CSE 120-01-4252 Intro to Programming with Python

Final Project

Rock, Paper, Scissors in Python

By

Andrew Merritt

April 25, 2025

Table of Contents

Description and Goal …… 4

Timeline……. 5

Responsibilities……..6

Code……...7

Explanation Of Code……..…….8-16

Learning Outcome……16-17

Challenges……17

Video Link ……..17

References……18

List of Figures

Figure Pages

1. Get User Choice……8-10
2. Get Computer Choice……11-12
3. Determine Winner……12-13
4. Play Game……13-16

Description and Goal

The Game of Rock, Paper, Scissors has been created by the Chineses but spread worldwide around the late 20th century. Two opponents play in an space they could play with three choses: Rock, Paper, Scissors. Where rock beats scissors, paper beats rock and scissors beats paper. The objective of the game is for each opponent to choose one of the three options and if they chose the same option they will continue until they haven't. The game of Rock, Paper, Scissors has become commonplace in which it can settle petty disputes between friends and has been used a game to build bonding in a friend group while it outcomes are only three a win, a loss, or a tie the game has been cultural important particularly for Japan as it has settle disputes because according to legend it was originally used to settle disputes between warring samurai as the winner was seen as intelligent to lead an army into battle and the use of the Rock, Paper, Scissors game in Japan hasn’t dissipated today but is used to settle disputes between friends and can also determine who pays the bill at a restaurant. In South Korea on the other hand the game of Rock, Paper, Scissors was used as an act of punishment and the loser of the game would have to perform or execute an embarrassing task but here in recent years the game has gained national interest as national tournaments are held drawing more attention to the sport. In the United States it is played by all age groups and is commonly used to settle disagreements. There are also tournaments in the United States that are played where the competitors compete for prizes and bragging rights. A rare few people have developed sophisticated strategies and techniques for the game which has allowed them to have an advantage to help them in these tournaments. Rock, Paper, Scissors no matter how you play it and what to play it for you cannot deny its cultural significance around the globe which is why I wanted to create a python program of my favorite game to play to settle disputes in my family.

Timeline

Deciding the project took one week to decide in which during this time I had to think about what my project would be and how I was going to approach my project in which I had to organize my time efficiently as well. Requirements needed for this project took me a week to figure out what I needed and what I might have needed to assist with my project which includes what questions I needed to ask my computer so I can find information Information gathering took me a week so that I could find adequate information and relevant information for my project. Implementation of code took four weeks to complete and the bulk of my time as I needed to figure out what types of code I needed in which it took the longest amount of my time because the length of the code I needed testing and correcting code took me a week to correct the code to get the outputs I wanted which was the hardest part of this project as in each of my lines of code needed fixing to get the outcome I wanted. Final report is taking me a week to complete and accurately describe to ensure my completeness on the Final report.

Responsibilities

My personal responsibilities on this project included a wide range of data that I needed to asset me on this project. The first part of my responsibilities was to make sure that I stayed on my own created timeline when I submitted my project proposal which includes organization of what I would work on during each of the next nine weeks. The second part of my responsibilities was to make sure that by the end of the nine weeks that everything that I organized in the first part would be completed by the end of the nine weeks. The special part of the project included my python programing in which this part of the project was the hardest part and longest part of the project to complete as I had to provide complete program that would work but also test and retest the code until I get the outcome of the code that I wanted for my project. The responsibilities of my project made me create a schedule of what I needed to complete by the end of each week and the implementation of my python code along with a correction that I needed to perform.

Code

```python
import random


def get_user_choice():
    user_choice = input("Enter your choice (rock, paper, scissors): ").lower()
    while user_choice not in ["rock", "paper", "scissors"]:
        user_choice = input("Invalid choice. Please enter rock, paper, or scissors: ").lower()
    return user_choice


def get_computer_choice():
    return random.choice(["rock", "paper", "scissors"])


def determine_winner(user_choice, computer_choice):
    if user_choice == computer_choice:
        return "It's a tie!"
    elif (user_choice == "rock" and computer_choice == "scissors") or \
            (user_choice == "paper" and computer_choice == "rock") or \
            (user_choice == "scissors" and computer_choice == "paper"):
        return "You win!"
    else:
        return "You lose!"


def play_game():
    print("Welcome to Rock, Paper, Scissors!")
    while True:
        user_choice = get_user_choice()
        computer_choice = get_computer_choice()
        print(f"You chose: {user_choice}")
        print(f"Computer chose: {computer_choice}")
        print(determine_winner(user_choice, computer_choice))

        play_again = input("Do you want to play again? (yes/no): ").lower()
        if play_again != "yes":
            break
    print("Thanks for playing!")


# Start the game
play_game()
```

Explanation Of Code

Figure 1. Get User Choice

```
def get_user_choice():
    user_choice = input("Enter your choice (rock, paper, scissors): ").lower()
    while user_choice not in ["rock", "paper", "scissors"]:
        user_choice = input("Invalid choice. Please enter rock, paper, or scissors: ").lower()
    return user_choice
```

The get_user_choice() function is a key element of a simple Rock, Paper, Scissors game coded in Python. The function's purpose is to interact with the player, request the input from them, sanitize it, and then return it to be utilized in the rest of the game logic.

Here is this function broken down line by line with the purpose and reasoning for each one:

1. Function Definition

def get_user_choice():

This line declares a Python function named get_user_choice. Functions are reusable code blocks that execute specific functions. The parentheses () denote that this function does not accept any input arguments. The colon : marks the beginning of the function body. This function will: Ask the user for input, Check if the input is valid, Keep asking until valid input is given, Return the cleaned (lowercased) valid input.

2. Taking Input from the User

user_choice = input("Enter your choice (rock, paper, scissors): ").lower()

The input() function displays a prompt string in the terminal and waits for the user to type something and then press Enter. The prompt tells the user exactly what options are accepted: rock, paper, or scissors. The.lower() function is invoked over the input so that regardless of whether the user entered "Rock", "ROCK", or "rOcK", the output is being converted to "rock". This step of

normalization makes the process of comparison remain simple and steer clear of the case sensitivity shortcomings. The normalized value is stored in the variable user_choice.

3. Having a While Loop for Input Validations

while user_choice not in ["rock", "paper", "scissors"]:

user_choice = input("Invalid choice. Type rock, paper, or scissors: ").lower()

Here the user is prompted to enter one of the choices. Analysis: The if statement user_choice not in ["rock", "paper", "scissors"] is to check whether the input is not one of the three provided strings. If the user entered an invalid string like "banana", "rck", or even "rockk", the condition is true, and the loop runs. The user is prompted to enter a valid value again. As before,.lower() ensures that whatever value the user enters will be treated case-insensitively. This loop continues to run until a valid value is entered. This is known as input validation — it's a good practice in programming whenever you're getting input from a user, because users can (and will) make mistakes or provide unexpected input.

4. Returning the Valid Input

return user_choice

When the loop is exited (i.e., the user has entered a valid choice), the function returns said valid string ("rock", "paper", or "scissors"). This returned value is then used by other parts of the program — e.g., it is compared to the computer's choice in order to determine a winner.

Why is This Function Important? The get_user_choice() function is also important in ensuring the reliability and seamless running of the game. Here's why: User Interaction: It's the primary method through which the user interacts with the program. It collects player input in a polite and straightforward manner. Error Prevention: It saves the program from crashing or acting erratically because of invalid inputs. Reusability: Since it is a function, the code can be reused every time a

new game round starts. This avoids repetition and keeps the code tidy. User Experience: Warning messages and validation improve the player's experience by providing instructions and feedback when something goes wrong.

Possible Improvements and Enhancements

Even though this function is already quite good for a simple game, these are some improvements you may wish to make:

1. Support for Exit Command

Allow the user to type "quit" or "exit" to exit the game:

```
if user_choice == "quit":
  return "quit"
```

2. Improved Feedback

Display the invalid input the user provided, in an attempt to make them aware of their mistake:

```
print(f"'{user_choice}' is not a valid choice.")
```

3. Parameterizable Choices

Make the list of valid choices an argument to the function, to allow for flexibility to use future games with other sets of valid choices:

```
def get_user_choice(valid_choices):
```

The get_user_choice() function is short, but it is crammed with essential logic into a small amount of code. It demonstrates good programming practice like input validation, user-friendly interaction, and reuse of code. Understanding how this function works gives you an excellent foundation on which to build more complex games or programs that require trustworthy user input.

Figure 2. Get Computer Choice

```
def get_computer_choice():
    return random.choice(["rock", "paper", "scissors"])
```

This function has the responsibility of simulating the computer making a move in Rock, Paper, Scissors. In contrast to the human player, the computer doesn't need to provide input — rather, it chooses one of the three allowable options randomly.

1. Function Definition

def get_computer_choice():

The def keyword is utilized to initialize a function. The function name is get_computer_choice, which clearly states what the function is meant to accomplish. It doesn't require any arguments — it works the same way when called. The colon : marks the start of the body of the function

2. With random.choice

return random.choice(["rock", "paper", "scissors"])

This line calls the choice() function of the random module to select an element from a list randomly. The list contains three strings: "rock", "paper", and "scissors". When this line is run, one of these strings is selected randomly and returned as the computer's move. The function instantly returns this value through the use of the return statement.

Why Is This Function Helpful?

The function is necessary because it allows the computer an even, unexpected method for playing the game. Let's go through its uses: Randomness: It keeps the game fair. The computer doesn't "know" what the player will choose, so the game is fun and competitive. Simplicity: Instead of repeating the random choice logic throughout your program, this function encapsulates that

behavior into one, reusable piece. Maintainability: If later you ever wanted to change the logic (e.g., have the computer play more smartly), you'd have to change only this one function.

Figure 3: Determine Winner

```
def determine_winner(user_choice, computer_choice):
    if user_choice == computer_choice:
        return "It's a tie!"
    elif (user_choice == "rock" and computer_choice == "scissors") or \
            (user_choice == "paper" and computer_choice == "rock") or \
            (user_choice == "scissors" and computer_choice == "paper"):
        return "You win!"
    else:
        return "You lose!"
```

This function plays a round of Rock, Paper, Scissors. It has two parameters:

user_choice: the input from the user (e.g., "rock")

computer_choice: the randomly generated computer move (e.g., "scissors")

The function checks these two and returns one of the following strings: "You win!", "You lose!", or "It's a tie!".

1. Check for a Tie

if user_choice == computer_choice:

  return "It's a tie!"

If user and computer choose the same one, the game is a tie.

Examples: "rock" vs "rock" , "paper" vs "paper" , "scissors" vs "scissors"

There is no need to search any further here — the outcome is revealed immediately.

2. Search for a Win

elif (user_choice == "rock" and computer_choice == "scissors") or \

   (user_choice == "paper" and computer_choice == "rock") or

(user_choice == "scissors" and computer_choice == "paper"):

return "You win!"

This elif statement tests whether the user has a winning combination.

Let's analyze each condition: Rock crushes scissors, Paper covers rock, Scissors cuts paper. The logical or operator permits any one of the three conditions to result in a win. If any of these are the case, the function returns "You win!"

This function is a clever summary of rules of the game in a streamlined, easy-to-understand fashion. Here is why it does: Simple reasoning: All that can happen has been accounted for — tie, win, loss. Reusable: It can repeatedly be called on and on and on for each round without its logic being rewritten elsewhere. Easy to read: The if, elif, and else are easy to read.

The determine_winner function is brief but effective. It: Has two arguments (user and computer choice), Checks the rules of Rock, Paper, Scissors, Returns a clear result.

Figure 4: Play Game

```
def play_game():
    print("Welcome to Rock, Paper, Scissors!")
    while True:
        user_choice = get_user_choice()
        computer_choice = get_computer_choice()
        print(f"You chose: {user_choice}")
        print(f"Computer chose: {computer_choice}")
        print(determine_winner(user_choice, computer_choice))

        play_again = input("Do you want to play again? (yes/no): ").lower()
        if play_again != "yes":
            break
    print("Thanks for playing!")
```

The play_game() function runs the game in a loop, where the player can play multiple rounds before choosing to quit.

Step 1: Welcome Message

print("Welcome to Rock, Paper, Scissors!")

This is a welcome message to let the player know that the game is starting. It's the first thing the user will see when they run the program.

Step 2: Start the Game Loop

while True:

This is an infinite loop — it will keep going forever until the user decides to end the game. Inside this loop, a whole round of the game is played.

We use while True because we don't want the game to ever naturally end on its own unless the user enters "no" when asked if he/she wants to play again.

Step 3: Get User and Computer Choices

user_choice = get_user_choice()

computer_choice = get_computer_choice()

Here the function calls

get_user_choice() — prompts the user for their move.

get_computer_choice() — randomly selects a move for the computer.

Both values are stored to be utilized in the round.

Step 4: Display the Choices

print(f"You chose: {user_choice}")

print(f"Computer chose: {computer_choice}")

These lines give feedback to the player, showing what both sides have chosen. This is needed to view the result of each round.

Step 5: Determine and Display the Result

print(determine_winner(user_choice, computer_choice))

This calls the determine_winner() function and prints out the result: "You win!", "You lose!", or "It's a tie!"

Page 2: Loop Control and User Interaction

Step 6: Ask to Play Again

play_again = input("Do you want to play again? (yes/no): ").lower()

The program prompts the player after each round whether he/she wishes to play another. The input is converted to lower case to accept answers like "YES" or "Yes" pleasantly.

Step 7: Break out of the Loop if Answer Isn't "Yes"

if play_again!= "yes":

    break

The moment the user enters anything but "yes," the loop ends. That is how the game smoothly exits.

User types in "no" → loop finishes.

User types in "y" → loop ends (despite being a yes!).

User types in "yes" → round starts over again.

Step 8: Final Goodbye

print("Thanks for playing!")

This message is printed only once, at the end of the loop. It is a friendly farewell after the game has ended.

The play_game() function is the program's controller or main loop. It doesn't make decisions about winners or create options — it outsources those jobs to other functions. Instead, it does the following: Manages the game's overall flow. Keeps the user in a cycle of rounds. Manages clean input/output.Provides a replay mechanism.

The play_game() function is the brains of the Rock, Paper, Scissors game. It ties everything together — input, output, and game rules — into an integrated, interactive loop that gives the user a fun experience. Even though it seems simple, it follows a good design pattern: loop + input + logic + replay.

Learning Outcome

Working on this Rock, Paper, Scissors application taught me a great deal about how Python functions and how to create a basic interactive game. Below is an explanation of the most important things that I learned:

I learned to define and use functions like get_user_choice(), get_computer_choice(), and determine_winner(). This taught me how to keep code organized by dividing it into small, reusable pieces. The game asks the player to enter rock, paper, or scissors. I learned how to use input() to get the player's input, and how to check it so that the game won't crash if the user enters something other than rock, paper, or scissors.Using the determine_winner() function, I had a better understanding of if, elif, and else statements. I used logic to compare the user's and computer's choices and decide who wins each round. I used a while True: loop to get the game to keep running until the user wanted to quit. This showed me how to do things over and over again in a program and how to know when to stop using a break statement. The computer's move is selected randomly using random.choice(). I had learned how to use Python's random module to make the game unpredictable and engaging. I used f-strings to print out what the player and computer chose, and to tell us the result of each round. That showed me how to format strings nicely and readably. Overall, I had a good insight into how games are created: taking input, applying logic, showing the result, and giving the user an option to play on. I also valued the significance of dividing up various segments of the program into separate functions.

After finishing this project, I'm learning the following much better: Functions, Conditionals ,Loops, User input, Randomization, Output formatting.

Challenges

I have said before that when I was coding that I have faced many challenges and these challenges include minor mistakes like missing a colon and missing spaces; to major fixes like missing entire lines of code and for a long time I thought as though I would never be able to get the outcome I wanted for this project but just as I was about to give up all hope on my code I fixed a couple of lines and put a few spaces here and there and my code was fixed. I have faced many trials and tribulations while putting my code together but in the end my challenges were well worth it.

Video Link

https://youtu.be/ROjUpBttQu4

References

admin. (2023, May 17). The cultural significance of rock paper scissors around the world. *World Rock Paper Scissors Association - Professional Rock Paper Scissors*. https://wrpsa.com/the-cultural-significance-of-rock-paper-scissors-around-the-world/

kriz. (2021, May 20). *Rock, Paper Scissors game help*. Discussions on Python.Org. https://discuss.python.org/t/rock-paper-scissors-game-help/8823

Wilkerson, C. (2021, January 18). Make your first python game: Rock, paper, scissors! *Real Python*. https://realpython.com/python-rock-paper-scissors/