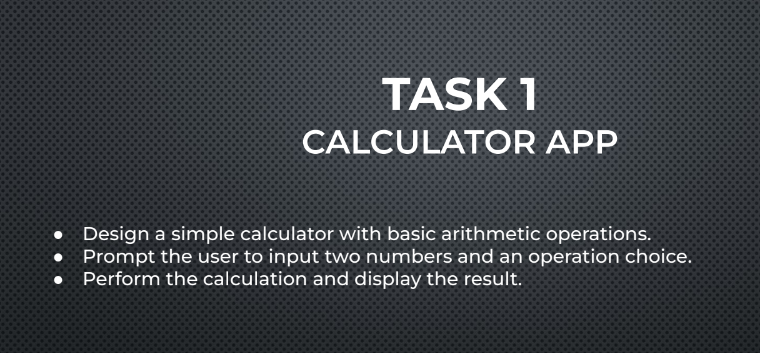
NAME :- J . Anil yadav

Gmail : - [yadava14282@gmail.com](mailto:yadava14282@gmail.com)



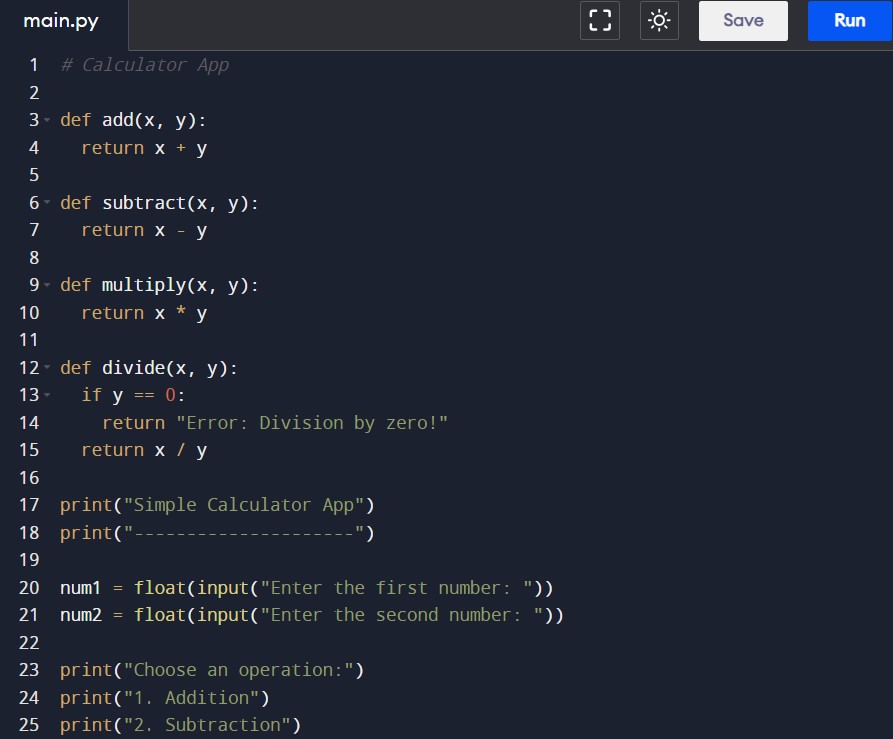


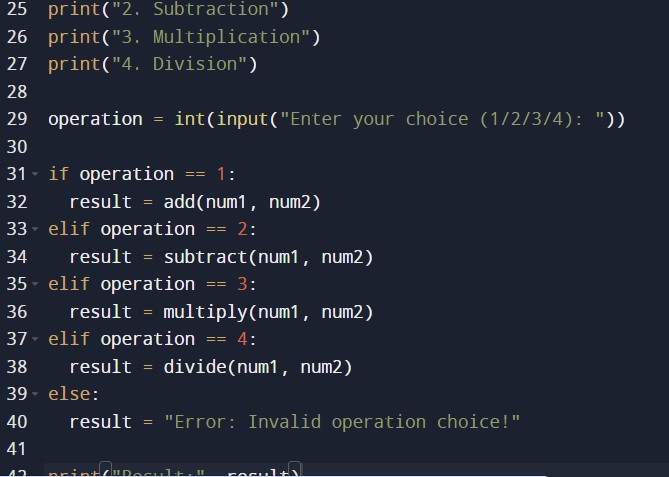
Here's how the app works:

1. The user is prompted to input two numbers, num1 and num2.
2. The user is then prompted to choose an operation by entering a number (1-4) corresponding to the desired operation.
3. Based on the user's choice, the app calls the corresponding function (add, subtract, multiply, or divide) and passes num1 and num2 as arguments.
4. The function performs the calculation and returns the result.
5. The app displays the result to the user.

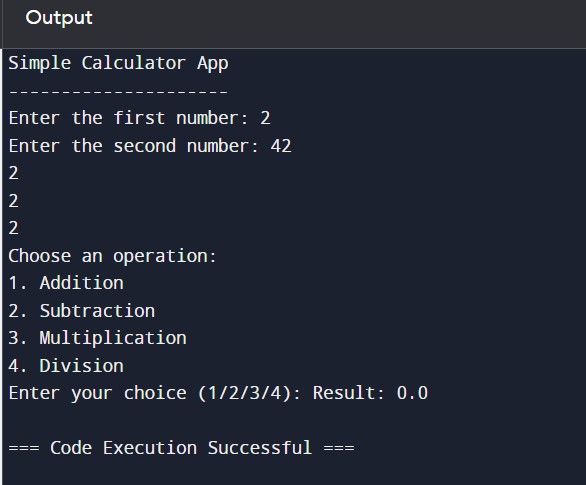
Note that I've added some basic error handling, such as checking for division by zero and invalid operation choices. You can modify or add more features as needed.

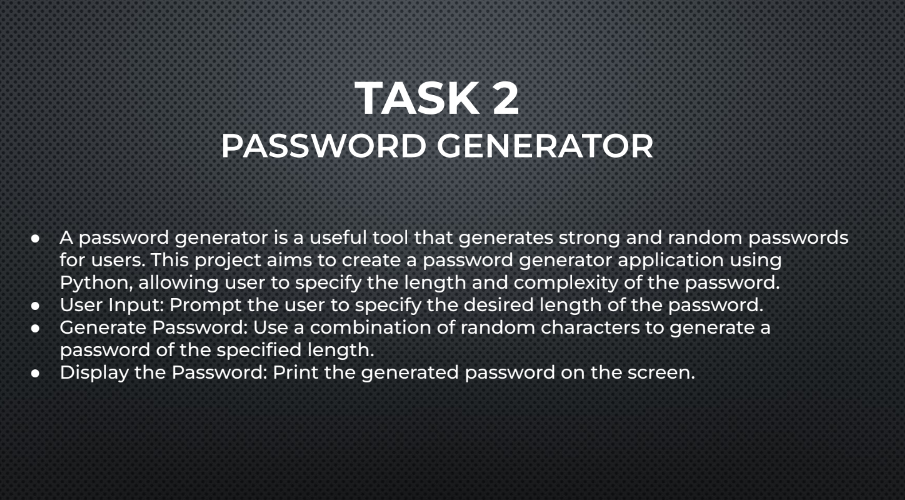
**INPUT : -**





**OUTPUT :-**





Here's how the app works:

The user is prompted to input the desired length of the password.

The generate\_password function is called with the user-input length as an argument.

The function checks if the length is at least 8 characters, and if not, it prints an error message and returns None.

The function generates a password of the specified length using a combination of random characters from the following character sets:

string.ascii\_letters: uppercase and lowercase letters (a-z, A-Z)

string.digits: digits (0-9)

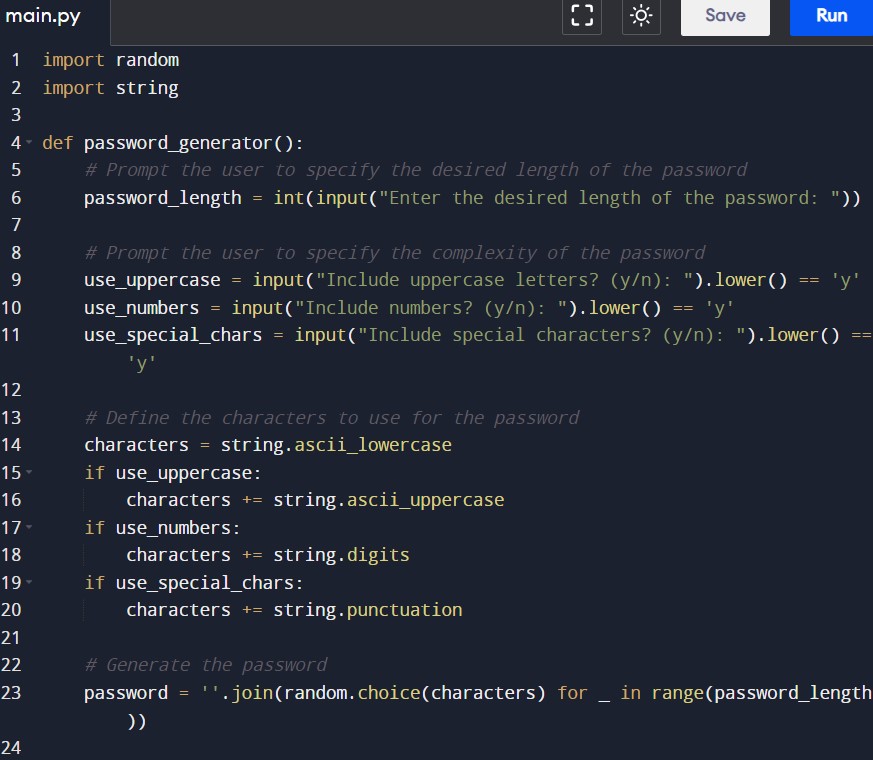
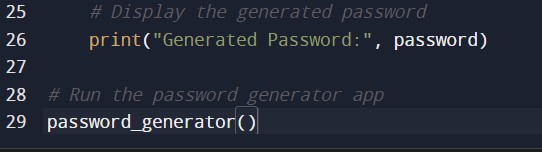
string.punctuation: special characters (!, @, #, $, etc.)

The generated password is returned as a string.

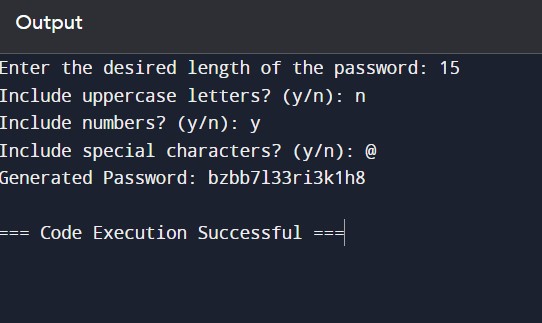
The app prints the generated password to the screen.

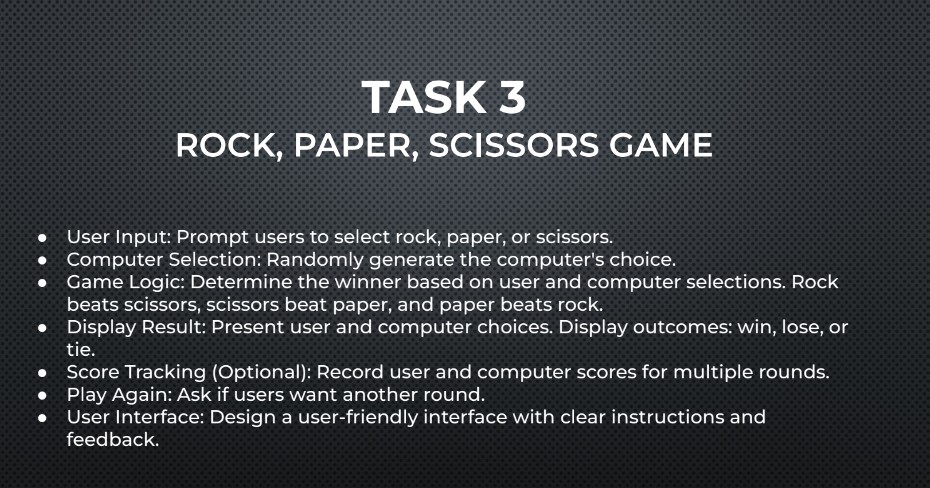
You can modify the generate\_password function to include additional character sets or to enforce specific password policies (e.g., requiring at least one uppercase letter, one digit, etc.).

**INPUT : -**



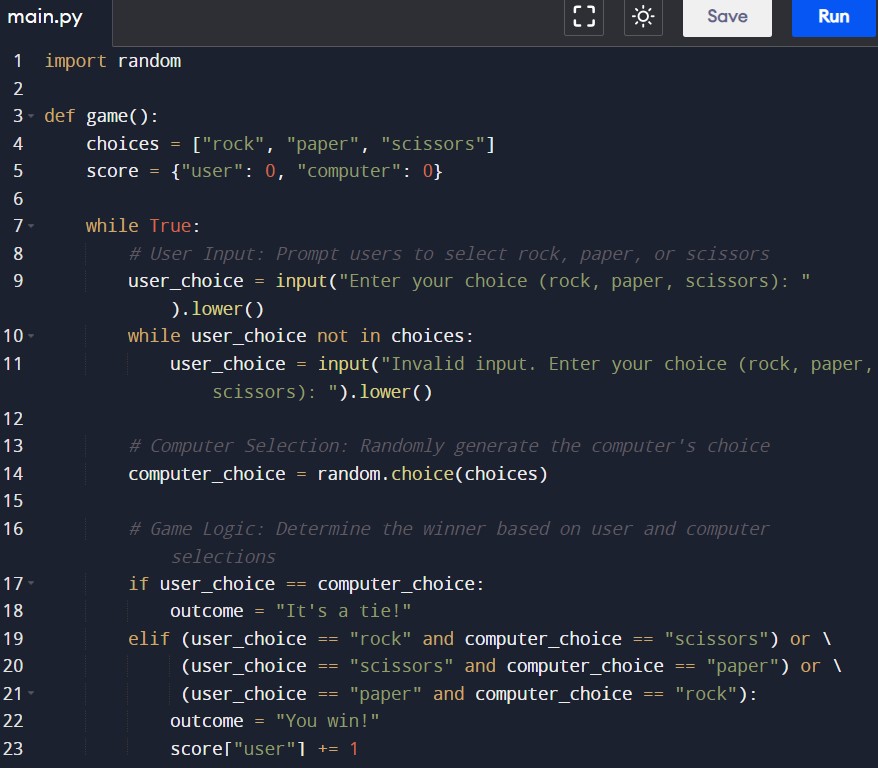
**OUTPUT : -**

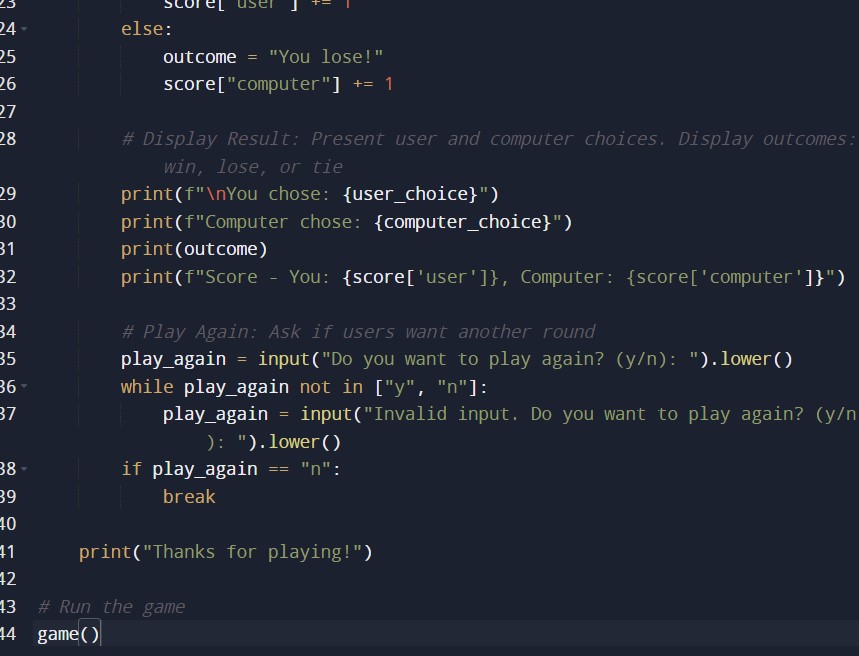


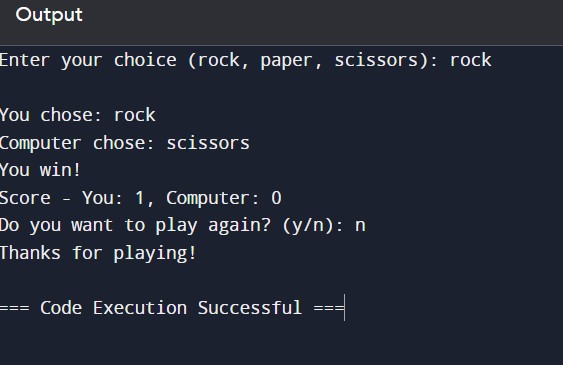


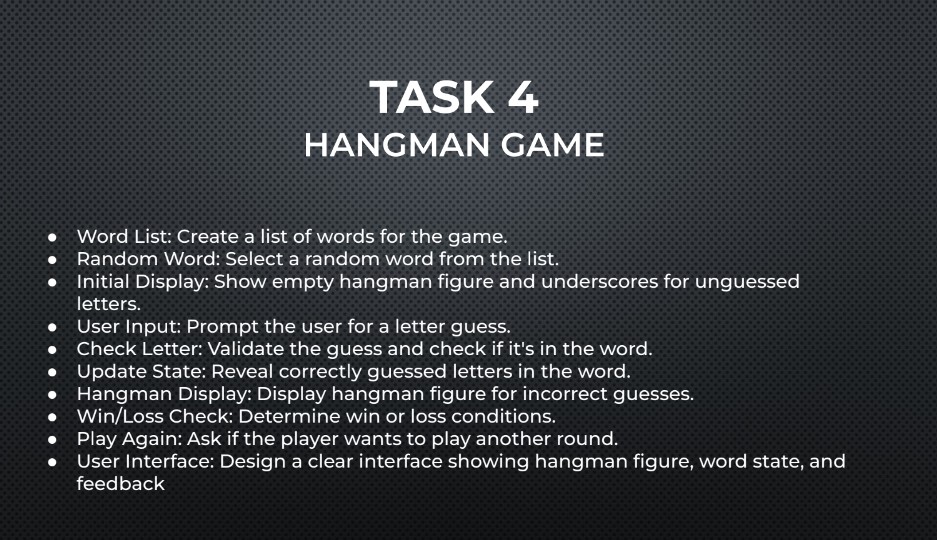
Here's how the game works:

1. The **get\_user\_choice** function prompts the user to input their choice and validates the input.
2. The **get\_computer\_choice** function generates a random choice for the computer.
3. The **determine\_winner** function determines the winner based on the user and computer choices.
4. The **play\_game** function runs the game loop, which:
   * Prompts the user to input their choice
   * Generates the computer's choice
   * Displays the choices and determines the winner
   * Updates the scores
   * Asks if the user wants to play again
5. The game continues until the user decides to quit.

**INPUT : -**



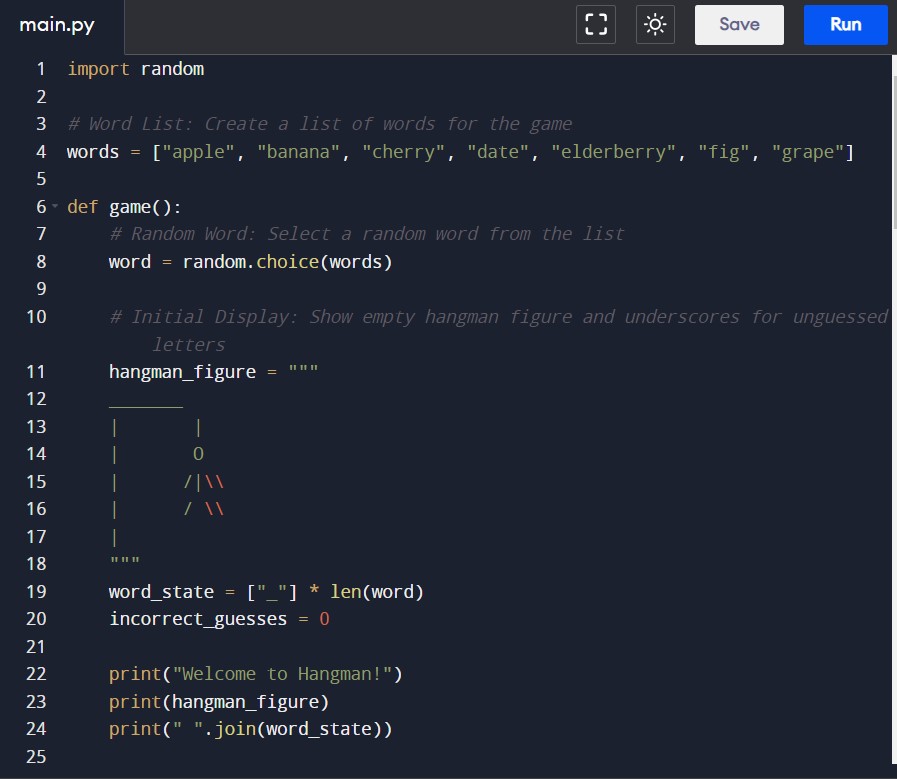
**OUTPUT : -**

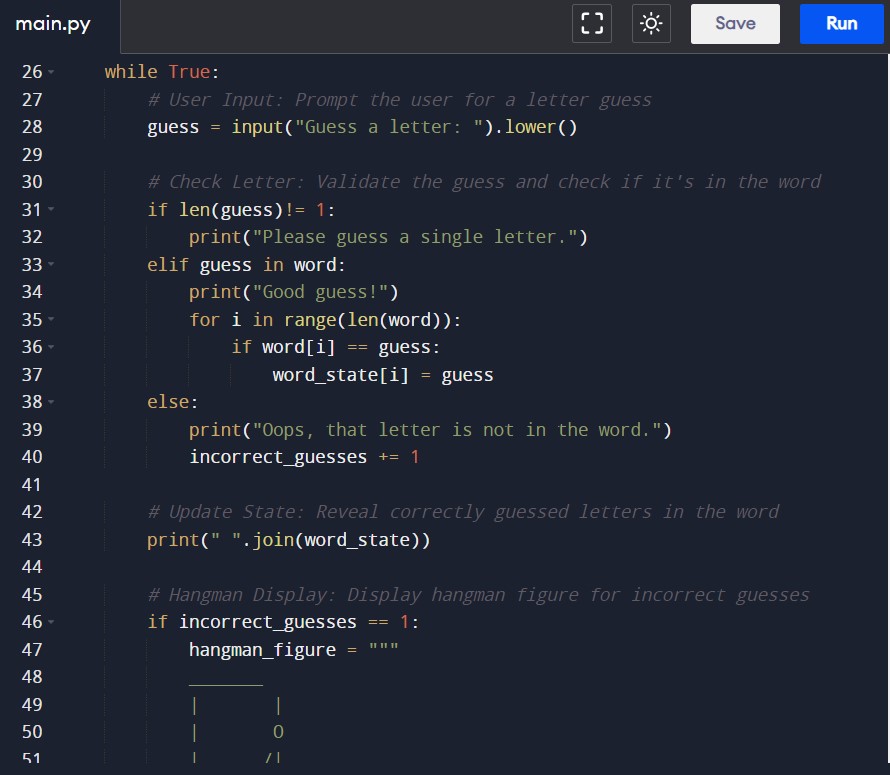


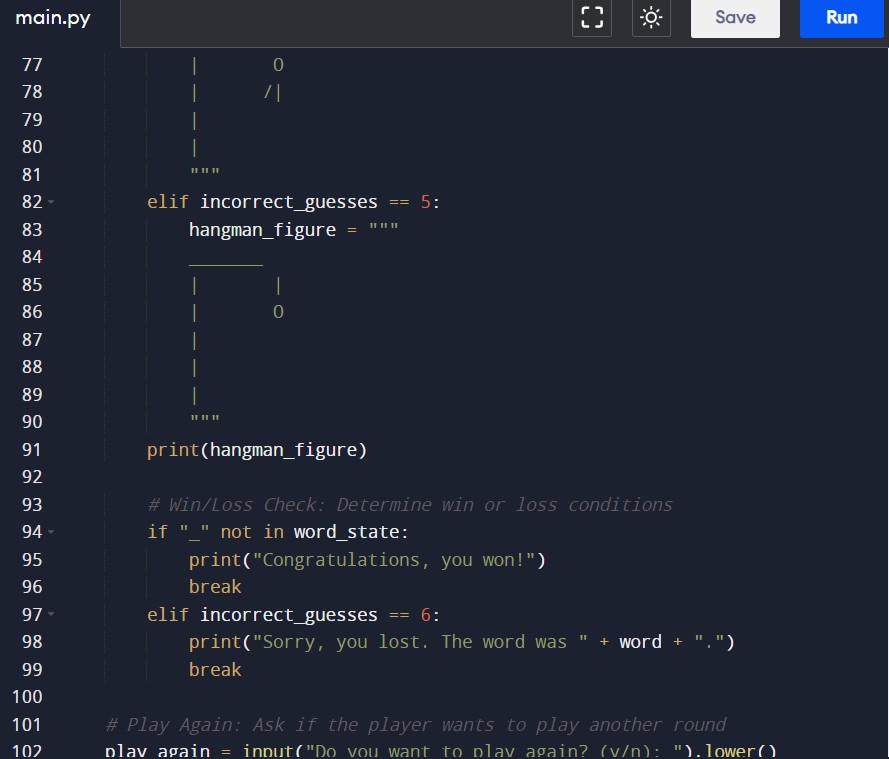
Here's how the game works:

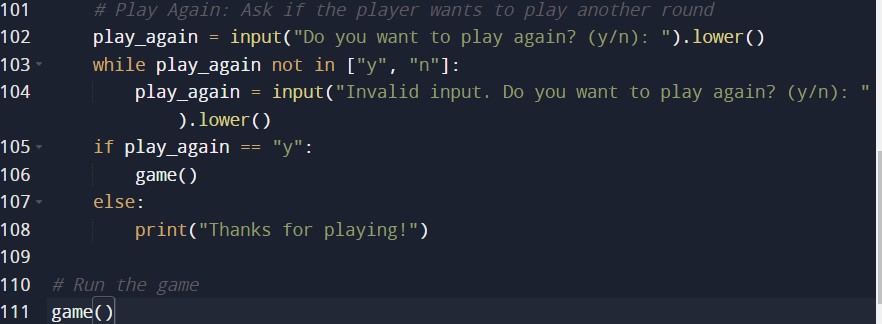
1. The **get\_random\_word** function selects a random word from the word list.
2. The **display\_hangman** function returns the hangman figure based on the number of incorrect guesses.
3. The **play\_game** function runs the game loop, which:
   * Displays the hangman figure and word state
   * Prompts the user to guess a letter
   * Checks if the guess is in the word and updates the word state
   * Increments the incorrect guesses count if the guess is incorrect
   * Checks for win or loss conditions
   * Asks if the user wants to play again
4. The game continues until the user decides to quit.

**INPUT :-**









**OUTPUT :-**



