ROCK, PAPER AND SCISSOR GAME (DECISION MAKING GAME)

Institute name: Besant Technologies

Submitted By: Bhagyashree B

Submitted To: Gowthami Shyam mam

Coordinator Name: Doorika

Roll No:

Date of Submission

### ****Acknowledgement****

I would like to express my sincere gratitude to all those who have supported and guided me during the completion of my project titled **“Rock, Paper, Scissors Simulation using Python Programming.”**

I am deeply thankful to my course instructor, **Miss Gowthami**, my project guider, **Ms**.**Durika**, and the organizer, **Mr.Mohamed Ismail**, for their valuable guidance, encouragement, and continuous support throughout this project. Their mentorship has greatly enhanced my learning experience and understanding of Python programming.

I am also grateful to **Besant Technologies** for providing me with the opportunity and resources to work on this project as part of my course completion.

Your Faithfull

Bhagyashree B

## ****Table of Contents****

| **S.No** | **Title / Section** | **Page No.** |
| --- | --- | --- |
| 1 | Cover page | 1 |
| 2 | Acknowledgement | 2 |
| 3 | Table of content | 3 |
| 4 | Introduction to python | 4 |
| 5 | Introduction of project | 5 |
| 6 | Concepts used in Project | 6 |
| 7 | Source code | 7 |
| 8 | Description about code | 8 |
| 9 | Screenshots of output | 9 |
| 10 | Conclusion | 10 |
| 11 | Bibliography | 11 |
| 12 | Git hub | 12 |

**INTRODUCTION TO PYTHON**

Python is a **high-level, interpreted, and general-purpose programming language** that has gained immense popularity due to its simplicity, readability, and versatility. Developed by **Guido van Rossum** in **1991**, Python emphasizes code readability and allows developers to express concepts in fewer lines of code compared to other programming languages.

Python supports multiple programming paradigms, including **procedural, object-oriented, and functional programming**, making it suitable for a wide range of applications—from web development and data science to artificial intelligence and game development. It comes with an extensive **standard library** and a large ecosystem of third-party packages that simplify complex tasks, such as file handling, mathematical computations, and graphics.

One of Python’s key features is its **ease of use for beginners**. Its clean syntax allows learners to focus on programming concepts rather than the complexity of the language itself. Additionally, Python is platform-independent; meaning programs written in Python can run on **Windows, macOS, Linux**, and even mobile platforms with minimal modifications.

In this project, Python is used to develop a **Rock, Paper, Scissors game**, which is a simple, interactive console-based game. The project demonstrates Python’s ability to handle **user input**, **conditional logic**, **loops**, and **randomization** efficiently. By using Python, the game can simulate a real-time interactive experience between the user and the computer, making it an ideal choice for learning programming fundamentals and logic implementation.

The Rock, Paper, Scissors project not only reinforces the concepts of **control flow, decision-making, and randomness in Python**, but also provides a foundation for developing more complex games and applications in the future. This project serves as an educational tool to help beginners gain practical experience in programming while enjoying the process of building a fun and interactive game

### ****Introduction to the Project****

The **Rock, Paper, Scissors** game is a simple yet classic hand game played worldwide. Traditionally, it involves two players who simultaneously select one of three options: **rock, paper, or scissors**. Each option defeats one of the others while being defeated by another, creating a cyclic balance: rock beats scissors, scissors beat paper, and paper beats rock. Although the rules are simple, the game is widely used for decision-making, conflict resolution, and entertainment.

With the advancement of computer technology, traditional games like Rock, Paper, Scissors have been adapted into **digital formats**, providing a fun and interactive way for users to engage with the game. This project focuses on developing a **console-based Rock, Paper, Scissors game using Python programming**, which allows a single user to play against the computer. The computer’s choices are generated randomly, creating an unpredictable and challenging opponent for the user.

#### ****Background****

Python is an ideal language for developing this project due to its simplicity, readability, and support for fundamental programming concepts. As a high-level language, Python allows beginners to focus on **logic and problem-solving** rather than complex syntax. It supports multiple programming paradigms, including procedural and object-oriented programming, making it flexible for a variety of applications.

Developing a game like Rock, Paper, Scissors in Python helps learners understand core programming concepts such as **variables, conditional statements, loops, and randomization**. Additionally, it provides a practical introduction to **algorithmic thinking**, which is essential for more advanced programming and software development projects.

#### ****Purpose of the Project****

The primary purpose of this project is to **create an interactive, user-friendly game** that demonstrates the application of Python programming in real-world scenarios. Specifically, the project aims to:

* Allow users to play multiple rounds against the computer.
* Implement randomization to simulate unpredictable computer choices.
* Track the outcome of each round (win, lose, or draw) and provide feedback to the player.
* Reinforce fundamental programming skills such as **decision-making, loops, and handling user input**.

Through this project, learners can **apply theoretical knowledge in a practical setting**, strengthening their understanding of Python while building a complete, functional program.

#### ****Scope of the Project****

This project focuses on a **single-player game** where the user competes against a computer. The game is implemented as a **console application**, making it accessible and easy to run on any platform with Python installed. While the current project focuses on the basic gameplay mechanics, it lays the groundwork for **future enhancements**, such as:

* Graphical User Interface (GUI) implementation using libraries like Tkinter or Pygame.
* Scoreboards and game statistics tracking.
* Multiplayer support over networks.
* Advanced AI strategies for computer moves.

#### ****Significance of the Project****

The project is significant for beginners and intermediate Python learners because it provides:

* **Practical experience** in coding, debugging, and running Python programs.
* **Understanding of control structures** like loops and conditional statements in real applications.
* A **fun and interactive learning environment**, motivating learners to experiment and improve their coding skills.

Additionally, this project demonstrates how programming can be used to **digitize traditional games**, providing a foundation for creating more complex interactive applications in the future. By combining **algorithmic logic with user interaction**, this project serves as a valuable educational tool that bridges theory and practical application.

## ****Concepts Used in the Project****

The **Rock, Paper, Scissors Simulation using Python Programming** project is built using several key programming concepts that form the foundation of Python. These concepts make the program interactive, logical, and efficient. Understanding them helps beginners grasp how programming languages translate human logic into working software.

In this project, we make use of fundamental Python concepts such as **variables, data types, control structures, loops, functions, randomization, input/output, and decision-making**. Each concept plays a specific role in the functioning of the game. Below is a detailed explanation of these concepts and how they are used.

### ****1. Variables and Data Types****

In programming, **variables** act as containers that store values that can be used and modified throughout the program. They make the program flexible and reusable.

In Python, variables do not require explicit declaration; the interpreter automatically assigns a data type based on the value. This dynamic typing feature simplifies programming and reduces code length.

In the Rock, Paper, Scissors project, variables are used to store:

* The **user’s choice**
* The **computer’s random choice**
* The **result** of each round

For example:

user\_choice = input("Enter your choice (rock, paper, scissors): ")

computer\_choice = random.choice(['rock', 'paper', 'scissors'])

Here:

* user\_choice and computer\_choice are **string variables**.
* The input() function captures user input.
* The random.choice() method generates the computer’s selection.

Python supports multiple **data types** such as int, float, str, bool, and list. In this project, **string data types** are mostly used to store textual data like “rock”, “paper”, and “scissors.”

Variables allow easy comparison and manipulation, helping in decision-making processes later in the program.

### ****2. Conditional Statements (if-elif-else)****

**Conditional statements** are used to make logical decisions. They check whether a particular condition is true or false, and based on that, execute a block of code.

In the Rock, Paper, Scissors project, conditions are used to determine who wins each round. For example, the program checks whether:

* Both user and computer made the same choice → **Draw**
* User’s choice beats the computer’s → **User Wins**
* Otherwise → **Computer Wins**

Example:

if user\_choice == computer\_choice:

print("It's a draw!")

elif (user\_choice == 'rock' and computer\_choice == 'scissors') or \

(user\_choice == 'scissors' and computer\_choice == 'paper') or \

(user\_choice == 'paper' and computer\_choice == 'rock'):

print("You win!")

else:

print("Computer wins!")

Here, **comparison operators** (==) and **logical operators** (and, or) are used together to check multiple conditions.

Conditional statements form the **core decision-making logic** of the game. Without them, the program would not be able to decide the outcome.

### ****3. Loops****

A **loop** allows the program to execute a block of code repeatedly until a condition is met. Loops are especially useful in games, where actions often need to be repeated multiple times.

In this project, a **while loop** is used to allow the user to play multiple rounds without restarting the program. The loop continues until the player chooses to quit.

Example:

while True:

# Game logic here

play\_again = input("Do you want to play again? (yes/no): ")

if play\_again.lower() != 'yes':

break

Explanation:

* The while True loop ensures continuous execution.
* The loop breaks only when the user inputs “no.”
* The break statement is used to exit the loop.

This concept adds **interactivity and flow control**, enabling multiple rounds of gameplay in one run.

### ****4. Randomization****

Randomness is a key component of this game. In real life, players make unpredictable moves; to mimic this in programming, Python’s **random module** is used.

The **random.choice()** function selects one item randomly from a list of possible moves — “rock”, “paper”, or “scissors.”

Example:

import random

computer\_choice = random.choice(['rock', 'paper', 'scissors'])

Each time the game runs, the computer’s choice changes, making the game unpredictable and fair. This demonstrates how randomness can simulate human-like behavior in programs.

Randomization is also widely used in **AI, simulations, testing, and game development**, making it an important concept for beginners to learn.

### ****5. Functions****

Functions are reusable blocks of code designed to perform specific tasks. They improve **code organization**, **readability**, and **modularity**.

In this project, a function such as play\_game() can be used to handle a single round of the game. This prevents code duplication and allows easy updates or debugging.

Example:

def play\_game():

user\_choice = input("Enter your choice: ")

computer\_choice = random.choice(['rock', 'paper', 'scissors'])

# determine winner here

Using functions makes the program more structured. It also demonstrates the principle of **modularity**, which is an important aspect of software design.

### ****6. User Input and Output****

Interaction between the program and user happens through **input and output** functions.

* The input() function takes the user’s move.
* The print() function displays results and prompts.

Example:

user\_choice = input("Enter your choice (rock, paper, scissors): ")

print("Computer chose:", computer\_choice)

These functions allow real-time feedback, which is essential in interactive applications like games. Proper user interaction improves **usability and engagement**.

### ****7. String Comparison and Logical Operators****

In this project, **string comparison** is used to check whether the user’s and computer’s choices match. Python’s comparison (==) and logical operators (and, or, not) help implement multiple conditions efficiently.

Example:

if (user\_choice == 'rock' and computer\_choice == 'scissors') or \

(user\_choice == 'scissors' and computer\_choice == 'paper') or \

(user\_choice == 'paper' and computer\_choice == 'rock'):

print("You win!")

Logical operators combine multiple conditions, allowing the program to make **complex decisions** in a single statement.

### ****8. Control Flow****

Control flow defines the **order in which code executes**. In this project, the program follows a clear and logical sequence:

1. Display a welcome message.
2. Accept user input.
3. Generate computer’s random move.
4. Compare choices using conditions.
5. Display the result.
6. Ask if the user wants to play again.

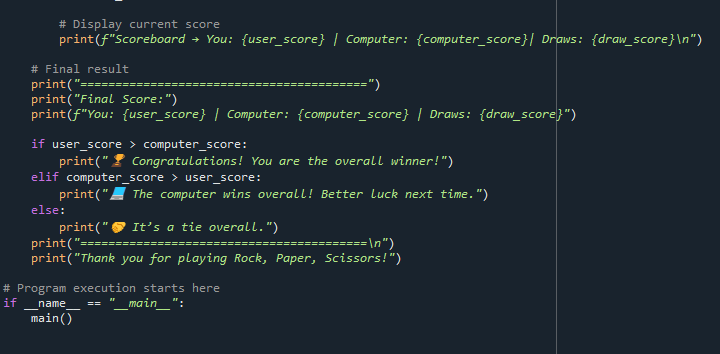
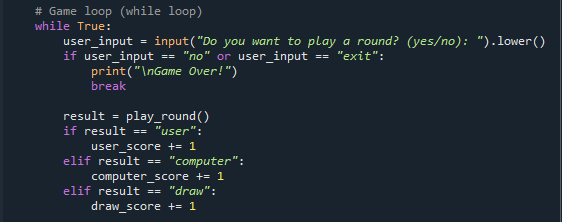
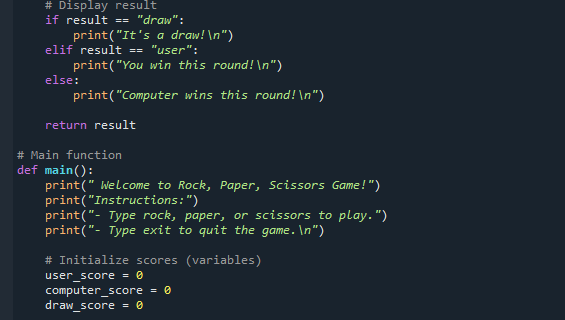
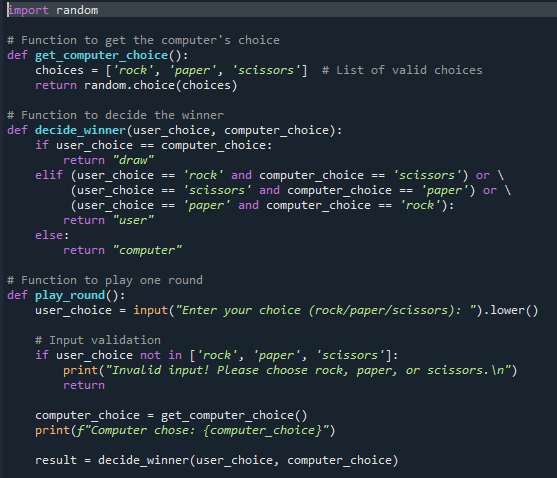
This structured flow ensures that each step executes in the right order, creating a smooth gaming experience.

### ****Conclusion****

The **Rock, Paper, Scissors Simulation using Python Programming** project applies several essential programming concepts — including variables, loops, conditionals, randomization, and user interaction — to create an engaging and interactive experience.

By using these core concepts together, the project not only strengthens understanding of Python basics but also demonstrates how logic and creativity can combine to build a functional application. This project serves as an excellent starting point for anyone looking to explore programming through practical, hands-on learning.

**Source Code:**

****

## ****Program Description: Rock, Paper, Scissors Game.****

This project is a simple yet interactive **Rock, Paper, Scissors game** developed using the **Python programming language**. The goal of the project is to demonstrate the use of **basic programming concepts** such as loops, functions, conditional statements, user input, randomization, and variables in a real-world mini project. It allows a user to play the classic Rock, Paper, Scissors game against the computer, keeping track of scores and declaring the overall winner.

### ****Overview of the Game****

Rock, Paper, Scissors is a popular hand game usually played between two people. The rules are straightforward:

* **Rock beats Scissors** (Rock crushes Scissors)
* **Scissors beats Paper** (Scissors cut Paper)
* **Paper beats Rock** (Paper covers Rock)

In this project, one player is the **user** and the other is the **computer**. The computer’s choice is generated randomly using Python’s built-in **random module**. After each round, the program decides the winner, displays the results, and keeps updating the score. The user can play multiple rounds and can exit the game anytime.

### ****Main Objectives****

The main objectives of this program are:

* To implement a **menu-driven Python program** that interacts with the user.
* To apply **decision-making** and **looping** constructs to manage the flow of the game.
* To demonstrate **randomization** for computer’s choices.
* To maintain and display scores after every round.
* To understand modular programming using **functions**.

### ****Explanation of the Code****

1. **Importing Required Module**  
   The program starts with importing the **random module**, which provides functions to generate random values.

import random (..code)

This module is essential for allowing the computer to make unpredictable choices among rock, paper, or scissors.

1. **Function for Computer’s Choice**  
   The function get\_computer\_choice() creates a list of valid moves (rock, paper, scissors) and uses random.choice() to select one randomly.  
   This ensures that every time the user plays, the computer’s move changes unpredictably.
2. **Decision Function**  
   The function decide\_winner(user\_choice, computer\_choice) compares the user’s input with the computer’s selection and determines the result.
   * If both choices are the same, the round is a draw.
   * Otherwise, the function applies the standard Rock-Paper-Scissors rules to decide who wins.  
     This part of the code mainly uses **if-elif-else conditional statements** to implement the logic.
3. **Playing One Round**  
   The play\_round() function handles one complete round of the game. It:
   * Takes the user’s input,
   * Validates if it is one of the accepted options (rock, paper, scissors),
   * Displays the computer’s random choice,
   * Calls the decide\_winner() function to calculate the result,
   * Displays the outcome of that round.

If the user enters an invalid word, the function notifies them and allows them to try again.  
This introduces **input validation**, which prevents the program from crashing due to incorrect input.

1. **Main Game Logic**  
   The main() function serves as the heart of the program. It:
   * Displays a welcome message and instructions,
   * Initializes score counters (user\_score, computer\_score, and draw\_score),
   * Uses a **while loop** to repeatedly ask the user if they want to continue playing.

Inside the loop, it calls play\_round() for each round and updates the scores based on who wins.  
After each round, the updated scoreboard is displayed, giving the user a real-time view of their performance.

1. **Exiting and Displaying Final Results**  
   When the user chooses to stop playing (by typing "no" or "exit"), the program prints the **final scores** and declares the **overall winner**:
   * If the user’s score is higher, they are declared the winner.
   * If the computer’s score is higher, the computer wins.
   * If both are equal, it’s declared a tie.

Finally, a thank-you message is printed to conclude the game.

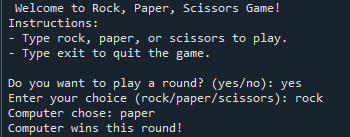
1. **Entry Point of Program**  
   The line

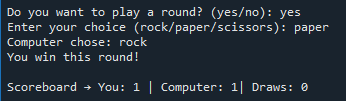
if \_\_name\_\_ == "\_\_main\_\_": (..code)

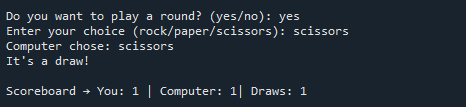
main()

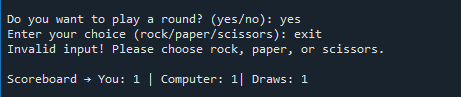
ensures that the program runs only when it is executed directly, not when imported as a module into another Python file.

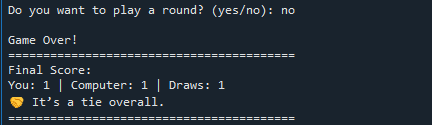
**OUTPUT**



****

****

****

****

## ****Conclusion****

The Rock, Paper, Scissors game project demonstrates how simple logic and basic Python concepts can be combined to create an interactive and enjoyable program. Through this project, we successfully implemented key programming principles such as **functions**, **loops**, **conditional statements**, **variables**, **input/output handling**, and **randomization**.

The program allows users to play multiple rounds against the computer, view their scores, and experience real-time decision-making and result generation. It not only offers a fun gaming experience but also helps in understanding the importance of **structured programming** and **modular code design**.

By completing this project, I have gained practical knowledge of how to:

* Break down a problem into smaller functions,
* Use logical conditions to make decisions,
* Handle user input effectively, and
* Display outputs in a user-friendly format.

This project serves as a strong foundation for learning **game development**, **algorithmic thinking**, and **Python programming**. It highlights how even a simple game can showcase the power and versatility of Python.

Overall, the project was an excellent learning experience that improved both my coding and problem-solving skills. It has inspired me to explore more complex projects and apply similar programming techniques in future applications.