


## Python programming (Programme No.1) Number guessing game with Python +

make a rock paper scissors game

Sure! Here's a simple **Rock, Paper, Scissors** game in Python that allows a player to play against the computer. The program will handle multiple rounds, track scores, and allow the player to quit whenever they want.

### Rock, Paper, Scissors Game Code:

python

 Copy code

```
import random

# Function to determine the winner
def determine_winner(player_choice, computer_choice):
    if player choice == computer choice:
```

Message ChatGPT


### Explain Code

explain the code

Certainly! Let's go through the code step-by-step to explain how it works:

### 1. Importing the `random` module

python

 Copy code

```
import random
```

## 1. Importing the random module

python

Copy code

```
import random
```

- The random module is used to generate random numbers or make random selections. In this game, we use it to let the computer randomly select "rock," "paper," or "scissors."

## 2. determine\_winner() function

python

Copy code

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

This function determines the outcome of the game based on the choices made by the player and the computer. Let's break it down:

- **Input:** The function takes two parameters:
  - **player\_choice:** The choice made by the player (either "rock", "paper", or "scissors").
  - **computer\_choice:** The choice made by the computer (randomly selected from "rock", "paper", or "scissors").
- **Logic:**
  - First, it checks if both the player and the computer made the same choice. If they did, it's a tie ("It's a tie!").
  - Then, it checks the winning conditions:

- Rock beats Scissors.
- Scissors beats Paper.
- Paper beats Rock.
- If any of the conditions are true, the player wins, and the function returns "You win!".
- If none of the above conditions hold, the computer wins, and the function returns "You lose!".

### 3. play\_game() function

python

Copy code

```
def play_game():
    print("Welcome to Rock, Paper, Scissors!")
```

- The `play_game()` function is the main function that runs the game. It begins by printing a welcome message.

python

Copy code

```
choices = ["rock", "paper", "scissors"]
player_score = 0
computer_score = 0
```

- **choices:** This is a list containing the three possible options in the game: "rock", "paper", and "scissors". The computer will randomly choose one of these during each round.
- **player\_score** and **computer\_score:** These variables are used to keep track of the player's and computer's scores, respectively. They are initialized to 0 at the beginning.

python

Copy code

```
while True:
    player_choice = input("\nChoose Rock, Paper, or Scissors (or
type 'quit' to stop playing): ").strip().lower()
```

- The `while True` creates an infinite loop, meaning the game will continue running until the player decides to stop (by typing 'quit').
- The `input()` function asks the player to enter their choice ("rock", "paper", or "scissors"). It also allows the player to type 'quit' to end the game.
- `.strip()` removes any leading or trailing spaces from the player's input.
- `.lower()` ensures that the input is case-insensitive (it converts the input to lowercase, so "Rock", "rock", or "ROCK" are all treated the same).

python

Copy code

```
    if player_choice == 'quit':
        print(f"\nGame Over! Final Score - You: {player_score},
Computer: {computer_score}")
        break
```

- This checks if the player typed 'quit' to end the game. If so, it prints the final score and uses `break` to exit the loop and stop the game.

python

Copy code

```
    if player_choice not in choices:
        print("Invalid choice! Please choose Rock, Paper, or
Scissors.")
        continue
```

- This block ensures the player enters a valid choice. If the input is not "rock", "paper", or "scissors", the game prints an error message and asks the player to try again (`continue` will skip the rest of the loop and prompt the player again).

python

Copy code

```
    computer_choice = random.choice(choices)
    print(f"Computer chose: {computer_choice}")
```

- `random.choice(choices)` randomly selects one of the three options (rock, paper, or scissors) for the computer.
- The computer's choice is then printed so the player can see it.

python

Copy code

```
result = determine_winner(player_choice, computer_choice)
print(result)
```

- The function `determine_winner()` is called with the player's and computer's choices as arguments. It returns a string indicating whether the player won, lost, or tied, and this result is printed to the screen.

python

Copy code

```
if result == "You win!":
    player_score += 1
elif result == "You lose!":
    computer_score += 1
```

- Based on the result of the round, the scores are updated:
  - If the result is "You win!", the player's score is incremented by 1.
  - If the result is "You lose!", the computer's score is incremented by 1.
- The `+=` operator increases the respective score by 1.

python

Copy code

```
print(f"Score - You: {player_score}, Computer: {computer_score}")
```

- After each round, the current score of both the player and the computer is displayed.

## 4. Starting the Game

python

Copy code

```
play_game()
```

- This line actually starts the game by calling the `play_game()` function. Without this line, the game wouldn't run.

## Example Output:

### *Scenario 1: Player Wins*

```
yaml
```

Copy code

```
Welcome to Rock, Paper, Scissors!
```

```
Choose Rock, Paper, or Scissors (or type 'quit' to stop playing): rock
```

```
Computer chose: scissors
```

```
You win!
```

```
Score - You: 1, Computer: 0
```

```
Choose Rock, Paper, or Scissors (or type 'quit' to stop playing):
```

```
paper
```

```
Computer chose: rock
```

```
You win!
```

```
Score - You: 2, Computer: 0
```

```
Choose Rock, Paper, or Scissors (or type 'quit' to stop playing): quit
```

```
Game Over! Final Score - You: 2, Computer: 0
```

### *Scenario 2: Invalid Input*

```
bash
```

Copy code

```
Welcome to Rock, Paper, Scissors!
```

```
Choose Rock, Paper, or Scissors (or type 'quit' to stop playing):
```

```
pizza
```

```
Invalid choice! Please choose Rock, Paper, or Scissors.
```

```
Choose Rock, Paper, or Scissors (or type 'quit' to stop playing): rock
```

```
Computer chose: scissors
```

You win!

Score - You: 1, Computer: 0

Choose Rock, Paper, or Scissors (or type 'quit' to stop playing): quit

Game Over! Final Score - You: 1, Computer: 0

### **Scenario 3: Tie Game**

yaml

Copy code

Welcome to Rock, Paper, Scissors!

Choose Rock, Paper, or Scissors (or type 'quit' to stop playing): rock

Computer chose: rock

It's a tie!

Score - You: 0, Computer: 0

Choose Rock, Paper, or Scissors (or type 'quit' to stop playing):

paper

Computer chose: scissors

You lose!

Score - You: 0, Computer: 1

Choose Rock, Paper, or Scissors (or type 'quit' to stop playing): quit

Game Over! Final Score - You: 0, Computer: 1

### **Key Concepts:**

- **Game Loop:** The game keeps running until the player types 'quit'.
- **Input Validation:** The program checks if the player enters a valid choice and asks them to re-enter if they make a mistake.
- **Random Selection:** The computer randomly picks one of the three choices using `random.choice()`.
- **Score Tracking:** The game keeps track of the scores and displays them after each round.

This breakdown should help you understand how the game works. Let me know if you have more questions or need further clarification!

## **Improving The code**

is there anything you can change in the code to make it better



Sure! While the code for the Rock, Paper, Scissors game is already functional, there are a few improvements and enhancements we can make to improve the user experience, readability, and functionality. Here are a few suggestions:

### **1. Input Validation Improvement**

Currently, the program checks if the player input is valid and prompts the player again if it's not. We could improve the user experience by using a loop that ensures valid input and avoids the need for manual checks every time.



### **1. Input Validation Improvement**

Currently, the program checks if the player input is valid and prompts the player again if it's not. We could improve the user experience by using a loop that ensures valid input and avoids the need for manual checks every time.

### **2. Clearer Prompt for Restarting the Game**

After the game ends, we could ask the player if they want to play again instead of immediately quitting.

### **3. Game Statistics**

Keep track of the total number of games played, wins, losses, and ties. This would give the player a better overview of their performance.

### **4. Randomized Introduction**

Add some fun elements like randomizing the welcome message or making the game start with a short introduction or sound effect to enhance the experience.



## 5. Better Handling of Special Characters or Case Sensitivity

While we are handling case sensitivity by using `.lower()`, we can make it even more robust by handling other variations like additional spaces or punctuation marks.

Here's an updated version of the code with some of these im