PROJECT TITLE

ROCK, PAPER ,SCISSOR GAME

OVERVIEW OF PROJECT

The Rock, Paper, Scissors (RPS) project is a classic beginner coding assignment in which players and computers select between three hand gestures—rock, paper, and scissors—with outcomes determined by simple rules

Features

User Input & Validation: Players choose rock, paper, or scissors, with code ensuring valid selection and prompting for mistakes.

Randomized Computer Move: The computer picks its move randomly for unpredictability, simulating real gameplay dynamics.

Game Logic: Implements the winner/tie rules: rock beats scissors, scissors beat paper, paper beats rock, and identical choices yield a draw.

Outcome Display: The result of each round is shown, specifying whether the player, computer, or neither won.

Technologies/tools used:

Rock, Paper, Scissors games are small projects that can use different technologies depending on whether they are console, GUI, web, or mobile based. Commonly they involve a programming language, a random library, and sometimes UI or hardware tools.

1.Programming Languages

Python: Very popular for beginner RPS projects; uses the random module for computer moves and sometimes Tkinter for GUI.

C / C++ / Java: Used for console-based or simple GUI versions, often in academic assignments and mini projects.

JavaScript: Used for browser-based RPS games with HTML and CSS for layout and styling.
Libraries and Modules

2.Randomization: random module in Python, rand() in C/C++, or Math.random() in JavaScript to generate the computer's choice.

3.GUI Libraries: Tkinter or Pygame in Python, Java Swing/JavaFX in Java, or browser DOM APIs for clickable buttons and visual feedback.

Development Tools and IDEs

Code Editors/IDEs: Tools like PyCharm, VS Code, IDLE, or other IDEs are commonly used to write and run the project.

Browser & DevTools: For web RPS, a modern browser plus its developer tools are used to test and debug HTML/CSS/JavaScript.

4.Web & Mobile Platforms

HTML/CSS/JavaScript Stack: For interactive web versions hosted on sites or used in tutorials.

App Inventor / Mobile Tools: Some RPS games are built as Android apps using MIT App Inventor, which provides blocks-based coding and simple mobile components.

Hardware / Special Variants (Optional)

Microcontrollers: Arduino or micro:bit versions use sensors, buttons, and LEDs to display moves and outcomes.

Simulation Platforms: Tinkercad Circuits or similar platforms are used to simulate hardware-based RPS circuits before physical implementation.

Steps to install

Steps to install&run the project

Step-by-Step Guide

1. Install Python

Download and install Python 3 from the official website (python.org).

Confirm installation by running python --version in your terminal.

2. Download Project Code

Obtain the project code from a repository (GitHub link or similar), or copy from a tutorial.

Save the files in a directory on your computer.

3. Install Required Libraries

For basic terminal projects, only the built-in random module is required.

For GUI or advanced versions, install additional packages using pip:

Tkinter (often included with Python on most platforms).

Pygame (pip install pygame).

Any other library mentioned in the project (e.g., PIL for image handling: pip install Pillow).

4. Run the Game

Open a terminal or command prompt.

Navigate to the project directory using cd path/to/project.

Run the main script (e.g., python rock_paper_scissors.py).

For GUI projects, a window should pop up; for console projects, you will be prompted for input.

Example: Basic Python Console Version

bash

```
# Download or clone code to your folder

# No extra installs required for random or input

python rock_paper_scissors.py
```

Example: GUI Version

bash

```
pip install Pillow pygame

python rock_paper_scissors_gui.py
```

If using advanced features (gesture recognition, AI), install OpenCV (pip install opencv-python) or TensorFlow (pip install tensorflow) as instructed by the project's documentation.

For web or mobile versions, follow creation and build instructions using HTML/CSS/JavaScript stacks or app platforms.

These steps provide everything needed to install and run a Rock, Paper, Scissors project for most popular implementations, whether console-based or graphical.

Instruction for testing

Step-by-Step Testing Instructions

1. Verify Correct Inputs and Game Logic

Run the game and play several rounds, choosing "rock," "paper," or "scissors" each time.

Ensure the game correctly announces the result (win, lose, draw) according to the rules: rock beats scissors, scissors beat paper, paper beats rock.

Confirm random computer choices appear balanced over multiple plays.

2. Test Input Validation

Enter invalid inputs (e.g., "spoon," numbers, blank) to check if the game correctly prompts again or displays an error message.

Confirm the game doesn't crash with unexpected or wrong input types.

3. Check Scorekeeping (if implemented)

After multiple rounds, verify that scores (player and computer wins, draws) increment accurately with each round.

Confirm score resets or game-over logic if available.

4. Repeatability and Flow

Play until asked whether you want to replay or exit; select replay to ensure the game loops correctly and restarting works without issues.

Select exit to verify that the game thanks the user and terminates gracefully.

5. Edge and Boundary Cases

Rapidly input choices or submit empty/whitespace input to check robustness.

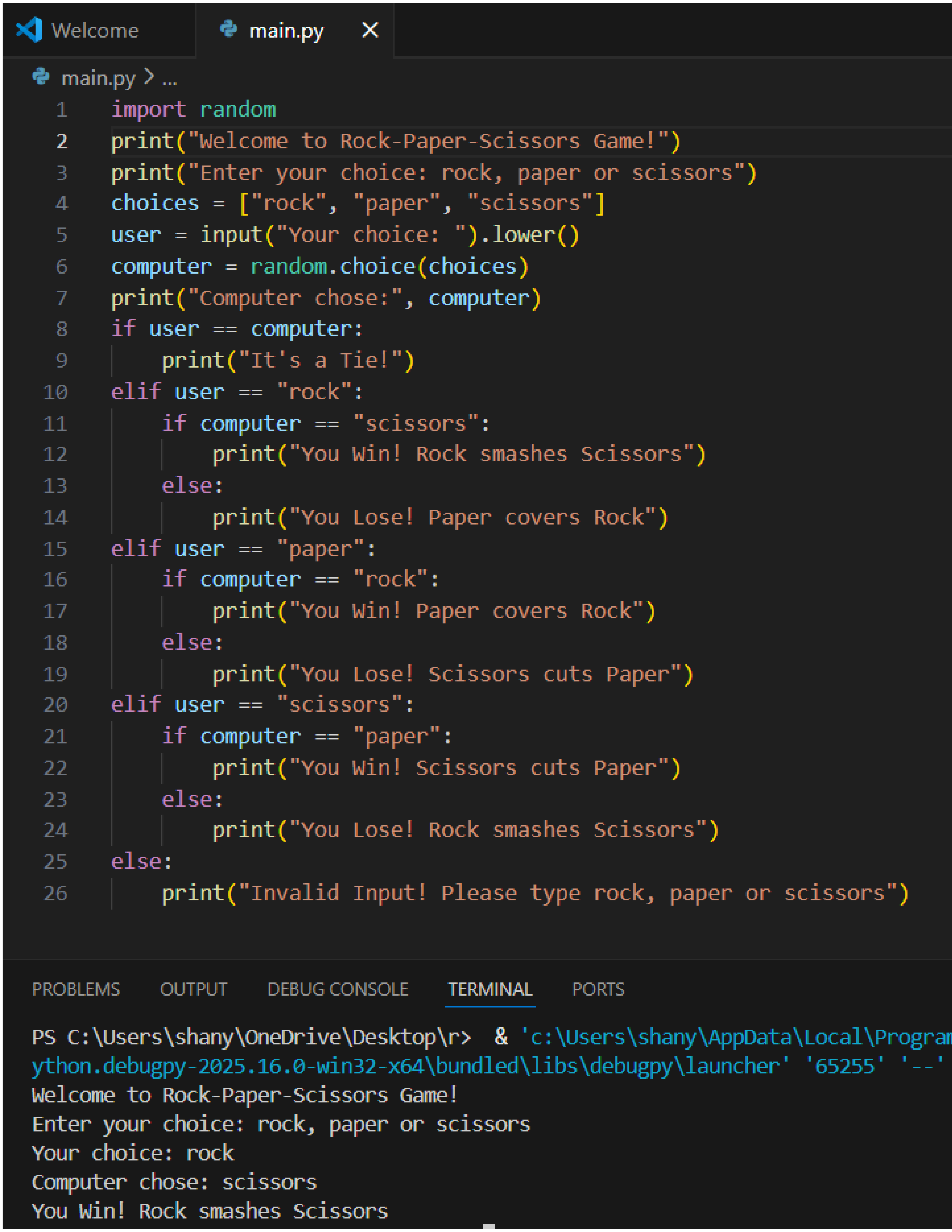
For multiplayer or GUI versions, test rapid button presses, fast game cycling, or switching players to ensure there are no bugs.

Additional Testing Tips

For GUI games, test all interface buttons, window resizing, and input boxes.

Review code or use automated test scripts for unit testing individual functions, like determining the winner given two choices.

screenshots



```python
import random
print("Welcome to Rock-Paper-Scissors Game!")
print("Enter your choice: rock, paper or scissors")
choices = ["rock", "paper", "scissors"]
user = input("Your choice: ").lower()
computer = random.choice(choices)
print("Computer chose:", computer)
if user == computer:
    print("It's a Tie!")
elif user == "rock":
    if computer == "scissors":
        print("You Win! Rock smashes Scissors")
    else:
        print("You Lose! Paper covers Rock")
elif user == "paper":
    if computer == "rock":
        print("You Win! Paper covers Rock")
    else:
        print("You Lose! Scissors cuts Paper")
elif user == "scissors":
    if computer == "paper":
        print("You Win! Scissors cuts Paper")
    else:
        print("You Lose! Rock smashes Scissors")
else:
    print("Invalid Input! Please type rock, paper or scissors")
```

```
PS C:\Users\shany\OneDrive\Desktop\r>  & 'c:\Users\shany\AppData\Local\Program
ython.debugpy-2025.16.0-win32-x64\bundled\libs\debugpy\launcher' '65255' '--'
Welcome to Rock-Paper-Scissors Game!
Enter your choice: rock, paper or scissors
Your choice: rock
Computer chose: scissors
You Win! Rock smashes Scissors
```