**MINI PROJECT REPORT**

**ON**

**Rock Paper Scissors**

Submitted by

**Lekshmi Nair**

**NC.SC.U4CSE24216**

For

23CSE101 – Computational Problem Solving

I Semester

B.Tech. CSE

**School of Computing**

**Amrita Vishwa Vidyapeetham, Nagercoil**

**INTRODUCTION**

The game of **Rock, Paper, Scissors** is a widely recognized hand game played around the world. It serves as a simple yet engaging way to make decisions, often used in both casual settings and as a basic tool for resolving disputes. The rules are straightforward: each player simultaneously forms one of three shapes with their hands—rock, paper, or scissors. The outcome is determined by the interaction between the shapes, where rock beats scissors, scissors beat paper, and paper beats rock. Despite its simplicity, the game can be a fun and strategic challenge when played repeatedly.

In the context of computer programming, **Rock, Paper, Scissors** is often used as an introductory project for beginners to practice basic programming concepts. It provides a hands-on opportunity to apply essential programming skills such as user input handling, decision-making using conditional statements, and working with loops and randomness. This project aims to implement a simple text-based version of Rock, Paper, Scissors where the user plays against a computer opponent.

The objective of the project is to develop a Python-based game where the user selects one of three choices—rock, paper, or scissors—and the computer randomly chooses one of the same options. The program will then determine the winner based on the rules of the game, keeping track of the score and displaying the result after each round. Additionally, the game will allow the user to continue playing multiple rounds or exit after a series of games.

Python, known for its simplicity and readability, is an ideal language for such a project. The **random** module is used to simulate the computer’s choice, making the game interactive and dynamic. By the end of this project, users will gain practical experience with Python’s **conditionals**, **loops**, and **functions**, while creating an interactive game that they can easily expand upon in future projects.

This game also serves as a foundational exercise for building more complex programs. For example, one can extend this project to include a graphical user interface (GUI), multiplayer options, or even incorporate machine learning algorithms for the computer to "learn" from the player's previous moves. By starting with a simple concept like Rock, Paper, Scissors, learners can gradually progress to more sophisticated game development and computer science topics.

**PROBLEM STATEMENT**

The **Rock, Paper, Scissors** game is a well-known hand game that is typically played by two participants. Each player chooses one of three options: rock, paper, or scissors. The winner is determined by the following rules: rock beats scissors, scissors beat paper, and paper beats rock. If both players choose the same option, the game results in a tie. Despite its simplicity, the game is often used in various decision-making processes, from casual disputes to more formal scenarios.

In this project, the primary problem is to create a **Rock, Paper, Scissors** game where the user plays against a computer, as opposed to a human opponent. This is a common problem for beginners in programming, as it requires the implementation of basic programming concepts such as user input handling, conditionals, and loops. The challenge lies in designing the game logic so that it is easy to play, interactive, and tracks the score through multiple rounds of gameplay.

The game must:

1. **Accept User Input**: The player should be able to choose between rock, paper, or scissors by typing the corresponding choice into the terminal.
2. **Generate Computer’s Choice**: The computer must randomly select one of the three choices (rock, paper, or scissors) using a randomization method.
3. **Determine the Winner**: The game must compare the user’s and computer’s choices and determine the winner based on the predefined rules. If both choices are the same, the round results in a tie.
4. **Track the Score**: The game should maintain a score counter that tracks the number of wins, losses, and ties for both the user and the computer across multiple rounds.
5. **Allow Replay**: After each round, the player should be given the option to continue playing or quit the game. If the player chooses to quit, the final score should be displayed.
6. **Handle Invalid Inputs**: The program must ensure that the user inputs a valid option (rock, paper, or scissors) and provide feedback if the input is incorrect.

**OBJECTIVES**

The primary objective of the Rock, Paper, Scissors project is to create a simple yet interactive game in Python where the player competes against the computer. This project aims to provide practical experience with basic programming concepts while allowing the user to engage in a fun and familiar game. The following are the key objectives that this project seeks to achieve:

1. **Understand and Implement Basic Python Syntax**:
   * The project will help the user familiarize themselves with Python syntax, such as variables, conditionals (if, elif, else), loops, and functions. By implementing a simple game, users can directly apply these concepts in an interactive setting.
2. **Handle User Input Effectively**:
   * One of the main objectives is to enable the program to receive user input for choosing between Rock, Paper, or Scissors. The player should be able to type their choice, and the program will process it to continue the game flow. This helps in learning how to handle string input and validate user choices.
3. **Implement Randomization for Computer’s Choice**:
   * The project will use Python's random module to simulate the computer's choice of Rock, Paper, or Scissors. By generating random choices, the computer’s selections will be unpredictable, making the game more interactive and engaging. This teaches the concept of randomization and its application in gaming scenarios.
4. **Apply Conditional Logic to Determine the Winner**:
   * A key part of the game is comparing the user’s and the computer’s choices to determine the winner. This requires an understanding of logical comparisons using conditionals. The program will implement the rules of Rock, Paper, Scissors to identify which player (user or computer) wins, or if the round results in a tie.
5. **Track and Display Scores**:
   * The project will keep track of the score throughout the game, recording the number of wins, losses, and ties for both the player and the computer. Displaying the scores in real-time after each round will teach the user how to update variables and maintain running totals in a program.
6. **Create a Loop for Multiple Rounds**:
   * To make the game engaging, the user will have the option to play multiple rounds without restarting the program. The program will loop back to the choice input prompt after each round, allowing the player to continue playing or exit the game. This involves using loops to create repetitive game flow and manage multiple rounds.
7. **Provide Option to Quit the Game**:
   * After each round, the program will ask if the player wants to continue or quit. By implementing a loop that checks the user’s input, the program will allow the user to decide when to end the game. This demonstrates handling conditional loops and user decisions in an interactive setting.
8. **Handle Invalid Inputs Gracefully**:
   * The program will ensure that the user provides a valid input (either Rock, Paper, or Scissors). If an invalid choice is entered, the game will prompt the user again without crashing. This requires implementing input validation and error handling to improve the user experience.
9. **Provide a Fun and Engaging User Experience**:
   * Beyond the technical aspects, an important objective is to make the game enjoyable to play. By creating a smooth user interaction with clear instructions and feedback after each round, the project will improve the player’s experience and demonstrate how to create user-friendly command-line applications.
10. **Create a Foundation for Further Game Development**:
    * Finally, the project lays the groundwork for more advanced game development. As the user becomes comfortable with Python’s basic concepts, they can expand on this project by adding features like multiplayer options, a graphical user interface (GUI), or more complex game mechanics, such as difficulty levels or additional choices.

**PYTHON LIBRARIES USED IN THE PROJECT**

In this **Rock, Paper, Scissors** game project, the primary Python library used is the **random** module. This library is crucial for generating the computer’s choice in the game, making the game dynamic and interactive. Here’s an explanation of the key library used:

1. **random**:
   * The random module is a built-in Python library that provides various functions for generating random numbers and performing random actions. In this project, the most important function used is random.choice(). This function is used to randomly select an element from a given sequence, in this case, the list ['rock', 'paper', 'scissors'].
   * By using random.choice(), the computer’s choice of Rock, Paper, or Scissors is unpredictable, simulating the randomness of a human opponent’s decision. This ensures fairness, as each of the three options has an equal chance of being selected in every round.
   * The random library is essential for making the game engaging and interactive, as it removes any bias or predictability in the computer's behaviour. Without it, the game would be static, and the computer's choice would not be randomized, making it less interesting for the user.

This game relies on the random module as the only external library, using it to add an element of chance to the game. While the project is simple and doesn’t require additional libraries for its functionality, incorporating randomness allows the game to remain fair and engaging.

While additional libraries could be used for more complex versions of the game (such as graphical user interfaces or advanced AI), the use of the random module is sufficient for the basic logic and operation of the Rock, Paper, Scissors game.

This simplicity allows beginners to focus on understanding key programming concepts without needing to learn additional libraries.

**MODULES OF THIS PROJECT**

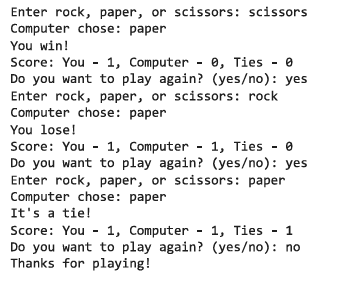
The Rock, Paper, Scissors game is divided into several logical modules that work together to create a smooth and interactive user experience. These modules help in organizing the code and improving the overall structure of the program. Below are the main modules involved in the project:

1. **User Input Handling**:
   * This module is responsible for capturing the user's choice of Rock, Paper, or Scissors. The program prompts the user to enter their choice using the input() function. It also ensures that the input is valid, i.e., it matches one of the allowed choices. If the user enters an invalid choice, the program will ask them to input again.
2. **Computer Choice Generation**:
   * The random module is used to generate the computer’s choice. This module selects one option (Rock, Paper, or Scissors) randomly from a predefined list using random.choice(). This ensures that the computer’s choice is unpredictable and mimics a human-like opponent.
3. **Game Logic**:
   * The game logic module is responsible for comparing the user’s choice and the computer’s choice. Using conditional statements (if, elif, else), the program determines whether the user wins, loses, or ties with the computer based on the traditional rules of the game (Rock beats Scissors, Scissors beats Paper, Paper beats Rock).
4. **Score Tracking**:
   * This module tracks the score of the game, keeping a record of the number of wins, losses, and ties for both the user and the computer. After each round, the score is updated and displayed to the player.
5. **Replay Option**:
   * After each round, the program asks the user if they want to play again or quit. This module controls the flow of the game, allowing it to loop through multiple rounds until the user chooses to exit.

**CODE**



**OUTPUT SCREENSHOTS**



**APPLICATIONS OF THE PROJECT**

The **Rock, Paper, Scissors** game may seem like a simple project, but it has several practical applications and can serve as a foundation for more complex systems. Below are some of the potential applications of this project:

1. **Introduction to Game Development**:
   * This project serves as an introduction to game development, teaching the basics of user input handling, game logic, and randomness. It provides a foundation for learners to build more complex games with graphical interfaces or advanced features.
2. **Programming Practice**:
   * The game helps practice core programming skills such as loops, conditionals, and functions. It is an excellent way for beginners to strengthen their understanding of Python syntax and debugging in a real-world scenario.
3. **Simulating Randomized Outcomes**:
   * The use of the **random** module to generate the computer’s choice introduces learners to randomization, a concept that’s useful in simulations, modelling, and testing scenarios, where unpredictability is needed.
4. **Multiplayer Expansion**:
   * While this version is single-player, the game can be expanded to support multiplayer functionality, allowing users to learn about networking or more advanced game logic.
5. **Command-Line Applications**:
   * The project also demonstrates how to create interactive command-line applications, which can be applied to other text-based utilities or games, offering insight into building interactive programs.

**LIMITATIONS OF THIS PROJECT**

While the **Rock, Paper, Scissors** game is a simple and engaging project, it has several limitations that can be addressed in future versions or expanded upon:

1. **Lack of Graphical Interface**:
   * The current version of the game is text-based, running in the command-line interface. This limits the user experience, as there is no visual representation of the game. A graphical user interface (GUI) with images for Rock, Paper, and Scissors could significantly enhance user engagement.
2. **Single Player**:
   * The game is designed for one player against the computer. It lacks multiplayer functionality, meaning two players cannot compete against each other in the same game. Implementing multiplayer options would make the game more interactive and allow for a more social experience.
3. **Basic AI**:
   * The computer's choice is made randomly, which can make the game predictable and less challenging. An advanced AI could be implemented to track the player's patterns and make smarter decisions, increasing the complexity of the game.
4. **No Difficulty Levels**:
   * There is no way to adjust the difficulty of the computer’s choices. The game could be improved by adding difficulty levels, where the computer makes more strategic decisions based on the player’s previous choices.
5. **Limited Game Features**:
   * The game only offers basic functionality. Additional features like a score history, a timer, or a more detailed rules explanation could make the game more interesting and user-friendly.

**BIBLIOGRAPHY**

* + [Rock Paper Scissors in Python - A Complete Step-By-Step Guide - AskPython](https://www.askpython.com/python/examples/rock-paper-scissors-in-python)
  + [Rock, Paper, Scissor game - Python Project - GeeksforGeeks](https://www.geeksforgeeks.org/rock-paper-scissor-game-python-project/)
  + [Make Your First Python Game: Rock, Paper, Scissors! – Real Python](https://realpython.com/python-rock-paper-scissors/)
  + [Rock Paper Scissors | LeetPython](https://www.leetpython.com/docs/Projects/beginner/rock-paper-scissors)
  + [Python Tkinter tutorial: Build a Rock-Paper-Scissors game](https://www.educative.io/blog/python-tkinter-tutorial-rock-paper-scissors)