Python Project 04: Number Guessing Game

1. Project Overview

In this project, we will build a **Number Guessing Game** using Python and Tkinter.

The computer generates a random number, and the player tries to guess it. After every guess, the game tells the player whether the guess is **too high** or **too low**.

Tkinter provides the GUI widgets and button callbacks, while Python's `random` module can generate the secret number. (Python documentation)

---

2. Features

- Random secret number
- Number input
- Guess button
- Too high / too low hints
- Attempt counter
- Score system
- New game button
- Input validation
- Difficulty levels
- Modern dark UI

---

3. Technologies

- Python 3

- Tkinter
- random
- Object Oriented Programming

No external packages are required.

---

4. Folder Structure

```
number_guessing_game/
□
□□□ main.py
□□□ ui.py
□□□ game.py
□□□ README.md
```

---

5. How the Game Works

The game generates a random number between the selected minimum and maximum values.

For example:

```
Secret Number = 67

Player guesses 40
      □
Too Low

Player guesses 80
      □
Too High
```

```
Player guesses 67
      
Correct!
```

The backend handles the secret number, guesses, attempts, hints and score.

The frontend handles the input field, buttons and messages.

---

6. Frontend Code

ui.py

```
import tkinter as tk
from game import NumberGuessingGame


class NumberGuessingUI:
    def __init__(self, root):
        self.root = root

        self.root.title("Number Guessing Game")
        self.root.resizable(False, False)
        self.root.configure(bg="#0f172a")

        self.game = NumberGuessingGame()

        self.create_ui()
        self.new_game()

        self.root.bind("<Return>", self.handle_enter)
```

```
    def create_ui(self):
        title = tk.Label(
            self.root,
            text="NUMBER GUESSING",
            font=("Arial", 24, "bold"),
            fg="#38bdf8",
            bg="#0f172a"
        )
        title.pack(pady=(25, 5))

        subtitle = tk.Label(
            self.root,
            text="Guess the secret number",
            font=("Arial", 11),
            fg="#94a3b8",
            bg="#0f172a"
        )
        subtitle.pack(pady=(0, 20))

        difficulty_frame = tk.Frame(
            self.root,
            bg="#0f172a"
        )
        difficulty_frame.pack()

        tk.Label(
            difficulty_frame,
            text="Difficulty:",
            font=("Arial", 11, "bold"),
            fg="white",
            bg="#0f172a"
        ).pack(side="left", padx=5)

        self.difficulty = tk.StringVar(
```

```
            value="Easy"
        )

        difficulty_menu = tk.OptionMenu(
            difficulty_frame,
            self.difficulty,
            "Easy",
            "Medium",
            "Hard",
            command=self.change_difficulty
        )

        difficulty_menu.config(
            font=("Arial", 10, "bold"),
            bg="#1e293b",
            fg="white",
            activebackground="#334155",
            activeforeground="white",
            relief="flat"
        )

        difficulty_menu["menu"].config(
            bg="#1e293b",
            fg="white"
        )

        difficulty_menu.pack(side="left", padx=5)

        self.range_label = tk.Label(
            self.root,
            text="Guess a number between 1 and 50",
            font=("Arial", 11),
            fg="#cbd5e1",
            bg="#0f172a"
```

```
        )
        self.range_label.pack(pady=15)

        self.entry = tk.Entry(
            self.root,
            font=("Arial", 20, "bold"),
            justify="center",
            width=12,
            bg="#1e293b",
            fg="white",
            insertbackground="white",
            relief="flat"
        )
        self.entry.pack(pady=5)

        self.guess_button = tk.Button(
            self.root,
            text="GUESS",
            command=self.make_guess,
            font=("Arial", 11, "bold"),
            bg="#2563eb",
            fg="white",
            activebackground="#1d4ed8",
            activeforeground="white",
            relief="flat",
            padx=30,
            pady=10,
            cursor="hand2"
        )
        self.guess_button.pack(pady=12)

        self.message_label = tk.Label(
            self.root,
            text="",
```

```
            font=("Arial", 14, "bold"),
            fg="#f8fafc",
            bg="#0f172a",
            wraplength=350
        )
        self.message_label.pack(
            padx=30,
            pady=10
        )

        self.attempt_label = tk.Label(
            self.root,
            text="Attempts: 0",
            font=("Arial", 11),
            fg="#94a3b8",
            bg="#0f172a"
        )
        self.attempt_label.pack()

        self.score_label = tk.Label(
            self.root,
            text="Score: 0",
            font=("Arial", 12, "bold"),
            fg="#38bdf8",
            bg="#0f172a"
        )
        self.score_label.pack(pady=8)

        self.new_game_button = tk.Button(
            self.root,
            text="New Game",
            command=self.new_game,
            font=("Arial", 10, "bold"),
            bg="#475569",
```

```
            fg="white",
            activebackground="#64748b",
            activeforeground="white",
            relief="flat",
            padx=20,
            pady=8,
            cursor="hand2"
        )
        self.new_game_button.pack(pady=(5, 25))

    def change_difficulty(self, level):
        self.new_game()

    def make_guess(self):
        value = self.entry.get().strip()

        if not value.isdigit():
            self.show_message(
                "Please enter a valid number.",
                "#f87171"
            )
            return

        result = self.game.guess(int(value))

        if result["status"] == "low":
            self.show_message(
                "Too Low! Try a higher number.",
                "#fbbf24"
            )

        elif result["status"] == "high":
            self.show_message(
                "Too High! Try a lower number.",
```

```
                "#fbbf24"
            )

        elif result["status"] == "correct":
            self.show_message(
                f"Correct! The number was {result['number']}.",
                "#4ade80"
            )

            self.guess_button.config(
                state="disabled"
            )

        elif result["status"] == "invalid":
            self.show_message(
                f"Enter a number between "
                f"{self.game.minimum} and {self.game.maximum}.",
                "#f87171"
            )

        self.attempt_label.config(
            text=f"Attempts: {self.game.attempts}"
        )

        self.score_label.config(
            text=f"Score: {self.game.score}"
        )

        self.entry.delete(0, tk.END)
        self.entry.focus()

    def show_message(self, message, color):
        self.message_label.config(
            text=message,
```

```
            fg=color
        )

    def new_game(self):
        level = self.difficulty.get()

        self.game.start_new_game(level)

        self.range_label.config(
            text=(
                f"Guess a number between "
                f"{self.game.minimum} and "
                f"{self.game.maximum}"
            )
        )

        self.message_label.config(
            text="Make your first guess!",
            fg="#f8fafc"
        )

        self.attempt_label.config(
            text="Attempts: 0"
        )

        self.score_label.config(
            text=f"Score: {self.game.score}"
        )

        self.guess_button.config(
            state="normal"
        )

        self.entry.delete(0, tk.END)
```

```
        self.entry.focus()

    def handle_enter(self, event):
        if self.guess_button["state"] == "normal":
            self.make_guess()
```

---

7. Backend / Game Logic Code

game.py

```
import random


class NumberGuessingGame:

    DIFFICULTIES = {
        "Easy": (1, 50),
        "Medium": (1, 100),
        "Hard": (1, 500)
    }

    def __init__(self):
        self.score = 0
        self.start_new_game("Easy")

    def start_new_game(self, level):
        self.minimum, self.maximum = self.DIFFICULTIES[level]

        self.secret_number = random.randint(
            self.minimum,
            self.maximum
        )
```

```
    self.attempts = 0
    self.game_over = False

  def guess(self, number):

    if self.game_over:
      return {
        "status": "finished"
      }

    if number < self.minimum or number > self.maximum:
      return {
        "status": "invalid"
      }

    self.attempts += 1

    if number < self.secret_number:
      return {
        "status": "low"
      }

    if number > self.secret_number:
      return {
        "status": "high"
      }

    self.game_over = True

    self.score += max(
      10 - self.attempts,
      1
    )
```

```
        return {
            "status": "correct",
            "number": self.secret_number
        }
```

---

8. Main File

main.py

```
import tkinter as tk
from ui import NumberGuessingUI


def main():
    root = tk.Tk()

    NumberGuessingUI(root)

    root.mainloop()


if __name__ == "__main__":
    main()
```

---

9. How Frontend and Backend Connect

main.py

NumberGuessingUI

NumberGuessingGame

Random Number
Attempts
Hints
Score
NumberGuessingUI
Updated Screen

The **frontend** receives the player's input and displays the result.

The **backend** decides whether the number is:

Too Low
Too High
Correct
Invalid

The backend also keeps the secret number hidden from the UI.

---

10. How to Run

Open the project folder:

```
cd number_guessing_game
```

Run:

```
python main.py
```

No external packages are required.

---

11. Basic Testing

Test these cases:

- Enter a number below the secret number.
- Enter a number above the secret number.
- Guess the correct number.
- Enter a number outside the selected range.
- Enter text instead of a number.
- Press Enter to submit a guess.
- Change difficulty.
- Start a new game.
- Check that the score increases after winning.
- Check that attempts increase after every valid guess.

---

12. Small Improvements

Students can extend the project by adding:

- Maximum number of attempts
- Hint button
- Timer
- High score
- Difficulty based on number of attempts
- Sound effects

- Guess history
- Progress indicator
- Computer difficulty modes

Project Goal

This project practices:

Random Data User Input Validation Conditions Game State Result

It also demonstrates how game logic can remain separate from the graphical interface.

haas.dev

https://dev-roast-app.vercel.app

Visit haas.dev for more resources and guides.