MINOR PROJECT

A GAME USING PYTHON

Guess The Number

Objective: Create a command-line game where the player tries to guess a randomly chosen number within a given range. The game provides feedback on each guess and informs the player when they win or lose.

Technology:

Python program to create a “Number Guessing Game using Python”. It is a program (python) based game which contain basics of python such as user input, while condition, if- else condition.

Requirements:

- Random number generation.
- User input handling.
- Feedback mechanism.
- Game loop with a win/lose condition.

Feedback: Provide feedback on whether the guess is too high, too low, or correct.

End Condition: The game ends when the player guesses correctly or runs out of attempts.

Steps to create a “Guess a Number” game using Python:

- Step-1: Set up the Environment
- Step-2: Create the Python file
- Step-3: Import Libraries
- Step-4: Generate a random number
- Step-5: Create a game loop and get player input
- Step-6: Check player guess
- Step-7: Display game result while running the code

How It Works:

1. The first player thinks of an integer within a known range.
2. The second player tries to guess a number.
3. If the guess is incorrect, then the first player tells the second player whether the guess was too high or too low.
4. Eventually, the second player guesses the correct number.
5. The second players score equals the number of guesses he made.
6. The players then reverse their roles and repeat the game.

7. The winner is the player who gets the correct answer with the fewest guesses.

SOURCE CODE

```
import random
def guess_the_number():
    print ("Welcome to 'Guess the Number'!")
    print ("I'm thinking of a number between 1 and 100.")
    number_to_guess = random.randint(1, 100)
    guess = None
    attempts = 0
    max_attempts = 10
    while guess !=number_to_guess and attempts < max_attempts:
        try:
         guess = int (input ("Enter your guess: "))
            attempts += 1
         if guess < number_to_guess:
            print ("Too low! Try again.")
         elif guess > number_to_guess:
            print ("Too high! Try again.")
         else:
```

```
            print (f"Congratulations! You've guessed the number {number_to_guess} correctly in {attempts} attempts.")
            break
        except ValueError:
            print ("Invalid input. Please enter a valid number.")

    if guess != number_to_guess:
        print (f"Sorry, you've used all {max_attempts} attempts. The number was {number_to_guess}. Better luck next time!")

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

Explanation of the Code:-

1. **Imports and Initialization:**
   - import random is used to generate a random number.
   - number_to_guess is a random integer between 1 and 100.
   - guess stores the player's current guess.

- attempts track the number of guesses the player has made.
- max_attempts are the maximum number of guesses allowed.

2. **Game Loop:**
   - The game continues while the player hasn't guessed the number and hasn't used all attempts.
   - User input is taken and checked if it is an integer. If not, an error message is displayed.
   - Feedback is given based on whether the guess is too high, too low, or correct.
3. **End of the Game:**
   - If the player guesses correctly, they are congratulated.
   - If the player runs out of attempts, the correct number is revealed.

Test the Game: -

- **Test with Various Inputs:** Ensure that the game handles numbers within and outside the range, invalid inputs, and correct guesses.
- **Check Feedback Messages:** Make sure the feedback messages are accurate and informative.

- **Verify End Conditions:** Ensure the game ends appropriately when the player guesses correctly or runs out of attempts.

Enhancements and Variations

1. **Difficulty Levels:**
    - Implement different difficulty levels with varying ranges and number of attempts.
2. **Replay Option:**
    - Add an option to play the game again without restarting the script.
3. **High Scores:**
    - Track and display high scores based on the number of attempts used.
4. **Hints:**
    - Provide additional hints after a certain number of incorrect guesses.
5. **GUI Version:**
    - Use a library like Tkinter to create a graphical version of the game.

guess.py - C:/Users/divyalekhya/OneDrive/文档/guess.py (3.12.4)

File Edit Format Run Options Window Help

```
import random

def guess_the_number():
    print("Welcome to 'Guess the Number'!")
    print("I'm thinking of a number between 1 and 100.")
    number_to_guess = random.randint(1, 100)
    guess = None
    attempts = 0
    max_attempts = 10
    while guess != number_to_guess and attempts < max_attempts:
          try:
           guess = int(input("Enter your guess: "))
           attempts += 1
           if guess < number_to_guess:
               print("Too low! Try again.")
           elif guess > number_to_guess:
               print("Too high! Try again.")
           else:
               print(f"Congratulations! You've guessed the number {number_to_guess} correctly in {attempts} attempts.")
               break
          except ValueError:
           print("Invalid input. Please enter a valid number.")

    if guess != number_to_guess:
       print(f"Sorry, you've used all {max_attempts} attempts. The number was {number_to_guess}. Better luck next time!")

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

IDLE Shell 3.12.4

File Edit Shell Debug Options Window Help

```
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/divyalekhya/OneDrive/文档/guess.py ==========
Welcome to 'Guess the Number'!
I'm thinking of a number between 1 and 100.
Enter your guess: 76
Too high! Try again.
Enter your guess: 67
Too high! Try again.
Enter your guess: 56
Too low! Try again.
Enter your guess: 60
Too low! Try again.
Enter your guess: 58
Too low! Try again.
Enter your guess: 59
Too low! Try again.
Enter your guess: 62
Too low! Try again.
Enter your guess: 65
Too high! Try again.
Enter your guess: 64
Congratulations! You've guessed the number 64 correctly in 9 attempts.
>>>
>>>
========== RESTART: C:/Users/divyalekhya/OneDrive/文档/gues
=
Welcome to 'Guess the Number'!
I'm thinking of a number between 1 and 100.
Enter your guess: 78
Too high! Try again.
Enter your guess: 65
Too low! Try again.
Enter your guess: 45
Too low! Try again.
Enter your guess: 55
Too low! Try again.
Enter your guess: 56
Too low! Try again.
Enter your guess: 67
Too low! Try again.
Enter your guess: 70
Too low! Try again.
Enter your guess: 72
Too low! Try again.
Enter your guess: 73
Too low! Try again.
Enter your guess: 75
Too low! Try again.
Sorry, you've used all 10 attempts. The number was 77. Better luck next time!
>>>
```

FUTURE SCOPE

1. Multiplayer Options

a. Local Multiplayer:

Allow multiple players to take turns guessing in the same session, or create a competitive mode where players can guess in turns.

b. Online Multiplayer:

Implement an online multiplayer feature where players can challenge each other to guess the number. This can involve server-client communication using libraries like socket or frameworks like Flask or Django.

2. Advanced Features

a. Leaderboard and High Scores:

- Maintain a leaderboard or high score system to track top scores and player performance. You can store scores in a database or file.

b. Statistics and Analysis:

- Provide detailed statistics on the player's performance, such as average number of attempts, win rate, and other metrics.

c. Customizable Game Settings:

- Allow players to customize game settings such as the range of numbers, number of attempts, and the appearance of the game interface.

3. Educational and Learning Tools

a. Tutorial Mode:

- Add a tutorial mode that guides new players through the rules and strategies of the game.

b. Practice Mode:

- Implement a practice mode where players can train with different ranges and difficulties to improve their guessing skills.

4. Integration with Other Technologies

a. Voice Commands:

- Integrate voice recognition using libraries like SpeechRecognition to allow players to make guesses using voice commands.

b. Machine Learning:

- Explore machine learning by implementing a system that learns from player behaviour and provides smarter hints or adjusts difficulty dynamically.

5. Community and Social Features

a. Sharing and Social Media Integration:

- Allow players to share their scores or challenge friends via social media.

b. Achievements and Badges:

- Implement an achievement system where players earn badges for completing certain challenges or milestones.

6. Deployment and Distribution

a. Packaging for Distribution:

- Package the game as a standalone executable using tools, making it easier for users to install and play without needing a Python interpreter.

b. Mobile App:

- Develop a mobile version of the game using frameworks to reach a broader audience.

Result: The "Guess the Number" game, as implemented in Python, provides a straightforward and engaging experience for players. It effectively demonstrates fundamental programming concepts, including Random Number Generation, User Input Handling, Control Flow, Exception Handling.

Conclusion: Overall, the "Guess the Number" game serves as a foundational project that can be expanded into more complex and feature-rich applications, making it a valuable learning experience and a springboard for further development.