**Imported Modules**

| **Code** | **Explanation** |
| --- | --- |
| import random | Imports Python’s random module so the program can make random choices (used for the computer’s move). |
| import sys | Imports the sys module so we can call sys.stdout.flush() to force immediate printing (used in the thinking animation). |
| import time | Imports time so the program can pause execution briefly with time.sleep() to create animation/timing effects. |
| from colorama import Fore, Style, init | Imports parts of the colorama package: Fore (text colors), Style (text styles), and init() (for initialization). Used for colored terminal output. |
| init(autoreset=True) | Initializes colorama. autoreset=True means color changes apply only to the printed string and reset automatically after each print. |

**Declared Variables**

| **Code** | **Explanation** |
| --- | --- |
| mylist = ["rock", "paper", "scissor"] | Creates a list of the three valid game choices. |
| wins = 0 | Tracks the number of wins. |
| losses = 0 | Tracks the number of losses. |
| draws = 0 | Tracks the number of draws. |

**Defining “Animation” Function**

def thinking\_animation():  
 print("\nComputer is thinking", end="")  
 for i in range(3):  
 time.sleep(0.3)  
 print(".", end="")  
 sys.stdout.flush()  
 time.sleep(0.3)  
 print("\n")

| **Line** | **Explanation** |
| --- | --- |
| def thinking\_animation(): | Defines a function to simulate computer “thinking.” |
| print("\nComputer is thinking", end="") | Prints the text and keeps the cursor on the same line. |
| for i in range(3): | Repeats the dot printing 3 times. |
| time.sleep(0.3) | Pauses for 0.3 seconds between dots. |
| print(".", end="") | Prints one dot without moving to a new line. |
| sys.stdout.flush() | Forces the dots to appear immediately on screen. |
| time.sleep(0.3) | Adds a short pause after the animation. |
| print("\n") | Moves to the next line after animation ends. |

**Defining the GUI Display Function**

def print\_gui(player, computer, result):  
 print("\n" + "="\*40)  
 print("ROCK–PAPER–SCISSORS".center(40))  
 print("=" \* 40)  
 print(f"You: {player.upper():<10} | Computer: {computer.upper()}")  
 print("-"\* 40)  
 if "Win" in result:  
 print(Fore.GREEN + result.center(40))  
 elif "Lose" in result:  
 print(Fore.RED + result.center(40))  
 else:  
 print(Fore.MAGENTA + result.center(40))  
 print("-" \* 40)  
 print(f"🏆 Score → Wins: {wins} | Losses: {losses} | Draws: {draws}")  
 print("=" \* 40 + "\n")

| **Section** | **Purpose** |
| --- | --- |
| print("\n" + "="\*40) | Prints a decorative header line. |
| .center(40) | Centers the title text in a 40-character width. |
| Fore.GREEN / Fore.RED / Fore.MAGENTA | Sets text color for Win/Lose/Draw results. |
| f"🏆 Score → Wins: {wins} ..." | Displays the updated scoreboard. |

**Welcome Message**

| **Code** | **Explanation** |
| --- | --- |
| print(Fore.CYAN + "🎮 Welcome to Rock–Paper–Scissors (Best of 5)!") | Welcomes the player with a cyan message. |
| print(Fore.YELLOW + "First to 3 wins becomes the Champion 🏅") | Displays rules in yellow. |
| print(Fore.WHITE + "Type 'quit' to exit anytime.\n") | Informs how to quit. |

**Main Game Loop**

while True:  
 playerChoice = str(input("Enter your choice (rock, paper, scissor) or 'quit' to exit: ")).lower()  
 thinking\_animation()  
 computerChoice = random.choice(mylist)

| **Explanation** |
| --- |
| while True: keeps the game running until broken by break. |
| input() asks for player’s choice. |
| .lower() converts input to lowercase for consistency. |
| random.choice(mylist) selects the computer’s move. |

**Quit Option**

if playerChoice == "quit":  
 print("Game Over 🏁")  
 print(f"Final Score → Wins: {wins} | Losses: {losses} | Draws: {draws}")  
 break

| **Explanation** |
| --- |
| Ends the game if the user types quit. |
| Displays the final scores and stops the loop. |

**Input Validation**

if playerChoice not in mylist:  
 print("Invalid input ⚠️ Try again.\n")  
 continue

| **Explanation** |
| --- |
| Ensures only “rock”, “paper”, or “scissor” are accepted. |
| If invalid, shows a warning and restarts the loop. |

**Game Logic**

Each block compares the player’s choice with the computer’s.

**If player chooses Rock:**

if "rock" in playerChoice:  
 if computerChoice == "rock":  
 print("The Computer picks", computerChoice)  
 result = "Draw!"  
 draws += 1  
 elif computerChoice == "paper":  
 print("The Computer picks", computerChoice)  
 result = "You Lose"  
 losses += 1  
 elif computerChoice == "scissor":  
 print("The Computer picks", computerChoice)  
 result = "You Win"  
 wins += 1

**If player chooses Scissor:**

elif "scissor" in playerChoice:

...

**If player chooses Paper**

elif "paper" in playerChoice:

...

|  |
| --- |
| **Explanation** |
| Each block compares combinations of player vs. computer choices and updates the result accordingly. |

**Display Round Result**

print\_gui(playerChoice, computerChoice, result)

Calls the function to print the formatted scoreboard after each round.

**Win Condition Checks**

if wins == 3:  
 print("You are the Champion! \n")  
 print(f"Final Score → Wins: {wins} | Losses: {losses} | Draws: {draws}")  
 break  
elif losses == 3:  
 print(" The Computer is the Champion! \n")  
 print(f"Final Score → Wins: {wins} | Losses: {losses} | Draws: {draws}")  
 break

| **Explanation** |
| --- |
| Checks if the player or computer reached 3 wins. |
| Prints champion message and final score, then ends the game. |

**Program Output Sample**

**A screenshot of a computer program

AI-generated content may be incorrect.**