

# **SYNOPSIS**

**Report on**

**Rock-Paper-Scissors Game**

**by**

**TUSHAR CHANDRA PANT (202410116100226)**

**TUSHAR MISHRA(202410116100228)**

**UTKARSH SANRE (202410116100231)**

**SHRESTH YADAV(202410116100201)**

**Session: 2024-2025 (II Semester)**

**Under the supervision of**

**Ms. Komal Salgotra (Assistant Professor)**

**KIET Group of Institutions, Delhi-NCR, Ghaziabad**



**DEPARTMENT OF COMPUTER APPLICATIONS  
KIET GROUP OF INSTITUTIONS, DELHI-NCR,  
GHAZIABAD-201206**

## TABLE OF CONTENTS

	Page Number
1. Introduction	3
2. Methodology	4-6
3. Typed Code	7
4. Output	8-11
5. Conclusion	12

# **Rock-Paper-Scissors Game**

## **Introduction**

The Rock-Paper-Scissors game is a well-known and simple game played between two players, often used as a decision-making tool. It is based on three choices—rock, paper, and scissors—each with a specific interaction that determines the outcome. Rock crushes scissors, scissors cut paper, and paper covers rock. Though simple, this game introduces key programming concepts such as conditional statements, loops, functions, and random number generation, making it an ideal beginner-friendly project.

This project aims to develop a Python-based Rock-Paper-Scissors game where a player competes against a computer. The computer's choice is randomly generated using Python's random module, ensuring unpredictability. The player's input is validated to prevent incorrect entries, creating a smooth and error-free experience. The game logic follows predefined rules to determine the winner and display the result accordingly.

Additionally, this project demonstrates the importance of modular programming by organizing the code into separate functions for user input, computer choice generation, and result determination. This structured approach enhances code readability and allows for future improvements, such as score tracking, multiplayer mode, or a graphical user interface (GUI). By implementing this game, we not only create an interactive program but also explore fundamental programming concepts that can be expanded into more complex applications.

## Methodology

The Rock-Paper-Scissors game is implemented using a modular approach, where the game logic is divided into multiple functions. This structured programming methodology ensures clarity, maintainability, and scalability, making it easier to modify and expand the game in the future. Such an approach allows for easy debugging, enhances readability, and facilitates potential upgrades such as multi-round gameplay, a graphical user interface (GUI), multiplayer mode, and AI-based opponents.

The implementation consists of four major steps:

1. Generating Computer Choice
2. Handling User Input
3. Determining the Winner
4. Executing the Game

Each step plays a crucial role in ensuring the game functions smoothly and provides an interactive user experience.

### 1. Generating Computer Choice

One of the key components of the game is generating the computer's choice, ensuring randomness and fairness.

- The function `get_computer_choice()` is responsible for generating a choice for the computer.
- It utilizes Python's built-in random module, specifically the `random.choice()` function, to select a random option from a predefined list: "rock", "paper", or "scissors".
- The use of randomness guarantees that the computer's selection is unbiased and unpredictable, preventing the user from anticipating the next move.

This randomness simulates real-world decision-making in the game, keeping it engaging and fair for the player. Future enhancements could involve AI-driven decision-making, where the computer learns from previous rounds and adapts its choices accordingly.

## 2. Handling User Input

Ensuring correct and valid input from the player is essential for a smooth and error-free gaming experience.

- The function `get_player_choice()` is responsible for taking user input and validating it.
- A while loop is used to repeatedly prompt the player for input until they enter a valid choice (i.e., "rock", "paper", or "scissors").
- If the user enters an invalid choice, they receive a friendly error message and are prompted to try again.

This mechanism enhances user experience and prevents unintended errors that could otherwise disrupt the gameplay. By enforcing input validation, we eliminate potential game-breaking scenarios caused by incorrect inputs.

## 3. Determining the Winner

Once both the player and computer have made their choices, the program determines the outcome based on predefined rules.

- The function `determine_winner()` takes two parameters: the player's choice and the computer's choice.
- The game follows a simple set of rules:
  - Rock crushes Scissors → Rock wins
  - Scissors cut Paper → Scissors win
  - Paper covers Rock → Paper wins
- If both choices are the same, the game results in a tie.
- A series of if-elif-else statements evaluate the player's choice against the computer's choice and return the appropriate result.

This function ensures quick and efficient decision-making, displaying the result of the round to the player.

## 4. Executing the Game

The main function, `play_game()`, orchestrates the different steps of the game by calling the necessary functions in sequence:

1. The player's choice is obtained using `get_player_choice()`.
2. The computer's choice is generated using `get_computer_choice()`.
3. The winner is determined by calling `determine_winner()`.
4. The result is then displayed to the user.

This function serves as the central controller of the program, ensuring a structured flow of execution.

## Typed Code (Final Implementation)

Below is the complete Python implementation of the Rock-Paper-Scissors game:

```
import random

def get_computer_choice():
    """Generates a random choice for the computer."""
    choices = ["rock", "paper", "scissors"]
    return random.choice(choices)

def get_player_choice():
    """Gets the player's choice and validates input."""
    while True:
        player_choice = input("Enter your choice (rock, paper, or scissors): ").lower()
        if player_choice in ["rock", "paper", "scissors"]:
            return player_choice
        else:
            print("Invalid choice. Please try again.")

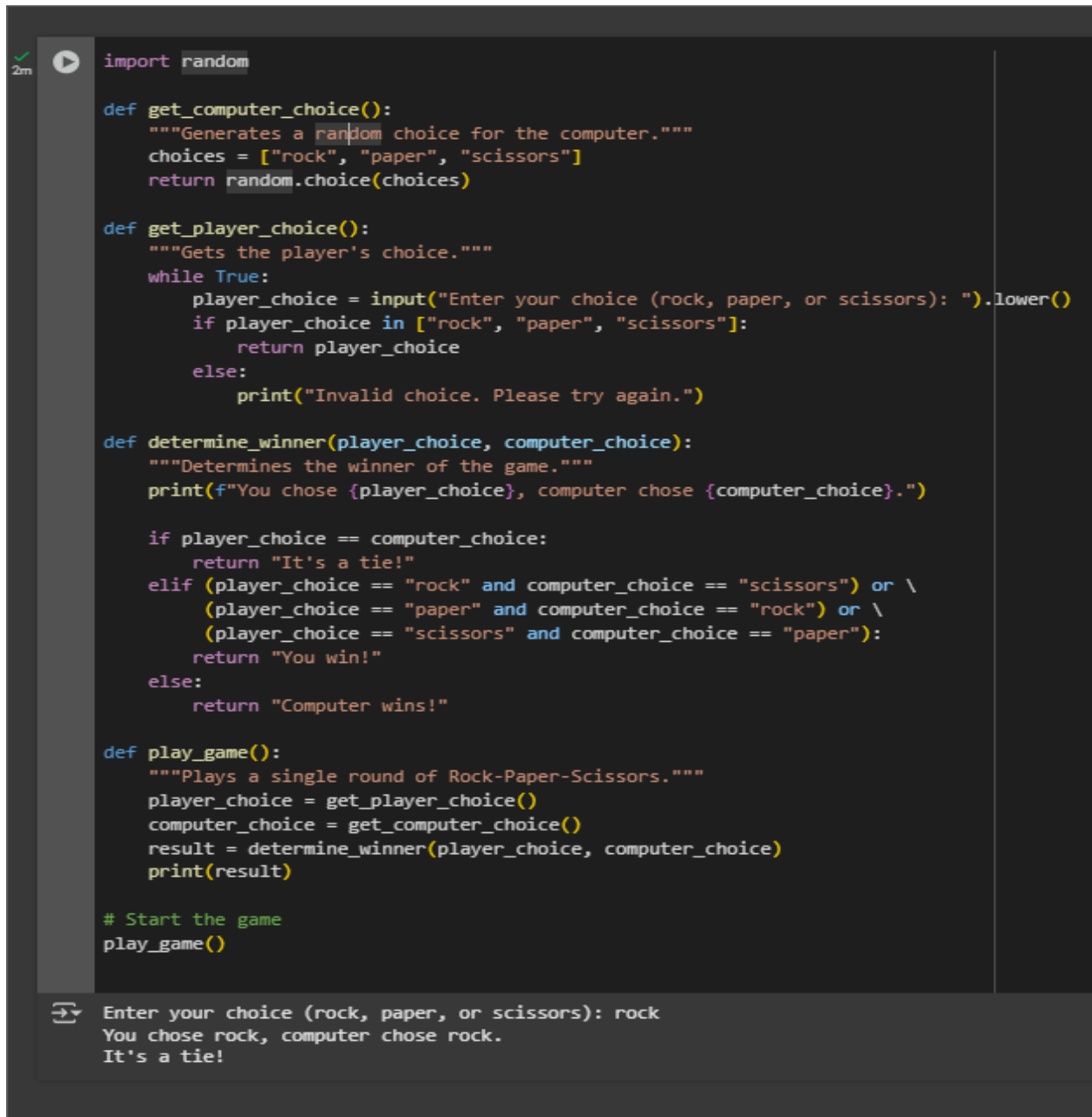
def determine_winner(player_choice, computer_choice):
    """Determines the winner of the game."""
    print(f"You chose {player_choice}, computer chose {computer_choice}.")

    if player_choice == computer_choice:
        return "It's a tie!"
    elif (player_choice == "rock" and computer_choice == "scissors") or \
         (player_choice == "paper" and computer_choice == "rock") or \
         (player_choice == "scissors" and computer_choice == "paper"):
        return "You win!"
    else:
        return "Computer wins!"

def play_game():
    """Plays a single round of Rock-Paper-Scissors."""
    player_choice = get_player_choice()
    computer_choice = get_computer_choice()
    result = determine_winner(player_choice, computer_choice)
    print(result)

# Start the game
play_game()
```

## Output



```
import random

def get_computer_choice():
    """Generates a random choice for the computer."""
    choices = ["rock", "paper", "scissors"]
    return random.choice(choices)

def get_player_choice():
    """Gets the player's choice."""
    while True:
        player_choice = input("Enter your choice (rock, paper, or scissors): ").lower()
        if player_choice in ["rock", "paper", "scissors"]:
            return player_choice
        else:
            print("Invalid choice. Please try again.")

def determine_winner(player_choice, computer_choice):
    """Determines the winner of the game."""
    print(f"You chose {player_choice}, computer chose {computer_choice}.")

    if player_choice == computer_choice:
        return "It's a tie!"
    elif (player_choice == "rock" and computer_choice == "scissors") or \
        (player_choice == "paper" and computer_choice == "rock") or \
        (player_choice == "scissors" and computer_choice == "paper"):
        return "You win!"
    else:
        return "Computer wins!"

def play_game():
    """Plays a single round of Rock-Paper-Scissors."""
    player_choice = get_player_choice()
    computer_choice = get_computer_choice()
    result = determine_winner(player_choice, computer_choice)
    print(result)

# Start the game
play_game()
```

Enter your choice (rock, paper, or scissors): rock  
You chose rock, computer chose rock.  
It's a tie!



The screenshot shows a Jupyter Notebook titled "ROCK\_PAPER.ipynb". The interface includes a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu bar is a "Commands" section with "+ Code" and "+ Text" options. The main area contains a code cell with the following Python code:

```
print("Invalid choice. Please try again.")

def determine_winner(player_choice, computer_choice):
    """Returns the winner of the game."""
    if (player_choice == "rock" and computer_choice == "scissors") or \
        (player_choice == "paper" and computer_choice == "rock") or \
        (player_choice == "scissors" and computer_choice == "paper"):
        return "You win!"
    elif player_choice == computer_choice:
        return "It's a tie!"
    else:
        return "Computer wins!"

def play_game():
    """Plays a single round of Rock-Paper-Scissors."""
    player_choice = get_player_choice()
    computer_choice = get_computer_choice()
    result = determine_winner(player_choice, computer_choice)
    print(result)

# Start the game
play_game()
```

A tooltip is visible over the first line of code, stating: "Run cell (Ctrl+Enter) cell executed since last change", "executed by Tushar", "12:05 AM (0 minutes ago)", and "executed in 7.414s".

Below the code cell, the output is displayed:

```
Enter your choice (rock, paper, or scissors): rock
You chose rock, computer chose scissors.
You win!
```

Commands + Code + Text

5a

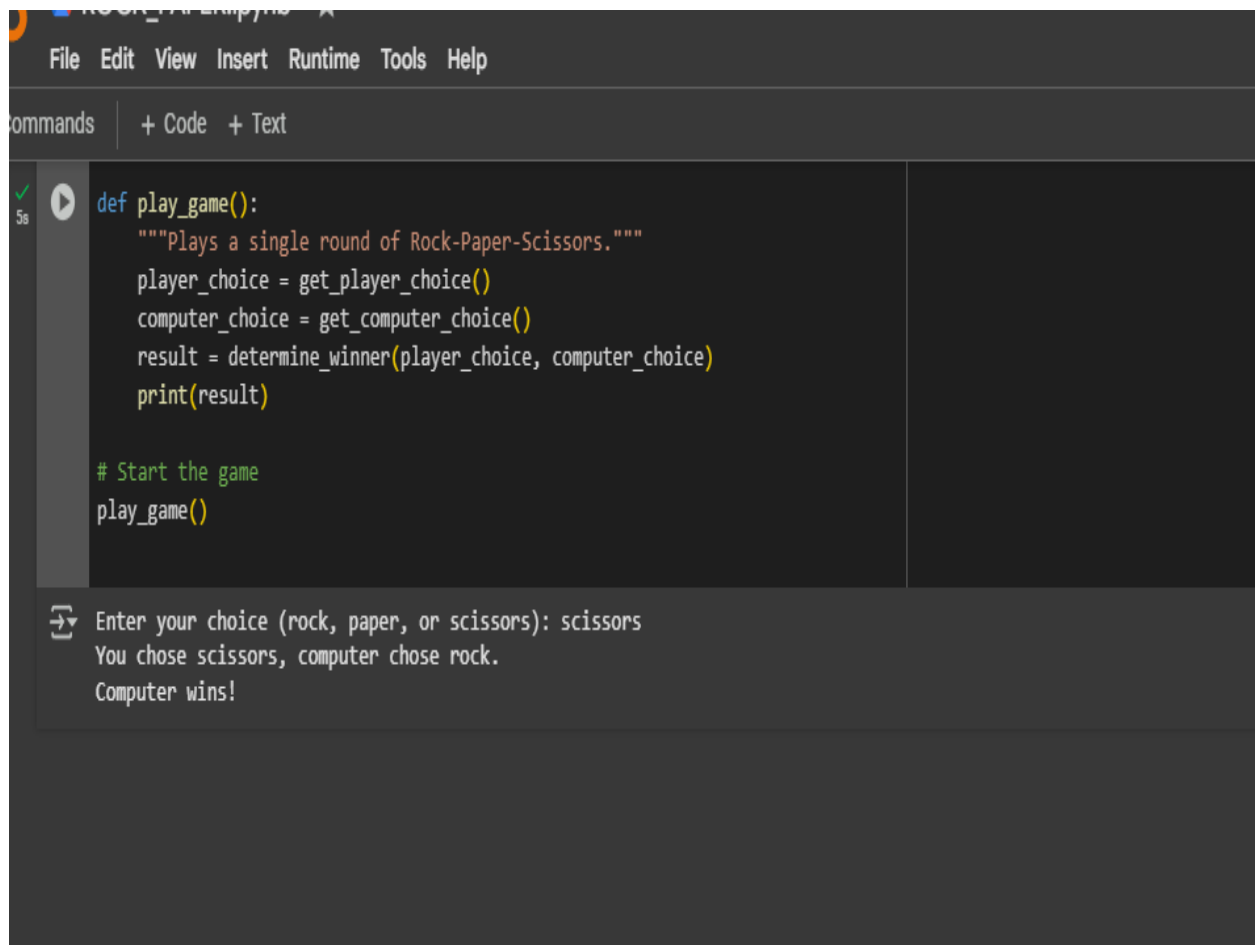
```
def play_game():
    """Plays a single round of Rock-Paper-Scissors."""
    player_choice = get_player_choice()
    computer_choice = get_computer_choice()
    result = determine_winner(player_choice, computer_choice)
    print(result)

# Start the game
play_game()
```

Enter your choice (rock, paper, or scissors): paper

You chose paper, computer chose rock.

You win!



The screenshot shows a Jupyter Notebook interface with a dark theme. The top menu bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the menu is a toolbar with 'Commands', '+ Code', and '+ Text' buttons. The main area contains a code cell with the following Python code:

```
def play_game():  
    """Plays a single round of Rock-Paper-Scissors."""  
    player_choice = get_player_choice()  
    computer_choice = get_computer_choice()  
    result = determine_winner(player_choice, computer_choice)  
    print(result)  
  
# Start the game  
play_game()
```

Below the code cell is a console output area showing the execution results:

```
Enter your choice (rock, paper, or scissors): scissors  
You chose scissors, computer chose rock.  
Computer wins!
```

## Conclusion

The Rock-Paper-Scissors game was successfully implemented using Python, demonstrating the use of functions, loops, conditionals, and randomness. The structured approach ensures readability, maintainability, and scalability for future enhancements. The use of input validation ensures a smooth user experience, while the random module introduces unpredictability, making the game fair and engaging.

Possible improvements include implementing multiple rounds with score tracking to determine an overall winner, making the game more competitive. Adding a Graphical User Interface (GUI) using Tkinter or Pygame could enhance user interaction by replacing text-based input with buttons and animations. A multiplayer mode allowing two users to compete locally or online would further increase engagement. Additionally, incorporating sound effects and animations would create a more immersive experience.

Beyond entertainment, this project serves as a learning tool for beginners, introducing key programming concepts such as functions, loops, and user input handling. It also provides a foundation for exploring advanced topics like game development and AI-based opponents. Expanding on this project allows developers to refine their coding skills and experiment with new features, making it a valuable stepping stone in Python programming and software development.