**Number Guessing Game Documentation**

**1. Introduction**

This documentation provides a detailed overview of a Number Guessing Game implemented in Python using the Tkinter library. The game prompts the player to guess a randomly generated number within a specified range and provides feedback on each guess. The player is encouraged to play again upon successful guessing.

**2. Project Structure**

**2.1. Import Statements**

Python code :

import tkinter as tk

from tkinter import messagebox

import random

* The code imports **tkinter** for GUI development.
* **messagebox** from **tkinter** is imported for displaying messages.
* **random** is imported to generate a random number for the game.

**2.2. Class: NumberGuessingGame**

**Python code :**

class NumberGuessingGame:

def \_\_init\_\_(self, master):

# ... (Constructor and initialization)

def start\_game(self):

# ... (Initialize the game and prompt for the player's name)

def make\_guess(self):

# ... (Handle the player's guess and provide feedback)

def show\_congratulations\_message(self):

# ... (Display a congratulatory message and ask if the player wants to play again)

def reset\_game(self):

# ... (Reset the game for a new round)

* The main class represents the Number Guessing Game.
* Methods include initialization, starting the game, handling guesses, showing a congratulatory message, and resetting the game.

**2.3. Function: main()**

Python code :

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

game = NumberGuessingGame(root)

root.mainloop()

* The script initializes the Tkinter root window and creates an instance of the **NumberGuessingGame** class.
* The main event loop starts with **root.mainloop()**.

**3. Class Methods**

**3.1. \_\_init\_\_(self, master)**

* **Description:** Initializes the Number Guessing Game with the specified master (root) window.
* **Parameters:**
  + **master**: The master (root) window for the application.
* **Components:**
  + Labels, entry widgets, and buttons for user interaction.
  + Attributes for storing the generated number and tracking guesses.

**3.2. start\_game(self)**

* **Description:** Initiates the game by obtaining the player's name.
* **Functionality:**
  + Retrieves the player's name from an entry widget.
  + Changes the display to show the game instructions and an entry widget for guesses.

**3.3. make\_guess(self)**

* **Description:** Handles the player's guess, provides feedback, and checks for a win.
* **Functionality:**
  + Retrieves the player's guess from an entry widget.
  + Validates the guess, provides feedback, and shows appropriate messages.
  + Calls **show\_congratulations\_message** if the guess is correct.

**3.4. show\_congratulations\_message(self)**

* **Description:** Displays a congratulatory message and asks if the player wants to play again.
* **Functionality:**
  + Creates a custom dialog to display a message.
  + Prompts the player to play again using a messagebox.
  + Calls **reset\_game** if the player chooses to play again.

**3.5. reset\_game(self)**

* **Description:** Resets the game for a new round.
* **Functionality:**
  + Generates a new random number and resets the guesses counter.
  + Adjusts the display to show the initial game entry screen.

**4. Function: main()**

**4.1. Description**

* Initializes the Tkinter root window and creates an instance of the **NumberGuessingGame** class.
* Starts the Tkinter main event loop.

**4.2. Execution**

* The script checks if it is the main module and, if so, calls the **main()** function to start the Number Guessing Game.

**5. Execution and Usage**

**5.1. Preferred Development Environment**

* You can use any Python-supported development environment such as VS Code, PyCharm, Jupyter, or a simple text editor.

**5.2. Necessary Packages and Libraries**

* Ensure that you have a Python environment installed.
* The code uses the built-in **tkinter** library, which is usually included in standard Python installations.

**5.3. Execute the Code**

* Copy the provided code into a Python file (e.g., **number\_guessing\_game.py**).
* Open a terminal or command prompt.
* Navigate to the directory containing the Python file.
* Run the script using **python number\_guessing\_game.py**.
* The Number Guessing Game GUI should appear, allowing you to interact with it.

**6. Implementation**

**6.1. Potential Enhancements**

* Customize the game further, such as allowing the user to set the range of numbers.
* Implement a scoring system based on the number of attempts.
* Add sound effects or visual elements to enhance the user experience.

**6.2. Test Scenarios**

* Guess the correct number within the specified range.
* Enter invalid input and observe the error handling.
* Reset the game and start a new round.

**6.3. Exploration**

* Explore customization options provided by Tkinter for GUI elements.
* Research additional Tkinter features to enhance the application.

**6.4. Learning Opportunities**

* Learn more about Tkinter's event-driven programming model.
* Experiment with adding additional features or functionalities to the game.

Feel free to experiment, modify, and expand the code to further your understanding of GUI programming with Tkinter!