

ASSIGNMENT - 4.3

2303A51355

Batch-10

Task-1

Prompt: Give me a program to zero-short prompting to check a leap year, and give instructions without providing examples

code :

```
def is_leap_year(year):    if (year % 4 == 0 and year % 100  
!= 0) or (year % 400 == 0):  
    return True    return False    year = int(input("Enter a  
year to check if it's a leap year: ")) if is_leap_year(year):  
    print(f"{year} is a leap year.")  
else:  
    print(f"{year} is not a leap year.") Output
```

:

```
File Edit Selection View Go Run Terminal Help < → Q AI_ASSISTANT_CODING  
EXPLORER lab-2.py lab-3.3.py lab-3.4.py lab-4.3.py  
AI ASSISTANT CODING  
lab-3.3.py  
lab-3.4.py  
lab-4.3.py  
lab1.py  
lab2.py  
#task-1  
#Give me a program to zero-short prompting to check leap year and giving instructions without providing examples  
def is_leap_year(year):  
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):  
        return True  
    return False  
year = int(input("Enter a year to check if it's a leap year: "))  
if is_leap_year(year):  
    print(f"{year} is a leap year.")  
else:  
    print(f"{year} is not a leap year.")  
PS C:\Users\NIRWANA\OneDrive\Desktop\AI_ASSISTANT_CODING> & C:\Users\NIRWANA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRWANA/OneDrive/Desktop/AI_ASSISTANT_CODING/lab-4.3.py  
Enter a year to check if it's a leap year: 2024  
2024 is a leap year.
```

Code Analysis:

- This program determines whether a given year is a leap year using a function.
- The function applies standard leap year rules and returns True or False.
- The user inputs a year, which is checked by the function.
- The result is printed as either a leap year or not.

Task-2

Prompt: generate a one-short prompt that changes cen metres to inches with one input and output using the correct mathematical formula

Code :

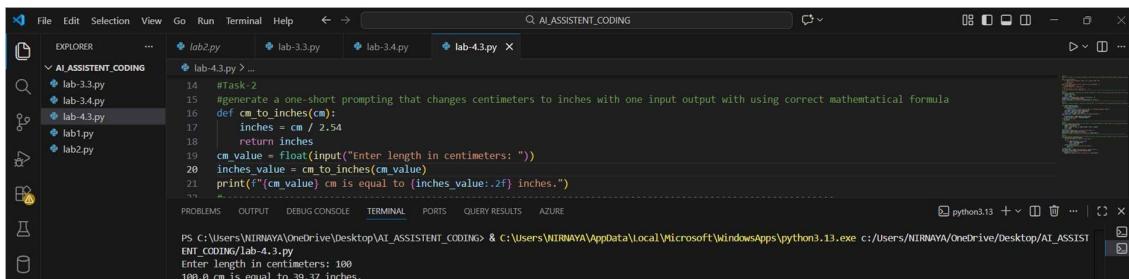
```
def cm_to_inches(cm):
    inches = cm / 2.54
    return inches

cm_value = float(input("Enter length in centimeters: "))

inches_value = cm_to_inches(cm_value)

print(f"{cm_value} cm is equal to {inches_value:.2f} inches.")
```

Output :



The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows files: lab2.py, lab-3.3.py, lab-3.4.py, lab-4.3.py, lab1.py, and lab2.py.
- Code Editor:** Displays the Python script for task-2. The code defines a function `cm_to_inches` that takes a centimeter value, divides it by 2.54 to get inches, and returns the result. It then prompts the user for a centimeter value, calls the function, and prints the result with two decimal places.
- Terminal:** Shows the command `python3.13 & c:\Users\WIRNAYA\Desktop\AI_ASSISTANT_CODING\lab-4.3.py` and the output: `Enter length in centimeters: 100`, `100.0 cm is equal to 39.37 inches.`

Code Analysis:

- This program converts a length from centimetres to inches using the correct mathematical formula.
- A function performs the conversion by dividing the value by 2.54.
- The user enters a value in centimetres, which is passed to the function.
- The converted result is displayed in inches.

Task-3

Prompt: generate a python program is few-short prompt that is name formating like accepting fullname as firstname,lastname.

Code :

```
def format_name(full_name):
    parts = full_name.split(',')
    if len(parts) != 2:
        raise ValueError("Please enter the name in 'Firstname,Lastname' format.")
    first_name = parts[0].strip().capitalize()
```

```

last_name = parts[1].strip().capitalize()    return f'{first_name} {last_name}'  full_name_input = input("Enter full name (Firstname,Lastname): ")

try:
    forma ed_name = format_name(full_name_input)
    print(f'Forma ed Name: {forma ed_name}')
except ValueError as e:
    print(e)

```

Output:

The screenshot shows a code editor interface with several files listed in the Explorer sidebar: lab2.py, lab-3.py, lab-3.4.py, lab-4.3.py, lab1.py, and lab2.py. The current file, lab-4.3.py, is open in the main editor area. The code is a Python script that prompts the user for a full name in 'Firstname,Lastname' format, splits it into first and last names, capitalizes them, and prints the result. A terminal window at the bottom shows the script being run and a sample input-output session where the user enters 'mula,nirmaya' and the output is 'Mula Nirmaya'.

```

#task-3
#generate a python program is few-short prompting that is name formating like accepting fullname as firstname,lastname.
def format_name(full_name):
    parts = full_name.split(',')
    if len(parts) != 2:
        raise ValueError("Please enter the name in 'Firstname,Lastname' format.")
    first_name = parts[0].strip().capitalize()
    last_name = parts[1].strip().capitalize()
    return f'{first_name} {last_name}'
full_name_input = input("Enter full name (Firstname,Lastname): ")
try:
    formatted_name = format_name(full_name_input)
    print(f"Formatted Name: {formatted_name}")
except ValueError as e:
    print(e)

```

Code Analysis :

- This program formats a full name entered as first name and last name.
- The input is split and validated to ensure the correct format.
- Each part of the name is cleaned and capitalised. The formatted full name is then displayed.

Task-4

Prompt: generate a comparative analysis for zero-short vs few-short prompting to count vowels in a string using a function:

Code :

```

def count_vowels(input_string):
    vowels = 'aeiouAEIOU'    count = sum(1 for char in
input_string if char in vowels)    return count
# Example usage

```

```

test_string = input("Enter a string to count vowels: ")
vowel_count = count_vowels(test_string) print(f"The
number of vowels in the string is: {vowel_count}") Output:

```

The screenshot shows a code editor interface with the following details:

- Explorer Pane:** Shows files in the 'AI ASSISTANT CODING' folder, including 'lab2.py', 'lab-3.3.py', 'lab-3.4.py', 'lab-4.3.py', 'lab1.py', and 'lab2.py'.
- Terminal Pane:** Displays the code for 'lab-4.3.py' and its execution output. The code defines a function 'count_vowels' that counts vowels in a string. An example usage is shown where the user inputs 'Nirnaya' and the program outputs 'The number of vowels in the string is: 3'.
- Status Bar:** Shows the path 'C:\Users\WIRNAYA\OneDrive\Desktop\AI_ASSISTANT_CODING & C:\Users\WIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/WIRNAYA/OneDrive/Desktop/AI_ASSISTANT_CODING/lab-4.3.py' and the terminal tab is active.

Code Analysis :

- The function counts vowels in a given string using a direct logic approach.
- Zero-shot prompting applies the logic without examples.
- Few-shot prompting helps by showing patterns before execution.
- The function returns the total number of vowels in the input string.

Task-5

Prompt: [generate a few short prompts for file handling to give a read text file, count the number of lines in the file, and line count by function](#)

```

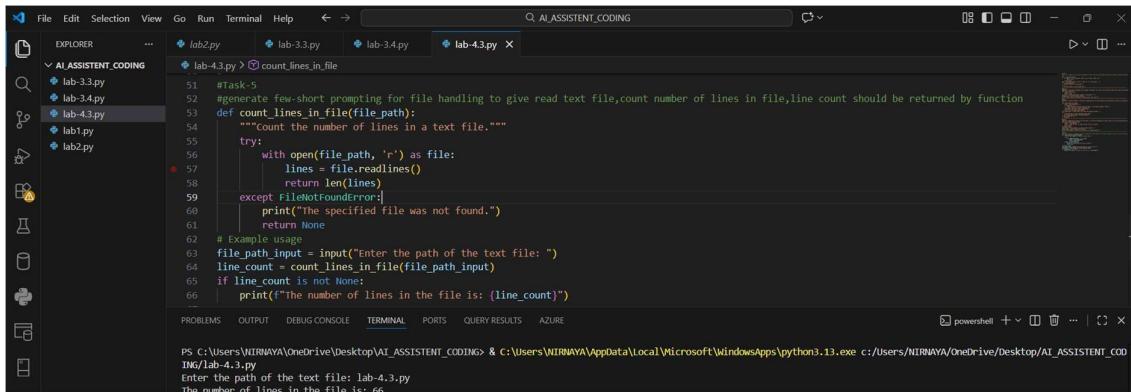
def count_lines_in_file(file_path):
    try:
        with open(file_path, 'r') as file:
            lines = file.readlines()
        return len(lines)
    except FileNotFoundError:
        print("The specified file was not found.")
    return None

# Example usage
file_path_input = input("Enter the path of the text file:
") line_count = count_lines_in_file(file_path_input)
if line_count is not None:

```

```
print(f"The number of lines in the file is: {line_count}")
```

Output :



The screenshot shows a code editor window titled "Q_AI_ASSISTANT_CODING". The left sidebar has a tree view under "EXPLORER" with files: lab-2.py, lab-3.py, lab-3.4.py, lab-4.py, lab-4.3.py, lab1.py, and lab2.py. The main editor area contains the following Python code:

```
51 #task-5
52 #generate few short prompting for file handling to give read text file,count number of lines in file,line count should be returned by function
53 def count_lines_in_file(file_path):
54     """Count the number of lines in a text file."""
55     try:
56         with open(file_path, 'r') as file:
57             lines = file.readlines()
58             return len(lines)
59     except FileNotFoundError:
60         print("The specified file was not found.")
61     return None
62 # Example usage
63 file_path_input = input("Enter the path of the text file: ")
64 line_count = count_lines_in_file(file_path_input)
65 if line_count is not None:
66     print(f"The number of lines in the file is: {line_count}")
--
```

The terminal tab at the bottom shows the command "PS C:\Users\NIRNAYA\Desktop\AI_ASSISTANT_CODING> & c:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRNAYA/OneDrive/Desktop/AI_ASSISTANT_CODING\lab-4.3.py" and the output "Enter the path of the text file: lab-4.3.py" followed by "The number of lines in the file is: 66".

Code Analysis :

- This program reads a text file and counts the number of lines.
- A function opens the file safely and calculates the line count.
- Error handling is used if the file does not exist.
- The final line count is returned and displayed.