# AI-Based Number Guessing Game

## Report


**Name: Shreya Mittal**

**Class Roll No : 20**

**Univ Roll No : 202401100300240**

**Branch: CSE AI**

**Library ID: 2428CSEAI268**



**Date: 11 March 2025**

# INTRODUCTION

The goal of this project is to create an AI-powered number guessing game where the AI attempts to guess a number that the user has thought of. The number is within a predefined range (1 to 100), and the AI uses a guessing algorithm to efficiently guess the number.

In the AI-based number guessing game, the AI's task is to repeatedly guess a number and refine its guess based on the user's feedback. The user provides feedback on whether each guess is too high, too low, or correct. Based on this feedback, the AI will narrow down the possible numbers and guess the correct one in the fewest possible attempts.

# METHADOLOGY

The code employs a strategy called **binary search** (or a variation of it) to efficiently find the target number. Here's how it works:

1. **Initial Range:** The AI starts with a search range from 1 to 100 (low and high variables).
2. **Guessing in the Middle:** In each iteration of the loop, the AI makes a guess by selecting a random number within the current search range (low to high). This helps to quickly narrow down the possibilities.
3. **Adjusting the Range:**
    - If the guess is too low, the AI knows that the target number must be higher. It updates the low boundary to guess + 1, effectively eliminating the lower half of the search range.
    - If the guess is too high, the AI knows that the target number must be lower. It updates the high boundary to guess - 1, eliminating the upper half of the search range.
4. **Repeating:** The AI repeats steps 2 and 3 until its guess matches the target number.

**Why Binary Search is Efficient:**

- **Reduces Search Space:** With each guess, the search space is roughly halved. This logarithmic reduction in search space makes the algorithm very efficient, especially for larger ranges.
- **Guaranteed Convergence:** Since the AI systematically narrows down the range, it's guaranteed to find the target number eventually.

**In this specific code:** The AI uses random number generation (random.randint) to make its guesses within the range. Ideally, for a pure binary search, the guess should be the middle element of the range. While this could further optimize the solution to have less attempts overall, the current approach still effectively utilizes the core principles of binary search by continuously narrowing the search space.

# Code

```
import random

def number_guessing_game():
    # Generate a random number between 1 and 100
    target_number = random.randint(1, 100)
    attempts = 0
    low = 1
    high = 100

    print("Welcome to the Number Guessing Game!")
    print("I will try to guess the number between 1 and 100.")

    while True:
        # The AI makes a guess in the middle of the range
        guess = random.randint(low, high)
        attempts += 1
        print(f"My guess is {guess}")

        # Check if the guess is too low
        if guess < target_number:
            print("Too low! I'll try again.")
            low = guess + 1

        # Check if the guess is too high
        elif guess > target_number:
            print("Too high! I'll try again.")
            high = guess - 1

        # Check if the guess is correct
```

```
        else:
            print(f"Congratulations to me! I've guessed the
number {target_number} in {attempts} attempts.")
            break

# Start the game
number_guessing_game()
```

## output

```
Welcome to the Number Guessing Game!
I will try to guess the number between 1 and 100.
My guess is 23
Too low! I'll try again.
My guess is 87
Congratulations to me! I've guessed the number 87 in 2 attempts.
```

```
Welcome to the Number Guessing Game!
I will try to guess the number between 1 and 100.
My guess is 64
Too high! I'll try again.
My guess is 14
Too low! I'll try again.
My guess is 62
Too high! I'll try again.
My guess is 20
Too low! I'll try again.
My guess is 28
Too high! I'll try again.
My guess is 21
Too low! I'll try again.
My guess is 22
Too low! I'll try again.
My guess is 24
Too low! I'll try again.
My guess is 26
Too low! I'll try again.
My guess is 27
Congratulations to me! I've guessed the number 27 in 10 attempts.
```

# References


1. **Binary Search:**
   - **Wikipedia:** You can find a comprehensive explanation of binary search on Wikipedia: [redacted link]
   - **GeeksforGeeks:** This website offers a detailed tutorial on binary search with examples and implementation in various programming languages: [redacted link]
2. **Random Number Generation in Python:**
   - **Python Documentation:** The official Python documentation provides information on the random module and its functions, including random.randint(): [redacted link]
   - **Real Python:** This tutorial on Real Python explains random number generation in Python with examples and practical applications: [redacted link]
3. **Number Guessing Game Implementations:**
   - **Various Online Resources:** You can find numerous examples and variations of number guessing game implementations in Python on websites like GitHub, Stack Overflow, and programming blogs. A simple search for "Python number guessing game" should yield many relevant results.