

Assignment-4

Task-1:

Function that checks whether a given year is a leap year

```
Ass 4.1.py
Ass 4.1.py > ...
1 # write a simple code to find whether a given year is a leap year using function
2 # ...existing code...
3 a=int(input("Enter a year: "))
4 def is_leap_year(year):
5     """Checks if a year is a leap year."""
6     return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)
7 print(is_leap_year(a))
8
```

Python Debug Console

```
PS C:\Aiassisted coding> & 'c:\Users\Poojasree\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Poojasree\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '53981' '--' 'C:\Aiassisted coding\Ass 4.1.py'
Enter a year: 2020
True
PS C:\Aiassisted coding>
```

CHAT

```
def is_leap_year(year):
    """Checks if a year is a leap year."""
    return (year % 4 == 0 and year % 100 != 0) or
           (year % 400 == 0)
print(is_leap_year(a)) explain this code
```

Ass 4.1.py:3-7

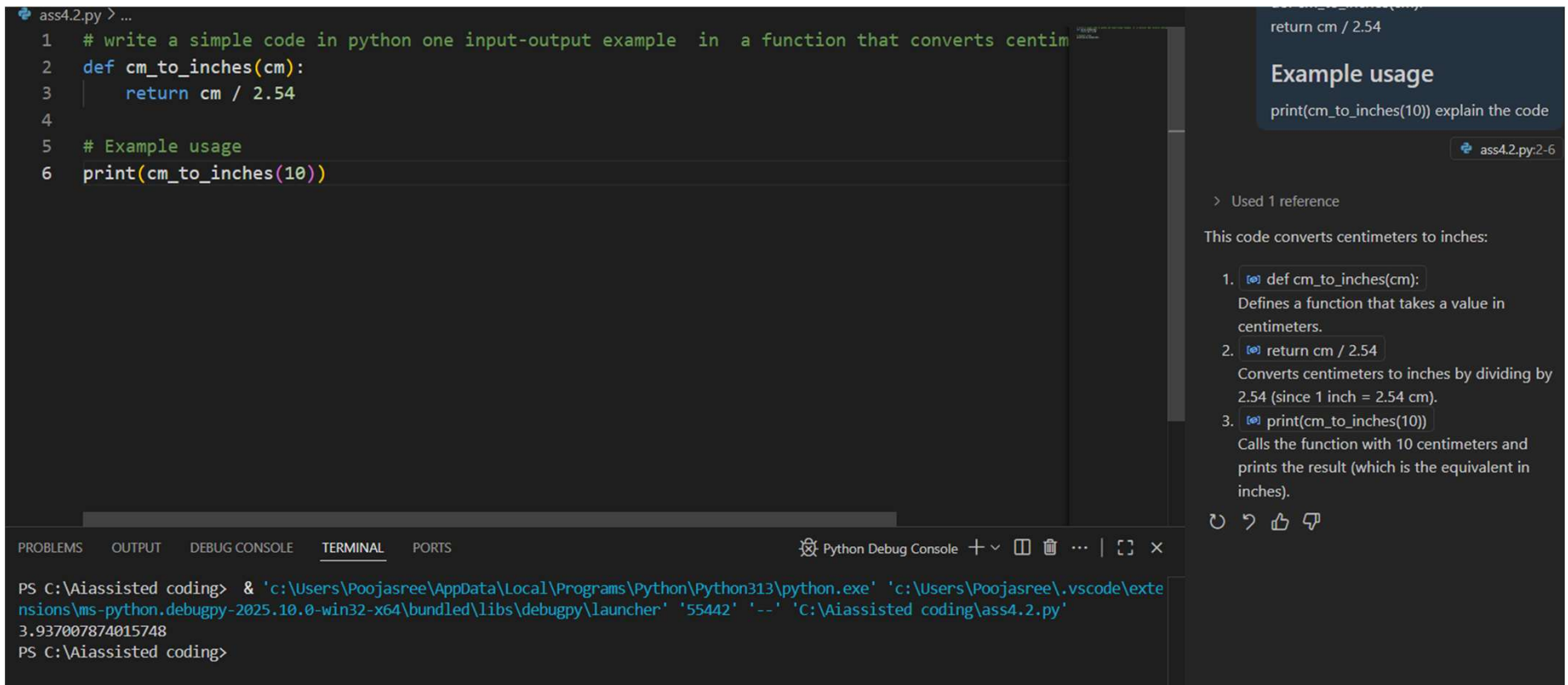
> Used 1 reference

This code checks if a given year is a leap year:

1. `a=int(input("Enter a year: "))`
Prompts the user to enter a year and converts the input to an integer.
2. `def is_leap_year(year):`
Defines a function to check leap year logic.
3. `return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)`
Returns `True` if the year is divisible by 4 but not by 100, or if it is divisible by 400 (leap year rules).
4. `print(is_leap_year(a))`
Calls the function with the user's input and prints `True` if it's a leap year, otherwise `False`.

Task-2:

write a simple code in python one input-output example in a function that converts centimeters to inches.



The screenshot shows a VS Code editor with a Python file named `ass4.2.py`. The code defines a function `cm_to_inches` that takes a centimeter value and returns its equivalent in inches. An example usage is provided, printing the result for 10 centimeters.

```
1 # write a simple code in python one input-output example in a function that converts centim
2 def cm_to_inches(cm):
3     return cm / 2.54
4
5 # Example usage
6 print(cm_to_inches(10))
```

Below the editor, the terminal shows the command used to run the script, which includes the Python interpreter path and the file path.

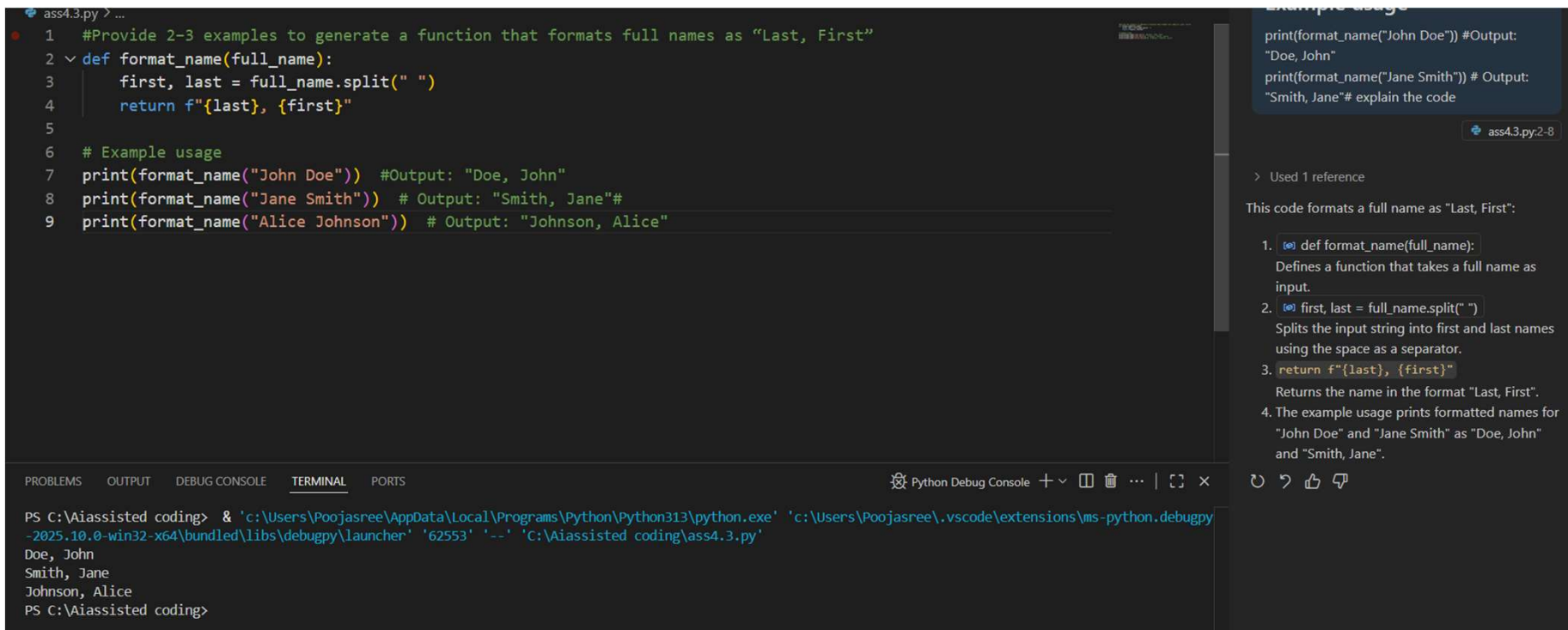
```
PS C:\Aiassisted coding> & 'c:\Users\Poojasree\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Poojasree\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '55442' '--' 'C:\Aiassisted coding\ass4.2.py'
3.937007874015748
PS C:\Aiassisted coding>
```

On the right side, a panel titled "Example usage" provides a detailed explanation of the code. It states that the code converts centimeters to inches and lists the three lines of code with their respective functions:

- `def cm_to_inches(cm):`: Defines a function that takes a value in centimeters.
- `return cm / 2.54`: Converts centimeters to inches by dividing by 2.54 (since 1 inch = 2.54 cm).
- `print(cm_to_inches(10))`: Calls the function with 10 centimeters and prints the result (which is the equivalent in inches).

Task-3:

#Provide 2–3 examples to generate a function that formats full names as “Last, First”



The screenshot shows a VS Code editor with a Python file named `ass4.3.py`. The code defines a function `format_name` that takes a full name as input and returns it in the format "Last, First". The function splits the input string by a space and then formats the last and first names. Below the function definition, there are three example usages: `print(format_name("John Doe"))`, `print(format_name("Jane Smith"))`, and `print(format_name("Alice Johnson"))`. The output of these calls is shown in the terminal: "Doe, John", "Smith, Jane", and "Johnson, Alice".

```
1 #Provide 2-3 examples to generate a function that formats full names as "Last, First"
2 def format_name(full_name):
3     first, last = full_name.split(" ")
4     return f"{last}, {first}"
5
6 # Example usage
7 print(format_name("John Doe")) #Output: "Doe, John"
8 print(format_name("Jane Smith")) # Output: "Smith, Jane"#
9 print(format_name("Alice Johnson")) # Output: "Johnson, Alice"
```

Example usage

```
print(format_name("John Doe")) #Output:
"Doe, John"
print(format_name("Jane Smith")) # Output:
"Smith, Jane"# explain the code
```

ass4.3.py:2-8

Used 1 reference

This code formats a full name as "Last, First":

1. `def format_name(full_name):`
Defines a function that takes a full name as input.
2. `first, last = full_name.split(" ")`
Splits the input string into first and last names using the space as a separator.
3. `return f"{last}, {first}"`
Returns the name in the format "Last, First".
4. The example usage prints formatted names for "John Doe" and "Jane Smith" as "Doe, John" and "Smith, Jane".

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python Debug Console

```
PS C:\Aiassisted coding> & 'c:\Users\Poojasree\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Poojasree\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '62553' '--' 'C:\Aiassisted coding\ass4.3.py'
Doe, John
Smith, Jane
Johnson, Alice
PS C:\Aiassisted coding>
```

Task:4

counts the number of vowels in a string.

1.Zero shot

```
a=input("enter a string")
def count_vowels(input_string):

    vowels = "aeiouAEIOU"
    vowel_count = 0
    for char in input_string:
        if char in vowels:
            vowel_count += 1
    return vowel_count
print(count_vowels(a))
```

here's a breakdown:

1. `a=input("enter a string")`: This line prompts the user to enter a string using the message "enter a string" and stores the entered string in the variable `a`.
2. `def count_vowels(input_string)`: This defines a function named `count_vowels` that takes one argument, `input_string`.
3. `vowels = "aeiouAEIOU"`: This line creates a string named `vowels` containing all the lowercase and uppercase vowels. This will be used to check if a character is a vowel.
4. `vowel_count = 0`: This initializes a variable `vowel_count` to 0. This variable will keep track of the number of vowels found in the input string.
5. `for char in input_string`: This starts a loop that iterates through each character in the `input_string`. In each iteration, the current character is assigned to the variable `char`.
6. `if char in vowels`: Inside the loop, this line checks if the current character (`char`) is present in the `vowels` string.
7. `vowel_count += 1`: If the character is a vowel (the condition in the `if` statement is true), this line increments the `vowel_count` by 1.
8. `return vowel_count`: After the loop finishes iterating through all the characters in the input string, this line returns the final value of `vowel_count`.
9. `print(count_vowels(a))`: This line calls the `count_vowels` function with the user's input string stored in variable `a` and prints the returned vowel count to the console.

Output:

```
enter a string poojasree
5
PS C:\Aiassisted coding> 
```

One shot:

```
0s  def count_vowels(input_string):  
    vowels = "aeiouAEIOU"  
    return sum(1 for char in input_string if char in vowels)  
  
# Example usage  
print(count_vowels("Hello World")) # Output: 3
```

3

Gemini

Here's a breakdown:

1. `def count_vowels(input_string):` : This defines a function named `count_vowels` that takes one argument, `input_string`.
2. `vowels = "aeiouAEIOU"` : This line creates a string named `vowels` containing all the lowercase and uppercase vowels. This will be used to check if a character is a vowel.
3. `return sum(1 for char in input_string if char in vowels)` : This is a concise way to count the vowels using a generator expression and the `sum()` function.
 - `for char in input_string` : This part iterates through each character in the `input_string`.
 - `if char in vowels` : This filters the characters, only considering those that are present in the `vowels` string (i.e., the vowels).
 - `1 for char in input_string if char in vowels` : For each character that passes the `if` condition (is a vowel), it generates the number `1`.
 - `sum(...)` : This sums up all the `1`s generated by the expression, giving the total count of vowels.
4. `print(count_vowels("Hello World"))` : This line calls the `count_vowels` function with the string "Hello World" and prints the returned vowel count (which is 3) to the console.

Would you like to try modifying this code, for example, to count consonants instead of vowels?

Zero-Shot Analysis:

In a zero-shot context, which means evaluating the code without any prior examples or context about the user's intent, the first function is superior. It uses a generator expression (`1 for char in input_string if char in vowels`) combined with the built-in `sum()` function. This approach is highly efficient because it doesn't create a temporary list in memory, as it generates the values on the fly.

A one-shot analysis:

where we consider the code's functionality based on a single example, reveals that both snippets are functionally correct. Both functions, `count_vowels(input_string)`, take a string as input and correctly return the number of vowels.

Task-5:

#Use few-shot prompting to generate a function that reads a .txt file and returns the number of lines

```

1 #Use few-shot prompting to generate a function that reads a .txt file and returns the number of lines
2 # The function from the prompt
3 def count_lines_in_file(file_path):
4     with open(file_path, 'r') as file:
5         return sum(1 for line in file)
6
7 # Example Usage:
8 # 1. First, create a sample file for demonstration
9 with open("sample.txt", "w") as f:
10     f.write("Line 1\n")
11     f.write("Line 2\n")
12     f.write("Line 3\n")
13     f.write("Line 4\n")
14
15 # 2. Call the function with the file path
16 file_path = "sample.txt"
17 line_count = count_lines_in_file(file_path)
18
19 # 3. Print the result
20 print(f"The number of lines in '{file_path}' is: {line_count}")

```

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python Debug Console +v [ ] [ ] [ ] [ ]

PS C:\Aiassisted coding> & 'c:\Users\Poojasree\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Poojasree\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher
'64362' '--' 'C:\Aiassisted coding\ass4.5.py'
The number of lines in 'sample.txt' is: 4
PS C:\Aiassisted coding>

```


Explanation:

Here's a breakdown:

1. `def count_lines_in_file(file_path):` : This defines a function named `count_lines_in_file` that takes one argument, `file_path`.
2. `with open(file_path, 'r') as file:` : This opens the file specified by `file_path` in read mode (`'r'`). The `with` statement ensures the file is automatically closed even if errors occur. The opened file object is assigned to the variable `file`.
3. `return sum(1 for line in file)` : This part iterates through each `line` in the `file` object. For each line, it generates the number `1`. The `sum()` function then adds up all these `1`s, effectively counting the number of lines.
4. `with open("sample.txt", "w") as f:` : This creates a sample file named `sample.txt` in write mode (`"w"`). If the file already exists, it will be overwritten. The file object is assigned to the variable `f`.
5. `f.write("Line 1\n")` (and similar lines): These lines write the specified strings to the `sample.txt` file. The `\n` at the end of each string creates a new line.
6. `file_path = "sample.txt"` : This sets the `file_path` variable to the name of the sample file.
7. `line_count = count_lines_in_file(file_path)` : This calls the `count_lines_in_file` function with the `file_path` and stores the returned line count in the `line_count` variable.
8. `print(f"The number of lines in '{file_path}' is: {line_count}")` : This prints a formatted string to the console showing the name of the file and the calculated number of lines.



Sample.txt file

```
I am poojasree
i am currently persuing cse aiml from sru
i am in 2nd year now
i am from hanamkonda
```