American International University-Bangladesh

Spring 2024-25

Course: PROGRAMMING IN PYTHON
Section: A Group: 07
Supervised By: DR. ABDUS SALAM

GROUP MEMBER

Name	ID
Monisha Sarkar	22-47063-1
Sha Mohammad Yeahia Idris	22-46787-1
Mahmud Reza Mahim	22-46210-1

Project Overview

Word Ladder Game is an interactive word puzzle using Python and Tkinter. The game features a simple graphical user interface that displays the word ladder. In the game player transforms a given start word into a specified end word by entering valid four-letter English words. Each new word entered must differ from the previous one by exactly one letter and must be a real English word, verified using the WordNet dictionary. Players can restart the game with a new word pair for endless challenges. The ladder consists of six steps the first and last words are provided by the game and the player must correctly fill in the four intermediate steps. The game provides immediate feedback on invalid words, ensuring that only legitimate and valid words are used.

Features

1. Predefined Word Ladders

In the Word Ladder Game, a set of predefined word ladders is provided. Each ladder consists of a pair of words, a start word and an end word. The player needs to transform the start word into the end word by changing one letter at a time. These word pairs are stored in a list called ladder_sets. A random ladder pair is selected at the beginning of each game.

Code snippet:

```
ladder_sets = [
 ["cold","worm"],["game","malt"],["bake","cure"],["lead","goof"],["fast","cash"],
 ["make","goon"],["spin","scar"],["word","goal"],["tail","pale"],["hunt","ward"],
 ["life","love"],["star","head"],["find","tome"],["long","bunk"],["coat","cake"],
 ["port","coat"],["fast","look"],["head","tail"],["word","pole"],["star","feat"]
]
```

2. Word Validation and Transformation Rules

To ensure meaningful gameplay, the project follows three main validation requirements for user-submitted words:

- The word must be a valid English word.
- The word must differ from the previous word by exactly one letter.
- The word must be a four letter word.

The function is_valid_word() uses **WordNet** from the nltk to check if a word exists in the English language.

Code snippet:

```
def is_valid_word(word):
    return len(wordnet.synsets(word)) > 0
```

In addition to being a real word, each guessed word must change exactly **one letter** from the previous word. This logic is implemented using the one_letter_diff() function:

Code snippet:

```
def one_letter_diff(w1, w2):
    if len(w1) != len(w2):
        return False
    return sum(a != b for a, b in zip(w1, w2)) == 1
```

Entered word must be 4 letter long

```
        if len(guess) != 4:
            msg.config(text="Enter a 4-letter word.")
            return
```

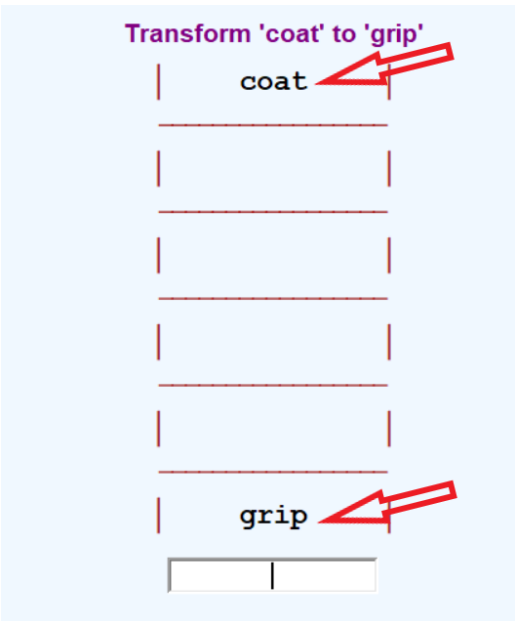
3. Random Word Ladder Picker:

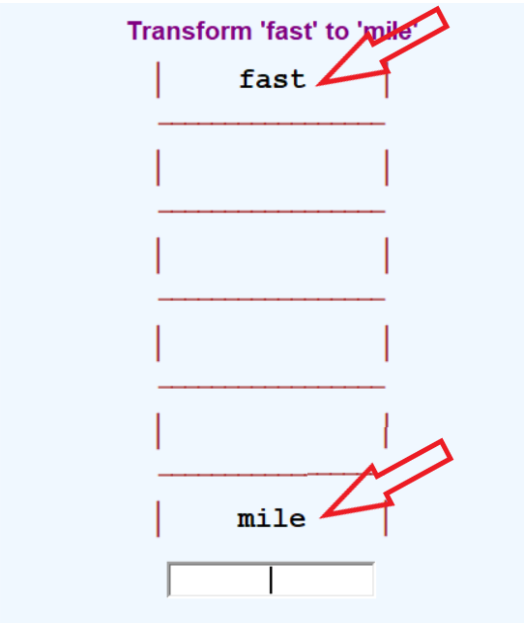
A random word ladder is chosen from the predefined list of word ladders at the beginning of each game. The pick_ladder() function randomly selects a start word and an end word from the ladder_sets list.

Code snippet:

```
def pick_ladder():
    return random.choice(ladder_sets)
```

Output:





4. GUI Setup - First Page & Second Page

The GUI is built using Tkinter and includes two main sections: the Main Menu and the Game Page. The Main Menu displays a welcome message, a short game description, and a ▶️ Start Game button. Clicking the button switches to the Game Page using pack_forget() and pack().On the Game Page, the Word Ladder is visually represented using a vertical sequence of labeled entries. Each step is initially shown as a placeholder (____) and updates as the user inputs valid transformations. The structure is styled to resemble a physical ladder using vertical lines (|) and horizontal rungs (¯¯¯).

Code snippet:

First Page:

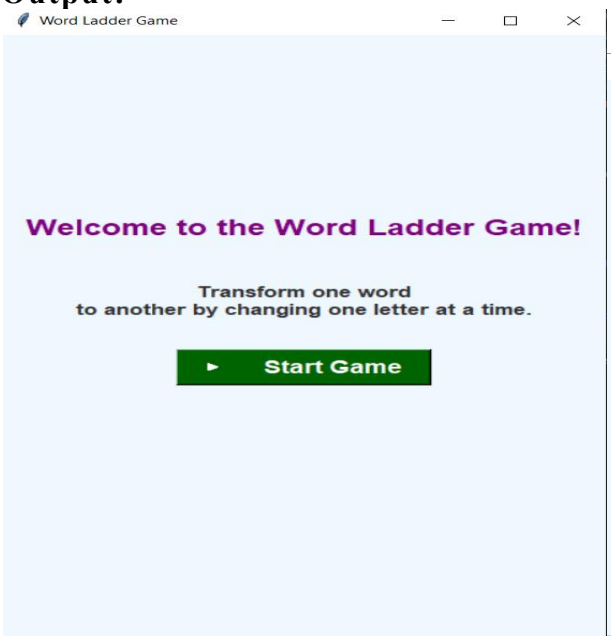
```
main_frame = tk.Frame(window, bg="#f0f8ff")
main_frame.pack(fill="both", expand=True)

tk.Label(main_frame, text=" ", font=("Arial", 16, "bold"),
bg="#f0f8ff").pack(pady=65, anchor="center")
headline = tk.Label(main_frame, text="Welcome to the Word Ladder Game!",
font=("Arial", 18, "bold"), bg="#f0f8ff", fg="purple")
headline.pack(pady=30, anchor="center")

description = tk.Label(main_frame, text="Transform one word\nto another by
changing one letter at a time.", font=("Arial", 13, "bold"), bg="#f0f8ff",
fg="#333333")
description.pack(pady=10, anchor="center")

start_btn = tk.Button(main_frame, text="▶️ Start Game", font=("Arial", 14,
"bold"), bg="dark green", fg="white", width=15, command=lambda:
show_game_page())
start_btn.pack(pady=20, anchor="center")
```

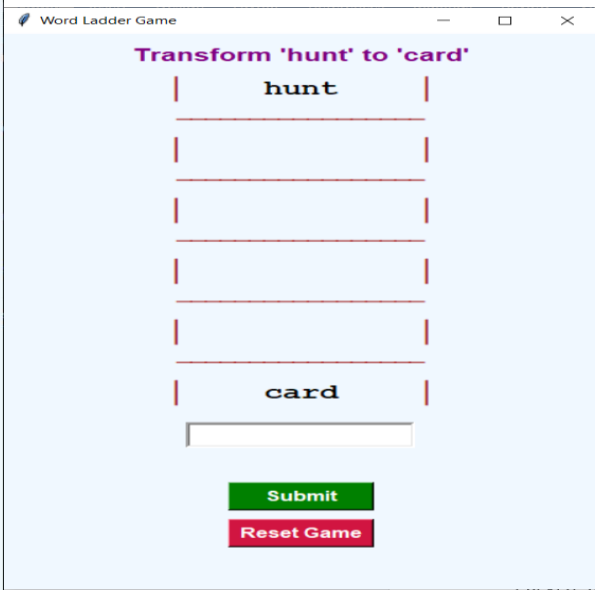
Second Page:

```
game_frame = tk.Frame(window, bg="#f0f8ff")

def show_game_page():
    main_frame.pack_forget()
    game_frame.pack(fill="both", expand=True)

    frame = tk.Frame(game_frame, bg="#f0f8ff")
    frame.pack(pady=10)

    global headline
    headline = tk.Label(frame, text=f"Transform '{start_word}' to
'{end_word}'", font=("Arial", 16, "bold"), bg="#f0f8ff", fg="purple")
    headline.pack()
```

```
    ladder_frame = tk.Frame(frame, bg="#f0f8ff")
    ladder_frame.pack(pady=5)

    global labels
    labels = []

    for i in range(max_steps):
        row_frame = tk.Frame(ladder_frame, bg="#f0f8ff")
        row_frame.pack()

        tk.Label(row_frame, text="|", font=("Courier", 22, "bold"),
bg="#f0f8ff", fg="brown").pack(side="left", padx=10)
        lbl = tk.Label(row_frame, text=" ", font=("Courier", 18, "bold"),
width=10, bg="#f0f8ff", fg="black")
        lbl.pack(side="left")
        labels.append(lbl)
        tk.Label(row_frame, text="|", font=("Courier", 22, "bold"),
bg="#f0f8ff", fg="brown").pack(side="left", padx=10)

        if i < max_steps - 1:
            tk.Label(ladder_frame, text="————————————————", font=("Courier",
14, "bold"), bg="#f0f8ff", fg="brown").pack(pady=2)

    labels[0].config(text=start_word)
    labels[-1].config(text=end_word)
```

Output:





5. Reset Game Button

The Reset Game button allows the player to restart the game at any time. When this button is clicked, the program selects a new random pair of start and end words from the predefined list. It then clears all user inputs and resets the entire ladder interface to its initial state. Additionally, the instructional headline at the top of the screen is updated to according to the new start and end words.

Code snippet:

```
def restart():
    global start_word, end_word, ladder_words, current_index
    start_word, end_word = pick_ladder()
    ladder_words = ["" for _ in range(max_steps)]
    ladder_words[0] = start_word
    ladder_words[-1] = end_word
    current_index = 1

    headline.config(text=f"Transform '{start_word}' to '{end_word}'")
    for i in range(max_steps):
        labels[i].config(text=" ")
    labels[0].config(text=start_word)
    labels[-1].config(text=end_word)
    msg.config(text="", fg="red")
    entry.delete(0, tk.END)
    entry.focus_set()
```

Output:

6. User Input (Submit Word)

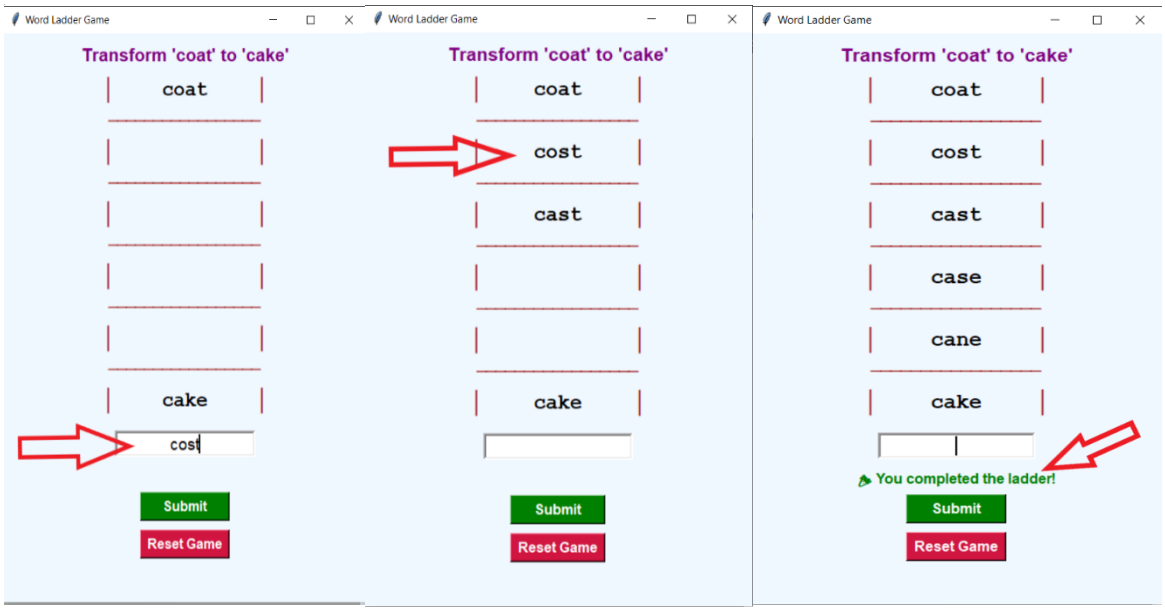
The Submit Word feature enables the player to input a guess that must be a valid four-letter English word differing by exactly one letter from the previous word. Upon submission, the game checks the word's validity, updates the ladder if correct, and provides feedback messages. If the second-to-last word is correctly placed, the player is congratulated.

Code snippet:

```
def submit_word():
        global current_index
        guess = entry.get().strip().lower()
        entry.delete(0, tk.END)

        if current_index >= max_steps - 1:
            msg.config(text="Game already finished!")
            return

        # Reset the message to empty after the word is entered
        msg.config(text="", fg="red")

        if len(guess) != 4:
            msg.config(text="Enter a 4-letter word.")
            return

        if not is_valid_word(guess):
            msg.config(text="Invalid English word.")
            return

        prev_word = ladder_words[current_index - 1]
        if not one_letter_diff(prev_word, guess):
            msg.config(text="Must differ by 1 letter.")
            return

        ladder_words[current_index] = guess
        labels[current_index].config(text=guess)

        if current_index == max_steps - 2:
            if one_letter_diff(guess, end_word):
                msg.config(text="🎉 You completed the ladder!", fg="green")
            else:
               msg.config(text="❌ End word not reachable from here!",fg="red")

        current_index += 1
```

```
    btn = tk.Button(frame, text="Submit", command=submit_word)
    btn.pack(pady=5)
```

Output:



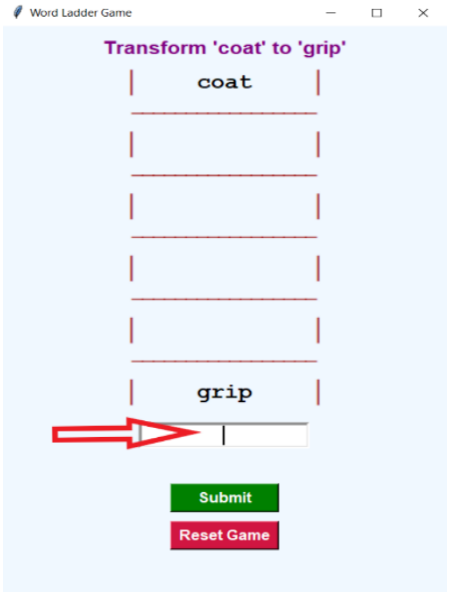
7. Auto Focus on Input

To enhance user experience, the game ensures that the input field (where players enter their word guesses) is automatically focused whenever the game starts or restarts. This eliminates the need for players to click the text box before typing, making gameplay faster and more intuitive. The focus is set right after creating the input widget and also after the game is restarted.

Code snippet:

```
entry.focus_set()
```

Output:



8. Feedback Message Display

The game provides dynamic feedback to guide the player. Messages appear below the input field to inform the player whether the word is valid, if it differs by only one letter,if the word is invalid or if the game has been successfully completed. The color of the text (red for errors, green for success) provides a visual cue as well.

Code snippet:

```
def submit_word():
    global current_index
    guess = entry.get().strip().lower()
    entry.delete(0, tk.END)

    if current_index >= max_steps - 1:
        msg.config(text="Game already finished!")
        return

    msg.config(text="", fg="red")

    if len(guess) != 4:
        msg.config(text="Enter a 4-letter word.")
        return

    if not is_valid_word(guess):
        msg.config(text="Invalid English word.")
        return

    prev_word = ladder_words[current_index - 1]
    if not one_letter_diff(prev_word, guess):
        msg.config(text="Must differ by 1 letter.")
        return
```

Output:

9. Keyboard Shortcuts

To improve usability, the game allows players to press the Enter (Return) key to submit their guess instead of clicking the "Submit" button. This is done by binding the Return key to the submit function, making input submission quicker and more natural during gameplay.

Code snippet:

```
window.bind('<Return>', lambda e: submit_word())
```

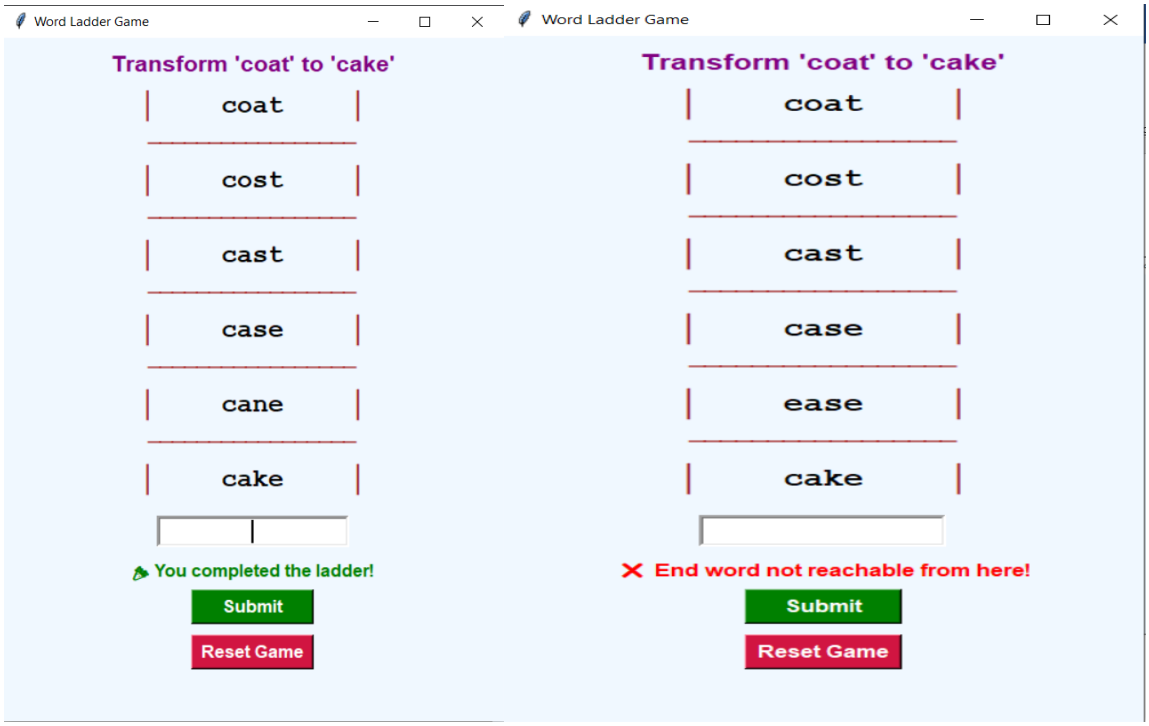
10. End-of-Game Logic

The game includes logic to check if the player has successfully completed the word ladder. When the second-to-last word is entered, it checks if it differs by exactly one letter from the end word. If the condition is met, a success message is shown; otherwise, the player is informed that the transformation isn't valid.

Code snippet:

```
  ladder_words[current_index] = guess
    labels[current_index].config(text=guess)

    if current_index == max_steps - 2:
        if one_letter_diff(guess, end_word):
            msg.config(text="🎉 You completed the ladder!", fg="green")
        else:
            msg.config(text="❌ End word not reachable from here!",
fg="red")

    current_index += 1
```

Output:


Word Ladder Game
Transform 'coat' to 'cake'
coat
cost
cast
case
cane
cake
You completed the ladder!
Submit
Reset Game
Word Ladder Game
Transform 'coat' to 'cake'
coat
cost
cast
case
ease
cake
End word not reachable from here!
Submit
Reset Game