

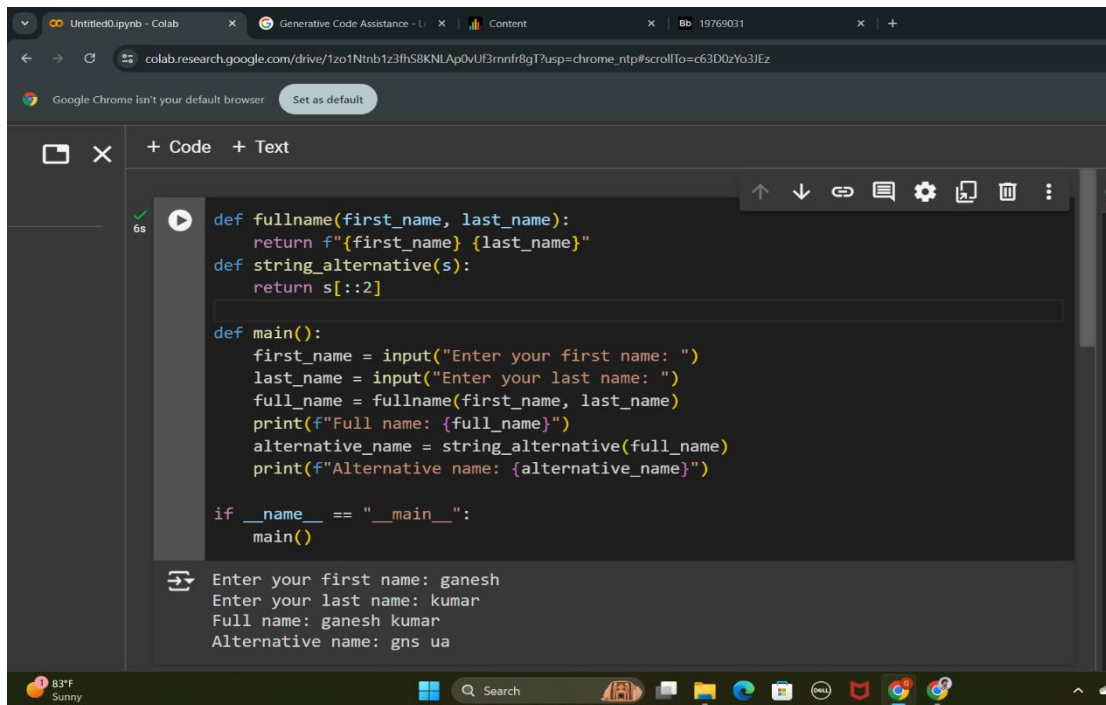
NAME:GANESH KUMAR KORRA

STUDENT ID:700761716

NEURAL NETWORKS ASSIGNMENT -2

GITHUB: [GaneshKumarKorra/ICP2 \(github.com\)](https://github.com/GaneshKumarKorra/ICP2)

1ST CODE

A screenshot of a Google Colab notebook interface. The top bar shows the notebook name 'Untitled0.ipynb - Colab' and the URL 'colab.research.google.com/drive/1zo1Ntnb123fhS8KNLAp0vUf3rnnfr8gT?usp=chrome_nrt#scrollTo=c63D0zY63JEz'. The main area contains a Python script with two functions: 'fullname' and 'string_alternative', and a 'main' function that takes user input. The output shows the user entering 'ganesh' and 'kumar', resulting in 'Full name: ganesh kumar' and 'Alternative name: gns ua'.

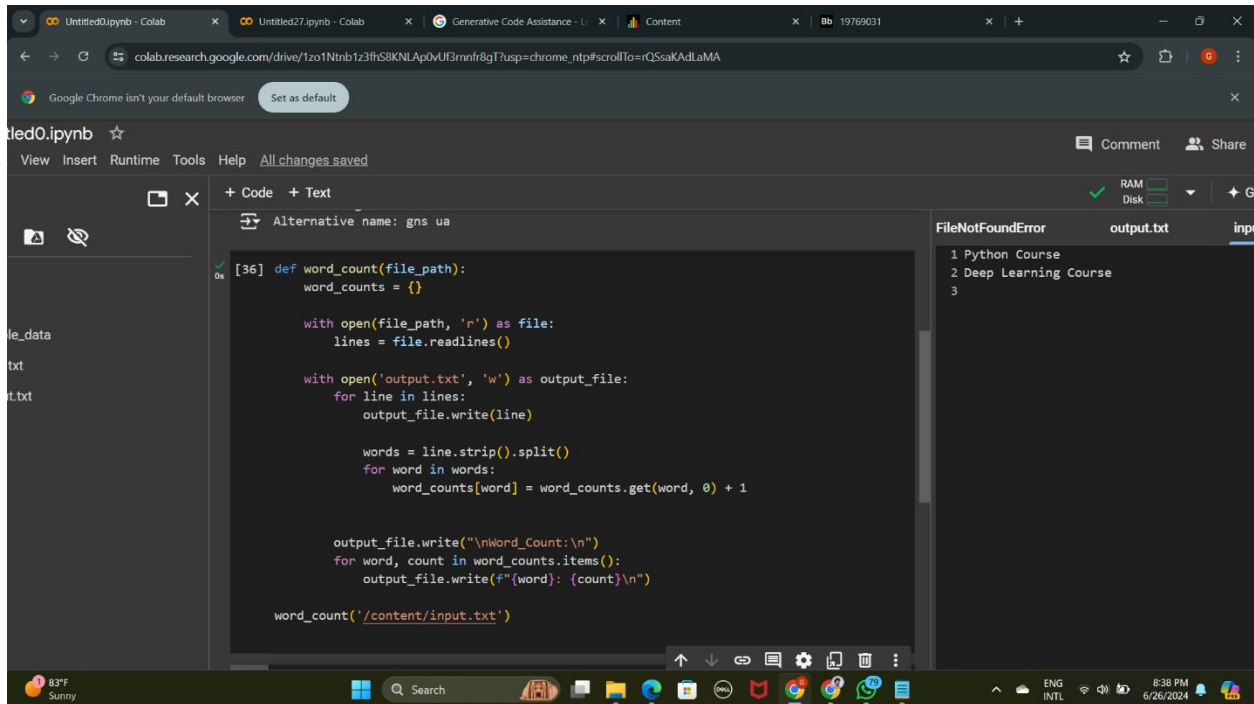
```
def fullname(first_name, last_name):  
    return f"{first_name} {last_name}"  
  
def string_alternative(s):  
    return s[::2]  
  
def main():  
    first_name = input("Enter your first name: ")  
    last_name = input("Enter your last name: ")  
    full_name = fullname(first_name, last_name)  
    print(f"Full name: {full_name}")  
    alternative_name = string_alternative(full_name)  
    print(f"Alternative name: {alternative_name}")  
  
if __name__ == "__main__":  
    main()
```

Enter your first name: ganesh
Enter your last name: kumar
Full name: ganesh kumar
Alternative name: gns ua

DESCRIPTION:

The provided Python script includes two main functions: `fullname`, which combines a first name and last name into a full name, and `string_alternative`, which returns every second character of a given string. The main function prompts the user for their first and last names, generates the full name using `fullname`, and prints it. It then generates an alternative version of the full name using `string_alternative` and prints this as well. The script ensures that main runs only when the script is executed directly, not when imported as a module. This concise program demonstrates basic string manipulation and user interaction in Python.

2ND CODE INPUT



The screenshot shows a Google Colab notebook interface. The browser address bar displays the URL: `colab.research.google.com/drive/1zo1Ntnb1z3fhS8KNIAp0vUf3rnnfr8gT?usp=chrome_nntp#scrollTo=rQ5saKAdLaMA`. The notebook has a tab titled "Untitled0.ipynb". The code editor shows a Python function `word_count` that reads a file, counts word occurrences, and writes the results to `output.txt`. The function is called with the path `"/content/input.txt"`. The output window on the right shows the content of `output.txt`:

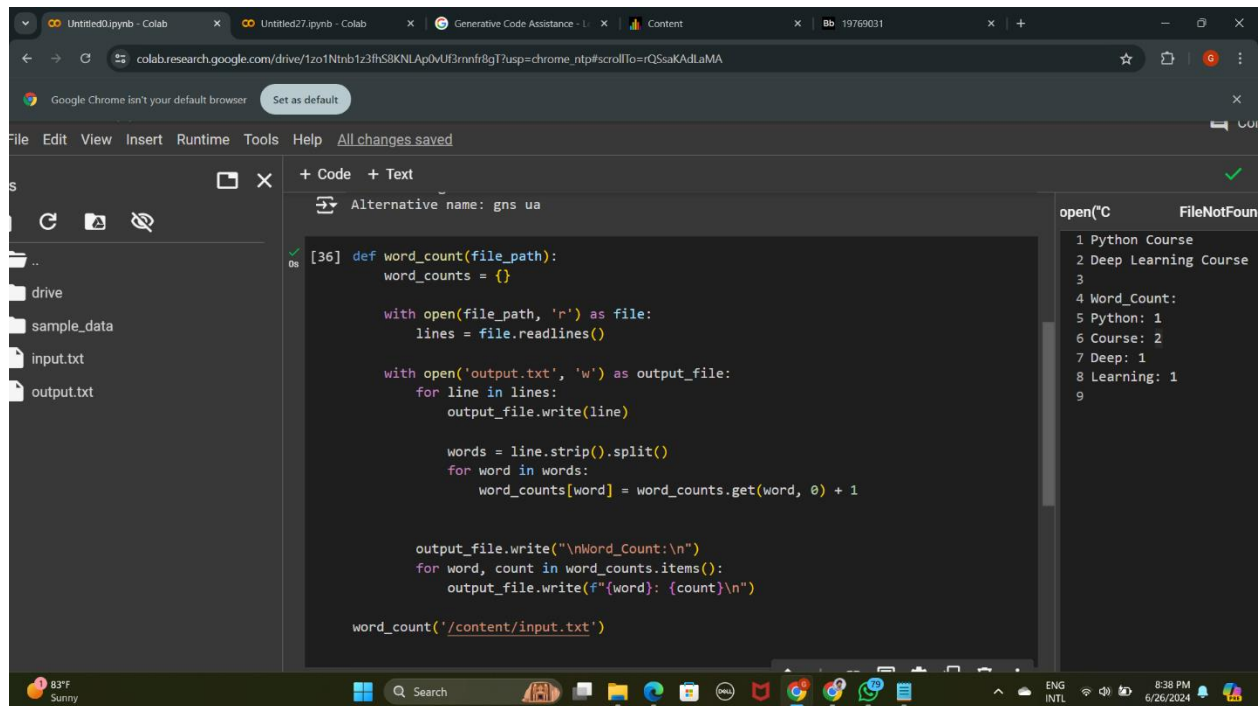
```
FileNotFoundError: output.txt
1 Python Course
2 Deep Learning Course
3
```

The Windows taskbar at the bottom shows the date and time as 8:38 PM on 6/26/2024.

DESCRIPTION:

The `word_count` Python script reads a text file, counts the occurrences of each word, and writes the results to an output file. It initializes a dictionary to store word counts, reads lines from the input file, and writes these lines to `output.txt`. For each line, the script splits it into words, updates the word counts in the dictionary, and finally writes the word counts to the output file in a formatted manner. The script is executed by calling the `word_count` function with the path to the input file.

2ND CODE OUTPUT



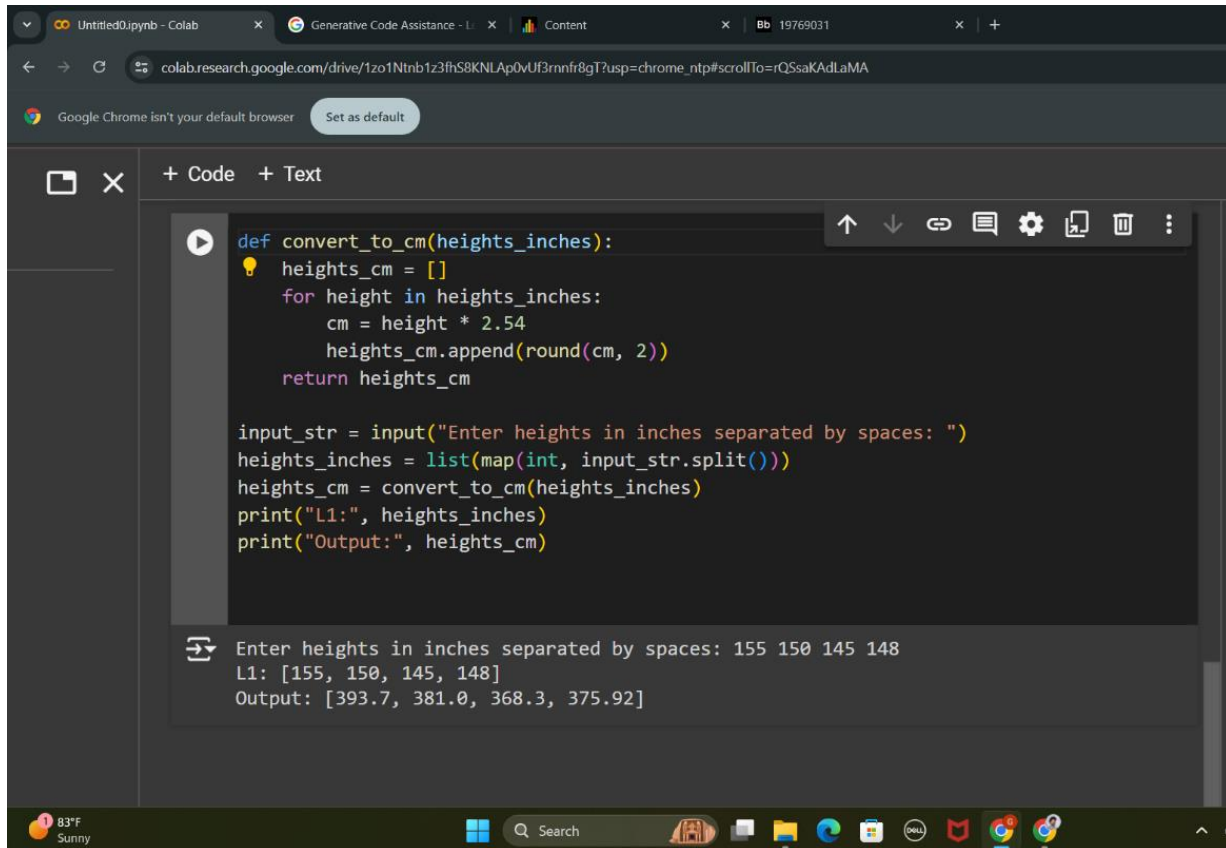
The screenshot shows a Google Colab notebook interface. The left sidebar displays a file explorer with folders 'drive' and 'sample_data', and files 'input.txt' and 'output.txt'. The main code editor contains a Python function `word_count` that reads lines from a file, counts the frequency of each word, and writes the results to 'output.txt'. The function is called with `word_count('/content/input.txt')`. The right sidebar shows the output of the code execution, which is a list of words and their counts.

```
[36] def word_count(file_path):  
    word_counts = {}  
  
    with open(file_path, 'r') as file:  
        lines = file.readlines()  
  
    with open('output.txt', 'w') as output_file:  
        for line in lines:  
            output_file.write(line)  
  
            words = line.strip().split()  
            for word in words:  
                word_counts[word] = word_counts.get(word, 0) + 1  
  
    output_file.write("\nWord_Count:\n")  
    for word, count in word_counts.items():  
        output_file.write(f"{word}: {count}\n")  
  
word_count('/content/input.txt')
```

open("C FileNotFoun

```
1 Python Course  
2 Deep Learning Course  
3  
4 Word_Count:  
5 Python: 1  
6 Course: 2  
7 Deep: 1  
8 Learning: 1  
9
```

3RD CODE



The screenshot shows a Google Colab notebook interface. The top bar includes tabs for 'Untitled0.ipynb - Colab', 'Generative Code Assistance - L...', 'Content', and a file named '19769031'. The address bar shows a Google Drive link. Below the browser window, the Colab interface has a '+ Code' and '+ Text' button. The code editor contains a Python function `convert_to_cm` that takes a list of heights in inches and returns a list of heights in centimeters. The function uses a `for` loop to iterate over the input list, calculate the conversion (`height * 2.54`), and append the result to a new list. The script then prompts the user for input, splits the input string into a list of integers, and uses the `convert_to_cm` function to generate the converted heights. Finally, it prints the original heights and the converted heights. The output shows the input list `[155, 150, 145, 148]` and the converted list `[393.7, 381.0, 368.3, 375.92]`. The bottom of the screen shows a Windows taskbar with a search bar and various application icons.

```
def convert_to_cm(heights_inches):  
    heights_cm = []  
    for height in heights_inches:  
        cm = height * 2.54  
        heights_cm.append(round(cm, 2))  
    return heights_cm  
  
input_str = input("Enter heights in inches separated by spaces: ")  
heights_inches = list(map(int, input_str.split()))  
heights_cm = convert_to_cm(heights_inches)  
print("L1:", heights_inches)  
print("Output:", heights_cm)
```

Enter heights in inches separated by spaces: 155 150 145 148
L1: [155, 150, 145, 148]
Output: [393.7, 381.0, 368.3, 375.92]

DESCRIPTION:

The provided Python script defines a function, `convert_to_cm`, which converts a list of heights from inches to centimeters by multiplying each height by 2.54 and rounding the result to two decimal places. The script then prompts the user to input heights in inches as a space-separated string, converts this input into a list of integers, and uses the `convert_to_cm` function to generate the corresponding heights in centimeters. Finally, it prints the original heights in inches and the converted heights in centimeters. This program effectively demonstrates basic list manipulation, user input handling, and unit conversion in Python.