***Number Generating Game Using Python***

Guessing a number is a fun game that involves us to relate the clues the system gives us. This fun, yet provoking game is a simple way to keep our mind stimulated.

Python Number Guessing Game Project:

We will create a simple number guessing game using python and display the scores the player gets if he/she wins. This project will serve as an introduction to functions and creating GUI using Tkinter.

Project Prerequisites:

The project makes use of Tkinter for GUI, random module for generating random targets and clues. Since we use Tkinter which is a built-in GUI library in python, we check it by importing it

### Project File Structure:

There are many GUI libraries supported by python such as PyQT5, Kivy, Pyside2 etc. Tkinter is widely used by many developers and is easy for beginners to practise.

Let’s have a look at the steps to build the project:

1. Importing modules: tkinter, random and declaring variables
2. Declaring clue functions
3. Declaring and defining guess and target number functions
4. Creating the user input interface

Feel free to play with the values and change your input methods.

#### **1. Importing necessary modules for Python Number Guessing Game**:

**Code Explanation:**

* **import random:** Random library generates a random number or a random subset in a string. We use randint function of random
* **from tkinter import \*:**To create the user interface, we import Tkinter. Tkinter contains provisions to build the interface with widgets such as buttons, input entry fields etc
* **from tkinter import messagebox:**To display the user with prompts, we use messagebox. Using this module, we can show warnings, errors, information etc to the user
* **Target, score, guess:**Declare the variables to set the target value and guess from the user in the forthcoming functions. Score is set to 10 which also implies the user gets 10 chances before he loses

#### **2. Declaring clue functions:**

* **clues():**Declare the function clues
* **switcher:**Define a dictionary switcher to simulate switch case where the values of the keys represent the clue functions
* **return switcher.get(random.randint(0,4)):**Take a random key from the switcher to generate a random clue using get() function
* **generate\_target\_number():**Declare the function generate\_target\_number
* **global target:**Make the variable global to preserve its value across other functions, ie not limiting to local scope (within the function)
* **target = random.randint(1,10):**Using randint, assign a random integer to target. The parameters are the lower and upper range, both inclusive
* **messagebox.showinfo():**To display the user with a prompt after generating the target value, we display a prompt using messagebox. Since this is an information, we use showinfo of messagebox with the message parameter set to a message.
* **random\_number\_button[‘state’], guess\_button[‘state’]:**After getting a target, we disable the random number button so the user plays a fair game without changing the target. Enable guess button to allow the user to guess the number after getting a target. Disable turns off the button and Normal allows to click the button. The state of the button attribute is altered to achieve this.
* **guess\_and\_score:**Declare the function guess\_and\_score
* **global score, global guess:**Make the variable global to preserve its value across other functions, ie not limiting to local scope (within the function)
* **guess =0,guess = int(guess\_entry.get()):**Initialise guess to 0 and read the user given input. We put this in a try…except block to raise an error if the user did not specify with a guess. Raise the error prompt using messagebox.showerror()
* **if guess == target:**Display the score to the player if he guesses it correct and disable the guess button. Enable the random number button to play another game
* **elif score == 0:**If the score is 0, it implies the user has run out of tries. Prompt the user with 0 score and show 0 moves to go
* **Else block:**The else block calls for clues. It calls clues functions which will call the functions created earlier. Display the clue using messagebox

#### **4. Creating  the user input interface:**

* **window  = Tk():**Create a window by assigning the tkinter class constructor to it. We can now add widgets and entry fields to our app
* **window.geometry(“350×200”):**Set the dimensions of the application window using geometry. The parameters are width and height
* **window.title(“TechVidvan’s Number Guessing Game”):**Set a title for the titlebar of the application using title parameter
* **title\_label, guess\_label:**To add any non editable text to the window, we use labels. The parameters are window and the text to display.
* **Random\_number\_button, guess\_button:**Buttons invoke a function when the user clicks on them. The command parameter contains the function to call. The other parameters are the name of the button and the window it will sit on. Use state parameter to set on or off a button
* **pack():**Placing a widget on a window makes it visible. So to position it, we use pack which puts it in the center of each row. First element is in the first row, second in the second row and so on
* **window.mainloop()**: Any line of command pertaining to the GUI will not be visible after mainloop. This line indicates the end of the application interface and allows to terminate the application when user closes it

Python Number Guessing Output

Play the number guessing game by starting the random number generator and get the output.

Thus, we created a simple project to understand and create our functions to provide clues and calculate the scores of the games. We also created a GUI using tkinter to provide an interface