Mini Project-1

(Number Guessing Game)
(Language PYTHON)

Problem Statement:

The number Guessing game in Python is a common project for programmers who have recently learned a new language and mastered topics like number generation and conditional statements with iteration.

Analysis:

There are a few tasks that we need to take care of while building a number-guessing game using Python. First of all, we need the user to select a range of numbers in which the number-guessing game will be played. Once the user has selected the range from X to Y, where X and Y are integers. Some random number will be selected by the integer and the user has to guess that integer in the minimum number of guesses.

So, How do we find this minimum number of guesses? Well, that minimum number of guesses depends upon range and we already have the formula for it.

Minimum number of guessing = log2(Upper bound – lower bound + 1)

Now, we have every possible information required for creating our Number Guessing game in Python. So, let's understand its Algorithmic flow.

Algorithm:

An algorithm is a step-by-step procedure used for solving a problem or performing a task. In the context of the number guessing game described above, the algorithm involves the following steps:

- Step1: The range's lower and upper bounds are taken as input by the user.
- Step 2: A random integer between the range is generated by the Python compiler and stored in a variable for later use.

- Step 3: An initialized while loop will be used for repeated guessing.
- Step 4: The user receives the message “Try Again! You guessed too high” if the guesses are higher than the randomly chosen number.
- Step 5: Else The user receives the output “Try Again! You guessed too small” if the user’s guess is less than the randomly chosen number.
- Step 6: And if the user correctly predicted the answer in the required number of attempts, they are congratulated.
- Step 7: At the end, the user will receive the output “Better Luck next time” if they failed to correctly guess the integer in the required number of attempts.

Pseudo Code:

```
function guessNumber(X, Y):

 # Step 1: System selects a random number N in the range [X, Y]
     N = random(X, Y)

 # Step 2: Initialize the low and high pointers for binary search
      low = X
      high = Y
      attempts = 0
      while True:

# Step 3: User makes a guess
     guess = (low + high) // 2 # User's guess using binary search
     attempts += 1

# Step 4: System provides feedback
 if guess < N:
     print("Too low")
     low = guess + 1

# Adjust the range to [guess + 1, high]
elif guess > N:
     print("Too high")
     high = guess - 1
 # Adjust the range to [low, guess - 1]
else:
   print("Correct! You guessed the number in", attempts, "attempts.")
   Break
```

Time Complexity:

The time complexity of this code is **O(n)** as the number of iterations of the loop is not fixed and depends on the number of guesses taken by the user to get the right answer.

Space Complexity:

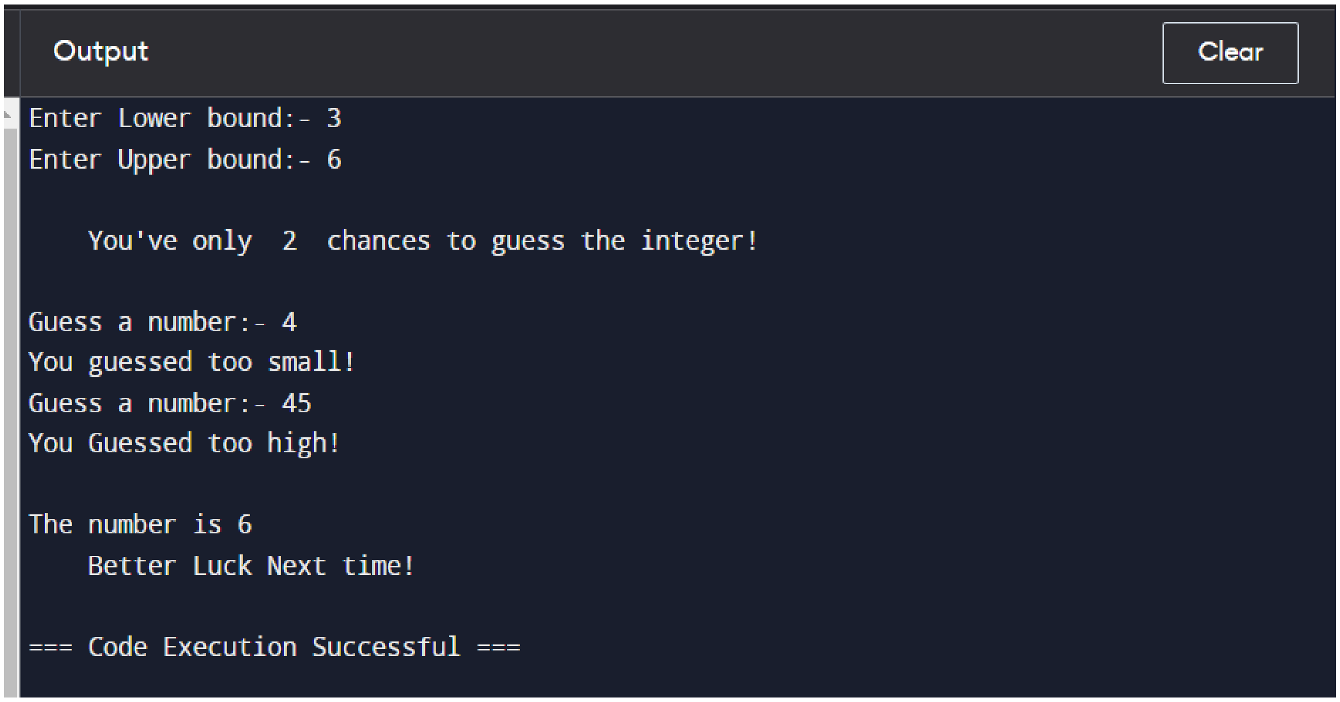
The space complexity of this code is **O(1)** as all the variables used in this code are of the same size and no extra space is required to store the values.

Python 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:



```
Enter Lower bound:- 3
Enter Upper bound:- 6

	You've only  2  chances to guess the integer!

Guess a number:- 4
You guessed too small!
Guess a number:- 45
You Guessed too high!

The number is 6
	Better Luck Next time!

=== Code Execution Successful ===
```

Conclusion:

Although we created a simple number guessing game in Python we got to learn a lot of things while building this project.

- Firstly, we learned about Binary Search Upper and Lower Bound, followed by the use of while loop in Python.
- Finally, we got to brainstorm and as a result, we understood the thought process while building a project from scratch.