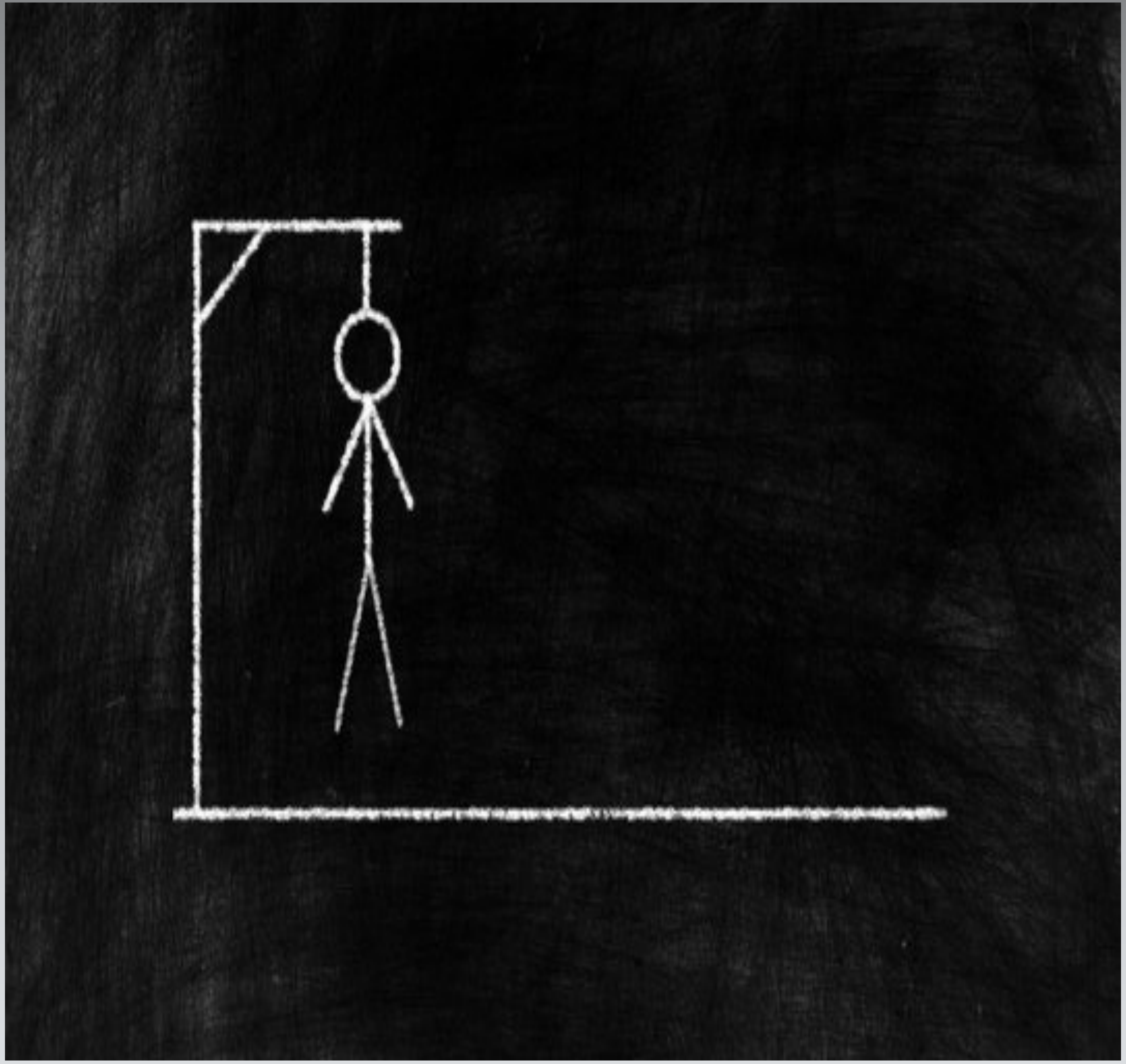


HANGMAN GAME

AGENDA

The main objective is to know application of python in game fields

The hangman game is a very essential game for children’s so that by playing the game, at the   same time they can learn the new English words

By using the python , we can make this game more interesting and efficent

INTRODUCTION

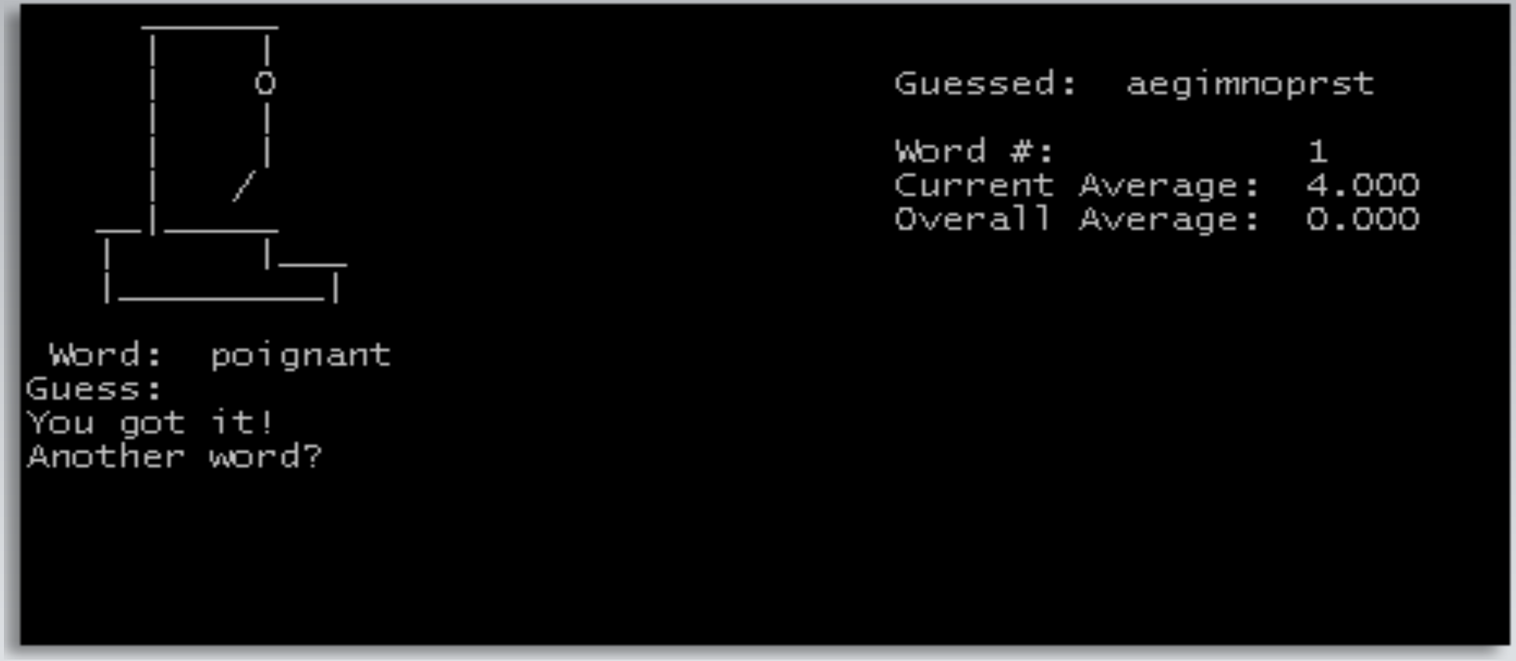
A Hangman Game On Python is about guessing letters (A-Z) to form the words. If the player guesses the right letter that is within the word, the letter appears at its correct position. The user has to guess the correct word until a man is hung, then the game is over.

ABSTRACT

- The Hangman program randomly selects a secret word from a list of secret words. The random module will provide this ability, so line 1 in program imports it.
- The Game: Here, a random word (a fruit name) is picked up from our collection and the player gets limited chances to win the game.
- When a letter in that word is guessed correctly, that letter position in the word is made visible. In this way, all letters of the word are to be guessed before all the chances are over.
- In this Python project, we will build a GUI-based Hangman game using the Tkinter, random, screeninfo and PIL modules..

EXISTING SOLUTIONS

- A hangman game is the first game that most programmers make, but they make it command-line based, and we are going to make it GUI based.



```
                                        Guessed:  aegimnoprst

                                        Word #:             1
                                        Current Average:  4.000
                                        Overall Average:  0.000

 Word:  poignant
Guess:
You got it!
Another word?
```

PROPOSED SYSTEM

- The code for this game is already existing python and the new implementations done by team are the following.
- Further enhancement of program by adding timer after every Guess
- Giving hints from the beginning to make the task a bit easier for user
- Decreasing the chance by one only if player’s guess is WRONG. If the guess is right,player’s chance is not reduced

SOFTWARE AND HARDWARE REQUIREMENTS

SOFTWARE REQUIREMENTS

PYCHARM IDE is used

HARDWARE REQUIREMENTS

- Graphics Card: ATI FireGL T2-128
- CPU: Intel Pentium 4 2.00GHz
- File Size: 135 MB
- OS: Windows 8 / Windows 10

The Python libraries:

To build this project, we will need the following libraries:

1. Tkinter – To create the GUI.
2. PIL (Pillow) – This is Python’s Image Library used for image manipulation. Here, we will use it to convert the Hangman progress images to PhotoImage format, which allows them to be displayed on Tkinter.
3. screeninfo.get_monitors() – To get the info of the screen’s dimension to set the height and width of the GUI window and its elements.
4. random.choice() – To get a random word from our list that the user of the game will guess.
5. os – To exit the script.

Tkinter Elements used:

- 1. The Tk() class is used to initialize a GUI window. The attributes and methods that need to be set during initialization are:
  a.title() method is used to set a title to the window.
  b.geometry() method is used to set the geometry of the window.
  c.resizable() method is used to allow/forbid the user to resize the window.
  d.update() and .mainloop() methods are used to prevent the window from closing milliseconds after it opens.
- 2. The Label widget is used to display static text to the window.
  a. The image parameter is used to specify the PhotoImage object that will be used in place of/along with text in the Label.

- The Button class is used to add a button to the window that executes a function as a command when pressed.
  a. The command parameter is the function to be executed when the button is pressed.
- The .place() method is used to place the widget it is associated with as though it is on a Cartesian plane.
  a. The anchor parameter specifies the origin of the Cartesian for that widget. It can be a corner or a side of the root window.
  b. The relx, rely parameters are used to set the horizontal and vertical offsets of the widget as a float number between 0.0 and 1.0
- The x, y parameters are used to set the horizontal and vertical offsets of the widge
  a. The master parameter is the parent widget of the class it is associated with.
  b. The text parameter is the text to be displayed.
  c. The font parameter is used to specify the font family and effects to the text.
  d. The bg parameter is used to give a background color.

Project File Structure:

These are the files used in this project:

1. main.py – This will be our main python file.

2. words.txt – This is the word file where all the words to guess will be there.

3. Steps Images (folder): In this folder, we have kept all the progress images that will be displayed as the user guesses an incorrect letter

a. Step 0.png – Image to be displayed in the beginning where only the basic setup is visible.

b. Step 1.png – Image to be displayed after 1 incorrect guess. No body part is visible.

c. Step 2.png – Image to be displayed after 2 incorrect guesses. Only the head is visible.

Project File Structure:

c. Step 2.png – Image to be displayed after 2 incorrect guesses. Only the head is visible.
d. Step 3.png – Image to be displayed after 3 incorrect guesses. The torso appears.
e. Step 4.png – Image to be displayed after 4 incorrect guesses. The arms appear.
f. Step 5.png – Image to be displayed after 5 incorrect guesses. The legs appear and Man gets scared!
g. Step 6.png – Image to be displayed after the sixth incorrect guess. Man is hung in this step and the user loses!
h. Winner.png – Image to be displayed after the user has successfully guessed the word with 1 or more incorrect guesses remaining. A banner with Congratulations appears, and man thanks you for saving his life.

Sample Source Code

Here are the steps needed to execute to build this project:

1. Importing all the necessary modules
2. Setting the variables and getting the monitor’s height and width
3. Creating all the functions
4. Initializing the GUI window, placing components in it and asking if the user wants to play or not

1. Importing all the necessary modules:

```
from tkinter import *

import tkinter.messagebox as mb

from PIL import ImageTk, Image

from screeninfo import get_monitors

from random import choice

from os import system
```

2. Setting the variables and getting the monitor's height and width:

```
# Variables
buttons_details = [["a", "A", 25, 800], ["b", "B", 125, 800], ["c", "C", 225, 800], ["d", "D", 325, 800],
                  ["e", "E", 425, 800], ["f", "F", 525, 800], ["g", "G", 625, 800], ['h', "H", 725, 800],
                  ['i', "I", 825, 800], ['j', "J", 925, 800], ['k', "K", 1025, 800], ['l', "L", 1125, 800],
                  ['m', "M", 1225, 800], ['n', "N", 25, 900], ['o', "O", 125, 900], ['p', "P", 225, 900],
                  ['q', "Q", 325, 900], ['r', "R", 425, 900], ['s', "S", 525, 900], ['t', "T", 625, 900],
                  ['u', "U", 725, 900], ['v', "V", 825, 900], ['w', "W", 925, 900], ['x', "X", 1025, 900],
                  ['y', "Y", 1125, 900], ['z', "Z", 1225, 900]]

word = get_word()
word_length = len(word)-1

correct_guesses = 0
incorrect_guesses = 0
```

```
li_word = [char.upper() for char in word]
# Getting the computer screen's dimensions
monitor = get_monitors()[0]

width = monitor.width - 600
height = monitor.height - 100
```

3. Creating all the functions:

```
# Basic Hangman function
def get_word():
    words = []

    with open('words.txt', 'r') as wordlist:
        for line in wordlist:
            words.append(line)

    chosen_word = choice(words)

    return chosen_word
```

```
def main_function(btn: Button):
    global image_lbl, li_word, correct_guesses, incorrect_guesses
    global step_1, step_2, step_3, step_4, step_5, step_6

    if btn['text'] in li_word:
        btn.config(bg='SpringGreen', fg='Black')
        indices = [indx for indx, val in enumerate(li_word) if val == btn['text']]
        for index in indices:
            exec(f'ltr_{index}["text"] = btn["text"]')
            correct_guesses += 1

    else:
        btn.config(bg='Red', fg='gray')
        incorrect_guesses += 1
        exec(f'image_lbl.config(image=step_{incorrect_guesses})')

    btn.config(state='disabled')
```

```
# Command functions for the buttons at the top-right of the screen
def instructions():
   mb.showinfo('',
               """The instructions are simple:
You have 6 chances of saving Man, if you get 6 incorrect guesses while trying to guess the word, Man's going
to be hung.
If you make less than 6 incorrect guesses while guessing the word, you will be successful in saving Man's life
and he will be eternally grateful to you.""")


def end_game():
   surety = mb.askyesno(
      '', 'Are you sure you want to exit the game? \nYour progress will not be saved')

   if surety:
      root.destroy()
```

```
def reset():
  surety = mb.askyesno('', "Are you sure you want to reset your game?")
  if surety:
     root.destroy()
     system('python main.py')
```

4. Initializing the GUI window, placing components in it and asking if the user wants to play or not:

```
# Initializing the window
root = Tk()
root.title('Hangman')
root.geometry(f'{width}x{height}')
root.resizable(0, 0)
root.config(bg='gray')
root.positionfrom('user')

# Title Label
Label(text='HANGMAN GAME', font=("Comic Sans MS", 22),
    bg='gray').place(relx=0.35, y=15)
```

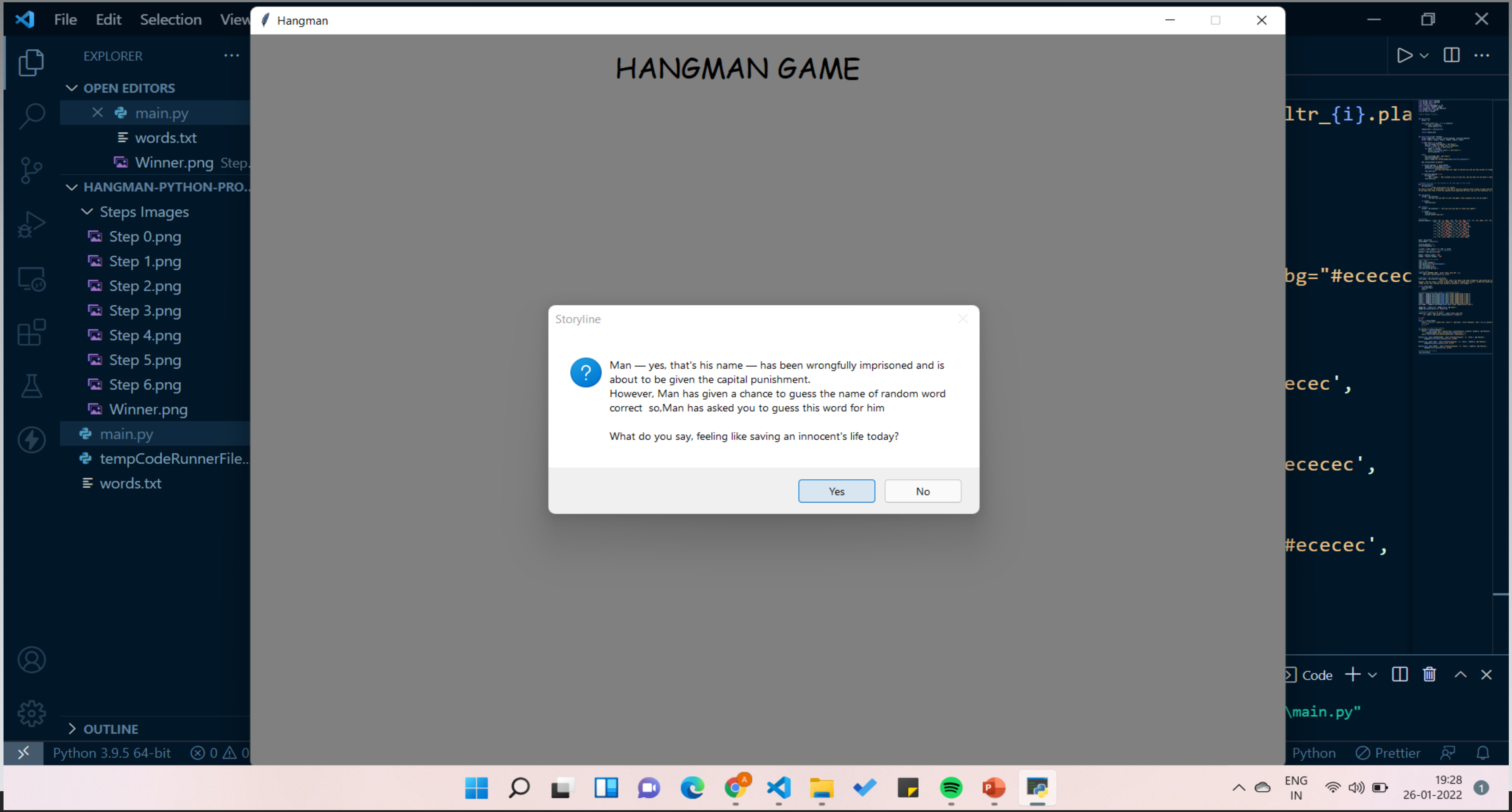
```
# Asking if the user wants to play
start_ques = mb.askyesno('Storyline',
                        '''Man — yes, that's his name — has been wrongfully imprisoned and is about
to be given the capital punishment.
However, Man has given a chance to guess the name of random word correct  so,Man has
asked you to guess this word for him
\nWhat do you say, feeling like saving an innocent's life today?''')

if not start_ques:
    root.destroy()
    exit()
```

```
# Creating all the steps images in PhotoImage format
step_0 = ImageTk.PhotoImage(Image.open('Steps Images/Step 0.png'))
step_1 = ImageTk.PhotoImage(Image.open('Steps Images/Step 1.png'))
step_2 = ImageTk.PhotoImage(Image.open('Steps Images/Step 2.png'))
step_3 = ImageTk.PhotoImage(Image.open('Steps Images/Step 3.png'))
step_4 = ImageTk.PhotoImage(Image.open('Steps Images/Step 4.png'))
step_5 = ImageTk.PhotoImage(Image.open('Steps Images/Step 5.png'))
step_6 = ImageTk.PhotoImage(Image.open('Steps Images/Step 6.png'))
won_image = ImageTk.PhotoImage(Image.open('Steps Images/Winner.png'))

image_lbl = Label(root, image=step_0, bg='gray')
image_lbl.place(relx=0.15, rely=0.09)
```

```
# Placing the word on the window
Label(root, text='Word to guess:', font=("Comic Sans MS",
     20, 'bold'), bg='gray').place(relx=0.6, rely=0.4)

x = 0.6
i = 0
while i < word_length:
   exec(f'''ltr_{i} = Label(root, text="_", bg="gray", font=("Georgia", 30)); ltr_{i}.place(relx={x},
rely=0.45)''')
   x += 0.03
   i += 1
```

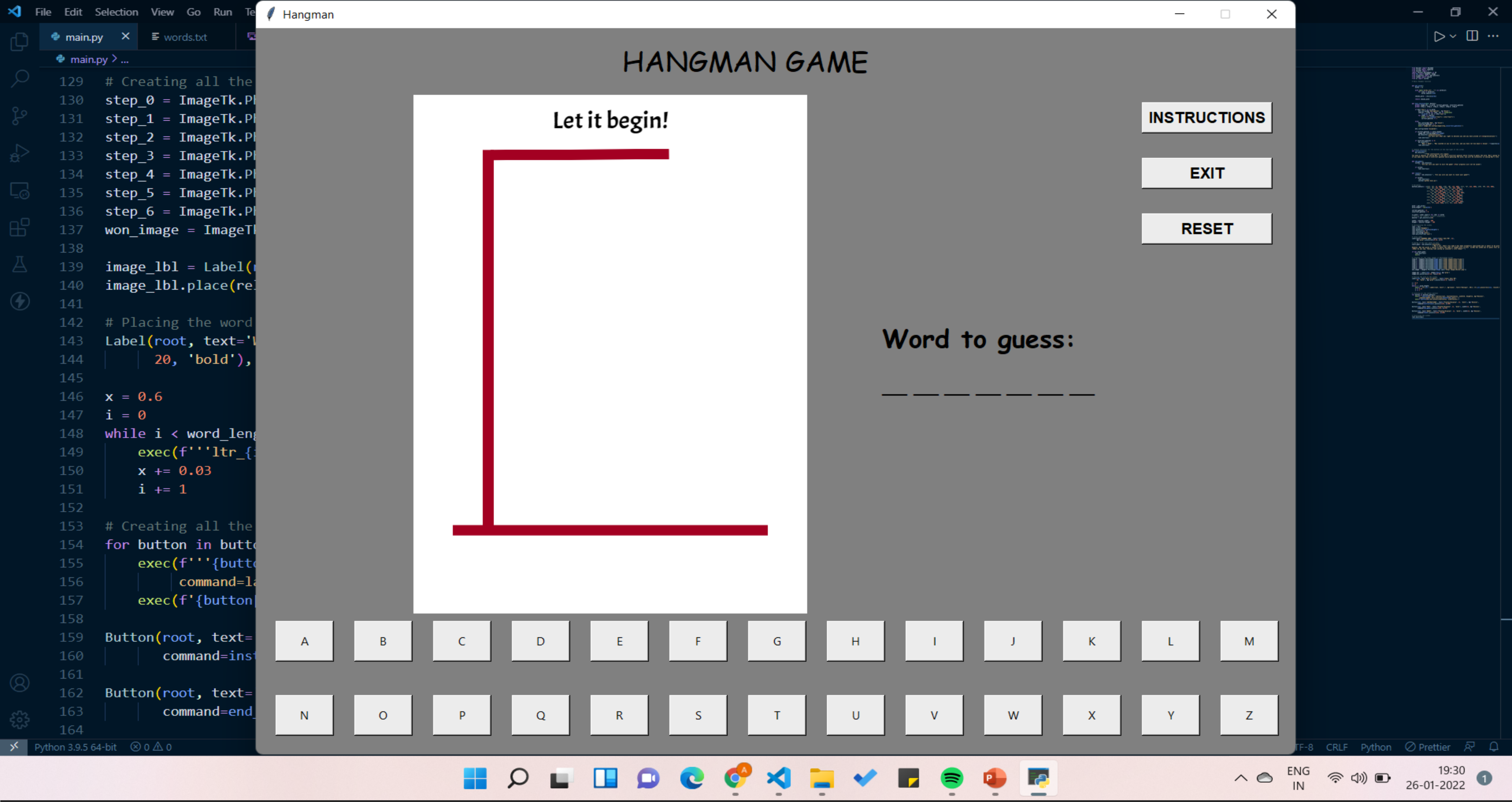
```
# Creating all the letter buttons
for button in buttons_details:
   exec(f'''{button[0]}_btn = Button(root, text=button[1], width=8, height=2,
bg="#ececec",
       command=lambda: main_function({button[0]}_btn))''')
   exec(f'{button[0]}_btn.place(x=button[2], y=button[3])')

Button(root, text='INSTRUCTIONS', font=("PlayfairDisplay", 12, 'bold'), bg='#ececec',
     command=instructions).place(x=1125, y=100)

Button(root, text='EXIT', font=("PlayfairDisplay", 12, 'bold'), width=13, bg='#ececec',
     command=end_game).place(x=1125, y=175)

Button(root, text='RESET', font=("PlayfairDisplay", 12, 'bold'), width=13,
bg='#ececec',
     command=reset).place(x=1125, y=250)

# Finalizing the window
root.mainloop()
```

Screenshots of Hangman Game Output



Man — yes, that's his name — has been wrongfully imprisoned and is about to be given the capital punishment.
However, Man has given a chance to guess the name of random word correct so,Man has asked you to guess this word for him

What do you say, feeling like saving an innocent's life today?



```
# Creating all the
step_0 = ImageTk.P
step_1 = ImageTk.P
step_2 = ImageTk.P
step_3 = ImageTk.P
step_4 = ImageTk.P
step_5 = ImageTk.P
step_6 = ImageTk.P
won_image = ImageT

image_lbl = Label(
image_lbl.place(re

# Placing the word
Label(root, text='
        20, 'bold'),

x = 0.6
i = 0
while i < word_len
    exec(f'''ltr_{
    x += 0.03
    i += 1

# Creating all the
for button in butt
    exec(f'''{butt
            command=l
    exec(f'{button

Button(root, text=
        command=ins

Button(root, text=
        command=end
```

HANGMAN GAME

Let it begin!

INSTRUCTIONS

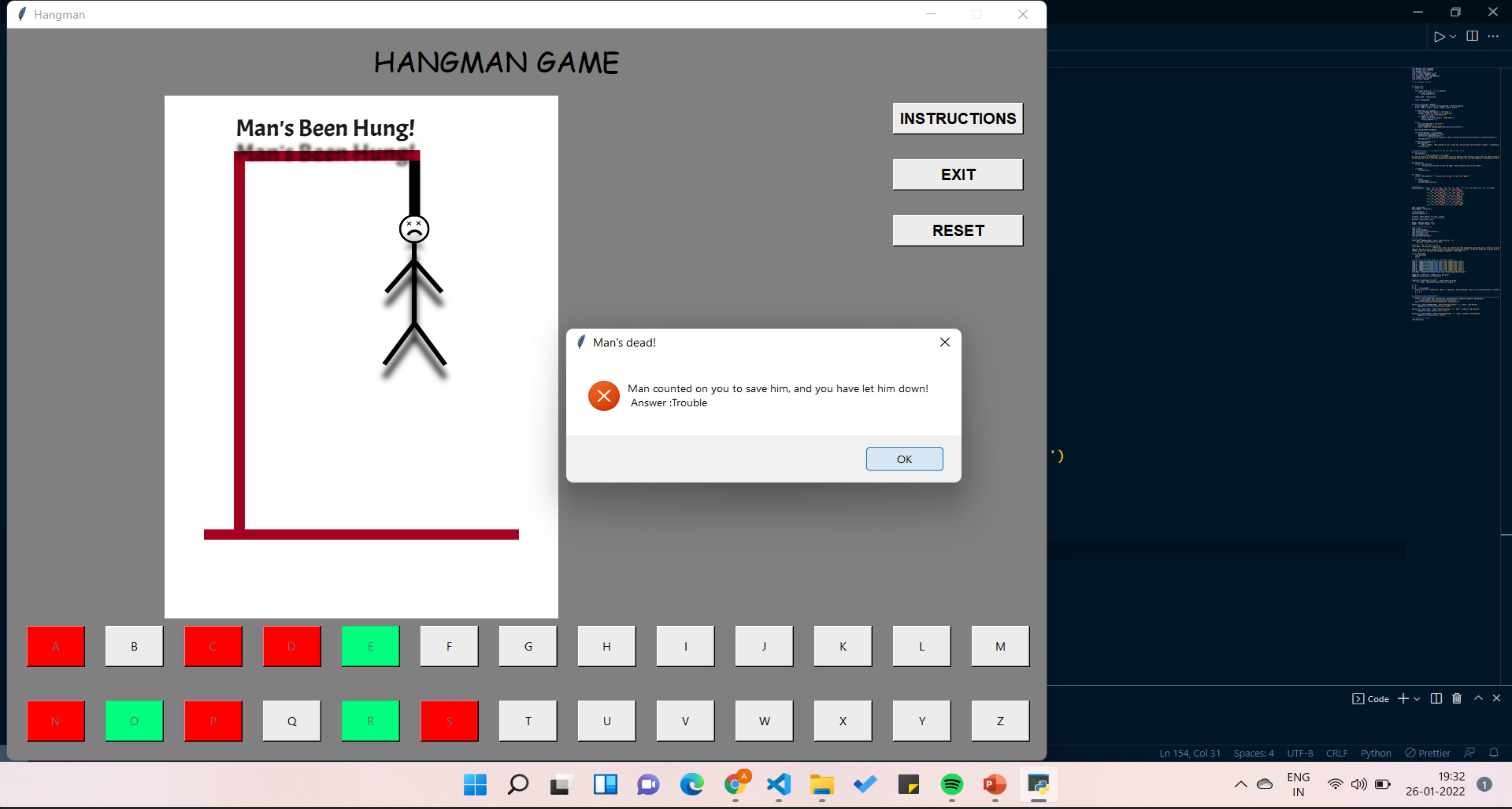
EXIT

RESET

Word to guess:

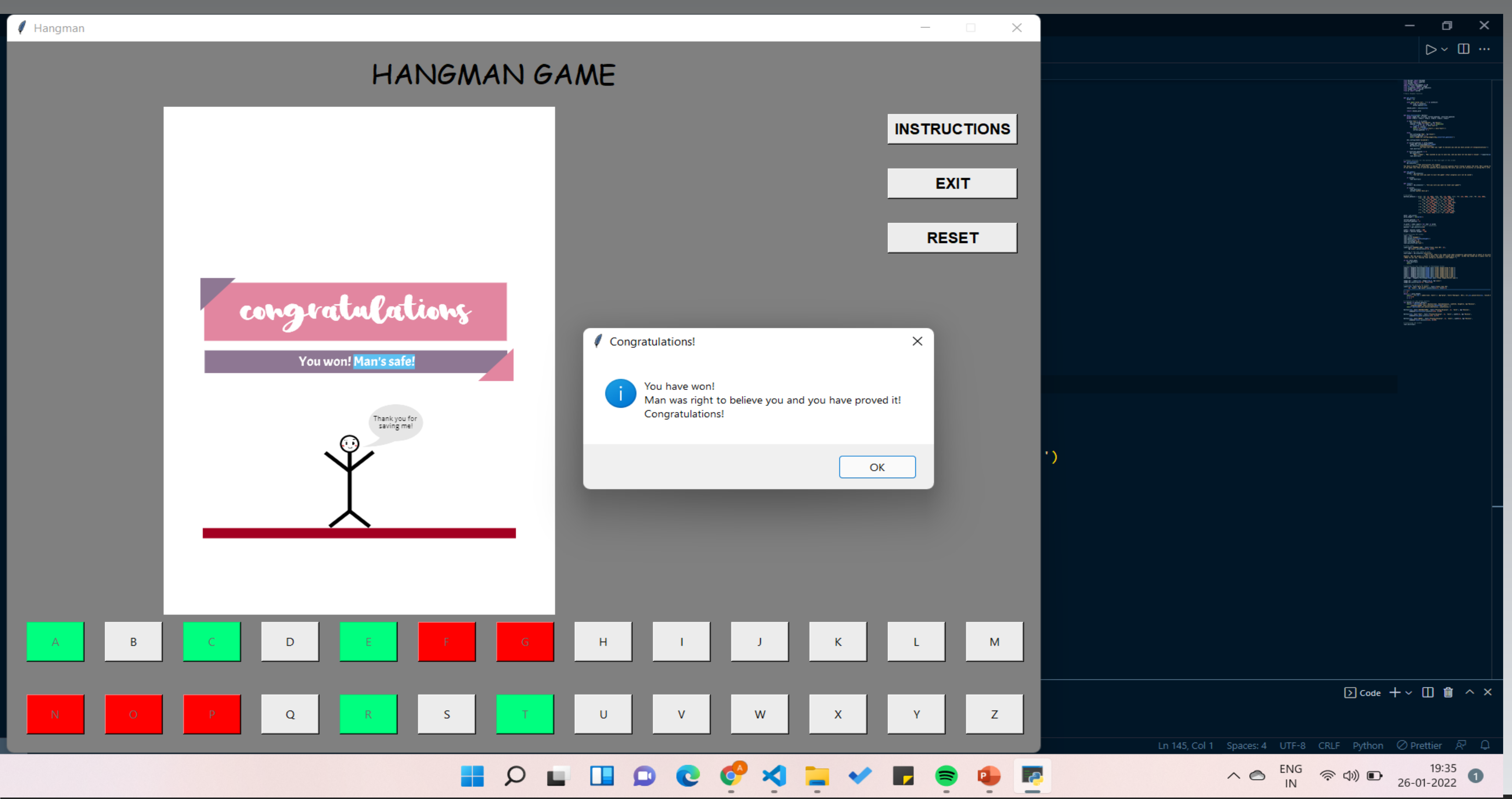
_ _ _ _ _ _ _

A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z


Hangman
HANGMAN GAME
Man's Been Hung!
INSTRUCTIONS
EXIT
RESET
A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z
OK
Ln 154, Col 31 Spaces: 4 UTF-8 CRLF Python Prettier
ENG IN 19:32 26-01-2022

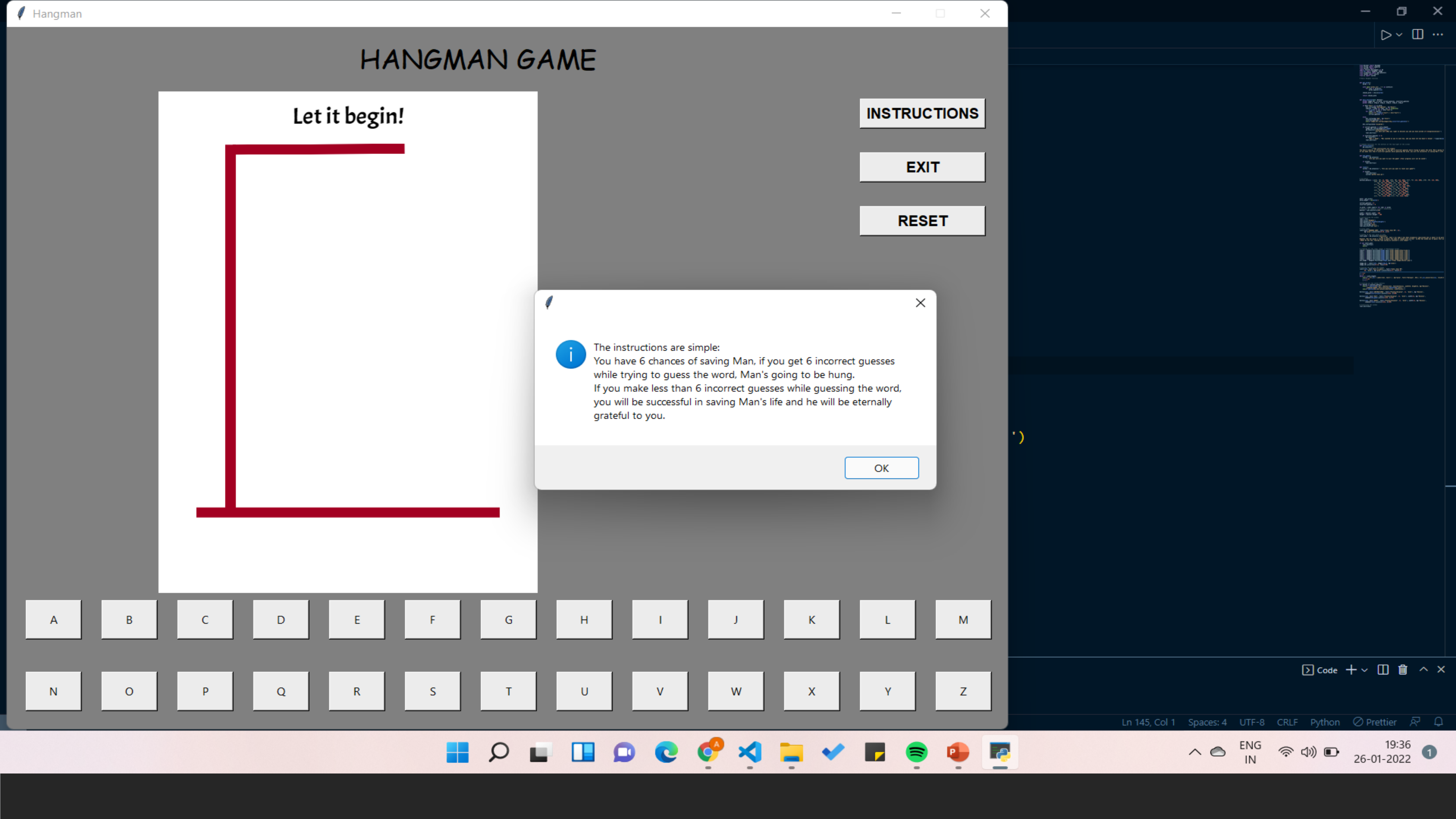
Man's dead!

Man counted on you to save him, and you have let him down!
Answer :Trouble

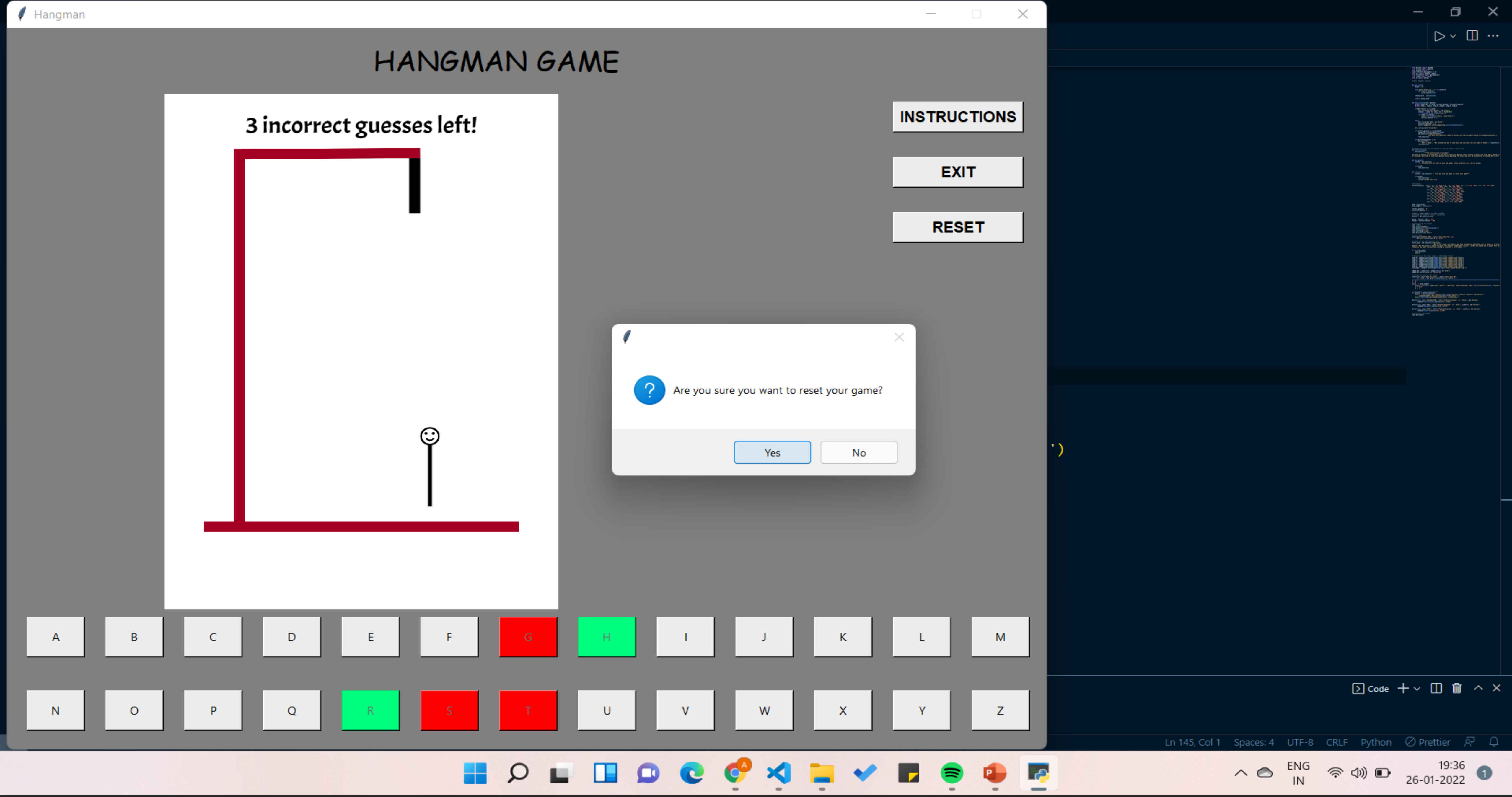


Congratulations!

You have won!
Man was right to believe you and you have proved it!
Congratulations!

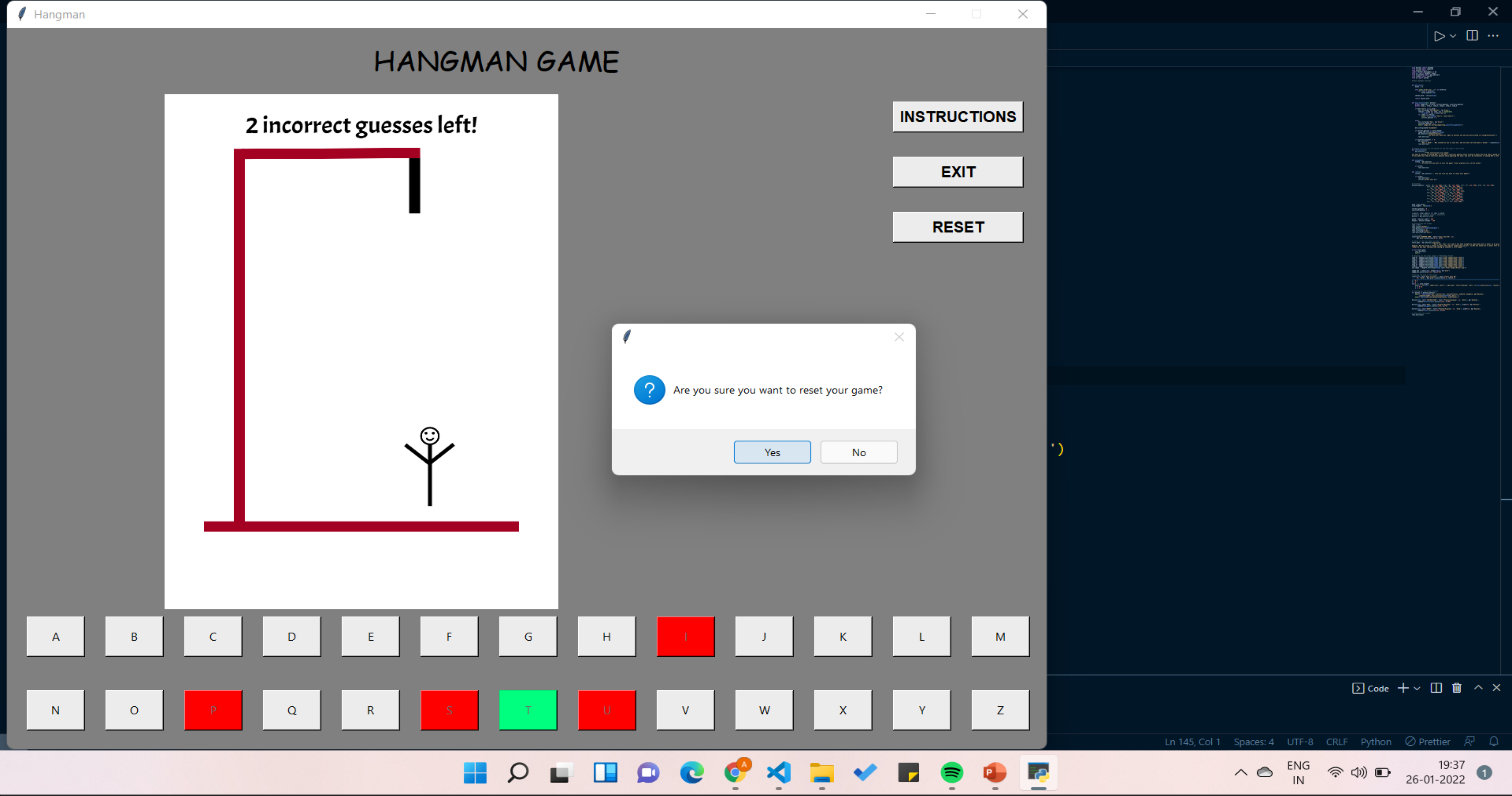


The instructions are simple:
You have 6 chances of saving Man, if you get 6 incorrect guesses while trying to guess the word, Man's going to be hung.
If you make less than 6 incorrect guesses while guessing the word, you will be successful in saving Man's life and he will be eternally grateful to you.



3 incorrect guesses left!

Are you sure you want to reset your game?



Are you sure you want to reset your game?

Conclusion

- We have created a GUI based Hangman game. With the help of intermediate understanding of Tkinter library and PIL’s GUI elements, and basic understanding of the random and screeninfo libraries.
- This is an incredible pass time that also enhances vocabulary and player get to be a superhero who saved an innocent’s life!

Refrences

https://www.tutorialspoint.com/python/python_gui_programming.htm

https://www.geeksforgeeks.org/libraries-in-python/

THANK
YOU