import tkinter as tk

def add\_task():

task = entry.get()

if task:

listbox.insert(tk.END, task)

entry.delete(0, tk.END)

def remove\_task():

selected\_task = listbox.curselection()

listbox.delete(selected\_task)

app = tk.Tk()

app.title("Dynamic To-Do List")

entry = tk.Entry(app, width=30)

entry.pack()

add\_button = tk.Button(app, text="Add Task", command=add\_task)

add\_button.pack()

remove\_button = tk.Button(app, text="Remove Task", command=remove\_task)

remove\_button.pack()

listbox = tk.Listbox(app, width=40, height=10)

listbox.pack()

app.mainloop()

import random

import tkinter as tk

class Hangman:

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

self.root = root

self.word\_list = ['python', 'hangman', 'dynamic', 'code', 'computer']

self.word = random.choice(self.word\_list)

self.guessed\_letters = []

self.incorrect\_guesses = 0

self.max\_incorrect\_guesses = 6

self.display\_word = ['\_'] \* len(self.word)

self.hangman\_parts = [

'O', '|', '/', '\\', '/', '\\'

]

self.init\_ui()

def init\_ui(self):

self.word\_label = tk.Label(self.root, text=" ".join(self.display\_word), font=("Arial", 20))

self.word\_label.pack()

self.guess\_entry = tk.Entry(self.root, font=("Arial", 20))

self.guess\_entry.pack()

self.guess\_button = tk.Button(self.root, text="Guess", font=("Arial", 20), command=self.make\_guess)

self.guess\_button.pack()

self.hangman\_label = tk.Label(self.root, text="", font=("Arial", 20))

self.hangman\_label.pack()

def make\_guess(self):

guess = self.guess\_entry.get().lower()

self.guess\_entry.delete(0, tk.END)

if len(guess) == 1 and guess.isalpha():

if guess in self.word and guess not in self.guessed\_letters:

self.guessed\_letters.append(guess)

self.update\_word(guess)

elif guess not in self.word and guess not in self.guessed\_letters:

self.guessed\_letters.append(guess)

self.incorrect\_guesses += 1

self.update\_hangman()

if '\_' not in self.display\_word:

print("You win!")

self.reset\_game()

if self.incorrect\_guesses >= self.max\_incorrect\_guesses:

print("You lose!")

self.reset\_game()

def update\_word(self, guess):

self.display\_word = [guess if letter == guess else self.display\_word[i] for i, letter in enumerate(self.word)]

self.word\_label.config(text=" ".join(self.display\_word))

def update\_hangman(self):

hangman = ''.join(self.hangman\_parts[:self.incorrect\_guesses])

self.hangman\_label.config(text=hangman)

def reset\_game(self):

self.word = random.choice(self.word\_list)

self.display\_word = ['\_'] \* len(self.word)

self.guessed\_letters = []

self.incorrect\_guesses = 0

self.word\_label.config(text=" ".join(self.display\_word))

self.hangman\_label.config(text="")

root = tk.Tk()

game = Hangman(root)

root.mainloop()