# SOFTWARE UNIT TESTING REPORT

## Table of Contents

## Introduction

Rock paper scissor is a popular game. This game is mainly played as a decider among teammates. Rock paper scissor game is mostly played as a hand game between two players. Boccignone *et al* (2022)The players say **"rock", "paper"**, and **"scissors"** and continuously form their hands into the shape of rock (fist), paper( palm facing downward), or scissors (extended two fingers). In general, the rules are simple that is **"rock"** smashes scissors, **"paper"** covers rock and **"scissors"** cut the paper. Using this game principle, the game has been built using python. The program is based on the rules of the original game played as a hand game. Game development in python has many benefits like simplicity and code vulnerability. Python is a great choice for prototyping. All the resultant output can be displayed immediately and deliverable faster to potential investors. Python has several libraries and frameworks that make game development easier.

Github Link - https://github.com/ridhamnadoda4/RockPaperScissor

## Objectives

- To build a rock scissor paper game in python
- To study the game development skills in python

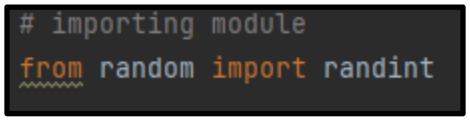
## Requirements

- The program randomly picks an option ***(rock, paper or scissor)*** as the computer choice.
- The user can provide the number of rounds as per choice.
- The user is asked to provide an option as rock, paper, scissors (one at a time)
- Point's criteria have been mentioned.
- A total number of rounds are displayed.
- After the completion of rounds, the user is asked to continue or quit.

## Automated testing tools

Automated testing tools are used for execution of test cases by using a script rather than human. One of the testing modules in python is unittest. This is a manual testing module used in python environment. This module requires much and effort. Rather using nose2 can be a much more effective module for testing purpose. This is a extension of unittest module.

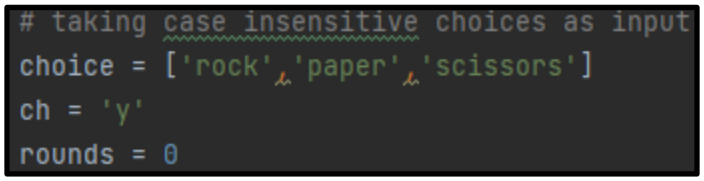
## Process



```python
# importing module
from random import randint
```

**Figure 1: Importing the module**

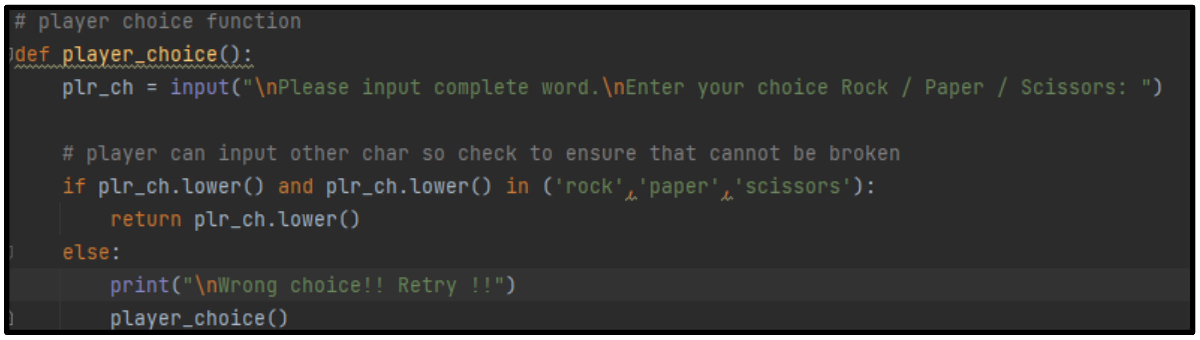
This enables to use of various tools inside the program that randomizes the computer event in the game.



```python
# taking case insensitive choices as input
choice = ['rock','paper','scissors']
ch = 'y'
rounds = 0
```

**Figure 2: Making user input case insensitive**

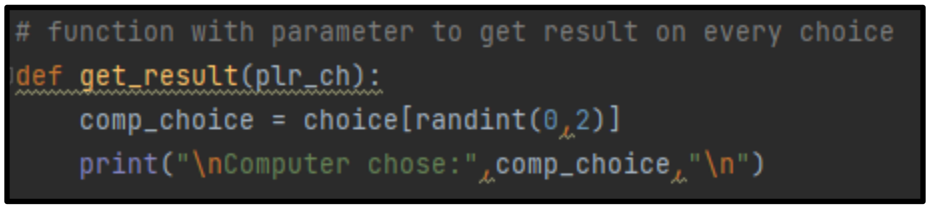
The above image explains the choice variable that is being used for user input. All the characters are assigned in lowercase strings.



```python
# player choice function
def player_choice():
    plr_ch = input("\nPlease input complete word.\nEnter your choice Rock / Paper / Scissors: ")

    # player can input other char so check to ensure that cannot be broken
    if plr_ch.lower() and plr_ch.lower() in ('rock','paper','scissors'):
        return plr_ch.lower()
    else:
        print("\nWrong choice!! Retry !!")
        player_choice()
```

**Figure 3: Function for user choice**

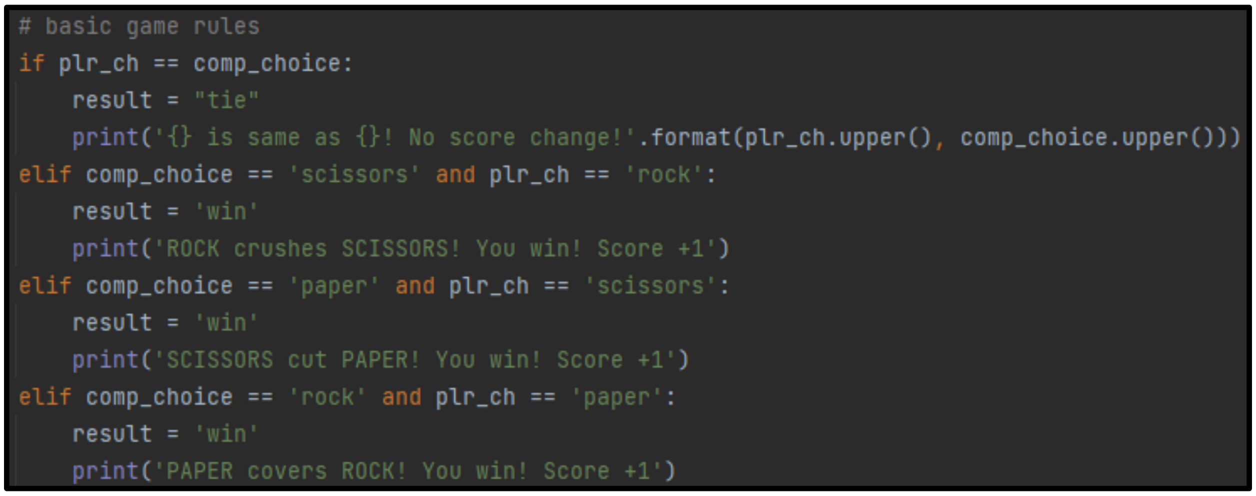
The above image shows the user choice function. Sharma *et al* (2022) This will allow the user to select ***rock, paper, and scissors*** from the menu. Then it checks for the user choice input that is similar to the main choice variable. If the condition goes right then it returns the user choice or else again asks for the choice.



```
# function with parameter to get result on every choice
def get_result(plr_ch):
    comp_choice = choice[randint(0,2)]
    print("\nComputer chose:",comp_choice,"\n")
```

**Figure 4:  get result function**

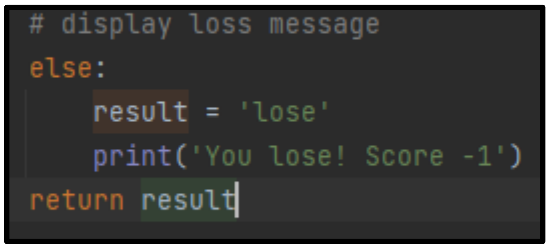
The above image describes the function for getting results for each choice.



```
# basic game rules
if plr_ch == comp_choice:
    result = "tie"
    print('{} is same as {}! No score change!'.format(plr_ch.upper(), comp_choice.upper()))
elif comp_choice == 'scissors' and plr_ch == 'rock':
    result = 'win'
    print('ROCK crushes SCISSORS! You win! Score +1')
elif comp_choice == 'paper' and plr_ch == 'scissors':
    result = 'win'
    print('SCISSORS cut PAPER! You win! Score +1')
elif comp_choice == 'rock' and plr_ch == 'paper':
    result = 'win'
    print('PAPER covers ROCK! You win! Score +1')
```

**Figure 5: Implementation of game rules**

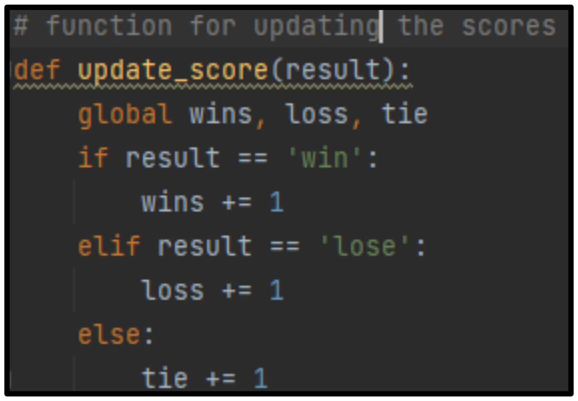
The above image explains the basic rule of the game which is the logic of the game. After taking input from the user ***(plr_choice)*** the condition statement checks for the computer choice ***(comp_choice)***. Here the condition is mentioned for ***"tie"*** and ***"win"***.



```
# display loss message
else:
    result = 'lose'
    print('You lose! Score -1')
return result
```

**Figure 6: displaying loss message**

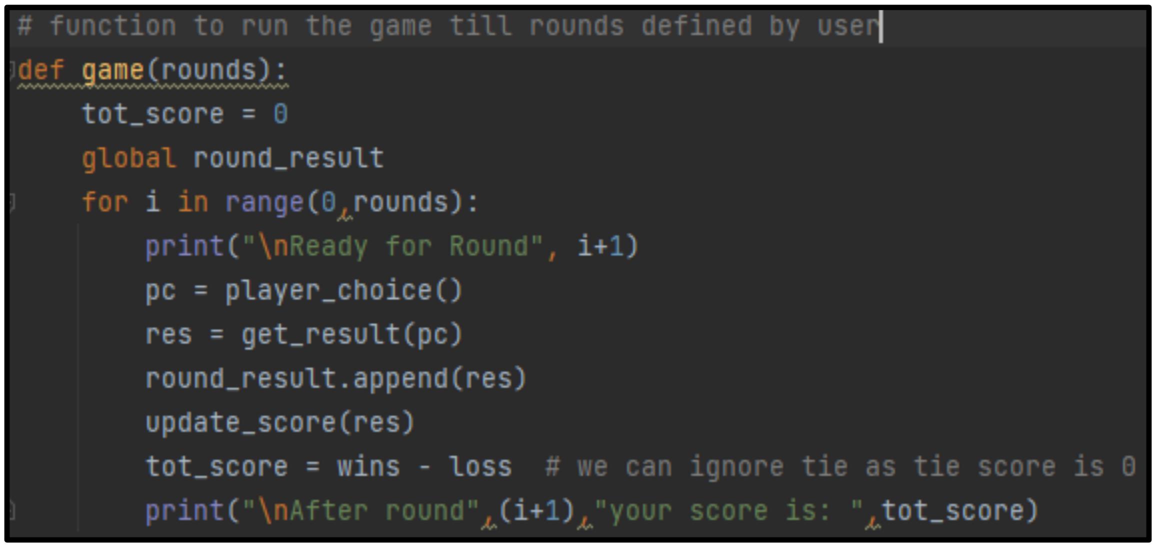
The image above shows the printing of loss condition that -1 for losing. This condition takes place when the result equals loss.



```python
# function for updating the scores
def update_score(result):
    global wins, loss, tie
    if result == 'win':
        wins += 1
    elif result == 'lose':
        loss += 1
    else:
        tie += 1
```

**Figure 7: Score updation**

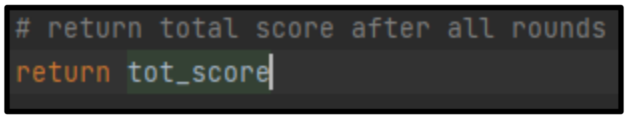
The above image shows the condition for updating scores. For updating scores, the result has been passed as a parameter for condition checking.



```python
# function to run the game till rounds defined by user
def game(rounds):
    tot_score = 0
    global round_result
    for i in range(0,rounds):
        print("\nReady for Round", i+1)
        pc = player_choice()
        res = get_result(pc)
        round_result.append(res)
        update_score(res)
        tot_score = wins - loss  # we can ignore tie as tie score is 0
        print("\nAfter round",(i+1),"your score is: ",tot_score)
```

**Figure 8: Executing game till user-defined rounds**

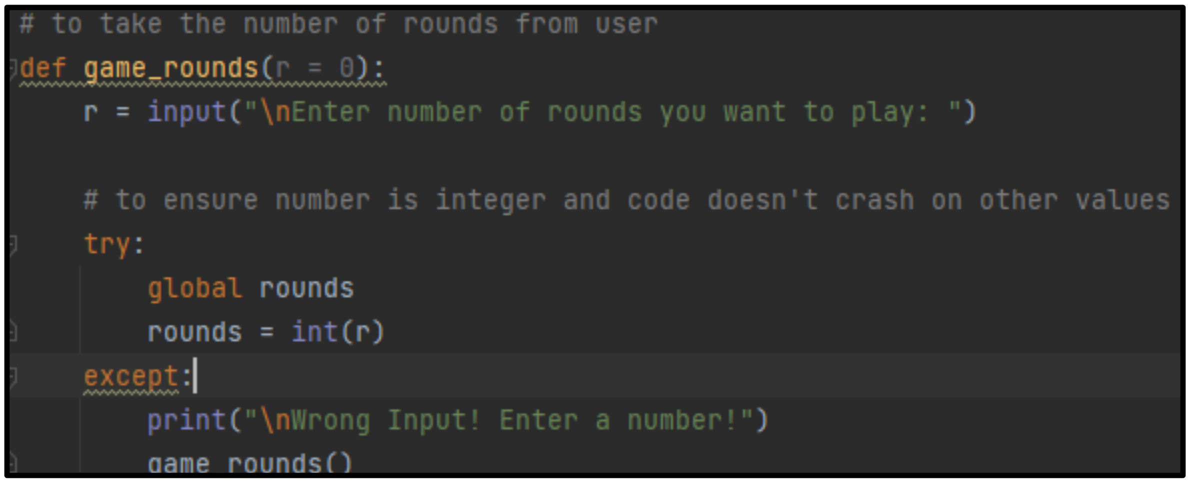
The above image describes the function for game rounds that will be provided by the user at the beginning of the game. Here **"rounds"** has been taken as an argument for playing certain rounds inside the game. Automatically this function updated the scores with each round completion.



```python
# return total score after all rounds
return tot_score
```

**Figure 9: Total score**

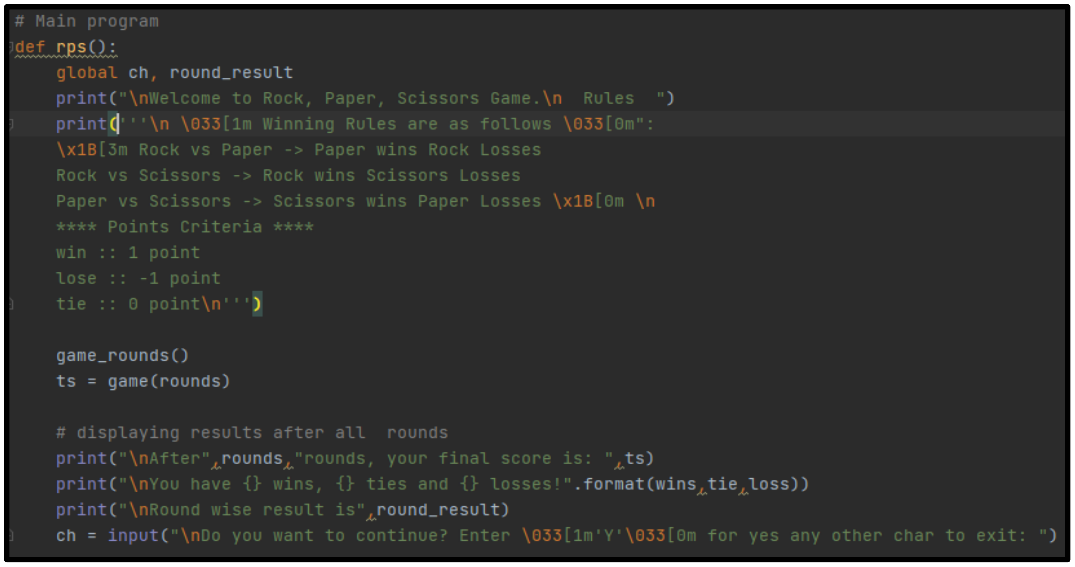
The image above prints the total score after completion of all the rounds. The total score has been denoted by **“tot_score”** variable.



```python
# to take the number of rounds from user
def game_rounds(r = 0):
    r = input("\nEnter number of rounds you want to play: ")

    # to ensure number is integer and code doesn't crash on other values
    try:
        global rounds
        rounds = int(r)
    except:
        print("\nWrong Input! Enter a number!")
        game_rounds()
```

**Figure 10: Implementing the number of rounds**

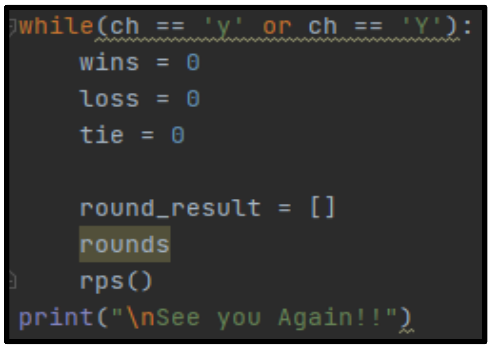
The image above shows the implementation of game rounds that will be provided by the user. Also, there is an exceptional case if the wrong input is provided.



```python
# Main program
def rps():
    global ch, round_result
    print("\nWelcome to Rock, Paper, Scissors Game.\n  Rules  ")
    print('''\n \033[1m Winning Rules are as follows \033[0m":
    \x1B[3m Rock vs Paper -> Paper wins Rock Losses
    Rock vs Scissors -> Rock wins Scissors Losses
    Paper vs Scissors -> Scissors wins Paper Losses \x1B[0m \n
    **** Points Criteria ****
    win :: 1 point
    lose :: -1 point
    tie :: 0 point\n''')

    game_rounds()
    ts = game(rounds)

    # displaying results after all  rounds
    print("\nAfter",rounds,"rounds, your final score is: ",ts)
    print("\nYou have {} wins, {} ties and {} losses!".format(wins,tie,loss))
    print("\nRound wise result is",round_result)
    ch = input("\nDo you want to continue? Enter \033[1m'Y'\033[0m for yes any other char to exit: ")
```

**Figure 11: Main program**

The image above describes the main program for the python code. Tao *et al* (2022) the main program calls all the functions individually. All the game rules and point criteria have been mentioned in the main program. After implementing all the functions, the program asks the user to continue or not.

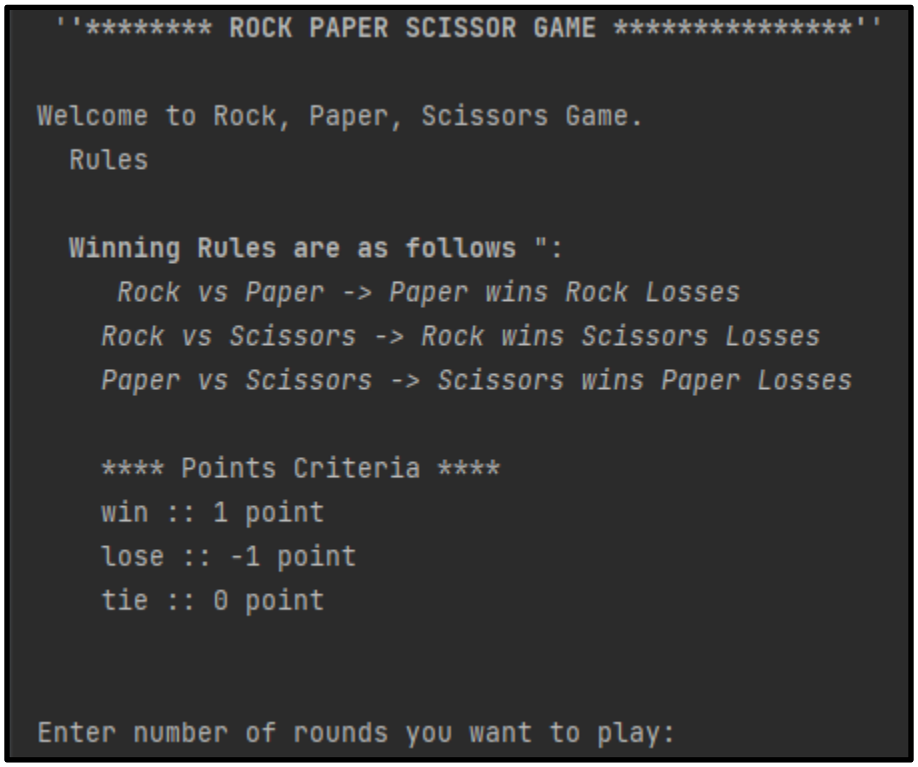


```
while(ch == 'y' or ch == 'Y'):
    wins = 0
    loss = 0
    tie = 0

    round_result = []
    rounds
    rps()
print("\nSee you Again!!")
```

**Figure 12: Condition for exit**

The image above shows the condition for character checking that compares with user input. We have implemented both the conditions for case sensitivity that **‘y’** and **‘Y’**. So the user can input both options.

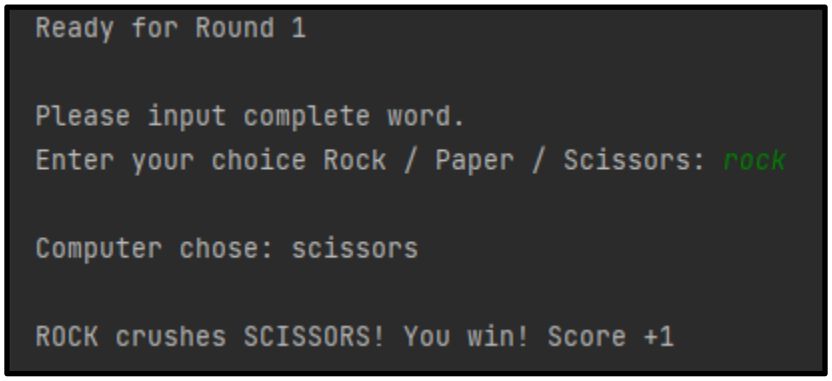
## Output



```
 ''******** ROCK PAPER SCISSOR GAME ***************''

Welcome to Rock, Paper, Scissors Game.
  Rules

  Winning Rules are as follows ":
     Rock vs Paper -> Paper wins Rock Losses
    Rock vs Scissors -> Rock wins Scissors Losses
    Paper vs Scissors -> Scissors wins Paper Losses

    **** Points Criteria ****
    win :: 1 point
    lose :: -1 point
    tie :: 0 point


Enter number of rounds you want to play:
```

**Figure 13: Output console for the game intro**

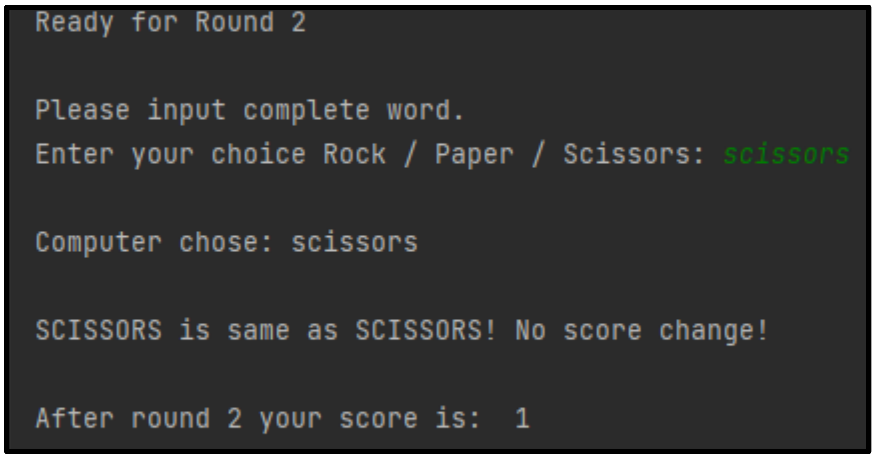
The image shows the output screen for the rock scissor paper game. The output shows the intro of the game that asks the user to enter the number of rounds.



```
Ready for Round 1

Please input complete word.
Enter your choice Rock / Paper / Scissors: rock

Computer chose: scissors

ROCK crushes SCISSORS! You win! Score +1
```

**Figure 14: Round 1**

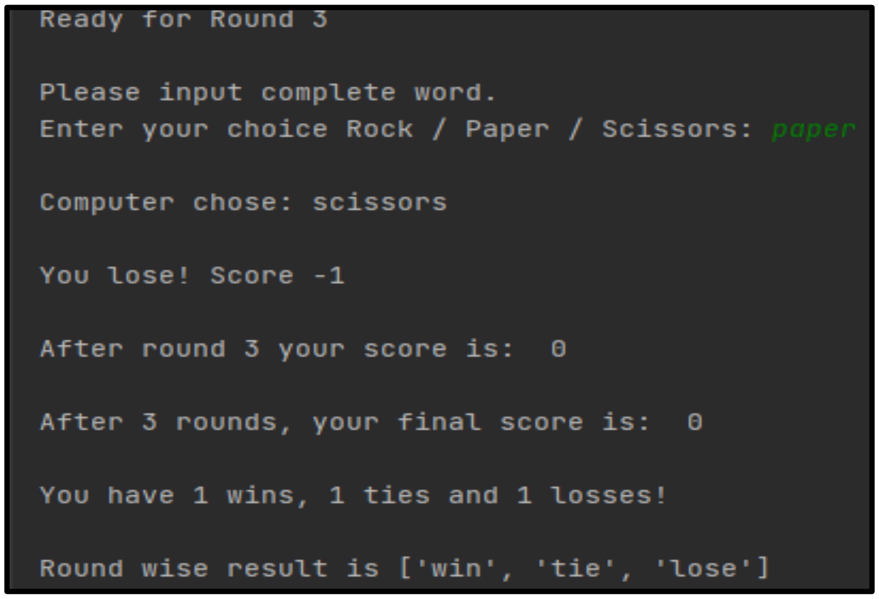
The image shows the output of round 1 where user has selected rock as the option.



```
Ready for Round 2

Please input complete word.
Enter your choice Rock / Paper / Scissors: scissors

Computer chose: scissors

SCISSORS is same as SCISSORS! No score change!

After round 2 your score is:  1
```

**Figure 15: Round 2**

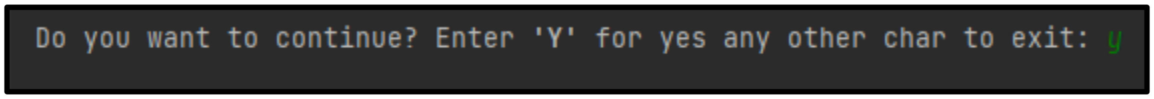
The image shows the output of round 2 where the user has selected scissors as option an also it shows the computer option and the score.



```
Ready for Round 3

Please input complete word.
Enter your choice Rock / Paper / Scissors: paper

Computer chose: scissors

You lose! Score -1

After round 3 your score is:  0

After 3 rounds, your final score is:  0

You have 1 wins, 1 ties and 1 losses!

Round wise result is ['win', 'tie', 'lose']
```

**Figure 16: Round 3**

The image shows the output for round 3 where user selected paper and the computer again chose scissors. After completion of all the rounds total score has been displayed.



```
Do you want to continue? Enter 'Y' for yes any other char to exit: y
```

**Figure 17: After completion of the rounds (continue/quit)**

## Conclusion

The above project has been accomplished by using the concept of object-oriented programming in python. For each specific event, individual functions have been implemented for a clear understanding of the program and its functionality. After specifying all the functions, all the functions are called inside the main program. Functions are an efficient way to isolate large chunks of code into smaller understandable segments. From the above python game, it can be concluded that all the number of events in the game can be carried out with minimum effort and the rock paper scissors game has been built from scratch.

## References

Boccignone, G., Conte, D., Cuculo, V., Alessandro D'Amelio, Grossi, G., Lanzarotti, R. & Mortara, E. 2022, "pyVHR: a Python framework for remote photoplethysmography", *PeerJ Computer Science,* .

Sharma, M., Mishra, T.K. & Kumar, A. 2022, "Source code auto-completion using various deep learning models under limited computing resources", *Complex & Intelligent Systems,* vol. 8, no. 5, pp. 4357-4368.

Tao, G., Jiang, Q., Shi, C., Chen, C. & Jiang, Z. 2022, "Coupling coordination relationship between geology–geomorphology and ecology in Northeast China", *PLoS One,* vol. 17, no. 4.