

Neural Networks & Deep Learning ICP-2

Venkata Suraj, Gamini
700744962

GitHub link: <https://github.com/SurajGamini18/Neural-Networks-Deep-Learning-Assignments>

Video Link: <https://drive.google.com/file/d/163fs1Zg0OzK5-jtXd-R5TyYPwWuSrUb/view?usp=sharing>

Question 1:

```
# Define a function to concatenate first_name and last_name into a full name
def fullname(first_name, last_name):
    full_name = first_name + " " + last_name
    return full_name

# Define a function to return every other character in a given string
def string_alternative(full_name):
    alternate_chars = ""
    for i in range(0, len(full_name), 2):
        alternate_chars += full_name[i]
    return alternate_chars

# Main function to collect user input, process it, and display results
def main():
    # Prompt the user to enter their first name
    first_name = input("Enter your first name: ")

    # Prompt the user to enter their last name
    last_name = input("Enter your last name: ")

    # Call the fullname function to concatenate the first and last names
    full_name = fullname(first_name, last_name)

    # Display the full name
    print("Full Name:", full_name)

    # Call the string_alternative function to get alternate characters in the full name
    alternate_chars = string_alternative(full_name)

    # Display the alternate characters in the full name
    print("Alternate Characters in Full Name:", alternate_chars)

# Check if the script is run as the main program
if __name__ == "__main__":
    main()
```

Explanation:

1. fullname Function:

- Purpose: Concatenates the first name and last name with a space in between to form a full name.
- Parameters: first_name (string), last_name (string).
- Returns: full_name (string) which is a combination of first_name and last_name.

2. string_alternative Function:

- Purpose: Creates a new string consisting of every other character from a given string, starting with the first character.
- Parameters: full_name (string).
- Returns: alternate_chars (string) which is a string of alternate characters from the full_name.

3. main Function:

- Acts as the entry point for the script when executed.
- It prompts the user to enter their first and last name.
- Uses the fullname function to combine these names.
- Prints the user's full name.
- Calls the string_alternative function to generate a string of alternate characters from the full name.
- Prints this alternate character string.

4. **Script Execution Flow**:

- If the script is the main program (not imported as a module), it will execute the `main` function.
- The `main` function orchestrates the user input, processing (using `fullname` and `string_alternative` functions), and output display.

Output:

```
Enter your first name: Venkata Suraj
Enter your last name: Gamini
Full Name: Venkata Suraj Gamini
Alternate Characters in Full Name: VnaaSrjGmn
```

Question 2:

```
import re

# Function to count words in a line
def count_words(line, word_counts):
    # Use regular expression to split the line into words and exclude punctuation
    words = re.findall(r'\b\w+\b', line.lower())
    for word in words:
        word_counts[word] = word_counts.get(word, 0) + 1

# Read input from input.txt file
with open("/content/input.txt", "r") as input_file:
    lines = input_file.readlines()

# Initialize a dictionary to store word counts
word_counts = {}

# Count words in the entire file
for line in lines:
    count_words(line.strip(), word_counts)

# Print the original lines
print("Input:")
for line in lines:
    print(line.strip())

# Print the word counts, displaying each word only once
print("Word_Count:")
displayed_words = set()
for line in lines:
    words = re.findall(r'\b\w+\b', line.lower())
    for word in words:
        if word not in displayed_words:
            print(f"{word}: {word_counts[word]}")
            displayed_words.add(word)

# Write the output to output.txt file
with open("output.txt", "w") as output_file:
    output_file.write("Input:\n")
    for line in lines:
        output_file.write(line.strip() + "\n")
    output_file.write("Word_Count:\n")
    for word in displayed_words:
        output_file.write(f"{word}: {word_counts[word]}\n")

print("Output saved to output.txt")
```

Explanation:

1. Function count_words:

- This function takes a line of text and a dictionary (`word_counts`) as inputs.
- It uses a regular expression (`re.findall(r'\b\w+\b', line.lower())`) to extract words from the line. The `\b\w+\b` pattern matches word boundaries to ensure only whole words are selected, and `line.lower()` ensures the process is case-insensitive.
- Each word found is either added to the `word_counts` dictionary or its count is incremented if it already exists.

2. Reading the Input File:

- The script opens and reads a file (`input.txt`) line by line.

3. Counting Words in the Entire File:

- The script initializes an empty dictionary `word_counts` to store the frequency of each word.

- It then iterates over each line in `lines`, calling `count_words` to update `word_counts`.

4. Printing the Input:

- The script prints "Input:" followed by each line from the input file.

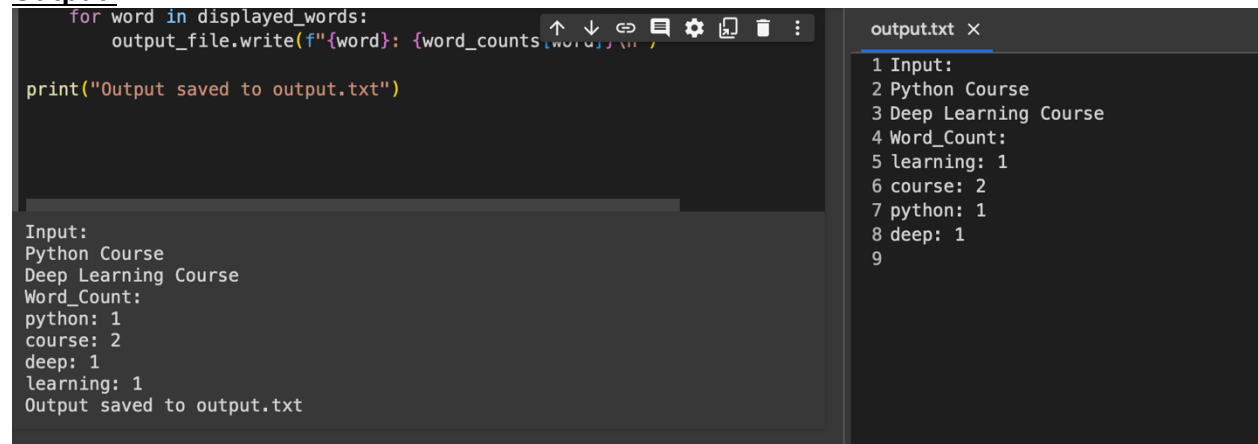
5. Printing the Word Count:

- "Word_Count:" is printed to signify the start of the word count output.
- It initializes an empty set `displayed_words` to track which words have been printed.
- For each line, it extracts words again and checks if each word has already been displayed. If not, it prints the word and its count, and adds the word to `displayed_words`.

6. Writing to Output File:

- The script creates/opens an `output.txt` file.
- It writes the original text lines under "Input:\n" and then the word counts under "Word_Count:\n".

Output:



```
for word in displayed_words:
    output_file.write(f"{word}: {word_counts[word]}\n")
print("Output saved to output.txt")
```

Input:
Python Course
Deep Learning Course
Word_Count:
python: 1
course: 2
deep: 1
learning: 1
Output saved to output.txt

output.txt x
1 Input:
2 Python Course
3 Deep Learning Course
4 Word_Count:
5 learning: 1
6 course: 2
7 python: 1
8 deep: 1
9

Question 3:

```
import ast

def centimeters_to_inches(centimeters):
    return centimeters / 2.54

# Function to read a list of heights from user input
def get_heights():
    input_string = input("Enter a list of heights in centimeters: ")
    try:
        # Safely evaluate the input string to a list
        heights = ast.literal_eval(input_string)
        if isinstance(heights, list) and all(isinstance(height, int) for height in heights):
            return heights
        else:
            raise ValueError
    except (ValueError, SyntaxError):
        print("Invalid input. Please enter a valid list of integers.")
        return []

# Read heights from user
heights_cm = get_heights()

# Convert to inches using a nested loop
heights_in_inches_loop = []
for height in heights_cm:
    inches = centimeters_to_inches(height)
    heights_in_inches_loop.append(round(inches, 2))

# Convert to inches using list comprehension
heights_in_inches_comprehension = [round(centimeters_to_inches(height), 2) for height in heights_cm]

# Output
print("Heights in Inches (Nested Loop):", heights_in_inches_loop)
print("Heights in Inches (List Comprehension):", heights_in_inches_comprehension)
```

Explanation:

1. Convert Centimeters to Inches:

- The function `centimeters_to_inches` takes a height in centimeters and converts it to inches.

2. Read Heights from User Input:

- The function `get_heights` prompts the user to enter a list of heights in centimeters.
- It uses `ast.literal_eval` to safely convert the input string into a Python list.
- The function checks if the input is a valid list of integers. If not, it prompts an error message and returns an empty list.

3. Convert Heights to Inches:

- The script reads the list of heights in centimeters from the user.
- It then converts these heights to inches using two methods:
 - A nested loop: Each height is converted individually and added to the `heights_in_inches_loop` list.
 - List comprehension: A more concise method to achieve the same result, stored in `heights_in_inches_comprehension`.

4. Output Results:

- Finally, the script prints the converted heights in inches using both methods for comparison.

This script effectively demonstrates two common ways to process lists in Python (a loop and list comprehension) and applies basic user input handling along with safe string evaluation using ``ast.literal_eval``.

Output:

```
Enter a list of heights in centimeters: [150,155, 145, 148]
Heights in Inches (Nested Loop): [59.06, 61.02, 57.09, 58.27]
Heights in Inches (List Comprehension): [59.06, 61.02, 57.09, 58.27]
```