UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

Number Guessing Game in Linux

A Project Report

Submitted to

Mr. Navdeep Singh

Submitted by

Nancy Sharma (24MCA20168)

in partial fulfilment for the award of the degree of

Master of Computer Application

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Chandigarh University

Aug 2024 – Nov 2026

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

DECLARATION

I, Nancy Sharma , hereby declare that this project report titled "Number guessing game " is original work carried out by me under the supervision of Er. Navdeep Singh . I further declare that this work has not been submitted to any other institute/university for the award of the degree of Master of Computer Applications.

Student Name: Nancy Sharma

Roll No: 24MCA20168

ACKNOWLEDGEMENT

I express my sincere gratitude to my project guide, Er. Navdeep Singh, for invaluable guidance and support throughout this project. I also extend my thanks to Chandigarh University for the opportunity to undertake this project and to my classmates and family for their continuous encouragement.

Nancy Sharma

24MCA20168

Introduction

The Number Guessing Game is a simple, interactive Python application developed using the Tkinter library to provide a graphical interface for Linux. The game generates a random number between 1 and 100, which the player must guess correctly within an unlimited number of attempts. With each guess, feedback is provided, guiding the player towards the correct answer.

This project is an exploration of GUI development fundamentals in Python using Tkinter and provides a hands-on exercise in user experience (UX) design and basic game development principles.

Objectives

The primary objectives of this project are:

1. Develop a simple GUI application: Create a user-friendly interface using Tkinter that allows users to interact with the game intuitively.

2. Implement Guessing Logic: Design logic to check guesses and provide feedback on whether each guess is too high, too low, or correct.

3. Practice Python and Tkinter skills: Improve Python programming skills, focusing on GUI creation and basic algorithmic problem-solving.

4. Ensure Cross-Platform Compatibility: Test the game on different Linux distributions to ensure compatibility.

Methodology

The project was approached through the following steps:

1. Planning and Design: Outline the game flow and design the user interface layout. A simple and intuitive layout was chosen for easy user interaction.

2. Random Number Generation: Use the `random` library to generate a random number each time the game starts.

3. Input Validation and Feedback: Implement logic to validate user input and provide helpful feedback based on the guess.

4. Game Flow Management: Use conditional checks to disable input and display a congratulatory message when the correct number is guessed.

Code Explanation and Design

1. Code Structure

This project consists of a single Python file with the following components:

- Import Libraries: Import `tkinter` for the GUI and `random` to generate a random number.
- Global Variables: The random number (`secretno`) and `attempts` are defined globally for game-wide access.
- Tkinter Widgets: Widgets such as `Label`, `Entry`, and `Button` were used to interact with the user and display messages.

2. Code Snippet

```
import tkinter as tk
import random
# Generate a secret number between 1 and 100
secretno = random.randint(1, 100)
attempts = 0
# Initialize the Tkinter root window
root = tk.Tk()
root.title("Number Guessing Game")
# Instruction Label
inst = tk.Label(root, text="Guess the number between 1 to 100:")
inst.pack()
# Entry field for user's guess
entry = tk.Entry(root)
entry.pack()
# Label to display results and feedback
resultlabel = tk.Label(root, text="")
resultlabel.pack()
```

3. Explanation of Functions:

1. checkguess(): This function retrieves the user's guess, validates it, and provides feedback.

- Input Validation: Checks if the input is numeric. If not, it displays an error message.
- Comparison Logic: Compares the guess to the target number and adjusts feedback accordingly:
- If the guess is correct, it displays a congratulatory message and disables the entry.
- If the guess is too low or too high, it provides a hint.

```
def checkguess():
    global attempts, secretno
    guess = entry.get()
    if not guess.isdigit():
        resultlabel.config(text="Enter a valid number")
        return
    guess = int(guess)
    attempts += 1
    if guess == secretno:
        resultlabel.config(text="Congratulations! You've guessed the correct number!")
        entry.config(state="disabled")
    elif guess < secretno:
        resultlabel.config(text="Too low! Try again.")
    else:
        resultlabel.config(text="Too high! Try again.")
```

4. Main GUI Window Setup: The main window is created using Tkinter's `Tk()` method, and each widget is packed onto it.

```
guessbutton = tk.Button(root, text="Check Guess", command=checkguess)
guessbutton.pack()
root.mainloop()
```

User Interface and Experience

1. Interface Overview: The application interface is minimalist, focusing on usability and clarity. The user is prompted to guess a number, and feedback is displayed below the input field, updating with each guess. The use of real-time feedback and clear messages helps keep users engaged and informed throughout the game.
2. Accessibility: Efforts were made to ensure the game is accessible by using large text sizes, intuitive button labels, and immediate error handling for invalid inputs.

Challenges Faced and Solutions

1. Input Validation: Ensuring that only numeric input was accepted required implementing a validation check.

- Solution: The function `checkguess()` includes logic to check if the input is a digit and provide appropriate feedback if not.

2. Game Ending: Handling the end of the game gracefully was challenging.

- Solution: The input entry is disabled, and a congratulatory message is displayed upon guessing the correct number.

Future Improvements

1. Limited Number of Attempts: Add a feature to limit the number of guesses allowed (e.g., 10 attempts).

2. Restart Option: Include a “Play Again” button to reset the game after a win or loss.

3. Difficulty Levels: Implement different difficulty levels by adjusting the number range or the maximum number of attempts.

Conclusion

This project provided an excellent opportunity to develop a graphical application using Python's Tkinter library on Linux. The resulting game demonstrates effective input handling, feedback display, and user engagement, successfully achieving the objectives set out in the planning phase. Through this project, Python skills were strengthened, particularly in GUI development and handling basic game logic.

Screenshot: