–--------

## **Project Title:**

## Rock,Paper,Scissors Game

**Introduction :**

This Python program is a fun and interactive implementation of the classic "Rock, Paper, Scissors" game. Designed with simplicity in mind, it allows players to compete against the computer. The game uses basic programming concepts such as loops, conditionals, and random choice generation to decide the computer's move. It serves as a great project for college students to practice and demonstrate their coding skills.

**Methodology Used :**

**Random :**

The random module in Python is a built-in library that provides functions for generating pseudo-random numbers. These numbers aren't truly random in the sense that they are determined by a mathematical algorithm, starting from an initial value called the seed. However, the sequences they generate appear random enough for many applications. This is because the algorithms are designed to produce sequences with statistical properties that match true randomness.

Here's a breakdown of key aspects:

• Pseudo-randomness: The numbers produced are not truly random; they are deterministic. Given the same seed value, you'll get the same sequence of numbers. This can be useful for reproducibility in simulations or testing.

• Seed: The random.seed() function allows you to set the initial value (seed) for the random number generator. This enables you to control the sequence generated, ensuring reproducibility of your results. If you don't set a seed, Python uses the system's current time as a default seed, so each run will be different.

• Common Functions: The random module offers a variety of functions, including:

\* random(): Returns a random floating-point number between 0.0 (inclusive) and 1.0 (exclusive).

\* randint(a, b): Returns a random integer N such that a <= N <= b.

\* randrange(start, stop[, step]): Returns a randomly selected element from range(start, stop, step).

\* uniform(a, b): Returns a random floating-point number N such that a <= N <= b.

\* choice(seq): Returns a randomly chosen element from a non-empty sequence.

\* sample(population, k): Returns a list of k unique elements chosen from the population sequence.

\* shuffle(x[, random]): Shuffles the elements of a list in place.

\* random.choices(population, weights=None, \*, cum\_weights=None, k=1): Allows you to select multiple random elements from a population, potentially with weights to control probabilities of selecting specific elements.

• Applications: The random module is used extensively in:

\* Simulations: Modeling random events (e.g., simulating dice rolls, stock prices, weather patterns).

\* Games: Generating random numbers for game elements (e.g., card shuffling, enemy placement).

\* Statistics: Generating random samples, performing Monte Carlo simulations, and hypothesis testing.

\* Machine Learning: Shuffling data, initializing weights in neural networks, and performing random search for hyperparameter optimization.

While the numbers generated by random are not truly random, they are sufficiently unpredictable for most applications. If true randomness is absolutely critical (e.g., cryptography), you might need to use other methods (like accessing a hardware-based random number generator).

**Boolean Method :**

In Python, "Boolean method" isn't astandard term. Instead, we talk about Boolean values and methods that /. return Boolean values.

• Boolean Values: These are a data type representing truth or falsehood, with two possible values: True or False (note the capitalization). Boolean values are fundamental to conditional statements and logical operations in Python.

• Methods Returning Boolean: Many methods in Python, particularly on strings, lists, and other objects, return Boolean values. These methods often test for a condition or presence of a specific element.

\* Example (String methods):

\* str.startswith(prefix): Returns True if the string starts with the specified prefix, False otherwise.

\* str.endswith(suffix): Returns True if the string ends with the specified suffix, False otherwise.

\* str.isalpha(): Returns True if all characters in the string are alphabetic, False otherwise.

\* str.isdigit(): Returns True if all characters are digits, False otherwise.

\* Example (List methods):

\* list.\_contains\_(element): Returns True if the list contains the specified element, False otherwise. This is used when you check element in list.

\* Example (Comparison Operators):

\* a == b, a != b, a > b, a < b, a >= b, a <= b: These operators return True or False depending on the truthiness of the expression.

In Summary:

Instead of a "Boolean method," understand that Python utilizes:

1. Boolean Values (True and False) to represent truth and falsehood.

2. Methods that return Boolean values to test conditions and perform logical checks. These methods are extensively used for controlling program flow based on conditions.

**Software Requirements :**

**Python :**

Python is a high-level, general-purpose programming language known for its clear syntax and readability. It's designed to be easy to learn and use, making it a popular choice for beginners and experienced programmers alike. Here's a breakdown of its key features and characteristics:

Key Features:

• Readability: Python emphasizes code readability with its clean syntax, using indentation to define code blocks instead of curly braces ({}) or keywords like begin and end found in other languages. This makes Python code easier to read, understand, and maintain.

• Interpreted Language: Python code is executed line by line by an interpreter, unlike compiled languages where the entire code is translated into machine code before execution. This makes development and debugging faster, as errors can be identified and corrected more quickly.

• Dynamically Typed: You don't need to explicitly declare the data type of a variable. Python infers the type at runtime. This simplifies coding but can sometimes lead to runtime errors if types are not handled carefully.

• Object-Oriented Programming (OOP): Python supports OOP principles, allowing you to structure code using classes and objects. This enables modularity, reusability, and better organization of large codebases.

• Extensive Libraries: Python boasts a vast collection of libraries (pre-written code modules) that provide functionalities for various tasks, including web development, data science, machine learning, scientific computing, and more. This drastically reduces development time and effort. Popular libraries include NumPy, Pandas, Scikit-learn, TensorFlow, and PyTorch.

• Cross-Platform: Python code can run on various operating systems (Windows, macOS, Linux) with minimal or no modifications. This enhances portability and reduces platform-specific issues.

• Large and Active Community: Python has a massive and supportive community of developers, providing ample resources, tutorials, and assistance for learners and experienced users.

Common Uses:

• Web Development: Python frameworks like Django and Flask are used to build web applications.

• Data Science and Machine Learning: Python's extensive libraries (NumPy, Pandas, Scikit-learn) make it a dominant language in this field.

• Scientific Computing: Python is widely used for scientific simulations, data analysis, and visualization.

• Scripting and Automation: Python is often used to automate repetitive tasks and streamline workflows.

• Game Development: Game engines like Pygame allow for creating 2D games.

• Desktop Applications: Python can be used to create GUI applications.

Advantages:

• Easy to learn and use.

• Readable and maintainable code.

• Large and supportive community.

• Extensive libraries for diverse applications.

• Cross-platform compatibility.

Disadvantages:

• Can be slower than compiled languages.

• Global Interpreter Lock (GIL) can limit performance in multithreaded applications.

• Dynamic typing can lead to runtime errors.

In summary, Python's ease of use, extensive libraries, and strong community make it a powerful and versatile language suitable for a wide range of applications. Its readability and ease of learning contribute to its popularity among beginners, while its power and libraries attract experienced developers.

**Google Colab :**

Google Colab, short for Google Colaboratory, is a free online platform provided by Google that allows you to write and execute Python code using Jupyter Notebooks. It's particularly well-suited for machine learning, data science, and education because it offers several key advantages:

Key Features and Benefits:

• Free Access to Resources: Colab provides free access to computational resources, including powerful CPUs, GPUs, and even TPUs (Tensor Processing Units). This is especially valuable for tasks that require significant computing power, such as training large machine learning models, which can be expensive on personal computers.

• Jupyter Notebook Environment: Colab uses the Jupyter Notebook interface, a web-based interactive computing environment. This allows you to combine code, text, images, and other media in a single document, making it ideal for sharing your work, documenting your code, and creating interactive tutorials.

• Easy Setup and Use: Colab requires minimal setup. You can start using it immediately with a Google account. There's no need to install Python or any libraries locally; Colab handles all the dependencies.

• Collaboration Features: Colab facilitates collaboration with others. Multiple users can work on the same notebook simultaneously, enabling teamwork and shared learning.

• Seamless Integration with Google Services: Colab integrates easily with other Google services like Google Drive, allowing you to easily save and load your notebooks, share them with others, and access data stored in Google Drive.

• Pre-installed Libraries: Many popular Python libraries for data science and machine learning (e.g., NumPy, Pandas, TensorFlow, PyTorch) are pre-installed, saving you the time and effort of setting up your environment.

Who Uses Google Colab?

Colab is used by a wide range of individuals and organizations, including:

• Students and Educators: It's a great platform for learning and teaching data science and machine learning.

• Researchers: It provides convenient access to computing resources for conducting research.

• Data Scientists and Machine Learning Engineers: It's used for prototyping, experimentation, and deploying models.

• Developers: It can be used for various Python programming tasks.

Limitations:

While Colab offers many advantages, it does have some limitations:

• Resource Limits: While it provides free resources, there are limits on the amount of time you can use the GPUs and TPUs, and there might be occasional queueing times depending on availability.

• Internet Dependency: Because it's a cloud-based service, you need an internet connection to use Colab.

• Limited Customization: Compared to a locally installed Jupyter Notebook environment, you have less control over the system settings and configurations.

Despite these limitations, Google Colab remains a powerful and accessible tool for anyone working with Python code, especially in the fields of data science and machine learning.

**Program Code :**

* **Aim**
* **Algorithm**
* **Program**
* **Output**

**Aim:**

To create a program for rock,paper,scissor game using python

**Algorithm:**

Step-1 : start

Step-2 : input the random library

Step-3 : create a list for rock, paper, scissor

Step-4 : create a loop function

Step-5 : create a variable for none

Step-6 : create a variable for random choice using random.choice()

Step-7 : create nested while write a syntax for player input

Step-8 : define if class if player equal to computer the game has been tie

Step-9 : if player not equal to computer the player will be win otherwise computer will be win

Step-10 : create a loop function for play again if our input is y the program will be run again if our input is n the program will be end

Step-11 : define a print function thanks for playing

Step-12 : end

**Program :**

import random

options=("rock","paper","scissors")

player=None

computer=random.choice(options)

running=True

while running:

while player not in options:

player=input("enter a choice(rock,paper,scissors):")

print(f"player:{player}")

print(f"computer:{computer}")

if player==computer:

print("its a tie!")

elif player=="rock" and computer=="scissors":

print("you win!")

elif player=="paper" and computer=="rock":

print("you win!")

elif player=="scissors" and computer=="paper":

print("you win!")

else:

print("you lose!")

if not input("play again?(y/n):").lower()=="y":

running='false'

print("thanking for playing!")

**Output:**

enter a choice(rock, paper, scissor):rock

Player : rock

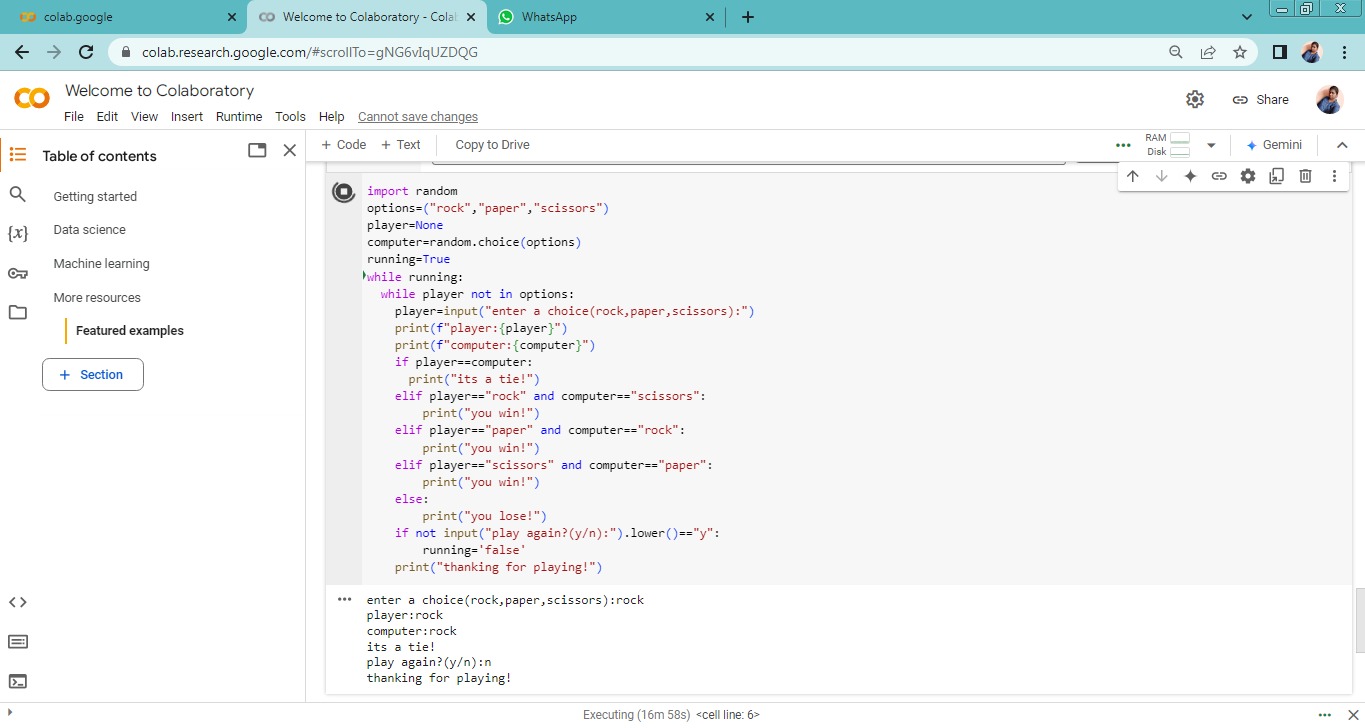
Computer : rock

It's a tie!

Play again?(y/n) : n

Thanks for playing!

**Output Screenshot :**



**Future Outcomes :**

As a college student, we can further enhance this project by introducing new features and expanding its scope. Possible improvements include:

Scoreboard: Add a feature to display the scores after each round to make the game more engaging.

Replay Option: Modify the replay functionality to provide a cleaner user experience, such as displaying a summary after multiple rounds.

GUI Integration: Use libraries like Tkinter or Pygame to create a graphical interface, turning this project into a visually appealing desktop application.

Mobile-Friendly Version: Explore frameworks like Kivy to convert this game into a mobile application.

Two-Player Mode: Add functionality for two users to play against each other on the same system.

AI Improvements: Implement AI logic for the computer's move to simulate different levels of difficulty.

Educational Tool: Use this game as a base to learn and practice other programming concepts like file handling (e.g., saving game history) and object-oriented programming.

**Conclusion :**

This "Rock, Paper, Scissors" game is a simple project showcasing Python basics like loops, conditionals, and user input. Enhancing it with features like score tracking or a GUI can make it more engaging and a great addition to your programming portfolio.

**-Done By Our Team**

**Team Members :**

* **Swathy.R**
* **Kavishree.V**
* **Kishore Kumar.B**
* **Kugan.R**

**BCA-Artificial Intelligence & Data Science**

**Department of computer application**