Computer.
- The user have option to choose an option as **0 for Rock, 1 for Paper** and **2 for Scissor**.
- When user press 5 as input, the game will ask user to really end game with **Yes and No** option.
- If user press **Yes** then game will continue to play otherwise game will be exited and shows the result with **Win**, **Lose** and **Tie** count and the **Winning Percentage**.
- If user enters wrong value rather than explained above, it will give the appropriate message.

Github Link for Source Code and Report - https://github.com/gangajaliya-sandeep/RPSGame

=> Process

1. **rockPaperScissorGame.py:**

   This file contains program of game.

```
# Import random module for random selection for computer
import random
```

Import Random Module

This module will help to get random selection as a computer input between 0 to 2.

```
# These objects will hold winning, lose and tie count
winCount = loseCount = tieCount = 0.0
```

Win, lose and tie object

The above objects holds winning, lose and tie count for a game. Whenever program exited, all objects will become 0.

```
# Take User's choice as input
def playerChoice():

    try:
        # Take choice from user as input
        userChoice = int(
            input("Select your choice => 0: Rock, 1: Paper, 2: Scissor, 5: Exit => "))

        # Validate User's choice
        if (validateUserInput(userChoice)):
            return userChoice
        else:
            playerChoice()

    except ValueError:
        # Throw error if selected wrong option
        print("Please enter an number \n")
```

User Input

PlayerChoice function will take input from user in the form of **0: Rock, 1: Paper, 2: Scissor** and **5: Exit** and call validate function. Also this function return user's choice if choice is validated.

```
# Validate User's choice
def validateUserInput(userChoice):

    # Compare User's choice
    if ((userChoice <= 2 and userChoice >= 0) or userChoice == 5):
        return True
    else:
        # Wrong choice from user, try again
        print(
            "\nPlease enter an integer less than 3 and greater than or equalt to 0 or 5 to exit. \n")
        return False
```

Validate User Input

This function will be used to validate the user's choice. User's choice should be 0, 1, 2 or 5. Otherwise function will return false and will ask again to input choice to user again.

```
# Function to play game
def playGame():

    # Choices is a static array
    choices = ["Rock", "Paper", "Scissor"]

    # This object will decide to play again or exit from game
    playContinue = True

    # Play continue
    while (playContinue):

        # User choice
        userChoice = playerChoice()

        # Computer choice with random between 0 to 2
        computerChoice = random.randint(0, 2)

        # Check user choice if valid or not
        if (userChoice <= 2 and userChoice >= 0):

            result = checkResult(userChoice, computerChoice)
            print("Your choice is %s" % choices[userChoice])
            print("The computer choose %s" % choices[computerChoice])
            print("You %s" % result)

        elif (userChoice == 5):
            # User choose to exit from game
            playContinue = False
```

Heart Of Game

PlayGame is a heart of game and main login of game will be called from this function. This function is used to compare user's choice and check for result.

```
# Compare choice and decide result
def checkResult(user, computer):
```

Logic

CheckResult will compare the user and computer's choice and decides the result.

```
# Caluculate and set win, lose and tie count
def countStats(result):
    global winCount
    global loseCount
    global tieCount

    if (result == 0):
        winCount += 1
    elif (result == 1):
        loseCount += 1
    else:
        tieCount += 1
```

Store Stats

This function will store the states of game.

```
# Announce results
def finaliseResult():

    # Get global object
    global winCount, loseCount, tieCount
    print("These are your final result: ")

    # Calculate winning percentage
    if (winCount+loseCount+tieCount != 0):
        winningPercentage = "{percent:.2%}".format(
            percent=(winCount / (winCount+loseCount)))
    else:
        winningPercentage = "{percent:.2%}".format(percent=0)

    # Pring final result
    print("You won %d game(s), you lose %d game(s) and %d tied game(s)!" %
          (winCount, loseCount, tieCount))
    print("You have a %s winning percentage" % winningPercentage)
```

Finalise Result

This function calculate the result by winCount, loseCount and tieCount and also calculate winning percentage based on win and lose. After calculating, this function announce result.

```
# This function will call at very first time
def main():
```

Main Function

This function will be called first and will call PlayGame function.

=> TDD

TDD (Test Driven Development) is a great approach to test program, project or software. Basically TDD is a software development practice that allow us to write test cases before developing the program or software.

In this game, **unittest module** is used for automation unit test. Below code snapshot explains how TDD is implemented.

```
import unittest
import xmlrunner
from game.rockPaperScissorGame import playerChoice
```

Imports

These imports are useful to write code for TDD.

```
ROCK = 0
PAPER = 1
SCISSOR = 2
FAILURE = 'Incorrect input'
SUCCESS = 'Correct input'
```

```
class RockPaperScissorGameTest(unittest.TestCase):
    def setUp(self):
        self.playerChoice = playerChoice()

    def user_choice_rock_init(self):
        value = self.playerChoice.userChoice
        self.assertEqual(value, 0, FAILURE)

    def user_choice_paper_init(self):
        value = self.playerChoice.userChoice
        self.assertEqual(value, 1, SUCCESS)

    def user_choice_scissor_init(self):
        value = self.playerChoice.userChoice
        self.assertEqual(value, 2, SUCCESS)

    def user_choice_exit_init(self):
        value = self.playerChoice.userChoice
        self.assertEqual(value, 1, SUCCESS)

    def user_choice_failure1_init(self):
        value = self.playerChoice.userChoice
        self.assertEqual(value, -1, FAILURE)
```

Test Cases

This code is used for automation testing. All function will provide result according to inputs.

=> Conclusion

The above game has been developed by using the concept of object-oriented programming in **Python**. Python provides supports structured and function oriented programming. Python also provides dynamic memory management which makes program and code efficient.

All functions and objects (Variables) are implemented for a clear understanding of the game and program. After declaring all methods, all methods will be called from main function. Functions are written in an efficient way by dividing large chunk of code into smaller parts that can be understood easily.

This Python game illustrate that all of the events can be done with minimum effort, and that the Rock, Paper and Scissor game is built form scratch.