```
"""
A GUI game where the user must guess
the program's secret number.
"""
import random
import tkinter as tk
from tkinter import ttk

DEBUG = True


class GuessingGame():
    """ One object of this class represents a running guessing game."""

    def __init__(self, input, output):
        """The constuctor method creates an instance of the tkVar StringVar
        subclass and is assigned to the variable, self.outputVar.
        This method also receives into its parameters (input and output)
        an entry widget and a label widget respectively.
        This label widget is assigned to self.output and its text is
        assigned the self.outputVar variable. The constructor method
        also holds variables to be mutated and or utilized in
        further methods: secretNumber, NUM_ALLOWED_TRIES, etc.
        It also initializes gameWon as FALSE so that the game may commence.
        """

        # List any instance variables,including constants, here
        self.input = input
        self._output = output
        self.NUM_ALLOWED_TRIES = 8
        self.MIN_POSSIBLE_GUESS = -64
        self.MAX_POSSIBLE_GUESS = 64
        self.timesGuessed = 0
        self.guessedNumbersList = []
        self.gameWon = False
        self.outputVar = tk.StringVar()
        self._output["textvariable"] = self.outputVar
        self.secretNumber = ""

        if(DEBUG):
            print("Created a GuessingGame object")

    def getRandomNumber(self):
        """If you need to get a number for the user to guess(randomly),
        call the method below and it will create one for you.
        The number may be as low as MIN_POSSIBLE_GUESS, and as high as
        MAX_POSSIBLE_GUESS
        (This will be 16 separate numbers that the user might try to guess)
        Of course, you will have to create a variable space to hold this number
        when it is returned(e.g., secretNumber)"""
        max = self.MAX_POSSIBLE_GUESS - self.MIN_POSSIBLE_GUESS
        zeroToMax = random.randrange(max + 1)
        return zeroToMax + self.MIN_POSSIBLE_GUESS

    def output(self, outputStr, append=False):
        ''' The output method receives strings to be displayed on
        the output label and mutates or appends the strings
        accordingly. It then sets the outputVar, which is
        displayed on the output label, to the appropriate string. '''
        self.outputStr = outputStr
        self.append = append

        if append:
            mo = self.outputVar.get()
            self.outputVar.set(mo + self.append)
        else:
            self.outputVar.set(self.outputStr)

    def getInput(self):
        """Checks so the input is valid (an int between min & max).
        Calls processInput only with valid input."""
        min = self.MIN_POSSIBLE_GUESS
        max = self.MAX_POSSIBLE_GUESS
        guess = self.input.get()
        errorMsg = (f"\nPlease enter a whole number between " +
                    f"{self.MIN_POSSIBLE_GUESS} and {self.MAX_POSSIBLE_GUESS}.")

        try:
            mo2 = int(guess)
        except ValueError:
            self.output(False, errorMsg)
            return
        if mo2 <= max and mo2 >= min:
            self.processInput(guess)
        else:
            self.output(False, errorMsg)

    def playGuessingGame(self):
        """This method calls the output method to reset the output
        label's text. It also reinitializes the secretNumber variable
        by calling the getRandomNumber function and assigning
        that number to secretNumber, it reinitializes the
        guessedNumbersList, and reinitializes gameWon to FALSE."""

        greeting = (
            f'''Playing the game.\nA new number has been picked.
        Welcome!
        Rules:
        1) You have {self.NUM_ALLOWED_TRIES} guesses.
        2) If you guess the wrong number, a hint will be displayed.
        3) Your guess has to be between ''' +
            f'''{self.MIN_POSSIBLE_GUESS} and {self.MAX_POSSIBLE_GUESS}.''')

        self.output(greeting)
        self.timesGuessed = 0
        self.secretNumber = self.getRandomNumber()
        if(DEBUG):
            print("secretNumber:", self.secretNumber)
        self.guessedNumbersList = []
        self.gameWon = False

        # TODO: complete this method

    def processInput(self, pinput):
        """The processInput method first checks to see if the game is over
        or is to be continued.The method also does a number of
        calculatory checks: It checks to see if a guess already exists in
        the guessedNumbersList. The method will either append the guess
        to the list or send the appropriate message if the guess
        is already listed. The method also checks to see if a guess is
        higher or lower than secretNumber and provides the necessary
        clue. If the guess is correct, the method will send the
        appropriate message and give the option to end or play again.
        And if the NUM_ALLOWED_TRIES are spent, it will display the
        message informing the player that the game is over and what
        the correct number was."""

        self.pinput = pinput
        mo3 = int(self.pinput)
        rem = 7 - self.timesGuessed
        str1 = f"You guessed: {self.pinput}"
        str3 = f"\nPrevious guess(es) {str(self.guessedNumbersList)}"
        str4 = f"\nYou have {str(rem)} guess(es) remaining"
        errorMsg = "\nYou already guessed that number. \nEnter a new value."

        if (self.gameWon or (self.timesGuessed >= self.NUM_ALLOWED_TRIES)):
            # play again
            self.playGuessingGame()
            return

        if mo3 in self.guessedNumbersList:
            self.output(False, errorMsg)
            return

        if mo3 > self.secretNumber:
            str2 = "\n Hint: The secret number is smaller than you guessed."
            self.output(str1 + str2 + str3 + str4)
            self.timesGuessed += 1
            self.guessedNumbersList.append(mo3)

        elif mo3 < self.secretNumber:
            str2 = "\n Hint: The secret number is larger than you guessed."
            self.output(str1 + str2 + str3 + str4)
            self.timesGuessed += 1
            self.guessedNumbersList.append(mo3)

        elif mo3 == self.secretNumber:
            str2 = "\nYou won! \nPress the button to start again."
            self.outputVar.set(str1 + str2)
            self.gameWon = True

        if rem == 0:
            self.outputVar.set(
                str1 + f'''
                You've run out of guesses.
                The correct number was {self.secretNumber}.
                Press the button to start again''')


class GuessingGameApp(tk.Tk):
    """One object of this class represents a GUI application root window
    for the Guessing Game.
    To win: Start with 0. Then, given the clue, guess halfway towards the
    maximum or minimum value.
    Given the following clues, continue guessing halfway to the most
    prudent maximum or minimum value.
    """
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        ''' The constructor method creates four widgets
        to be displayed on the GUI '''

        self.title("Guessing Game")
        self.resizable(width=True, height=True)

        self.label = ttk.Label(self, text="Output: ")
        self.label.grid(row=0, column=0, rowspan=2)

        output = ttk.Label(self, font=("TkDefaultFont", 18), width=80,)
        output.grid(row=0, column=1, rowspan=2)

        input = ttk.Entry(self, width=3)
        input.grid(row=0, column=3, rowspan=2)

        input_button = ttk.Button(self, text="OK")
        input_button.grid(row=0, column=4, rowspan=2)

        gg = GuessingGame(input, output)
        input_button.config(command=gg.getInput)
        gg.playGuessingGame()


if __name__ == "__main__":

    app = GuessingGameApp()
    app.mainloop()
```