

Video Link:

https://drive.google.com/file/d/1FXASMcbJktLhOXYJN8ixQ0FwOYxDYqre/view?usp=drive_link

Git-hub link: [Sabitha-C/Neural-networks \(github.com\)](https://github.com/Sabitha-C/Neural-networks)

1. Code:

```
def fullname(first_name, last_name):
    full_name = first_name + " " + last_name
    return full_name

def string_alternative(full_name):
    return full_name[::2]

def main():
    first_name = input("First_name = ")
    last_name = input("Last_name = ")
    full_name = fullname(first_name, last_name)
    print("Full_Name:", full_name)
    str_alt = string_alternative(full_name)
    print("Every other character in the full_name string:", str_alt)
main()
```

Output:

```
First_name = Sabitha
Last_name = Cheekla
Full_Name: Sabitha Cheekla
Every other character in the full_name string: SbtaCeka
```

Description: I defined two functions, `fullname` and `string_alternative`. In `fullname`, I concatenated `first_name` and `last_name` to form and return `full_name`, while in `string_alternative`, I used slicing `[::2]` on `full_name` to return an alternative string. In `main`, I prompted the user for their first and last names, used `fullname` to get and print `full_name`, then called `string_alternative` to print every other character, and finally executed the program by calling `main`.

2. Code:

```

def file_word_count(line, word_count):
    for word in line.split():
        word_count[word] = word_count.get(word, 0) + 1

def main():
    input_file = '/content/drive/My Drive/input.txt'
    output_file = '/content/drive/My Drive/output.txt'

    word_count = {}
    lines = []

    try:
        with open(input_file, 'r') as input:
            for line in input:
                line = line.strip()
                if line:
                    lines.append(line)
                    file_word_count(line, word_count)

        with open(output_file, 'w') as output:
            if len(lines) > 0:
                output.write(lines[0] + '\n\n')
            if len(lines) > 1:
                output.write(lines[1] + '\n\n')

            output.write("Word_Count:\n\n")
            for word, count in word_count.items():
                output.write(word + ": " + str(count) + "\n")

        print("Output written to" + output_file)

    except FileNotFoundError:
        print("Error: File " + input_file + "not found.")

main()

```

· Output written to/content/drive/My Drive/output.txt

Output:

```
output.txt X
1 Python Course
2
3 Deep Learning Course
4
5 Word_Count:
6
7 Python: 1
8 Course: 2
9 Deep: 1
10 Learning: 1
11
```

Description: I defined two functions, `file_word_count` and `main`. In `file_word_count`, I split a line into words and updated their counts in the `word_count` dictionary. In `main`, I specified paths of input and output files, then using `try-except` block I tried to open the input file and I appended the lines using `append` method and updated word counts using `file_word_count`, and wrote the first two lines and word counts to the output file, handled the file not found errors in `except` block and called `main` function to execute the program.

3.Code1: Nested Interactive loop:

```
def inches_to_cm(height):
    return round(height * 2.54, 2)

def main():
    height1 = [150, 155, 145, 148]
    height2 = []
    for height in height1:
        height2.append(inches_to_cm(height))

    print("Heights in inches:", height1)
    print("Heights in centimeters:", height2)

main()
```

Output:

```
Heights in inches: [150, 155, 145, 148]
Heights in centimeters: [381.0, 393.7, 368.3, 375.92]
```

Description:

I defined a function `inches_to_cm` that converts a height in inches to centimeters by multiplying it by 2.54 and rounding the result. In the main function, I created a list `height1` containing heights in inches and an empty list `height2` for the converted heights. I used a for loop to iterate over each height in `height1`, converting each height to centimeters using the `inches_to_cm` function, and appended the converted values to `height2` using method. Finally, I printed both the original heights in inches and the converted heights in centimeters and called main function to execute the program.

Code2: List Comprehensions:

```
def inches_to_cm(height):  
    return round(height * 2.54, 2)  
height = [150, 155, 145, 148]  
height1 = [inches_to_cm(height) for height in height]  
  
print("L1:", height)  
print("Output:", height1)
```

Output:

```
L1: [150, 155, 145, 148]  
Output: [381.0, 393.7, 368.3, 375.92]
```

Description: I defined a function `inches_to_cm` that converts a height in inches to centimeters using the formula and rounded it to 2 decimal points. Then, I created a list `height` with heights in inches and used a list comprehension to convert each height to centimeters and stored the results in `height1`. Finally, I printed both the original `height` list and the converted `height1` list.