Mini Project -1 (Number Guessing Game)

(Language PYTHON)

AIM:

- Build a number Guessing Game in which the user selects a range.
- Assume the user selected a range from X to Y where both X and Y are integers.
- So a random number in that range is selected by the system where the user needs to guess the number in minimum number of guesses.

[Note: There Should be a google doc with proper algorithm, pseudo code, analysis of the problem statement, code, all should be uploaded into Git Repository.**]**

1. Problem Statement:

Clearly define the problem: a number guessing game where the user selects a range (X to Y) and the system generates a random number within that range. The user needs to guess the number with the minimum number of attempts.

2. Algorithm:

Describe a Binary Search algorithm:

1. The system calculates the middle point of the range (X + Y) / 2.
2. If the guess is higher than the middle point, search the upper half of the remaining range (update X to the middle point + 1).
3. If the guess is lower than the middle point, search the lower half of the remaining range (update Y to the middle point - 1).
4. Repeat steps 2a and 2b until the guess matches the generated number.

3. Pseudocode:

```
-> function guessNumber(X, Y) // Generate a random number between X and Y (inclusive)
->         secretNumber = random(X, Y)
->         while (true)
->             guess = (X + Y) / 2 // Calculate the middle point
->             if (guess == secretNumber)
->                 break;   // Guessed correctly, exit loop
->             else if (guess > secretNumber)
->                 Y = guess - 1; // Search lower half
->             else X = guess + 1; // Search upper half
->         end while

->             return "You guessed the number in " + (Y - X + 1) + " tries!"
->  end function
```

4. Implementation[code]:

```
import random
import math
# Taking Inputs
lower = int(input("Enter Lower bound:- "))

# Taking Inputs
upper = int(input("Enter Upper bound:- "))

# generating random number between
# the lower and upper
x = random.randint(lower, upper)
print("\n\tYou've only ",
      round(math.log(upper - lower + 1, 2)),
      " chances to guess the integer!\n")

# Initializing the number of guesses.
count = 0

# for calculation of minimum number of guesses depends upon range
while count < math.log(upper - lower + 1, 2):
    count += 1

    # taking guessing number as input
    guess = int(input("Guess a number:- "))

    # Condition testing
    if x == guess:
        print("Congratulations you did it in ",
              count, " try")
        # Once guessed, loop will break
        break
    elif x > guess:
        print("You guessed too small!")
    elif x < guess:
        print("You Guessed too high!")

# If Guessing is more than required guesses, show this output.
if count >= math.log(upper - lower + 1, 2):
    print("\nThe number is %d" % x)
    print("\tBetter Luck Next time!")

```

Output:

```
PS D:\DAA> python daa.py
Enter Lower bound:- 3
Enter Upper bound:- 4

        You've only  1  chances to guess the integer!

Guess a number:- 5
You Guessed too high!

The number is 3
        Better Luck Next time!
```

5. Problems & Analysis:

- **Generating random number:** The `random.randint` function ensures a random number within the specified range.
- **Giving correct range:** The code validates user input to ensure the lower bound is less than the upper bound.
- **Limited guesses:** While not explicitly implemented here, you can add a maximum number of attempts before ending the game.
- **Giving hints:** You can modify the code to provide hints like "Higher" or "Lower" based on the user's guess.

6. Test Cases:

- **Wrong Range**: The code handles this with a `try-except` block and prompts the user to enter a valid range.
- **Correct Range, Wrong Guess**: The loop continues until the user guesses correctly, displaying the number of attempts.
- **Correct Range, Correct Guess**: The game ends successfully, displaying the number of attempts (should be minimal due to Binary Search).

Git Repository:

- Create a new Git repository on a platform like Github or GitLab.
- Add your Google Doc and any code files to the repository.
- Commit your changes and push them to the remote repository.