**Neural Networks & Deep Learning**

**ICP-1**

Sreeja Reddy Konda

700756597

GitHub Link: https://github.com/SreejaReddyKonda/Neural-Network-Sreeja/tree/main/Neural%20Networks/ICP-1

Video Link: https://drive.google.com/file/d/1lz\_k9IanpGhXDiK5Q51\_fiGKmumu73Y1/view?usp=sharing

A screen shot of a computer

Description automatically generated

Output-

A close up of a text

Description automatically generated

Explanation:

The code consists of two functions:

1. **full\_\_Name(fName, lName)**:
   * Combines the given fName (first name) and lName (last name) with a space in between.
   * Prints the full name.
2. **string\_alternative(str)**:
   * Takes a string str and extracts every second character using slicing (str[::2]).
   * Prints the alternate characters.

**Main Program**:

* Prompts the user to input their first and last names, then calls full\_\_Name to print the combined full name.
* Prompts the user for a string and calls string\_alternative to print the alternate characters.

A screenshot of a computer program

Description automatically generated

Output-

A white background with black dots

Description automatically generated

Explanation:

**File Operations**:

* open("testinput.txt", "w"): Opens or creates testinput.txt in write mode, allowing data to be written to the file.
* f.write(): Writes lines of text to the file.
* f.close(): Closes the file, ensuring all data is saved.
* open("testinput.txt", "r"): Reopens the file in read mode to read its contents.
* f.read(): Reads and prints the entire content of the file.

**Word Counting**:

* from collections import Counter: Imports Counter, a dictionary subclass from the collections module that counts hashable objects like words.
* file.readlines(): Reads all lines from testinput.txt into a list of strings.
* line.strip().split(): Removes leading/trailing whitespace from a line and splits it into individual words.
* Counter(words): Creates a dictionary-like object where each word is a key, and its count is the value.
* wordcountperline.append(): Adds the word count for each line to the wordcountperline list.

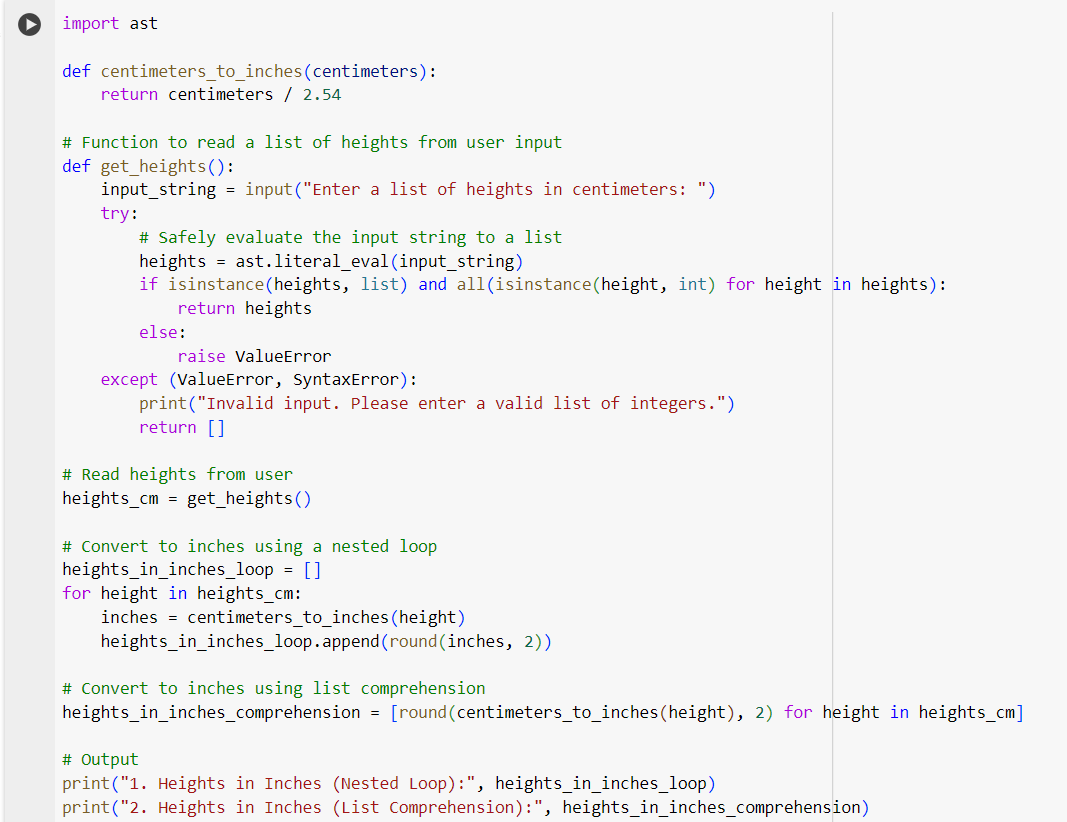
**Word Count Aggregation**:

* Counter(word for wc in wordcountperline for word in wc): Combines the word counts from all lines into a single Counter object, tallying the occurrences of each word across all lines.

**Output to a File**:

* The word counts are printed and then written to a new file testoutput.txt, appending the word counts below the original text.

3.



Output-

A black text on a white background

Description automatically generated

Explanation:

**centimeters\_to\_inches(centimeters) Function**: Converts a height from centimeters to inches. It uses the conversion factor where 1-inch equals 2.54 centimeters.

**get\_heights() Function**:

* Prompts the user to enter a list of heights in centimeters as a string.
* Uses ast.literal\_eval to safely evaluate the string into a Python list.
* Checks if the result is a list of integers. If not, it prints an error message and returns an empty list.

**Conversion Using a Loop**:

* Iterates over the list of heights.
* Converts each height to inches using centimeters\_to\_inches.
* Rounds the result to two decimal places and appends it to a list (heights\_in\_inches\_loop).

**Conversion Using List Comprehension**:

* Converts the heights to inches using a single line of code that performs the same operations as the loop.
* The result is stored in heights\_in\_inches\_comprehension.

**Output**:

* Prints the heights in inches obtained from both the nested loop and the list comprehension methods.