

# Project Report: Number Guessing Game

## 1. General Info

**Project Title:** Number Guessing Game with Performance Analysis

**Prepared By:** Alok Kumar Sharma

**Date:** November 23, 2025

**Institution:** VIT Bhopal

## 2. Introduction

This Python-based number guessing game challenges players to guess a secret number within limited attempts. It demonstrates concepts of input validation, control structures, and algorithmic strategies while providing educational feedback on player performance.

## 3. Problem Statement

Develop an interactive console game that generates a random secret number and limits user guesses. It must validate inputs, offer "higher/lower" hints, and analyze guessing efficiency by estimating winning probability and additional tries needed.

## 4. Functional Requirements

- Generate random number in range 1 100
- Accept and validate integer inputs
- Prevent duplicate guesses
- Provide hints after each guess
- Limit to 7 attempts
- Analyze probability of winning
- Calculate minimum additional guesses needed
- Display results and exit gracefully

## 5. Non-functional Requirements

- User-friendly console interface
- Handle invalid inputs gracefully
- Fast response time (1 second)
- Portable across Python 3+ environments
- Use only built-in modules
- Clear error messages

## 6. System Architecture

Modular design with main controller managing game logic, input validation, feedback, and analysis modules. Built-in modules `random`, `sys`, `time`, and `math` support functionality.

## 7. Design Diagrams

- **Use Case Diagram:** Player starts game, inputs guesses, receives feedback; system validates input, generates secret, analyzes guesses.
- **Workflow Diagram:** Game start → generate secret → loop guess input and validation → check guess → provide feedback → end game → analysis reported.
- **Sequence Diagram:** Player inputs guess → validated → compared → feedback → tracked → end → analysis displayed.
- **Class/Component Diagram:** Functions for input validation, game logic, analysis; use of built-in modules.
- **ER Diagram:** Not applicable (no persistent storage).

## 8. Design Decisions & Rationale

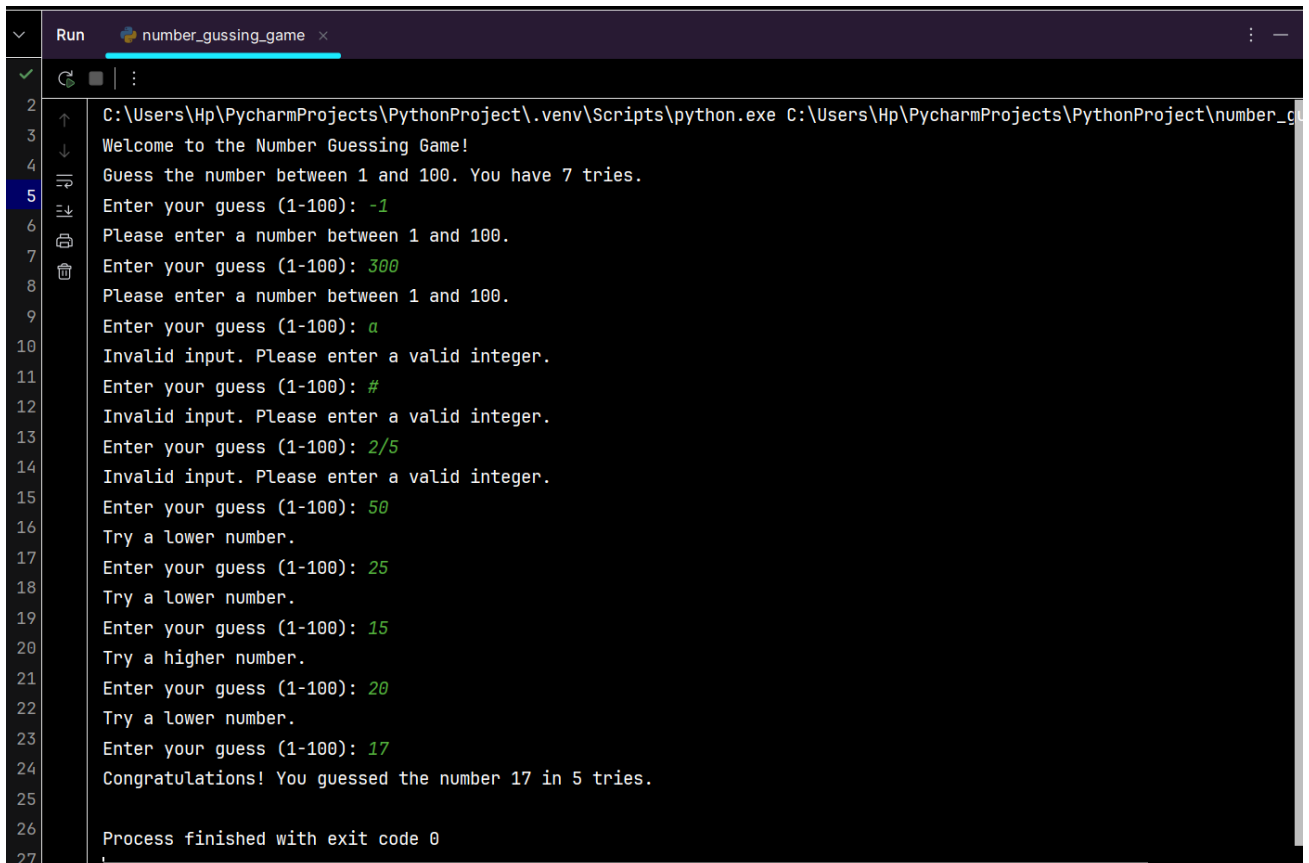
- Console interface chosen for simplicity
- 7 attempts chosen based on optimal binary search limit
- Range of 1-100 balances challenge and accessibility
- Input validation ensures robust, user-friendly gameplay
- Probability analysis educates on success likelihood
- Use of built-in modules avoids external dependencies

## 9. Implementation Details

- `get_valid_guess()` validates integer input, range, and duplicates
- `analyze_guesses()` calculates heuristic probability based on eliminated search space
- `min_additional_tries()` estimates further tries using binary search  $\log_2$  formula
- Main loop gathers guesses and provides feedback after each attempt

## 10. Screenshots / Results

Sample output demonstrates user guesses, error handling for invalid input, feedback hints, victory or defeat message, and final statistical analysis of performance.



```
Run number_guessing_game x
C:\Users\Hp\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Hp\PycharmProjects\PythonProject\number_gu
Welcome to the Number Guessing Game!
Guess the number between 1 and 100. You have 7 tries.
Enter your guess (1-100): -1
Please enter a number between 1 and 100.
Enter your guess (1-100): 300
Please enter a number between 1 and 100.
Enter your guess (1-100): a
Invalid input. Please enter a valid integer.
Enter your guess (1-100): #
Invalid input. Please enter a valid integer.
Enter your guess (1-100): 2/5
Invalid input. Please enter a valid integer.
Enter your guess (1-100): 50
Try a lower number.
Enter your guess (1-100): 25
Try a lower number.
Enter your guess (1-100): 15
Try a higher number.
Enter your guess (1-100): 20
Try a lower number.
Enter your guess (1-100): 17
Congratulations! You guessed the number 17 in 5 tries.
Process finished with exit code 0
```

```
Run number_gussing_game x
C:\Users\Hp\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Hp\PycharmProjects\PythonProject\number_gu
Welcome to the Number Guessing Game!
Guess the number between 1 and 100. You have 7 tries.
Enter your guess (1-100): 50
Try a lower number.
Enter your guess (1-100): 2
Try a higher number.
Enter your guess (1-100): 1
Try a higher number.
Enter your guess (1-100): 18
Try a lower number.
Enter your guess (1-100): 14
Try a higher number.
Enter your guess (1-100): 99
Try a lower number.
Enter your guess (1-100): 50
You already guessed 50. Try a different number.
Enter your guess (1-100): 22
Try a lower number.
Sorry, you've used all your tries. The number was 16. Try again!
Based on your answers, your probability of winning was approximately 97.00%.
Based on your guesses so far, you could have found the secret number in about 2 more optimally chosen guesses.

Process finished with exit code 0
```

## 11. Testing Approach

- Unit testing input validation and analysis functions
- Integration testing overall game flow
- Edge cases like repeated inputs, boundary values, and invalid types tested
- Manual playtesting verified correct feedback and results

## 12. Challenges Faced

- Handling various invalid inputs effectively
- Designing intuitive probability and optimal tries analysis
- Balancing game difficulty with user engagement
- Managing program flow with clean exit and delays

## 13. Learnings & Key Takeaways

- Mastering input validation and error handling
- Understanding binary search role in efficient guessing
- Applying modular design and clear function responsibilities
- Using built-in Python modules effectively for practical tasks

## 14. Future Enhancements

- Add GUI for improved user experience
- Introduce difficulty levels and multiplayer support
- Implement data persistence for statistics tracking
- Add hint system and scoring mechanism
- Use unit testing frameworks for automation

## 15. References

- Python Official Documentation (<https://docs.python.org/3/>)
- GeeksforGeeks, Number Guessing Game tutorial
- Stack Overflow discussions on guessing game algorithms
- Algorithm textbooks on binary search and probability
- Python built-in module guides

This concise report captures the core elements and structure of the Number Guessing Game project with clarity and precision.