

ASSIGNMENT - 4.3

2303A51525

Batch-10

Task-1

Prompt: Give me a program to zero-short prompt 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

•

The screenshot shows a Visual Studio Code (VS Code) interface. The top menu bar includes File, Edit, Selection, View, Go, Explorer, Open Editors, Outline, AI, and Help. The left sidebar has sections for Explorer, Open Editors (with one file unsaved), Outline, and AI. Under AI, several files are listed: lab-3.3.py, lab-3.4.py, lab-4.3.py (selected), lab-3.5.py, lab-5.4.py, lab-6.3.py, and lab-6.4.py. The main editor area contains Python code for checking if a year is a leap year. The terminal at the bottom shows the command to run the script and its output for the year 2024.

```
1 #Task-1
2 #Give me a program to zero-short prompting to check leap year and giving instructions without providing examples
3 def is_leap_year(year):
4     if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
5         return True
6     return False
7 year = int(input("Enter a year to check if it's a leap year: "))
8 if is_leap_year(year):
9     print(f"{year} is a leap year.")
10 else:
11     print(f"{year} is not a leap year.")
12 #
13 #Task-2
14 #generate a one-short prompting that changes centimeters to inches with one input output with using correct math
```

PROBLEMS 11 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SPELL CHECKER 13

PS C:\Users\hp\OneDrive\Desktop\AI> & C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/hp/OneDrive/Desktop\AI>.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.

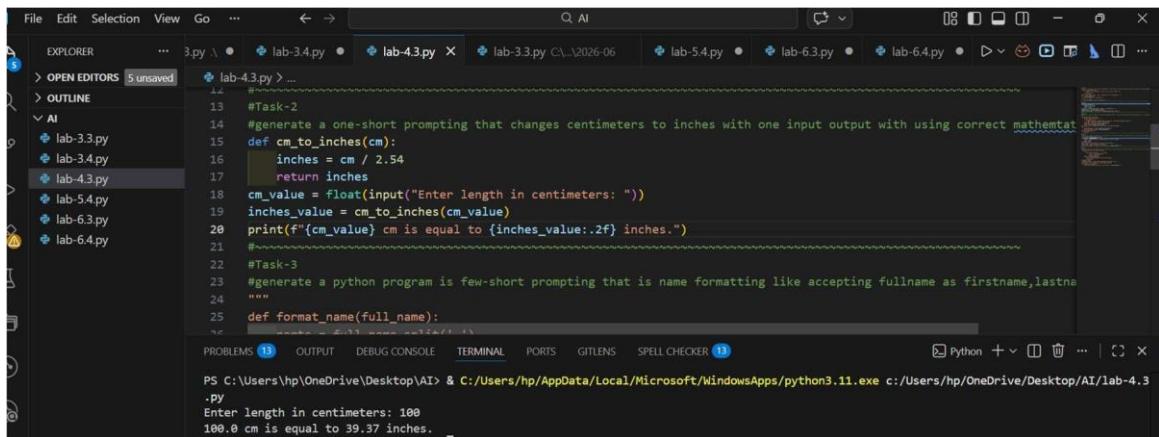
Task-2

Prompt: generate a one-short prompt that changes centimetres 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.")
```

{inches_value:.2f} inches.") Output :



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:
    formatted_name = format_name(full_name_input)
    print(f"Formatted Name: {formatted_name}")
except ValueError as e:
    print(e)

```

The screenshot shows a code editor interface with several tabs at the top. The active tab is 'lab-4.3.py'. The left sidebar shows an 'EXPLORER' view with files like 'lab-3.3.py', 'lab-3.4.py', 'lab-4.3.py', 'lab-5.4.py', 'lab-6.3.py', and 'lab-6.4.py'. The main code area contains the provided Python script. Below the code, the terminal window shows the command 'PS C:\Users\hp\OneDrive\Desktop\AI> & C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/hp/OneDrive/Desktop/AI/lab-4.py' followed by the user input 'Enter full name (Firstname,Lastname): puvati,vamshi' and the program's output 'Formatted Name: Puvati Vamshi'.

Code Analysis :

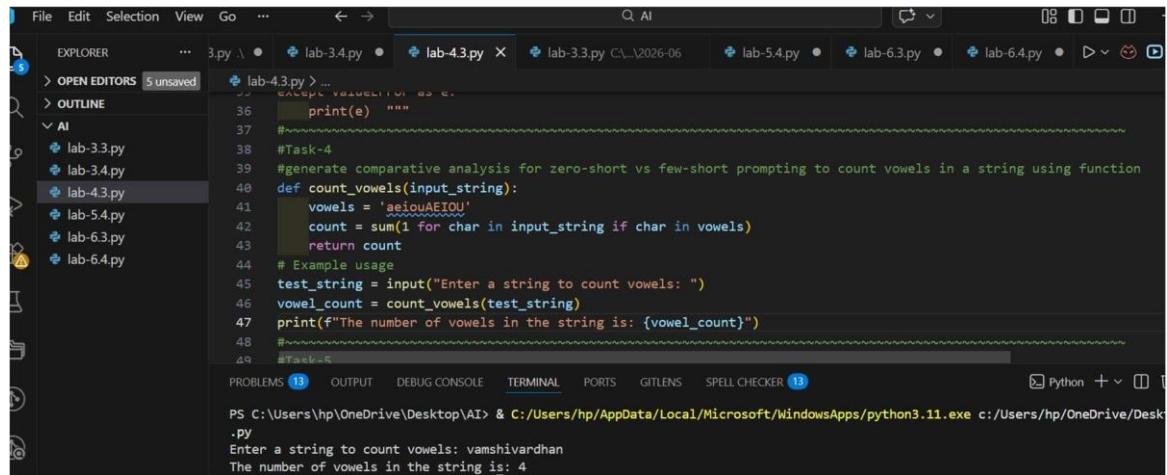
- This program formats a full name entered as first name and last name.
- The input is split and validated to ensure the correct format.
- Each part of the name is cleaned and capitalised.
- The formatted full name is then displayed.

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}")
```



The screenshot shows the Visual Studio Code interface. The left sidebar has an 'EXPLORER' view with several Python files listed under 'OPEN EDITORS'. The 'lab-4.3.py' file is currently open in the center editor area. The code in the editor is identical to the one provided above. Below the editor is a terminal window showing the command PS C:\Users\hp\OneDrive\Desktop\AI> & C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/hp/OneDrive/Desktop\AI>PY followed by the user's input 'Enter a string to count vowels: vamshividhan' and the output 'The number of vowels in the string is: 4'.

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
FileNotFoundException:
    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
:

```

```

File Edit Selection View Go ... ⌘ AI 08
EXPLORER 3.py · lab-3.4.py · lab-4.3.py X lab-3.3.py C:\2026-06 lab-5.4.py · lab-6.3.py · lab-6.4.py · ⌛
OPEN EDITORS 5 unsaved
> OUTLINE
> AI
lab-3.3.py
lab-3.4.py
lab-4.3.py
lab-5.4.py
lab-6.3.py
lab-6.4.py
File Path: C:\Users\hp\OneDrive\Desktop\AI> & C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/hp/OneDrive/Desktop/AI/lab-4.3.py
Enter the path of the text file: lab-4.3.py
The number of lines in the file is: 63

```

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.