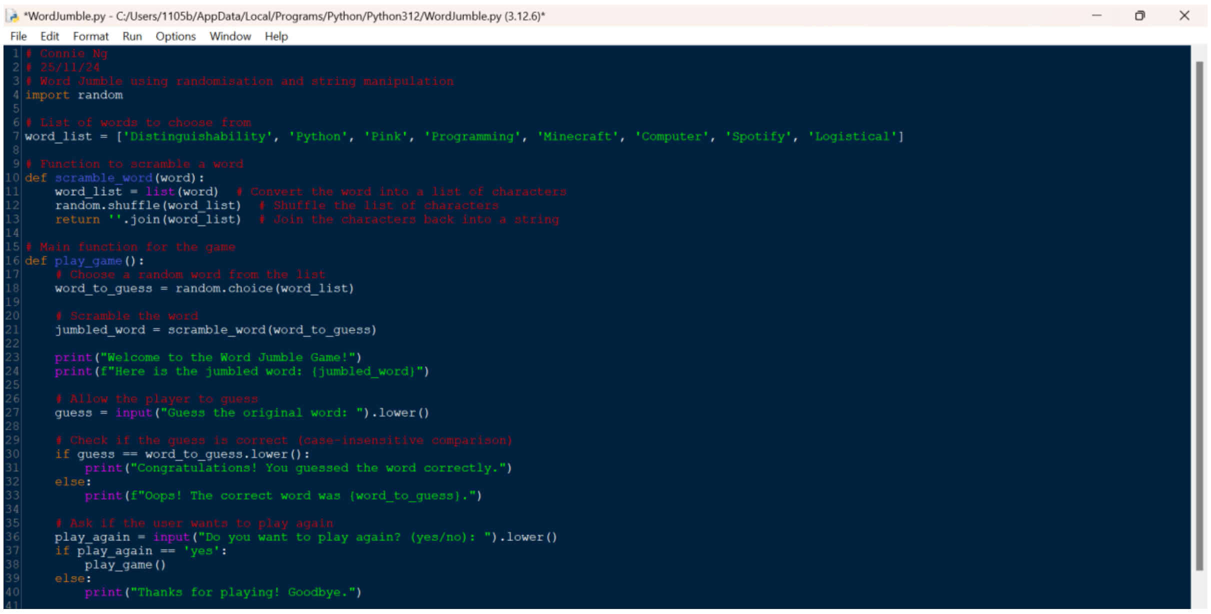
Program 1 – Word Jumble Game

The "Word Jumble Game" involves taking a word, scrambling its letters, and prompting the user to guess the original word.

In a broader sense, it could also be categorized as a type of word puzzle game. Depending on its specific design and features, it might be called a word guessing, but "Word Jumble" is the most accurate and widely recognized term for this type of game.


WordJumble.py - C:/Users/1105b/AppData/Local/Programs/Python/Python312/WordJumble.py (3.12.6)
File Edit Format Run Options Window Help

```python
# Connie Ng
# 25/11/24
# Word Jumble using randomisation and string manipulation
import random

# List of words to choose from
word_list = ['Distinguishability', 'Python', 'Pink', 'Programming', 'Minecraft', 'Computer', 'Spotify', 'Logistical']

# Function to scramble a word
def scramble_word(word):
    word_list = list(word)  # Convert the word into a list of characters
    random.shuffle(word_list)  # Shuffle the list of characters
    return ''.join(word_list)  # Join the characters back into a string

# Main function for the game
def play_game():
    # Choose a random word from the list
    word_to_guess = random.choice(word_list)

    # Scramble the word
    jumbled_word = scramble_word(word_to_guess)

    print("Welcome to the Word Jumble Game!")
    print(f"Here is the jumbled word: {jumbled_word}")

    # Allow the player to guess
    guess = input("Guess the original word: ").lower()

    # Check if the guess is correct (case-insensitive comparison)
    if guess == word_to_guess.lower():
        print("Congratulations! You guessed the word correctly.")
    else:
        print(f"Oops! The correct word was {word_to_guess}.")

    # Ask if the user wants to play again
    play_again = input("Do you want to play again? (yes/no): ").lower()
    if play_again == 'yes':
        play_game()
    else:
        print("Thanks for playing! Goodbye.")
```

For this program the objective is to take a word and have python jumble the letters, making it more challenging to guess. The program will present a scrambled version of the word and ask the user to guess the original word.

Step 1: Define Pythons Structure

Firstly, we will define basic python structures for our game

- A function to scramble a word.
- A main part of the program to interact with the user.

Step 2: Writing the Function to Jumble the Word

The core of the game is scrambling the word. We'll use Python's random module to shuffle the letters of the word. Here's the function to do that

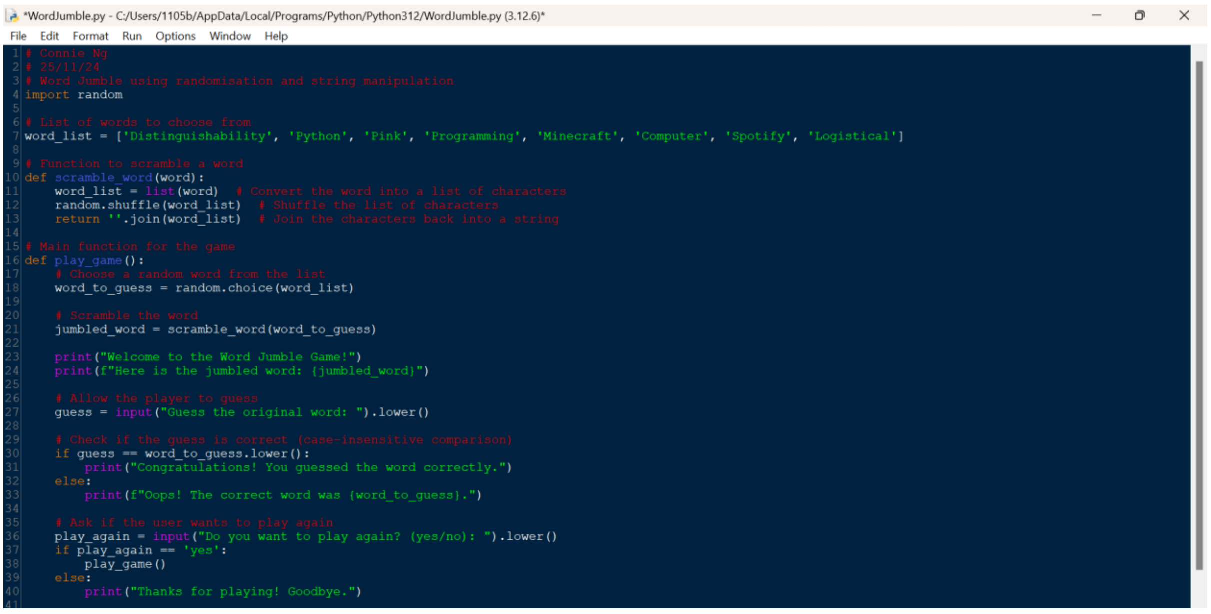
Explanation:

- list(word**):** This turns the word into a list of individual letters.
- random.shuffle(word_list): Randomly mixes up the letters in the list.
- ‘’.join(word_list): Converts the shuffled list of letters back into a single word.

Step 3: Building the Main Game

Now let’s put everything together in the main program. It will:

1. Pick a random word.
2. Scramble it using our jumble function.
3. Show the scrambled word to the player and ask them to guess what the word is.
4. Check if the player guessed correctly.


WordJumble.py - C:/Users/1105b/AppData/Local/Programs/Python/Python312/WordJumble.py (3.12.6)
File Edit Format Run Options Window Help

```python
# Connie Ng
# 25/11/24
# Word Jumble using randomisation and string manipulation
import random

# List of words to choose from
word_list = ['Distinguishability', 'Python', 'Pink', 'Programming', 'Minecraft', 'Computer', 'Spotify', 'Logistical']

# Function to scramble a word
def scramble_word(word):
    word_list = list(word)  # Convert the word into a list of characters
    random.shuffle(word_list)  # Shuffle the list of characters
    return ''.join(word_list)  # Join the characters back into a string

# Main function for the game
def play_game():
    # Choose a random word from the list
    word_to_guess = random.choice(word_list)

    # Scramble the word
    jumbled_word = scramble_word(word_to_guess)

    print("Welcome to the Word Jumble Game!")
    print(f"Here is the jumbled word: {jumbled_word}")

    # Allow the player to guess
    guess = input("Guess the original word: ").lower()

    # Check if the guess is correct (case-insensitive comparison)
    if guess == word_to_guess.lower():
        print("Congratulations! You guessed the word correctly.")
    else:
        print(f"Oops! The correct word was {word_to_guess}.")

    # Ask if the user wants to play again
    play_again = input("Do you want to play again? (yes/no): ").lower()
    if play_again == 'yes':
        play_game()
    else:
        print("Thanks for playing! Goodbye.")
```

But What is Happening Here?

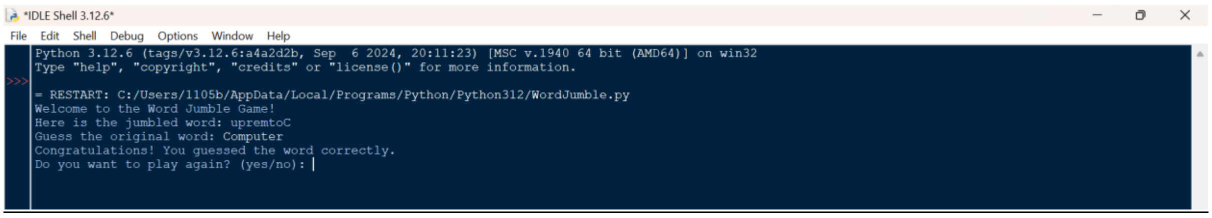
- random.choice(words): Picks a random word from the list.
- jumble_word(word_to_jumble): Jumbles the selected word.
- input(): Prompts the player to guess the word.
- .lower(): Makes the comparison case-insensitive, so "Python" and "python" are treated the same.

Step 4: Running the Game

When you run the game, this is how the program will play out:

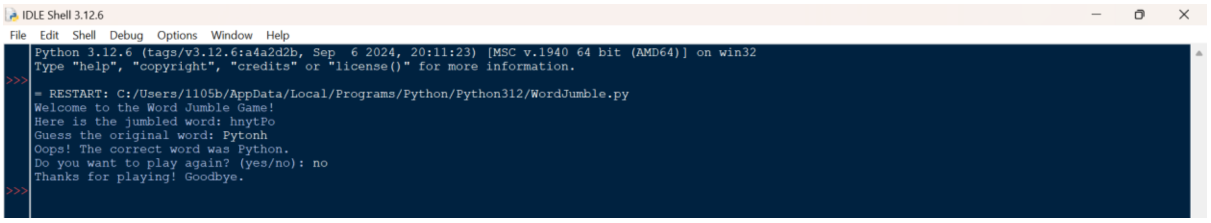
1. The program picks a random word from the list.
2. The program scrambles the letters of that word.
3. It will then show the jumbled word to the player.
4. The player will enter their guess.
5. The program will checks if the guess is correct and lets the player know either by saying "Opps! The correct word was (programmed word)" and ask if the player would like to play again, or it will let the user know that their guess was correct by having the output of "Congratulations! You guessed the word correctly."

Example if you guessed correctly.



```
Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep  6 2024, 20:11:23) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/1105b/AppData/Local/Programs/Python/Python312/WordJumble.py
Welcome to the Word Jumble Game!
Here is the jumbled word: upremtoC
Guess the original word: Computer
Congratulations! You guessed the word correctly.
Do you want to play again? (yes/no):
```

Example if you guessed incorrectly



```
Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep  6 2024, 20:11:23) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/1105b/AppData/Local/Programs/Python/Python312/WordJumble.py
Welcome to the Word Jumble Game!
Here is the jumbled word: hnytPo
Guess the original word: Pytonh
Oops! The correct word was Python.
Do you want to play again? (yes/no): no
Thanks for playing! Goodbye.
>>>
```

v

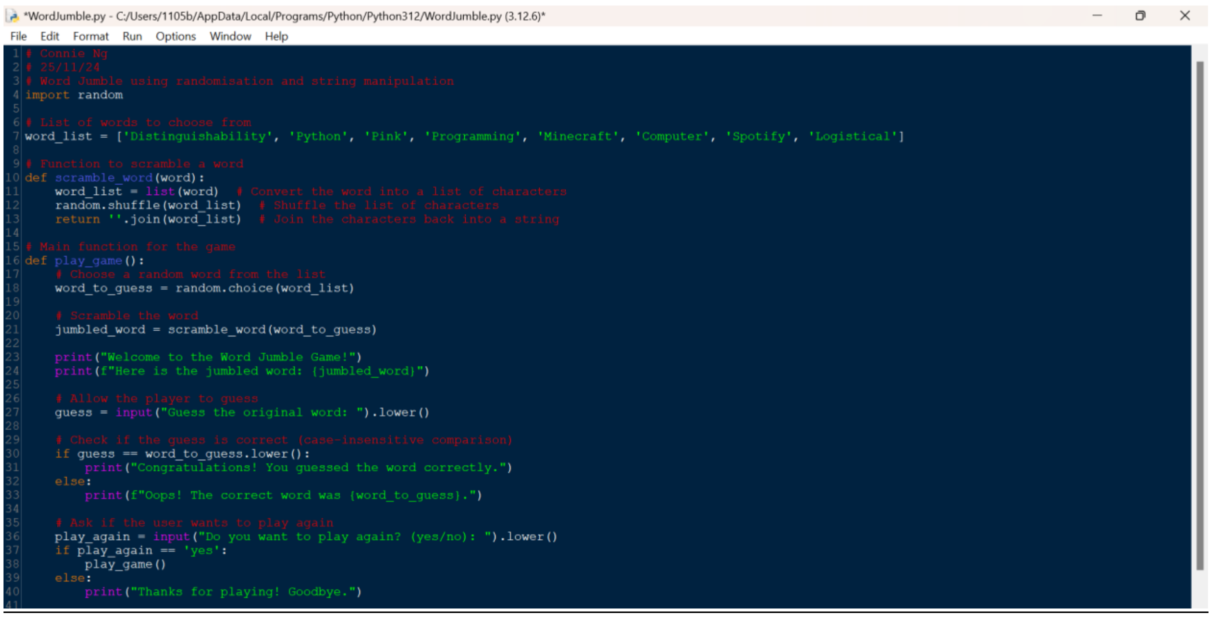
Step 5: Improving and Expanding the Game

Testing:

Before calling it finished, test the game. Try it out multiple times to make sure:

- The words are jumbled correctly.
- The game gives the right response when you guess correctly or incorrectly.
- The guesses are case-insensitive (it doesn't matter if you type "Python" or "python").

Which Leads our Final Outcome to


WordJumble.py - C:/Users/1105b/AppData/Local/Programs/Python/Python312/WordJumble.py (3.12.6)
File Edit Format Run Options Window Help

```python
# Connie Ng
# 25/11/24
# Word Jumble using randomisation and string manipulation
import random

# List of words to choose from
word_list = ['Distinguishability', 'Python', 'Pink', 'Programming', 'Minecraft', 'Computer', 'Spotify', 'Logistical']

# Function to scramble a word
def scramble_word(word):
    word_list = list(word)  # Convert the word into a list of characters
    random.shuffle(word_list)  # Shuffle the list of characters
    return ''.join(word_list)  # Join the characters back into a string

# Main function for the game
def play_game():
    # Choose a random word from the list
    word_to_guess = random.choice(word_list)

    # Scramble the word
    jumbled_word = scramble_word(word_to_guess)

    print("Welcome to the Word Jumble Game!")
    print(f"Here is the jumbled word: {jumbled_word}")

    # Allow the player to guess
    guess = input("Guess the original word: ").lower()

    # Check if the guess is correct (case-insensitive comparison)
    if guess == word_to_guess.lower():
        print("Congratulations! You guessed the word correctly.")
    else:
        print(f"Oops! The correct word was {word_to_guess}.")

    # Ask if the user wants to play again
    play_again = input("Do you want to play again? (yes/no): ").lower()
    if play_again == 'yes':
        play_game()
    else:
        print("Thanks for playing! Goodbye.")
```

Program Overview:

This Python program is a simple word jumble game. The program:

1. Selects a random word from a predefined list.
2. Scrambles the letters of the chosen word.
3. Prompts the user to guess the original word.
4. Compares the user's guess with the original word (in a case-insensitive manner).
5. Tells the user if they guessed correctly or not.
6. Asks if the user wants to play again, allowing them to repeat the game if they wish.

Code Explanation:

Let's break the code down into its key parts:

1. Importing the random Module:

- python

- Copy code

- import random

- The random module is imported to help us randomly select a word from the word_list and shuffle the letters of the word. This gives the game its random nature.

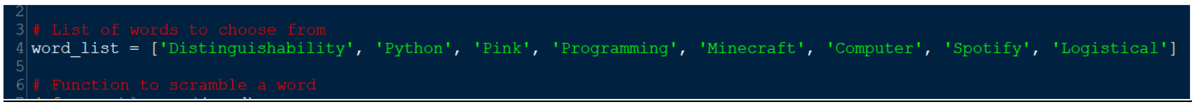
2. Word List (word_list):

- python

- Copy code

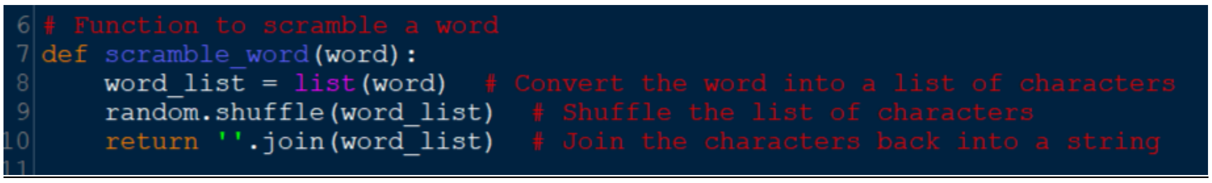
word_list = ['Distinguishability', 'Python', 'Pink', 'Programming', 'Minecraft', 'Computer', 'Spotify', 'Logistical']

- word_list is a list that contains several words. Each of these words is a potential choice for the game. The program will randomly select one word from this list to scramble and present to the player.



```python
# List of words to choose from
word_list = ['Distinguishability', 'Python', 'Pink', 'Programming', 'Minecraft', 'Computer', 'Spotify', 'Logistical']

# Function to scramble a word
```

3. Scramble Word Function.

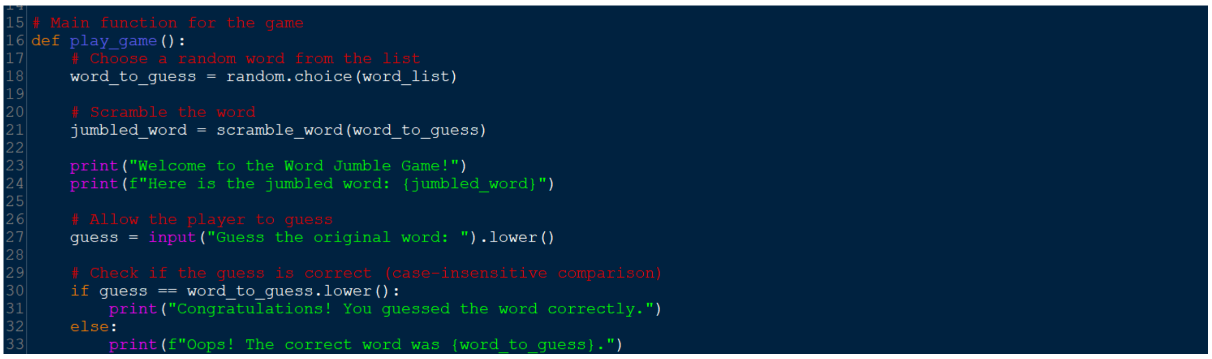


```python
# Function to scramble a word
def scramble_word(word):
    word_list = list(word)  # Convert the word into a list of characters
    random.shuffle(word_list)  # Shuffle the list of characters
    return ''.join(word_list)  # Join the characters back into a string
```

- **Purpose**: This function scrambles the letters of a given word.
- **How It Works**:

    1. **list(word)**: Converts the word (which is a string) into a list of individual characters. For example, 'Python' becomes ['P', 'y', 't', 'h', 'o', 'n'].

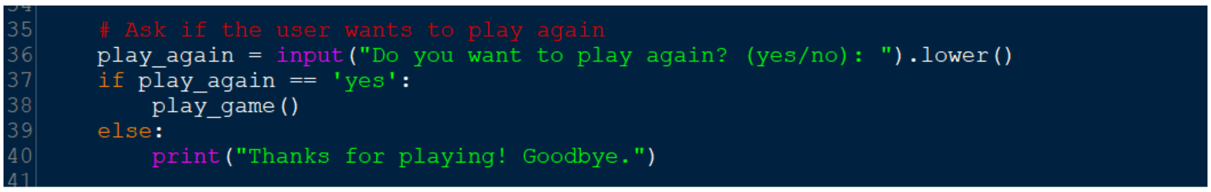
2. **random.shuffle(word_list)**: Randomly shuffles the list of characters.
3. **''.join(word_list)**: Joins the shuffled list of characters back together into a string, which is then returned. The result is a scrambled word (e.g., 'onPyth').



```python
# Main function for the game
def play_game():
    # Choose a random word from the list
    word_to_guess = random.choice(word_list)

    # Scramble the word
    jumbled_word = scramble_word(word_to_guess)

    print("Welcome to the Word Jumble Game!")
    print(f"Here is the jumbled word: {jumbled_word}")

    # Allow the player to guess
    guess = input("Guess the original word: ").lower()

    # Check if the guess is correct (case-insensitive comparison)
    if guess == word_to_guess.lower():
        print("Congratulations! You guessed the word correctly.")
    else:
        print(f"Oops! The correct word was {word_to_guess}.")
```

- **Purpose**: This is the main function that controls the flow of the game.
- **How It Works**:
    1. **random.choice(word_list)**: Selects a random word from the word_list.
    2. **scramble_word(word_to_guess)**: Calls the scramble_word function to scramble the letters of the selected word.
    3. **Display**: It prints a welcome message and shows the scrambled (jumbled) word to the player.
    4. **User Input**: The player is asked to guess the original word. The .lower() method is applied to the guess to make it case-insensitive (so "python" and "Python" are treated the same).
    5. **Guess Check**:
        - The program compares the player's guess with the original word (converted to lowercase using .lower()).
        - If the guess matches, the program prints a congratulatory message.
        - If the guess is incorrect, it tells the player the correct word.

5. Playing Again (Replay Option):



```python
    # Ask if the user wants to play again
    play_again = input("Do you want to play again? (yes/no): ").lower()
    if play_again == 'yes':
        play_game()
    else:
        print("Thanks for playing! Goodbye.")
```

- **Purpose**: After each round, the player is asked if they want to play again.
- **How It Works**:
    1. The user is asked to input "yes" or "no" to decide whether to continue playing.
    2. If the user enters "yes", the play_game() function is called again, starting a new round.
    3. If the user enters "no", a goodbye message is printed, and the game ends.

6. Starting the Game:

play_game()

- **Purpose**: This line starts the game by calling the play_game() function.
- **How It Works**: When the program runs, this is the first thing that gets executed, starting the game loop.
- **Program Flow:**
    1. The game begins by calling play_game().
    2. A random word is selected and scrambled.
    3. The scrambled word is displayed to the player.
    4. The player guesses the original word.
    5. If the guess is correct, the player is congratulated; if it's incorrect, the correct word is revealed.
    6. The player is asked if they want to play again, and the game repeats based on their response.

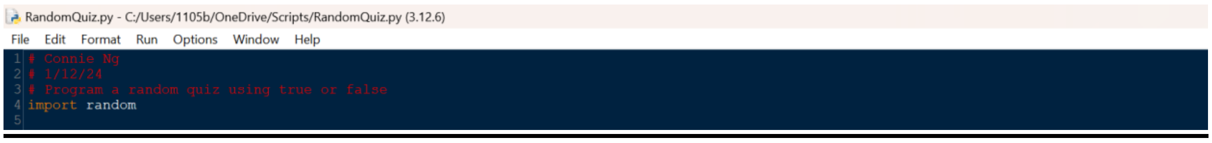
7.End

Finally, your game program is now a fully functional word jumble game that is both fun and flexible. The core functionality ensures that players can enjoy guessing scrambled words, and with the case-insensitive comparison and replay option, the game is user-friendly and interactive.

Program 2 – Random Quiz Game

In this Python code it is for a random simple quiz game that asks questions and checks whether the player's answers are correct.

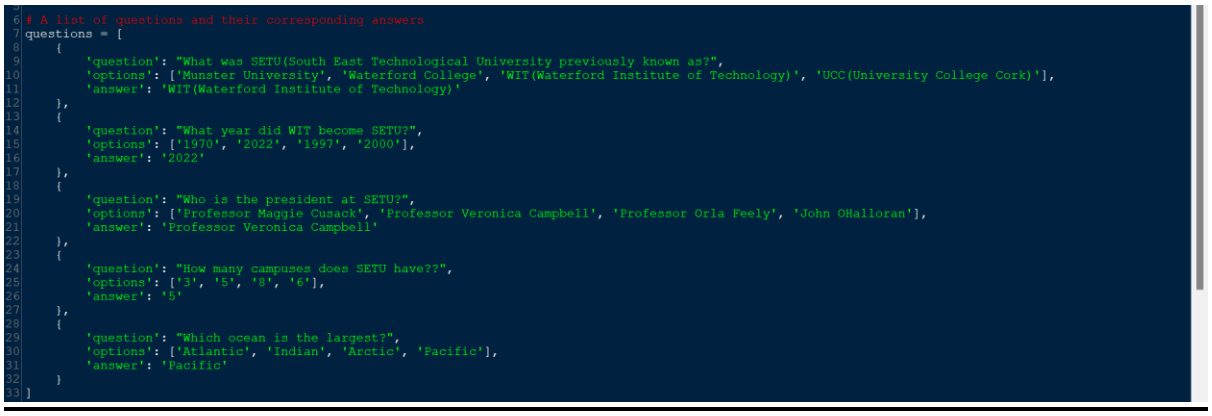
Importing the random Module


RandomQuiz.py - C:/Users/1105b/OneDrive/Scripts/RandomQuiz.py (3.12.6)
File Edit Format Run Options Window Help

```python
# Connie Ng
# 1/12/24
# Program a random quiz using true or false
import random

```

- Purpose: The random module is used to shuffle the order of questions in the quiz, ensuring that the questions appear in a different order each time the quiz is played. This adds variety to the game.

2. List of Questions and Answers:



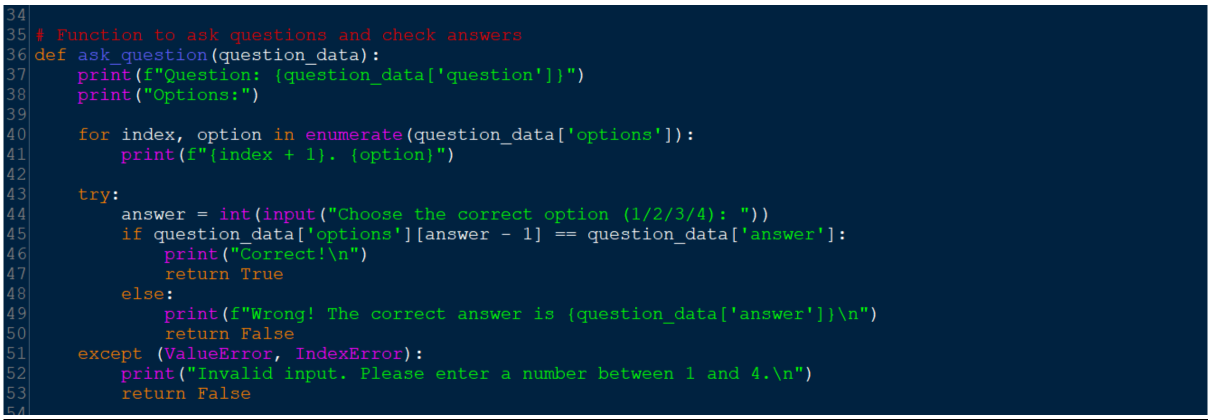
```python
# A list of questions and their corresponding answers
questions = [
    {
        'question': "What was SETU(South East Technological University previously known as?",
        'options': ['Munster University', 'Waterford College', 'WIT(Waterford Institute of Technology)', 'UCC(University College Cork)'],
        'answer': 'WIT(Waterford Institute of Technology)'
    },
    {
        'question': "What year did WIT become SETU?",
        'options': ['1970', '2022', '1997', '2000'],
        'answer': '2022'
    },
    {
        'question': "Who is the president at SETU?",
        'options': ['Professor Maggie Cusack', 'Professor Veronica Campbell', 'Professor Orla Feely', 'John OHalloran'],
        'answer': 'Professor Veronica Campbell'
    },
    {
        'question': "How many campuses does SETU have??",
        'options': ['3', '5', '8', '6'],
        'answer': '5'
    },
    {
        'question': "Which ocean is the largest?",
        'options': ['Atlantic', 'Indian', 'Arctic', 'Pacific'],
        'answer': 'Pacific'
    }
]
```

Purpose: This list contains the quiz questions along with their options and correct answers. Each question is represented as a dictionary, with:

- 'question': The question text.
- 'options': A list of possible answer choices.
- 'answer': The correct answer to the question.

Note: The list includes both general knowledge questions (like "Which ocean is the largest?") and specific questions related to SETU (South East Technological University), making it a themed quiz.

4: ask_question Function:



```python

# Function to ask questions and check answers
def ask_question(question_data):
    print(f"Question: {question_data['question']}")
    print("Options:")

    for index, option in enumerate(question_data['options']):
        print(f"{index + 1}. {option}")

    try:
        answer = int(input("Choose the correct option (1/2/3/4): "))
        if question_data['options'][answer - 1] == question_data['answer']:
            print("Correct!\n")
            return True
        else:
            print(f"Wrong! The correct answer is {question_data['answer']}\n")
            return False
    except (ValueError, IndexError):
        print("Invalid input. Please enter a number between 1 and 4.\n")
        return False
```

Purpose: This function is responsible for displaying the question, its options, and then checking if the player's answer is correct.

- Step 1: It prints the question and the possible options to the screen.

- Step 2: The player is prompted to enter a number between 1 and 4, corresponding to their chosen option.

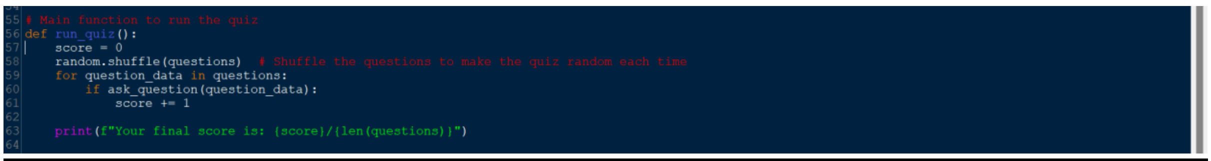
- Step 3: The code checks if the player's answer (from the options list) matches the correct answer.

    - If the answer is correct, it prints "Correct!" and returns True.
    - If the answer is wrong, it prints the correct answer and returns False.

- Error Handling: The function also handles invalid input:

    - If the input is not a number or is out of range (e.g., the player enters "5"), an error message is shown, and the function returns False.

Step 4: run_quiz Function:



```python
# Main function to run the quiz
def run_quiz():
    score = 0
    random.shuffle(questions)  # Shuffle the questions to make the quiz random each time
    for question_data in questions:
        if ask_question(question_data):
            score += 1

    print(f"Your final score is: {score}/{len(questions)}")
```

Purpose: This function is the main driver of the quiz game:

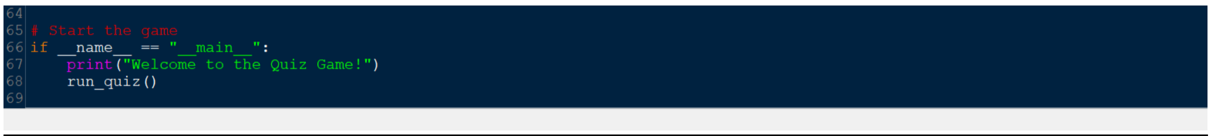
- Step 1: It initializes a score variable to 0, which will keep track of how many questions the player answers correctly.

- Step 2: The questions list is shuffled using random.shuffle() to randomize the order of questions every time the game is played.

- Step 3: The function iterates through each question in the shuffled list and calls the ask_question() function to present the question and check the answer. If the answer is correct (True), the score is incremented.

- Step 4: After all the questions have been asked, the player's final score is displayed, showing how many questions they answered correctly out of the total number of questions.

Step 5: Start the Game:

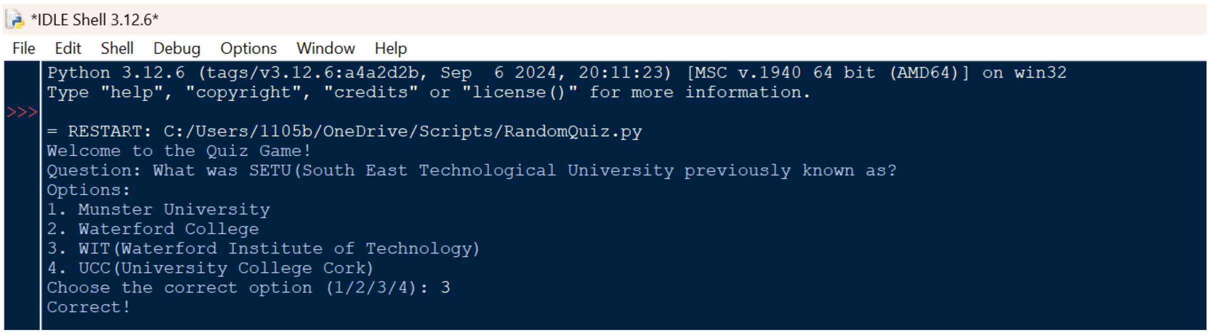


```python

# Start the game
if __name__ == "__main__":
    print("Welcome to the Quiz Game!")
    run_quiz()

```

Purpose: This section ensures that the game runs only when the script is executed directly (not when it is imported as a module in another program). It prints a welcome message and then calls the run_quiz() function to start the game.

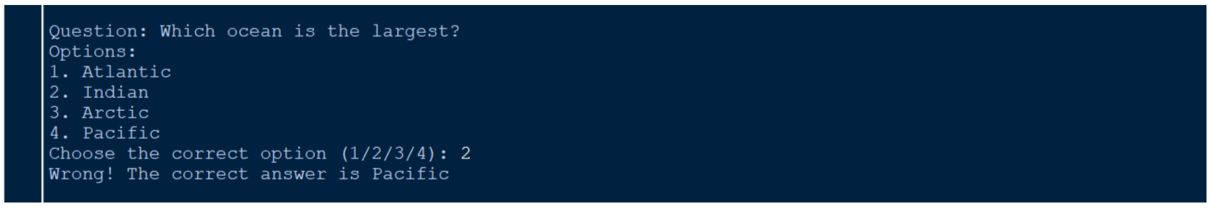
Output of the Game:

If your answer in the game is ‘Correct’ your game output should look like:



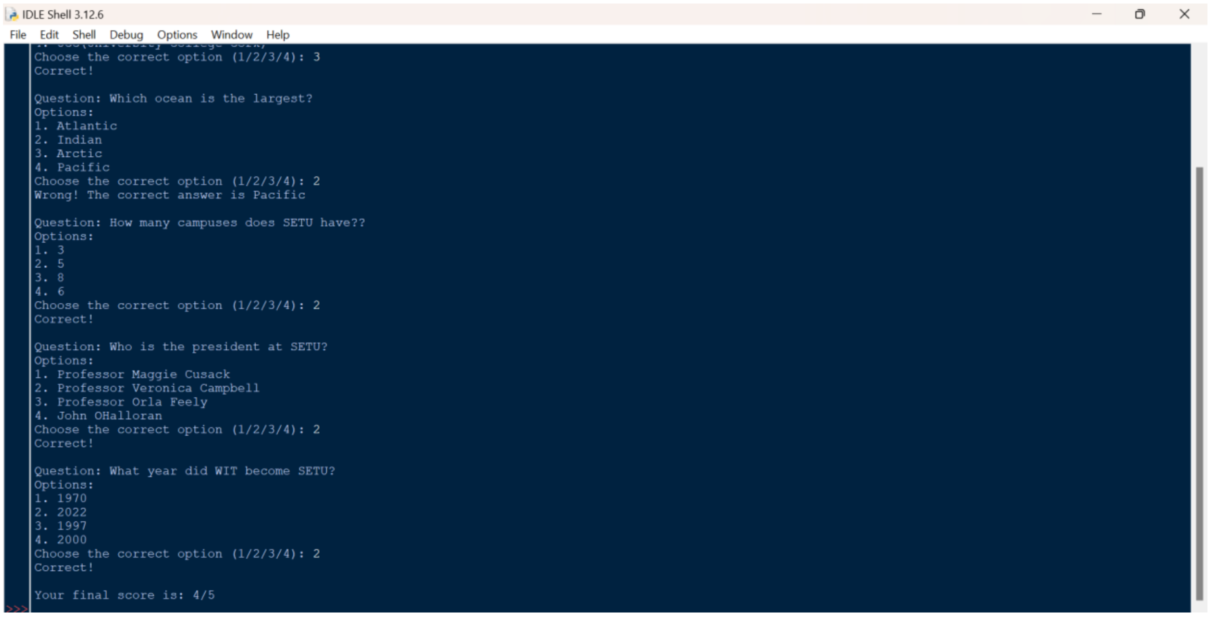
```
Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep  6 2024, 20:11:23) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/1105b/OneDrive/Scripts/RandomQuiz.py
Welcome to the Quiz Game!
Question: What was SETU(South East Technological University previously known as?
Options:
1. Munster University
2. Waterford College
3. WIT(Waterford Institute of Technology)
4. UCC(University College Cork)
Choose the correct option (1/2/3/4): 3
Correct!
```

And if your answer is ‘Incorrect’ your game output should look like:



```
Question: Which ocean is the largest?
Options:
1. Atlantic
2. Indian
3. Arctic
4. Pacific
Choose the correct option (1/2/3/4): 2
Wrong! The correct answer is Pacific
```

When you finish the quiz you should get a final answer of what you got out of 5



```
Choose the correct option (1/2/3/4): 3
Correct!

Question: Which ocean is the largest?
Options:
1. Atlantic
2. Indian
3. Arctic
4. Pacific
Choose the correct option (1/2/3/4): 2
Wrong! The correct answer is Pacific

Question: How many campuses does SETU have??
Options:
1. 3
2. 5
3. 8
4. 6
Choose the correct option (1/2/3/4): 2
Correct!

Question: Who is the president at SETU?
Options:
1. Professor Maggie Cusack
2. Professor Veronica Campbell
3. Professor Orla Feely
4. John OHalloran
Choose the correct option (1/2/3/4): 2
Correct!

Question: What year did WIT become SETU?
Options:
1. 1970
2. 2022
3. 1997
4. 2000
Choose the correct option (1/2/3/4): 2
Correct!

Your final score is: 4/5
>>>
```

Flow of the Program:

1. The game begins with a welcome message: "Welcome to the Quiz Game!"
2. The list of questions is shuffled.

3. The ask_question() function asks the player a series of multiple-choice questions.
4. The player's answer is evaluated:
    - If correct, the score is incremented.
    - If incorrect, the correct answer is displayed.
5. After all the questions, the player's final score is displayed (e.g., "Your final score is: 4/5").
6. The game ends.

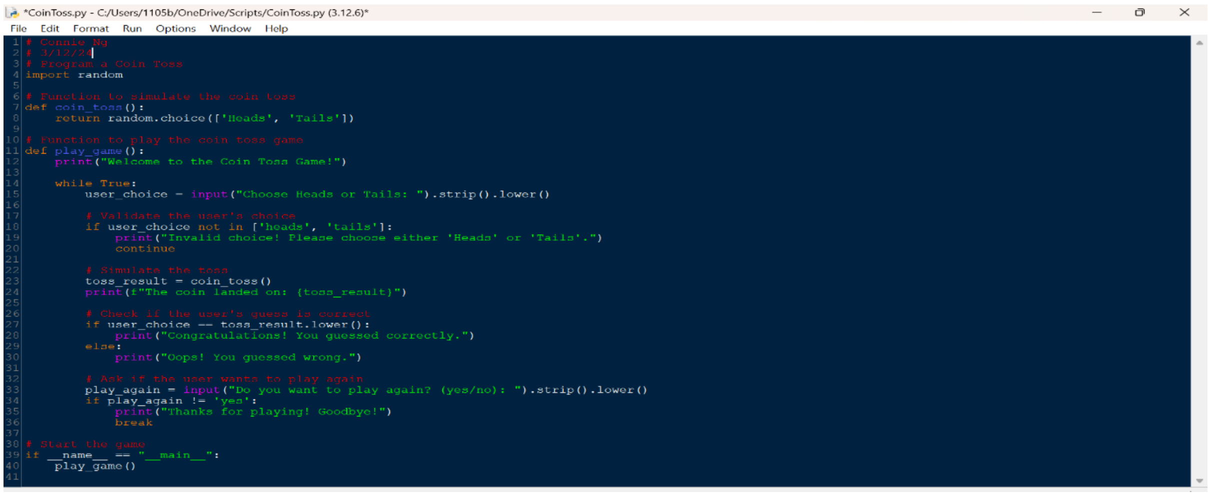
Key Points to Note:

- Dynamic Question Order: Thanks to the use of random.shuffle(), the quiz presents the questions in a random order each time, making the game more engaging.
- Error Handling: The code includes basic input validation, ensuring that the player enters valid responses (1, 2, 3, or 4).
- Score Tracking: The game tracks how many questions the player answers correctly, offering a final score at the end of the quiz.

This Python program creates a simple and engaging quiz game where users answer multiple-choice questions. It tests the player's knowledge and provides feedback on their answers. Through randomization, the quiz keeps things fresh each time it is played. Additionally, while this game doesn't directly use generative AI in the code, AI can be utilized to enhance question generation, feedback, and overall game experience.

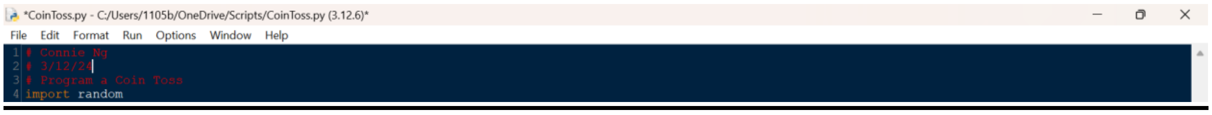
Program 3 – Coin Toss

In this Python program simulates a coin toss. When executed, the program will randomly generate a result of “Heads” or “Tails”, mimicking the behaviour of flipping a real coin. The program uses Python’s random module to ensure fairness and unpredictively in the result. It can be expanded to allow multiple tosses, track outcomes, or include user interaction for added functionality.


CoinToss.py - C:/Users/1105b/OneDrive/Scripts/CoinToss.py (3.12.6)

```python
# Connie Ng
# 3/12/24
# Program a Coin Toss
import random

# Function to simulate the coin toss
def coin_toss():
    return random.choice(['Heads', 'Tails'])

# Function to play the coin toss game
def play_game():
    print("Welcome to the Coin Toss Game!")

    while True:
        user_choice = input("Choose Heads or Tails: ").strip().lower()

        # Validate the user's choice
        if user_choice not in ['heads', 'tails']:
            print("Invalid choice! Please choose either 'Heads' or 'Tails'.")
            continue

        # Simulate the toss
        toss_result = coin_toss()
        print(f"The coin landed on: {toss_result}")

        # Check if the user's guess is correct
        if user_choice == toss_result.lower():
            print("Congratulations! You guessed correctly.")
        else:
            print("Oops! You guessed wrong.")

        # Ask if the user wants to play again
        play_again = input("Do you want to play again? (yes/no): ").strip().lower()
        if play_again != 'yes':
            print("Thanks for playing! Goodbye!")
            break

# Start the game
if __name__ == "__main__":
    play_game()
```

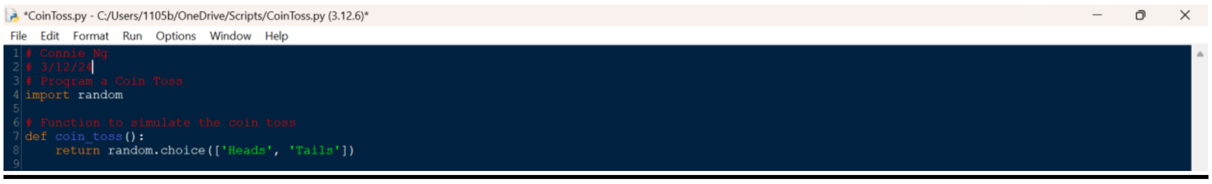
Step 1. Import random


CoinToss.py - C:/Users/1105b/OneDrive/Scripts/CoinToss.py (3.12.6)

```python
# Connie Ng
# 3/12/24
# Program a Coin Toss
import random
```

Purpose: The random module is used to simulate the randomness of the coin toss. It helps us pick a random outcome between "Heads" and "Tails."

Step 2. Coin Toss Fuction

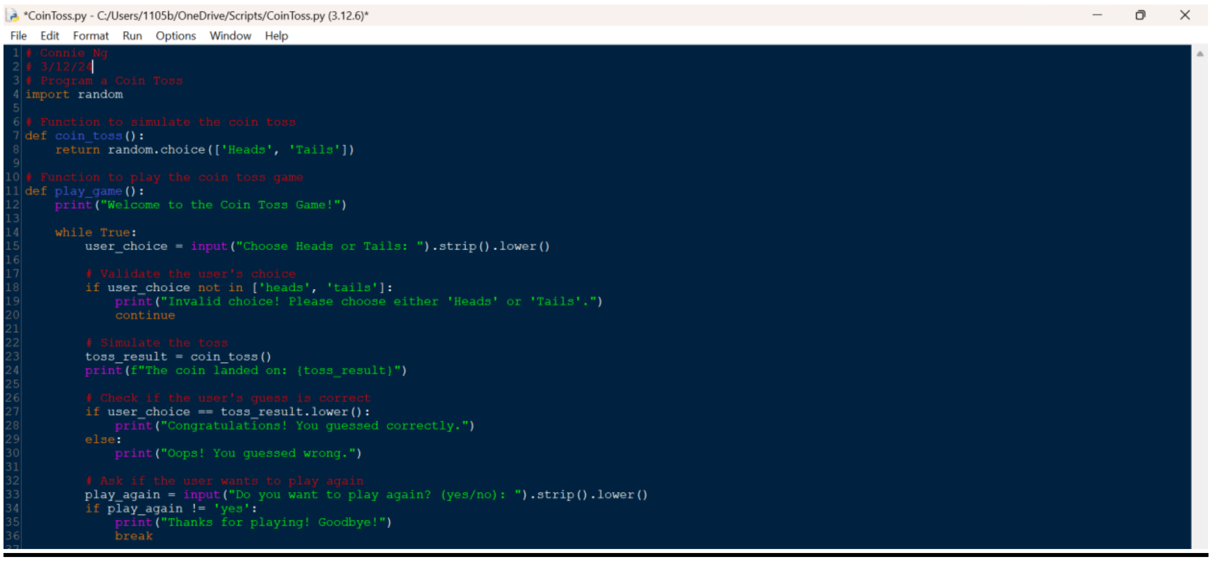

CoinToss.py - C:/Users/1105b/OneDrive/Scripts/CoinToss.py (3.12.6)

```python
# Connie Ng
# 3/12/24
# Program a Coin Toss
import random

# Function to simulate the coin toss
def coin_toss():
    return random.choice(['Heads', 'Tails'])
```

Purpose: This function simulates the coin toss. It uses random.choice() to randomly select one of the two outcomes: 'Heads' or 'Tails'.

How It Works:random.choice() takes a list of possible outcomes (in this case, ['Heads', 'Tails']) and returns a random element from that list.

Step 3. The play_game Function


CoinToss.py - C:/Users/1105b/OneDrive/Scripts/CoinToss.py (3.12.6)
File Edit Format Run Options Window Help

```python
# Connie Ng
# 3/12/24
# Program a Coin Toss
import random

# Function to simulate the coin toss
def coin_toss():
    return random.choice(['Heads', 'Tails'])

# Function to play the coin toss game
def play_game():
    print("Welcome to the Coin Toss Game!")

    while True:
        user_choice = input("Choose Heads or Tails: ").strip().lower()

        # Validate the user's choice
        if user_choice not in ['heads', 'tails']:
            print("Invalid choice! Please choose either 'Heads' or 'Tails'.")
            continue

        # Simulate the toss
        toss_result = coin_toss()
        print(f"The coin landed on: {toss_result}")

        # Check if the user's guess is correct
        if user_choice == toss_result.lower():
            print("Congratulations! You guessed correctly.")
        else:
            print("Oops! You guessed wrong.")

        # Ask if the user wants to play again
        play_again = input("Do you want to play again? (yes/no): ").strip().lower()
        if play_again != 'yes':
            print("Thanks for playing! Goodbye!")
            break
```

Purpose: This is the core function where the game takes place. It prompts the player to choose between "Heads" or "Tails" and compares the player's choice to the result of the coin toss.

How It Works:

User Input: The game asks the user to enter "Heads" or "Tails." The strip().lower() is used to ensure the input is case-insensitive and trimmed of extra spaces.

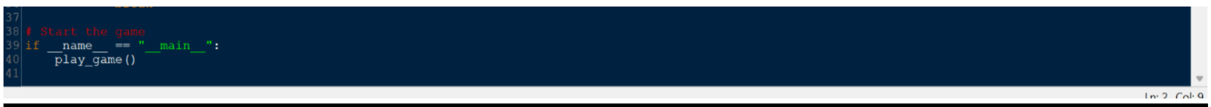
Input Validation: The if statement checks whether the user’s choice is valid (either "heads" or "tails"). If it’s invalid, the game prompts the user again without progressing further.

Coin Toss Simulation: The coin_toss() function is called to get a random result for the coin toss.

Comparison: The program checks if the user's guess matches the toss result. If the user is correct, it prints a success message; otherwise, it prints a failure message.

Replay Option: After each round, the user is asked if they want to play again. If the user answers anything other than 'yes', the game ends.

Step 4 Starting The Game

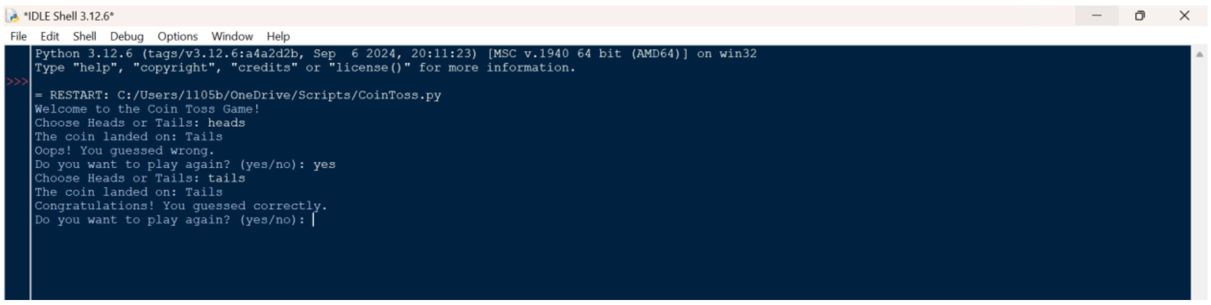


```
# Start the game
if __name__ == "__main__":
    play_game()
```

Purpose: This condition ensures that the game starts only when the script is run directly (not when imported as a module in another program).

How It Works: When the script is executed, it calls the play_game() function to start the game.

Output



```
Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep  6 2024, 20:11:23) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/1105b/OneDrive/Scripts/CoinToss.py
Welcome to the Coin Toss Game!
Choose Heads or Tails: heads
The coin landed on: Tails
Oops! You guessed wrong.
Do you want to play again? (yes/no): yes
Choose Heads or Tails: tails
The coin landed on: Tails
Congratulations! You guessed correctly.
Do you want to play again? (yes/no): 
```

Your final output should look something like this, with an answer od “Oops! You guessed wrong” if your guess was incorrect or “Congratulations! You guessed correctly” if your guess was correct.

AI Usage (Beyond the Coin Toss):

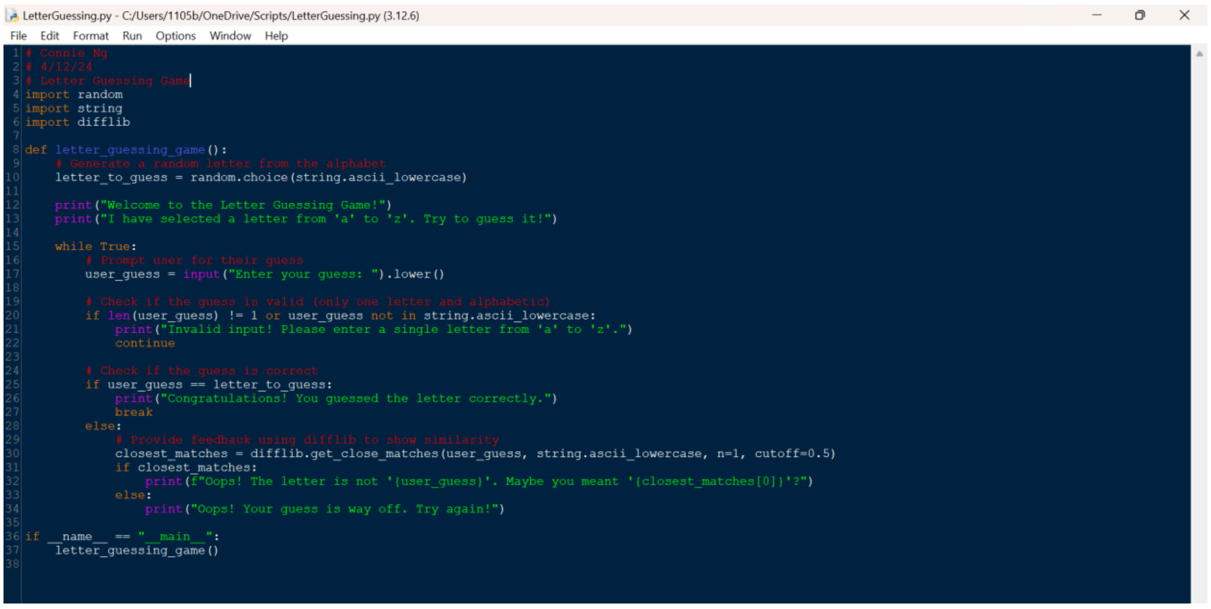
Predictive AI: If the game were extended, an AI component could be used to predict a user’s choices based on patterns in their previous decisions, though this would require more complex data tracking and analysis.

Final Outcome

This simple Coin Toss Game uses basic programming concepts like functions, loops, conditionals, and randomization. Although AI is not heavily involved, the use of randomness in the coin toss mirrors how randomness and probabilistic models are utilized in AI. The game’s core logic is straightforward and easy to follow, providing a good introduction to creating interactive Python applications.

Program 4 – Letter Guessing Game

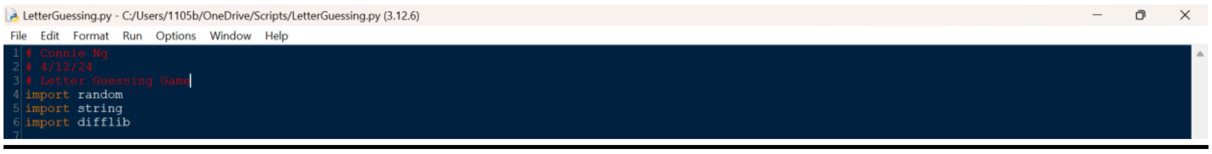
The Letter Guessing Game is a simple yet thrilling game that test your intuition or luck! The program randomly chooses a lower case letter and the user gets to guess which letter the program has randomised. With each guess he or she will get a hint to help navigate you closer to the target. This is how your program should look


LetterGuessing.py - C:/Users/1105b/OneDrive/Scripts/LetterGuessing.py (3.12.6)
File Edit Format Run Options Window Help

```python
# Connie Ng
# 4/12/24
# Letter Guessing Game
import random
import string
import difflib

def letter_guessing_game():
    # Generate a random letter from the alphabet
    letter_to_guess = random.choice(string.ascii_lowercase)

    print("Welcome to the Letter Guessing Game!")
    print("I have selected a letter from 'a' to 'z'. Try to guess it!")

    while True:
        # Prompt user for their guess
        user_guess = input("Enter your guess: ").lower()

        # Check if the guess is valid (only one letter and alphabetic)
        if len(user_guess) != 1 or user_guess not in string.ascii_lowercase:
            print("Invalid input! Please enter a single letter from 'a' to 'z'.")
            continue

        # Check if the guess is correct
        if user_guess == letter_to_guess:
            print("Congratulations! You guessed the letter correctly.")
            break
        else:
            # Provide feedback using difflib to show similarity
            closest_matches = difflib.get_close_matches(user_guess, string.ascii_lowercase, n=1, cutoff=0.5)
            if closest_matches:
                print(f"Oops! The letter is not '{user_guess}'. Maybe you meant '{closest_matches[0]}'?")
            else:
                print("Oops! Your guess is way off. Try again!")

if __name__ == "__main__":
    letter_guessing_game()
```

Explanation of the Program:

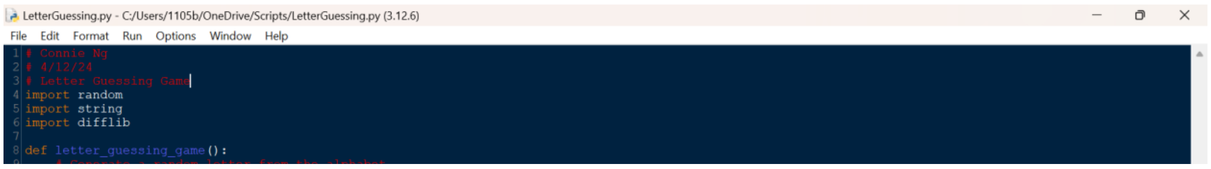
Step 1 Imports:


LetterGuessing.py - C:/Users/1105b/OneDrive/Scripts/LetterGuessing.py (3.12.6)
File Edit Format Run Options Window Help

```python
# Connie Ng
# 4/12/24
# Letter Guessing Game
import random
import string
import difflib
```

- **random**: This module allows us to generate random numbers or select random items. In this program, it's used to randomly choose a letter for the player to guess.
- **string**: The string module contains various useful constants like string.ascii_lowercase, which provides all lowercase letters ('a' to 'z'). We use it to generate the random letter and validate user input.
- **difflib**: This module provides tools for comparing sequences. Specifically, we use difflib.get_close_matches() to suggest letters that are close to the player's guess (if the guess is wrong), providing a hint.

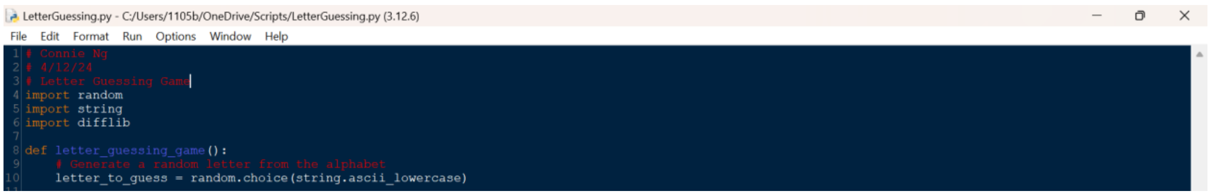
Step 2 Defining the Game Function:


LetterGuessing.py - C:/Users/1105b/OneDrive/Scripts/LetterGuessing.py (3.12.6)
File Edit Format Run Options Window Help

```python
# Connie Ng
# 4/12/24
# Letter Guessing Game
import random
import string
import difflib

def letter_guessing_game():
```

This defines the main game function that will be executed when the game starts. All the game logic is contained within this function.

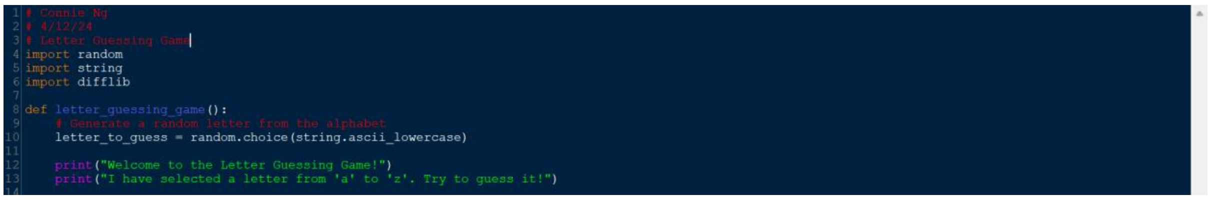
Step 3 Randomly Selecting a Letter:


LetterGuessing.py - C:/Users/1105b/OneDrive/Scripts/LetterGuessing.py (3.12.6)
File Edit Format Run Options Window Help

```python
# Connie Ng
# 4/12/24
# Letter Guessing Game
import random
import string
import difflib

def letter_guessing_game():
    # Generate a random letter from the alphabet
    letter_to_guess = random.choice(string.ascii_lowercase)
```

- **random.choice()**: This function selects a random element from a sequence.
- **string.ascii_lowercase**: This is a predefined constant in the string module that contains all lowercase letters ('a' to 'z').
- Here, random.choice(string.ascii_lowercase) randomly picks one letter from 'a' to 'z' and stores it in the variable letter_to_guess. This is the letter the player has to guess.

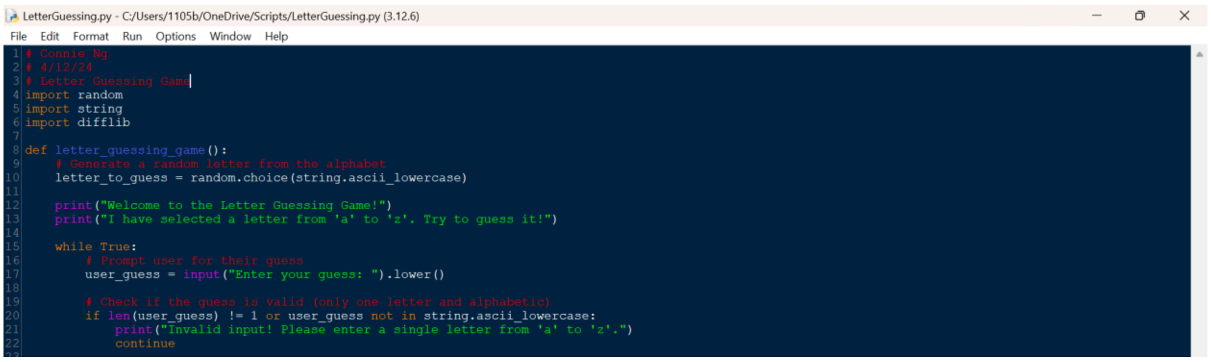
Step 4. Welcome Message and Instructions:



```python
# Connie Ng
# 4/12/24
# Letter Guessing Game
import random
import string
import difflib

def letter_guessing_game():
    # Generate a random letter from the alphabet
    letter_to_guess = random.choice(string.ascii_lowercase)

    print("Welcome to the Letter Guessing Game!")
    print("I have selected a letter from 'a' to 'z'. Try to guess it!")
```

These two print statements greet the player and provide instructions on how to play the game.

Step 5 Game Loop - Player Guessing:


LetterGuessing.py - C:/Users/1105b/OneDrive/Scripts/LetterGuessing.py (3.12.6)
File Edit Format Run Options Window Help

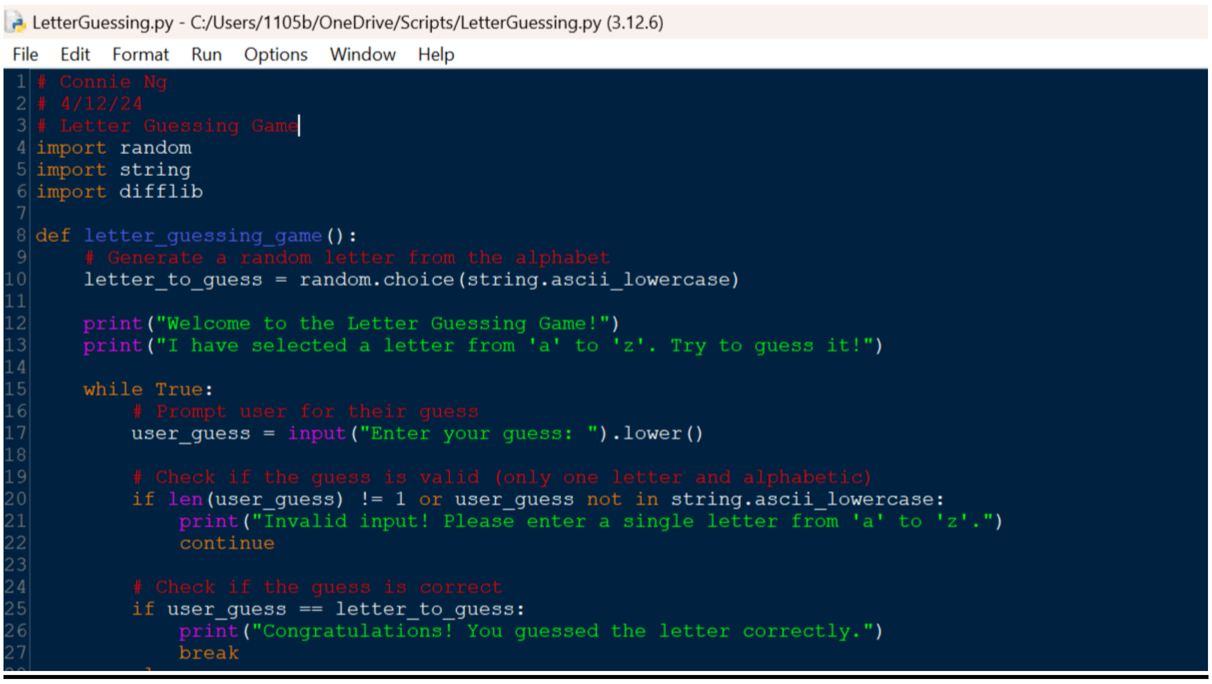
```python
# Connie Ng
# 4/12/24
# Letter Guessing Game
import random
import string
import difflib

def letter_guessing_game():
    # Generate a random letter from the alphabet
    letter_to_guess = random.choice(string.ascii_lowercase)

    print("Welcome to the Letter Guessing Game!")
    print("I have selected a letter from 'a' to 'z'. Try to guess it!")

    while True:
        # Prompt user for their guess
        user_guess = input("Enter your guess: ").lower()

        # Check if the guess is valid (only one letter and alphabetic)
        if len(user_guess) != 1 or user_guess not in string.ascii_lowercase:
            print("Invalid input! Please enter a single letter from 'a' to 'z'.")
            continue
```

- **while True:**: This starts an infinite loop, meaning the game will keep running until the player guesses the correct letter.
- **input()**: This function waits for the user to enter a guess and returns the input as a string.
- **.lower()**: This converts the input to lowercase so that the guess is case-insensitive. For example, if the player enters 'A', it will be treated the same as 'a'.

Step 6 Input Validation

- len(user_guess) != 1: Checks if the user entered more than one character. We only want the user to guess one letter at a time.
- user_guess not in string.ascii_lowercase: Ensures that the input is a valid lowercase letter from 'a' to 'z'.
- If the input is invalid (either not a single letter or not a letter at all), a message is printed, and the loop continues, asking the user to input again.

Step 7. Checking the Guess:

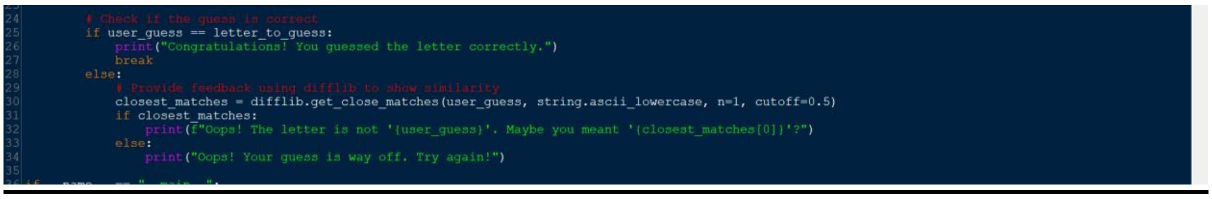

LetterGuessing.py - C:/Users/1105b/OneDrive/Scripts/LetterGuessing.py (3.12.6)
File Edit Format Run Options Window Help

```python
# Connie Ng
# 4/12/24
# Letter Guessing Game
import random
import string
import difflib

def letter_guessing_game():
    # Generate a random letter from the alphabet
    letter_to_guess = random.choice(string.ascii_lowercase)

    print("Welcome to the Letter Guessing Game!")
    print("I have selected a letter from 'a' to 'z'. Try to guess it!")

    while True:
        # Prompt user for their guess
        user_guess = input("Enter your guess: ").lower()

        # Check if the guess is valid (only one letter and alphabetic)
        if len(user_guess) != 1 or user_guess not in string.ascii_lowercase:
            print("Invalid input! Please enter a single letter from 'a' to 'z'.")
            continue

        # Check if the guess is correct
        if user_guess == letter_to_guess:
            print("Congratulations! You guessed the letter correctly.")
            break
```

user_guess == letter_to_guess: Checks if the player’s guess matches the randomly selected letter.

If the guess is correct, a congratulatory message is printed, and the break statement ends the loop, thus ending the game.

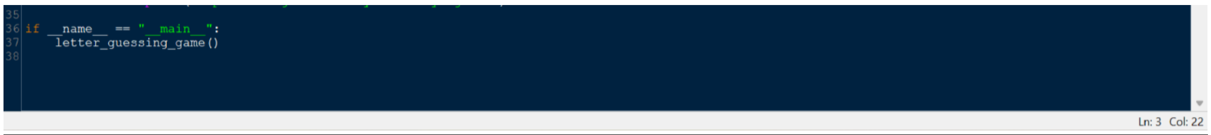
<u>Step 8. Providing Feedback for Incorrect Guesses:</u>



```python
        # Check if the guess is correct
        if user_guess == letter_to_guess:
            print("Congratulations! You guessed the letter correctly.")
            break
        else:
            # Provide feedback using difflib to show similarity
            closest_matches = difflib.get_close_matches(user_guess, string.ascii_lowercase, n=1, cutoff=0.5)
            if closest_matches:
                print(f"Oops! The letter is not '{user_guess}'. Maybe you meant '{closest_matches[0]}'?")
            else:
                print("Oops! Your guess is way off. Try again!")
```

- else:: If the guess is not correct, this block of code is executed.
- difflib.get_close_matches(): This function is used to find the closest matches to the player's guess from a list of possible letters (string.ascii_lowercase).
- user_guess: The player’s guess.
- string.ascii_lowercase: The sequence of letters to compare against.
- n=1: Limits the result to the best match (only one closest letter).
- cutoff=0.5: Specifies the minimum similarity ratio. If the similarity between the guess and the letter is less than 50%, no suggestion will be made.
- If closest_matches is not empty (i.e., there is a match), the program suggests the closest possible letter.
- If no close matches are found (when the guess is very far off), the program tells the user to try again.

<u>Step 9. Starting the Game:</u>



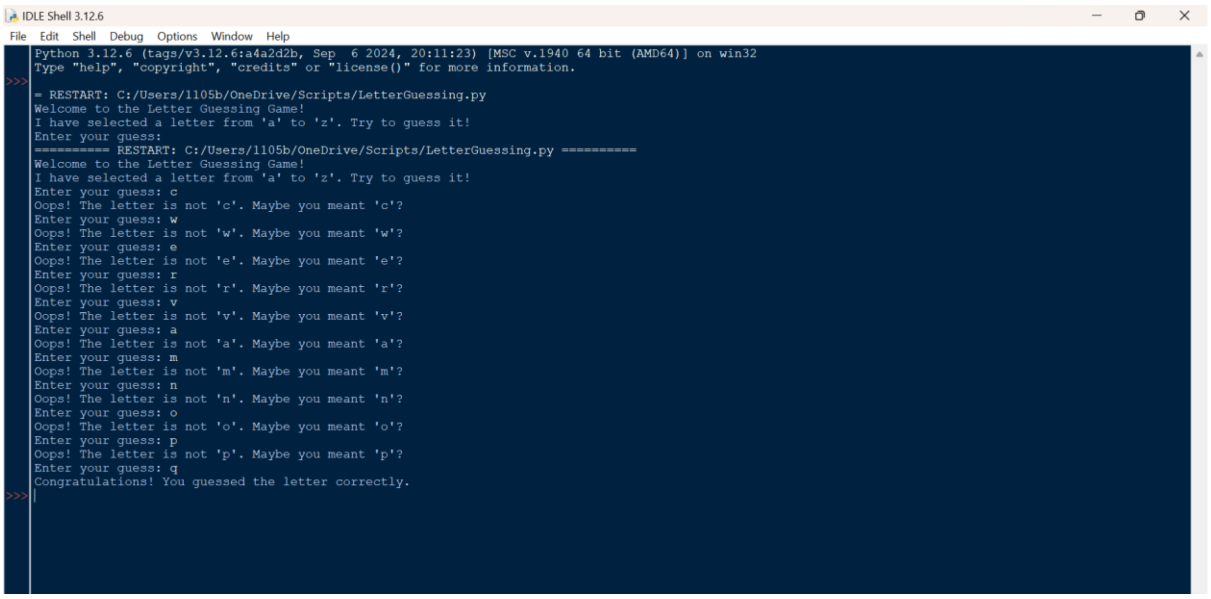
```
if __name__ == "__main__":
    letter_guessing_game()
```

Ln: 3 Col: 22

- if __name__ == "__main__":: This ensures that the letter_guessing_game() function is executed only when the script is run directly (not when imported as a module).
- letter_guessing_game(): This calls the function to start the game

Final Output.

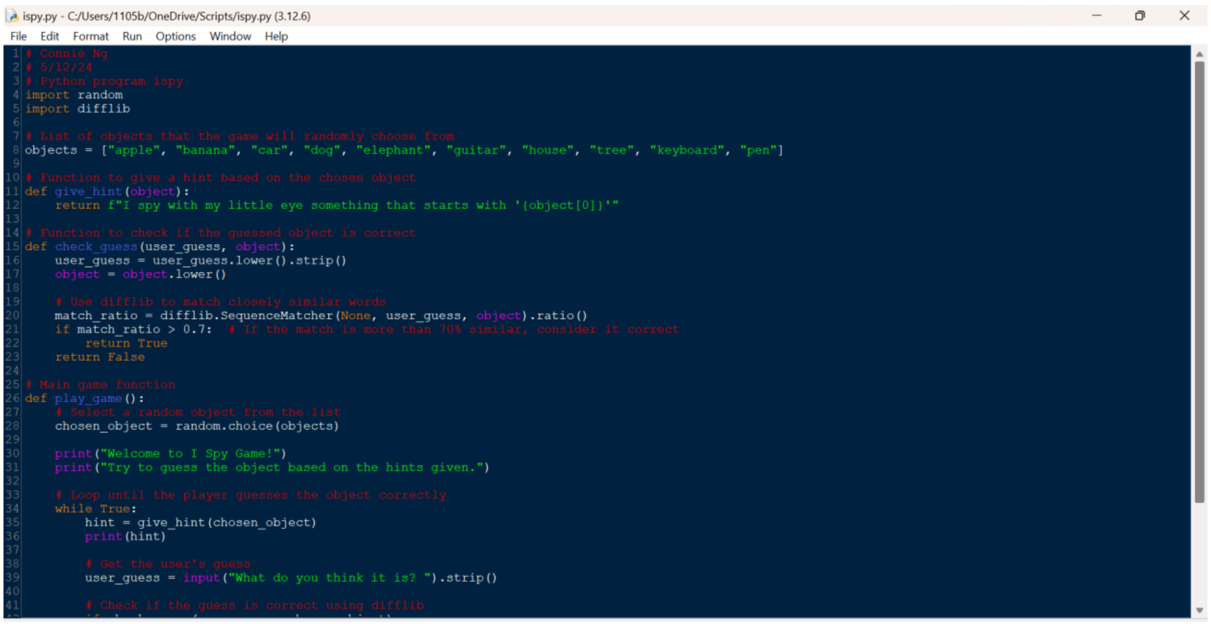
And finally, your output should look something like this



```
Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep  6 2024, 20:11:23) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
= RESTART: C:/Users/1105b/OneDrive/Scripts/LetterGuessing.py
Welcome to the Letter Guessing Game!
I have selected a letter from 'a' to 'z'. Try to guess it!
Enter your guess: 
========== RESTART: C:/Users/1105b/OneDrive/Scripts/LetterGuessing.py ==========
Welcome to the Letter Guessing Game!
I have selected a letter from 'a' to 'z'. Try to guess it!
Enter your guess: c
Oops! The letter is not 'c'. Maybe you meant 'c'?
Enter your guess: w
Oops! The letter is not 'w'. Maybe you meant 'w'?
Enter your guess: e
Oops! The letter is not 'e'. Maybe you meant 'e'?
Enter your guess: r
Oops! The letter is not 'r'. Maybe you meant 'r'?
Enter your guess: v
Oops! The letter is not 'v'. Maybe you meant 'v'?
Enter your guess: a
Oops! The letter is not 'a'. Maybe you meant 'a'?
Enter your guess: m
Oops! The letter is not 'm'. Maybe you meant 'm'?
Enter your guess: n
Oops! The letter is not 'n'. Maybe you meant 'n'?
Enter your guess: o
Oops! The letter is not 'o'. Maybe you meant 'o'?
Enter your guess: p
Oops! The letter is not 'p'. Maybe you meant 'p'?
Enter your guess: q
Congratulations! You guessed the letter correctly.
>>> 
```

Program 5 – Ispy

I spy is a very common game where you look at something and you don’t tell the other player, the other player would then try to guess what you are seeing by giving them hints. For the purpose of this program, we can build a digital version of "I Spy" where the computer picks an object from a list and gives hints to the user, and the user guesses the object based on those clues.

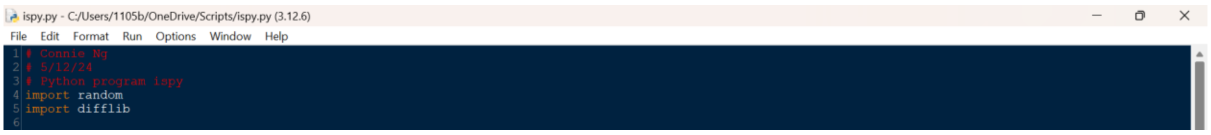

ispy.py - C:/Users/1105b/OneDrive/Scripts/ispy.py (3.12.6)
File Edit Format Run Options Window Help

```python
# Connie Ng
# 5/12/24
# Python program ispy
import random
import difflib

# List of objects that the game will randomly choose from
objects = ["apple", "banana", "car", "dog", "elephant", "guitar", "house", "tree", "keyboard", "pen"]

# Function to give a hint based on the chosen object
def give_hint(object):
    return f"I spy with my little eye something that starts with '{object[0]}'"

# Function to check if the guessed object is correct
def check_guess(user_guess, object):
    user_guess = user_guess.lower().strip()
    object = object.lower()

    # Use difflib to match closely similar words
    match_ratio = difflib.SequenceMatcher(None, user_guess, object).ratio()
    if match_ratio > 0.7:  # If the match is more than 70% similar, consider it correct
        return True
    return False

# Main game function
def play_game():
    # Select a random object from the list
    chosen_object = random.choice(objects)

    print("Welcome to I Spy Game!")
    print("Try to guess the object based on the hints given.")

    # Loop until the player guesses the object correctly
    while True:
        hint = give_hint(chosen_object)
        print(hint)

        # Get the user's guess
        user_guess = input("What do you think it is? ").strip()

        # Check if the guess is correct using difflib
```

<u>Explanation of the Program:</u>

<u>Step 1. Imports:</u>

- random: Used to randomly select an object from the list of objects.
- difflib: A Python library used to compare sequences. We use this to compare the user's guess to the correct answer and determine if the guess is close enough.


ispy.py - C:/Users/1105b/OneDrive/Scripts/ispy.py (3.12.6)
File Edit Format Run Options Window Help

```python
# Connie Ng
# 5/12/24
# Python program ispy
import random
import difflib

```

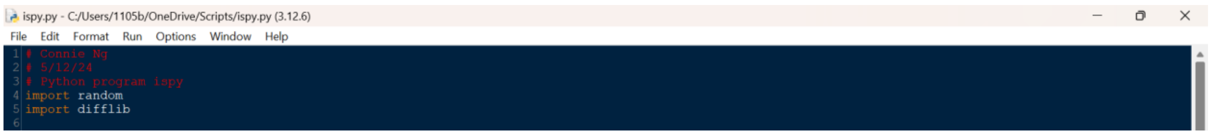
I Spy Game in Python:

The **I Spy** game is a simple guessing game where one player describes an object they see, and the other player tries to guess what it is. For the purpose of this program, we can build a digital version of "I Spy" where the computer picks an object from a list and gives hints to the user, and the user guesses the object based on those clues.

Explanation of the Program:

<u>Step 1. Imports</u>:

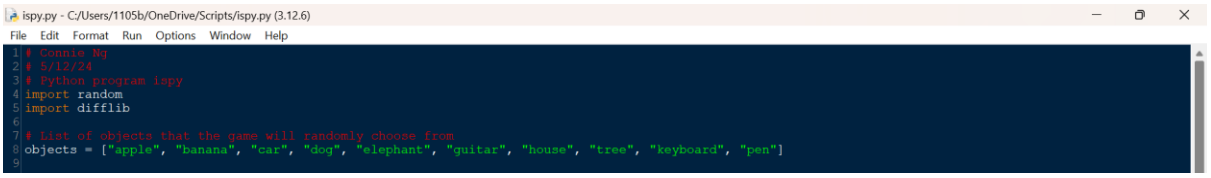
- random: Used to randomly select an object from the list of objects.
- difflib: A Python library used to compare sequences. We use this to compare the user's guess to the correct answer and determine if the guess is close enough.


ispy.py - C:/Users/1105b/OneDrive/Scripts/ispy.py (3.12.6)

```python
# Connie Ng
# 5/12/24
# Python program ispy
import random
import difflib

```

2. Object List:

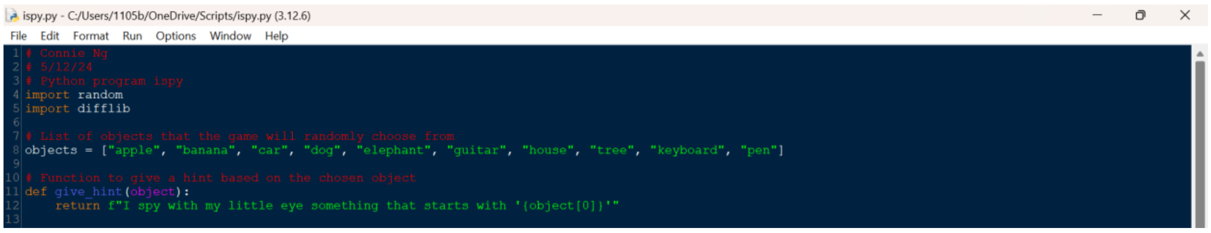
We define a list of random objects (objects) from which the program will randomly pick one for the player to guess.


ispy.py - C:/Users/1105b/OneDrive/Scripts/ispy.py (3.12.6)

```python
# Connie Ng
# 5/12/24
# Python program ispy
import random
import difflib

# List of objects that the game will randomly choose from
objects = ["apple", "banana", "car", "dog", "elephant", "guitar", "house", "tree", "keyboard", "pen"]

```

Step 3. Hint Function (give_hint):

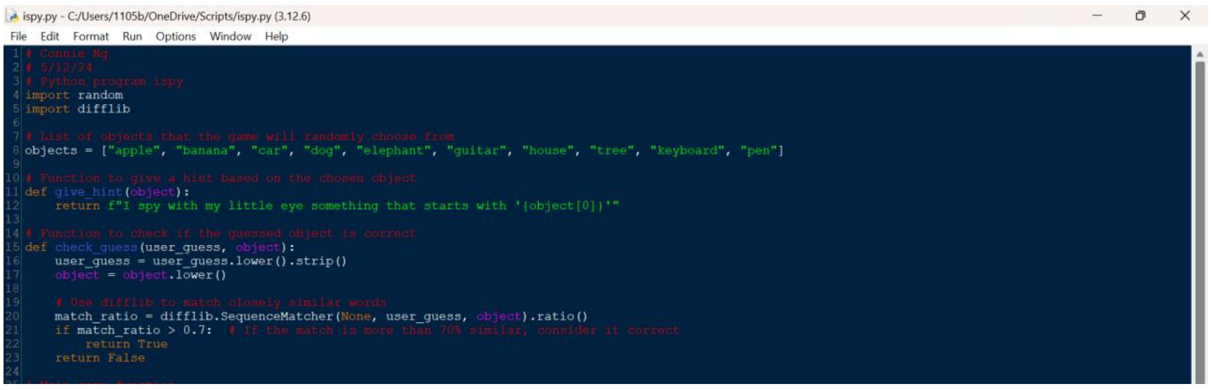
The function give_hint takes an object as input and gives the player a clue based on the first letter of the object. It uses the first character of the object as a hint (e.g., "I spy with my little eye something that starts with 'a'").


ispy.py - C:/Users/1105b/OneDrive/Scripts/ispy.py (3.12.6)

```python
# Connie Ng
# 5/12/24
# Python program ispy
import random
import difflib

# List of objects that the game will randomly choose from
objects = ["apple", "banana", "car", "dog", "elephant", "guitar", "house", "tree", "keyboard", "pen"]

# Function to give a hint based on the chosen object
def give_hint(object):
    return f"I spy with my little eye something that starts with '{object[0]}'"

```

Step 4. Guess Checking Function (check_guess):

The check_guess function compares the user's guess to the correct object. It uses the difflib.SequenceMatcher to check how similar the guess is to the object. If the similarity ratio is greater than 70%, it considers the guess correct.

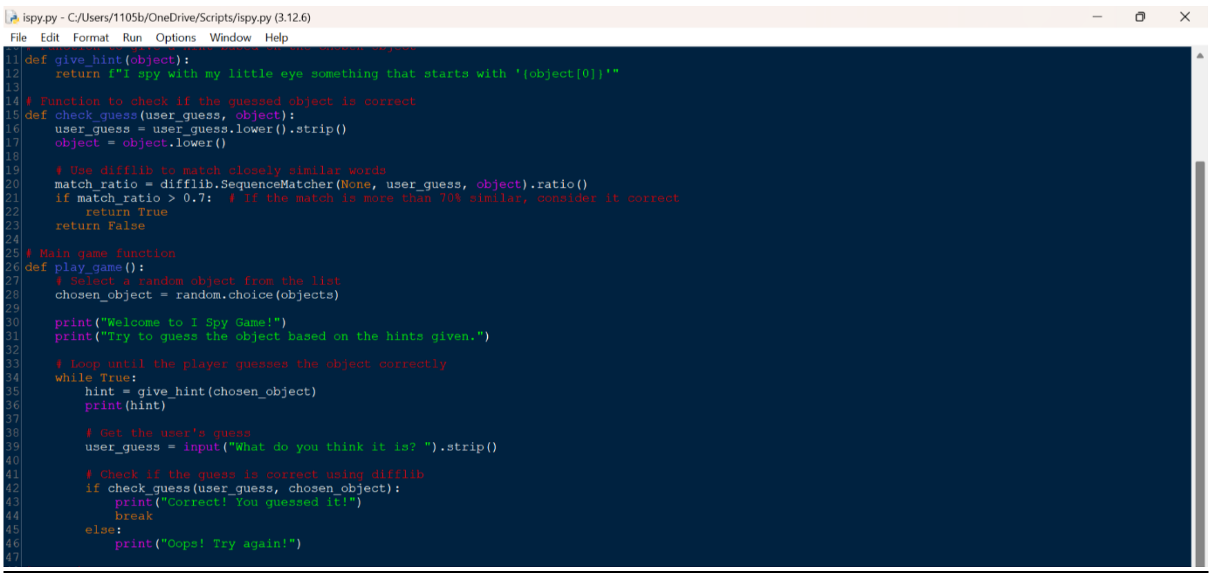

ispy.py - C:/Users/1105b/OneDrive/Scripts/ispy.py (3.12.6)

```python
# Connie Ng
# 5/12/24
# Python program ispy
import random
import difflib

# List of objects that the game will randomly choose from
objects = ["apple", "banana", "car", "dog", "elephant", "guitar", "house", "tree", "keyboard", "pen"]

# Function to give a hint based on the chosen object
def give_hint(object):
    return f"I spy with my little eye something that starts with '{object[0]}'"

# Function to check if the guessed object is correct
def check_guess(user_guess, object):
    user_guess = user_guess.lower().strip()
    object = object.lower()

    # Use difflib to match closely similar words
    match_ratio = difflib.SequenceMatcher(None, user_guess, object).ratio()
    if match_ratio > 0.7:  # If the match is more than 70% similar, consider it correct
        return True
    return False

```

- difflib.SequenceMatcher: This class is used to compare two strings and determine how similar they are. It returns a ratio between 0 and 1, with 1 meaning the strings are identical.
- We use a threshold of 0.7 (70% similarity) to consider guesses that are close enough to be correct, allowing for slight typos or misspellings.

Step 5 Main Game Loop (play_game):

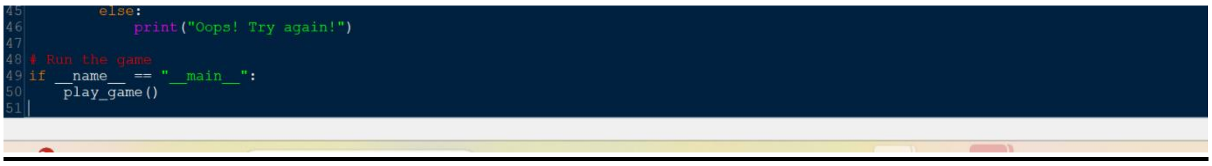
- In the play_game function, the game starts by picking a random object from the list.
- The program continuously provides hints to the player and asks for guesses until the player correctly guesses the object.
- The user inputs their guess, and the program checks if the guess is correct by calling the check_guess function.
- If the guess is correct, the game ends with a success message. If not, the game asks the user to try again.


ispy.py - C:/Users/1105b/OneDrive/Scripts/ispy.py (3.12.6)
File Edit Format Run Options Window Help

```python
def give_hint(object):
    return f"I spy with my little eye something that starts with '{object[0]}'"

# Function to check if the guessed object is correct
def check_guess(user_guess, object):
    user_guess = user_guess.lower().strip()
    object = object.lower()

    # Use difflib to match closely similar words
    match_ratio = difflib.SequenceMatcher(None, user_guess, object).ratio()
    if match_ratio > 0.7:  # If the match is more than 70% similar, consider it correct
        return True
    return False

# Main game function
def play_game():
    # Select a random object from the list
    chosen_object = random.choice(objects)

    print("Welcome to I Spy Game!")
    print("Try to guess the object based on the hints given.")

    # Loop until the player guesses the object correctly
    while True:
        hint = give_hint(chosen_object)
        print(hint)

        # Get the user's guess
        user_guess = input("What do you think it is? ").strip()

        # Check if the guess is correct using difflib
        if check_guess(user_guess, chosen_object):
            print("Correct! You guessed it!")
            break
        else:
            print("Oops! Try again!")

```

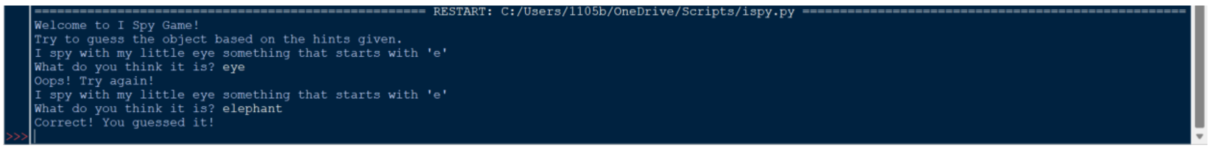
Step 6. Game Execution:

The game is executed in the __main__ block by calling play_game(). This ensures the game runs when the program is executed.



```python
        else:
            print("Oops! Try again!")

# Run the game
if __name__ == "__main__":
    play_game()

```

Final Output



```
==================================================== RESTART: C:/Users/1105b/OneDrive/Scripts/ispy.py ====================================================
Welcome to I Spy Game!
Try to guess the object based on the hints given.
I spy with my little eye something that starts with 'e'
What do you think it is? eye
Oops! Try again!
I spy with my little eye something that starts with 'e'
What do you think it is? elephant
Correct! You guessed it!
>>>
```

After you finish adding your code into python and run it, you should have a result that looks like this.

Summary:

This I Spy game is a simple guessing game in which the computer provides clues, and the player guesses the object based on those clues. The program uses difflib to measure the similarity of the user's guess, allowing for tolerance of minor spelling mistakes. While basic, the program can be enhanced with AI techniques such as NLP or machine learning for more advanced gameplay.