# Guess the Number Game Design Document

## 1. Game Overview

* **Game type:** Single-player, text-based guessing game
* **Purpose:** Provide a simple yet engaging game where a player attempts to guess a number randomly chosen by the computer within a specific range.

## 2. Game Objective

The player’s goal is to correctly guess the secret number selected by the system. The game tracks the number of attempts taken and encourages players to minimize their guesses.

## 3. Gameplay Mechanics

### 3.1 Setup

1. The program greets the player and explains the rules.
2. The system randomly selects a secret integer within a defined range (e.g., 1‑100).
3. A counter is initialized to track the number of guesses.

### 3.2 Player Input & Feedback

1. The player enters a guess.
2. The game compares the guess with the secret number and provides feedback:
   * **Too high:** The guess is higher than the secret number.
   * **Too low:** The guess is lower than the secret number.
3. The guess counter increments, and the loop repeats until the player guesses correctly.

### 3.3 Winning and Replay

1. When the correct number is guessed, the game congratulates the player and displays the number of attempts.
2. The player is asked if they want to play again. If yes, the game resets; otherwise, it ends.

## 4. Game Architecture

The game can be organized into small, reusable functions:

1. **generate\_secret\_number(range\_start, range\_end):** Returns a random integer within the defined range using Python’s random module.
2. **prompt\_for\_guess():** Prompts the player for input and returns their guess as an integer. Includes validation to ensure the input is numeric and within range.
3. **evaluate\_guess(guess, secret):** Compares the guess to the secret number and returns feedback (“too high,” “too low,” or “correct”).
4. **main():** Controls the game loop:
   * Calls generate\_secret\_number() to set the target.
   * In a loop, calls prompt\_for\_guess() and evaluate\_guess().
   * Tracks the number of attempts.
   * When the guess is correct, displays results and prompts to play again.

This modular approach allows easy extension—for example, adding difficulty levels or a scoring system.

## 5. Future Enhancements

* **Difficulty levels:** Allow players to choose the range (e.g., 1‑10, 1‑1000).
* **High‑score leaderboard:** Save and display the fewest attempts across sessions.
* **Graphical interface:** Create a simple GUI using tkinter or convert to a web-based game.
* **Multiplayer variant:** Let two players compete by guessing each other’s numbers.

This document provides the structure needed to implement the game in Python. The accompanying code sample demonstrates a basic version of this game.