

AI ASSISTED CODING

Roll no.: 2503A51L41

Batch: 24BTCAICSB20

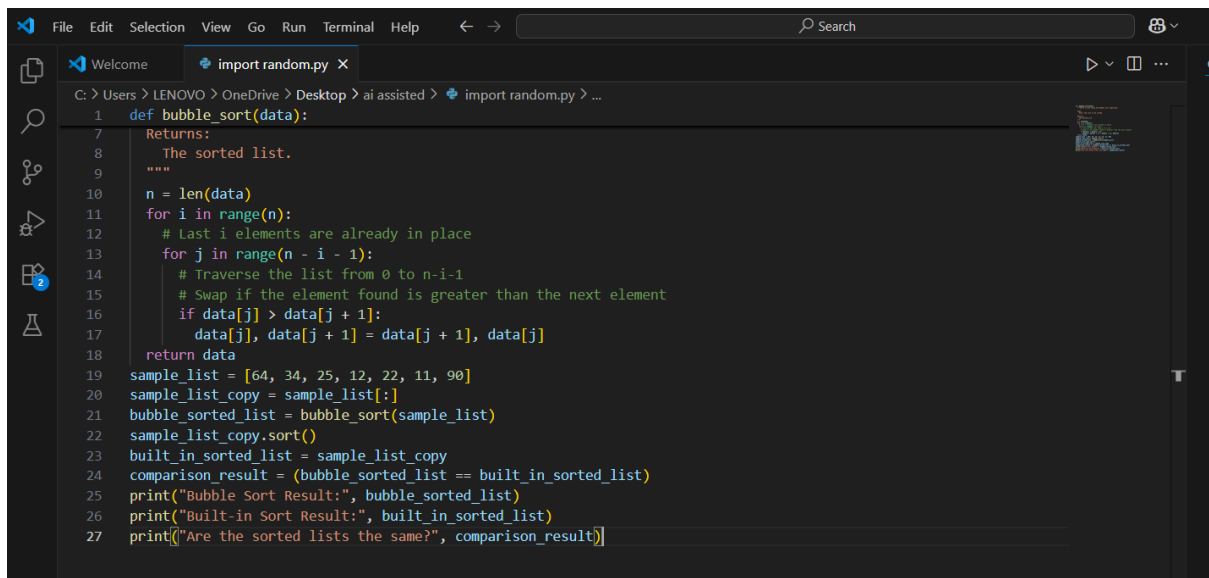
Name: G Hasini

Lab 2: Exploring Additional AI Coding Tools – Gemini (Colab) and Cursor AI

Task#1

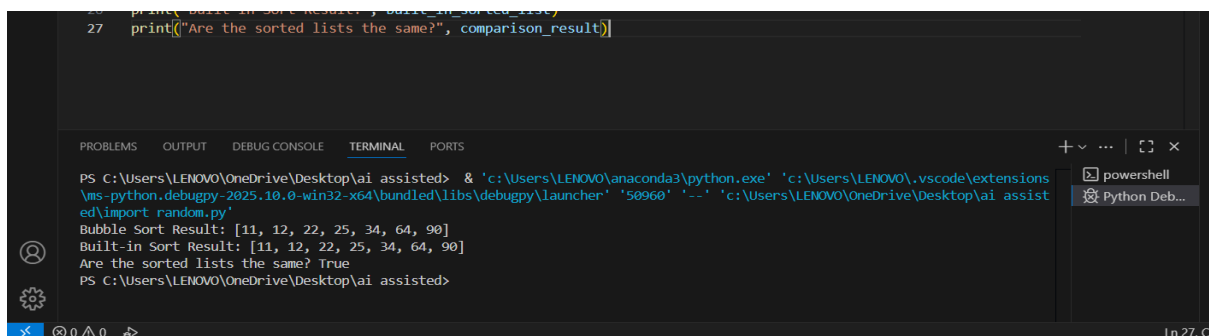
Prompt: Open Google Colab and use Google Gemini to generate Python code that performs sorting of a list using both the bubble sort algorithm and Python's built-in sort() function. Compare the two implementations

Code generated:



```
1 def bubble_sort(data):
7     Returns:
8         The sorted list.
9     """
10    n = len(data)
11    for i in range(n):
12        # Last i elements are already in place
13        for j in range(n - i - 1):
14            # Traverse the list from 0 to n-i-1
15            # Swap if the element found is greater than the next element
16            if data[j] > data[j + 1]:
17                data[j], data[j + 1] = data[j + 1], data[j]
18    return data
19 sample_list = [64, 34, 25, 12, 22, 11, 90]
20 sample_list_copy = sample_list[:]
21 bubble_sorted_list = bubble_sort(sample_list)
22 sample_list_copy.sort()
23 built_in_sorted_list = sample_list_copy
24 comparison_result = (bubble_sorted_list == built_in_sorted_list)
25 print("Bubble Sort Result:", bubble_sorted_list)
26 print("Built-in Sort Result:", built_in_sorted_list)
27 print("Are the sorted lists the same?", comparison_result)
```

Output:



```
26 print("Built-in Sort Result:", built_in_sorted_list)
27 print("Are the sorted lists the same?", comparison_result)

PS C:\Users\LENOVO\OneDrive\Desktop\ai assisted> & 'c:\Users\LENOVO\anaconda3\python.exe' 'c:\Users\LENOVO\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher' '50960' '--' 'c:\Users\LENOVO\OneDrive\Desktop\ai assist
ed\import_random.py'
Bubble Sort Result: [11, 12, 22, 25, 34, 64, 90]
Built-in Sort Result: [11, 12, 22, 25, 34, 64, 90]
Are the sorted lists the same? True
PS C:\Users\LENOVO\OneDrive\Desktop\ai assisted>
```

Observation:

1. Correct Implementation of Bubble Sort:

The function `bubble_sort(arr)` correctly implements the bubble sort algorithm using nested loops and value swapping.

Data Integrity Preserved:

The function uses `arr.copy()` to avoid modifying the original list, which is a good practice.

Clear Comparison with Built-in Sort:

The code includes a custom sort function and compares its result with Python's built-in `sorted()` function — great for validating correctness.

Readable and Well-Structured:

- The code is neatly organized into: ○

- Custom sort

- Built-in sort

- Example list ○ Comparison

Task#2

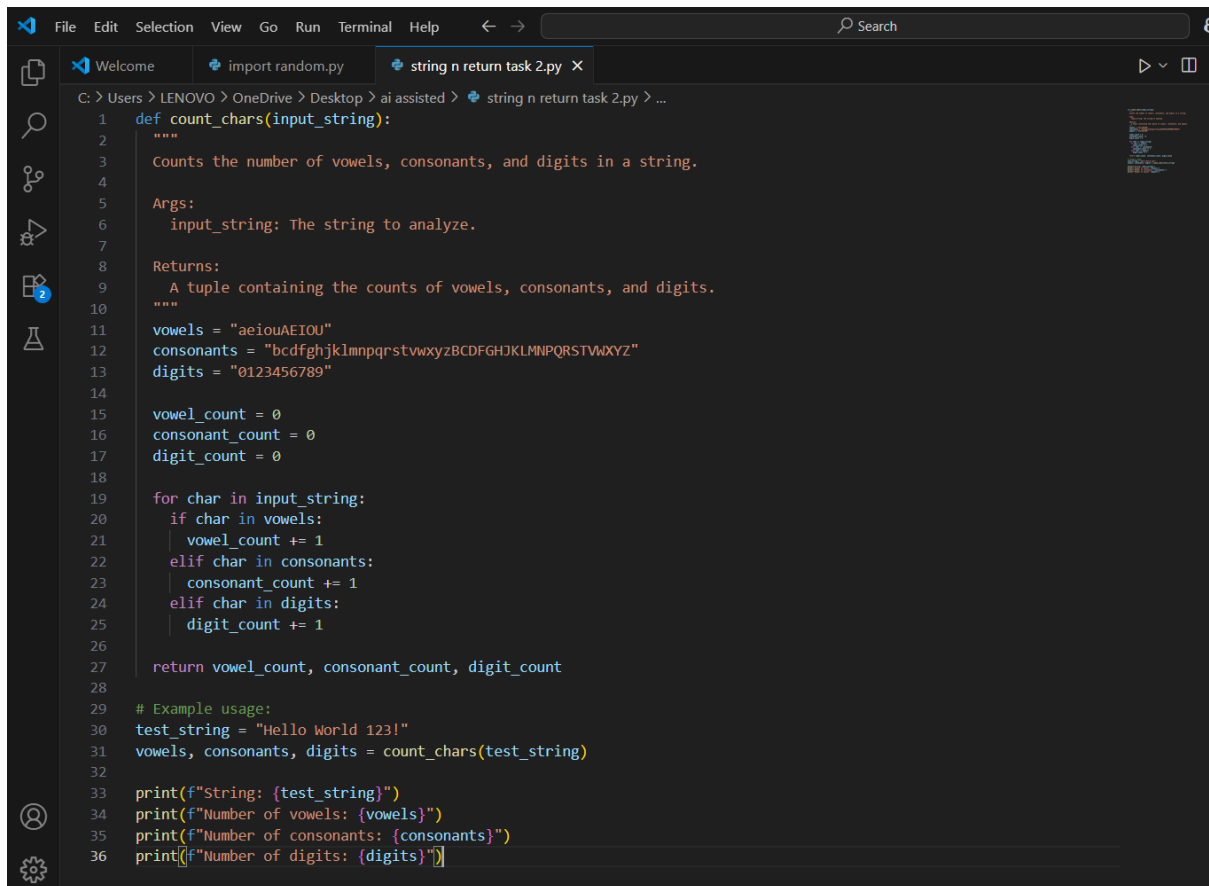
Prompt: In Colab, use Google Gemini to generate a Python function that takes a string and returns:

The number of vowels,

The number of consonants,

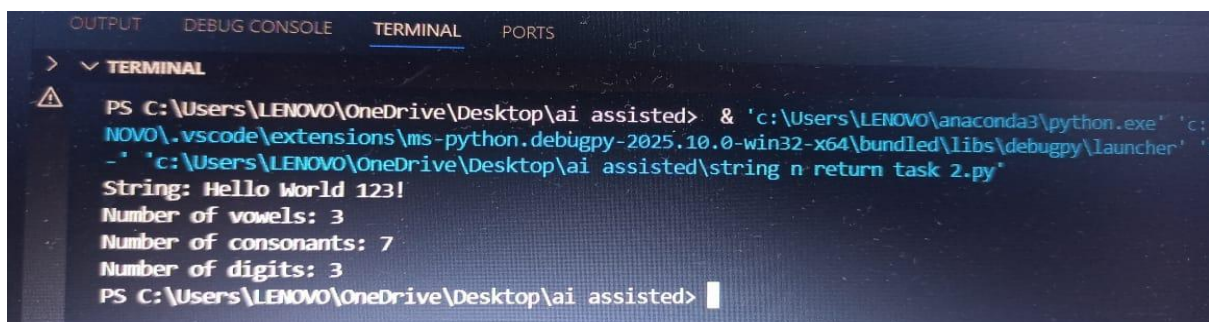
The number of digits in the string.

Code generated:



```
1 def count_chars(input_string):
2     """
3     Counts the number of vowels, consonants, and digits in a string.
4
5     Args:
6         input_string: The string to analyze.
7
8     Returns:
9         A tuple containing the counts of vowels, consonants, and digits.
10    """
11    vowels = "aeiouAEIOU"
12    consonants = "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ"
13    digits = "0123456789"
14
15    vowel_count = 0
16    consonant_count = 0
17    digit_count = 0
18
19    for char in input_string:
20        if char in vowels:
21            vowel_count += 1
22        elif char in consonants:
23            consonant_count += 1
24        elif char in digits:
25            digit_count += 1
26
27    return vowel_count, consonant_count, digit_count
28
29 # Example usage:
30 test_string = "Hello World 123!"
31 vowels, consonants, digits = count_chars(test_string)
32
33 print(f"String: {test_string}")
34 print(f"Number of vowels: {vowels}")
35 print(f"Number of consonants: {consonants}")
36 print(f"Number of digits: {digits}")
```

Output:



```
PS C:\Users\LENOVO\OneDrive\Desktop\ai assisted> & 'c:\Users\LENOVO\anaconda3\python.exe' 'c:\Users\LENOVO\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '- ' 'c:\Users\LENOVO\OneDrive\Desktop\ai assisted\string n return task 2.py'
String: Hello World 123!
Number of vowels: 3
Number of consonants: 7
Number of digits: 3
PS C:\Users\LENOVO\OneDrive\Desktop\ai assisted>
```

Observation:

Function Definition

def count_lines(filename):

- A function count_lines is defined that takes a filename (e.g., "example.txt") as input.

Try Block – Reading the File

- try: with open(filename, 'r') as file: return sum(1 for _ in file)

- - `open(filename, 'r')`: Tries to open the file in read mode.
 - `sum(1 for _ in file)`: Counts each line using a generator expression.
 - It iterates through each line and adds 1 per line.
 - If the file is found, it returns the line count.

1. Exception Handling – File Not Found

1. `except FileNotFoundError`:
2. If the file doesn't exist, this block is executed.

2. Created a Sample File

● `with open(filename, 'w') as file: file.write("Hello\n")`

`file.write("This is a test file\n")`

The file is opened in write mode ('w') which creates a new file.

Task#3

Prompt: Install and set up Cursor AI. Use it to generate a Python program that performs file handling:

Create a text file,

Write sample text,

Read and display the content.

Code generated:

```
Welcome 1 X
1 > ...
1 # Define the filename
2 filename = "sample_file.txt"
3
4 # Write to the file
5 with open(filename, "w") as file:
6     file.write("This is a sample text file.\n")
7     file.write("This is the second line.")
8
9 print(f"Successfully wrote to '{filename}'")
10
11 # Read from the file
12 with open(filename, "r") as file:
13     content = file.read()
14
15 print(f"\nContent of '{filename}':")
16 print(content)
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [ ] [ ] ... [ ] [ ] X
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (conda:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\Gundeti Hasini\OneDrive\Desktop\ai2> & C:/ProgramData/anaconda3/python.exe "c:/Users/Gundeti Hasini/OneDrive/Desktop/ai2/1"
Successfully wrote to 'sample_file.txt'

Content of 'sample_file.txt':
This is a sample text file.
This is the second line.
PS C:\Users\Gundeti Hasini\OneDrive\Desktop\ai2> |
```

Observation:

1.filename = "sample_file.txt": Sets the name of the file.

2. with open(filename, 'w') as file::

- Opens (or creates) the file in **write mode** ('w').
- If the file already exists, it will be **overwritten**.

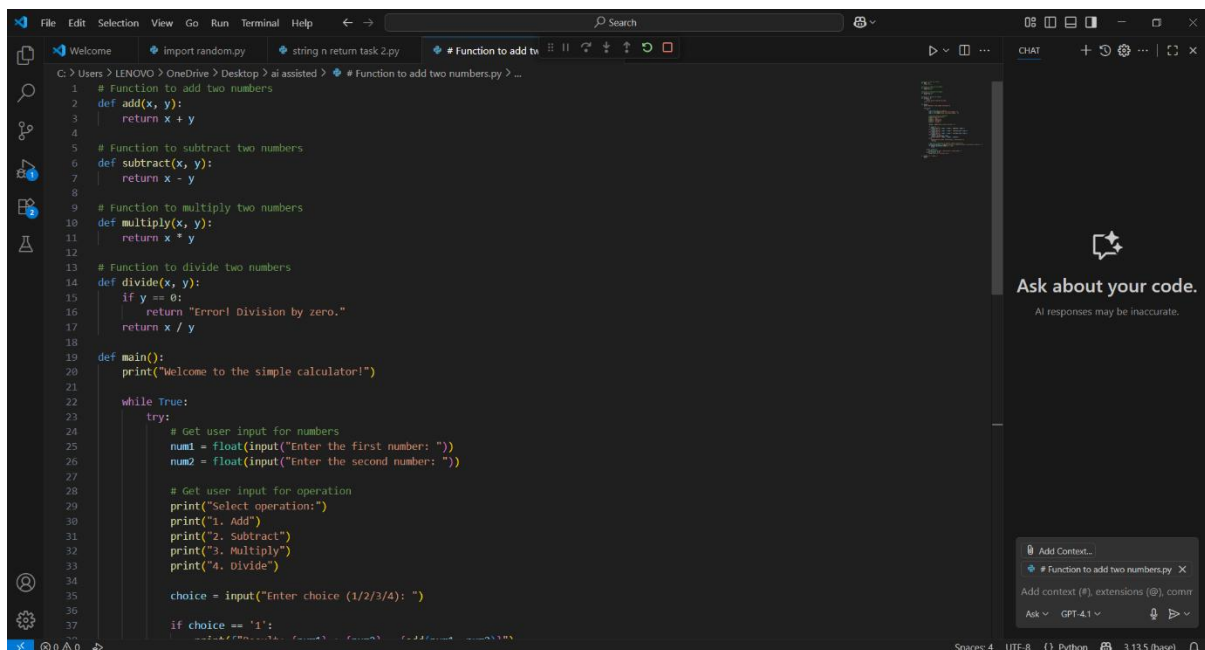
3. file.write(...): Writes 3 lines of text into the file, each ending with a newline (\n).

print(...): Confirms that the file was created and written successfully.

Task#4

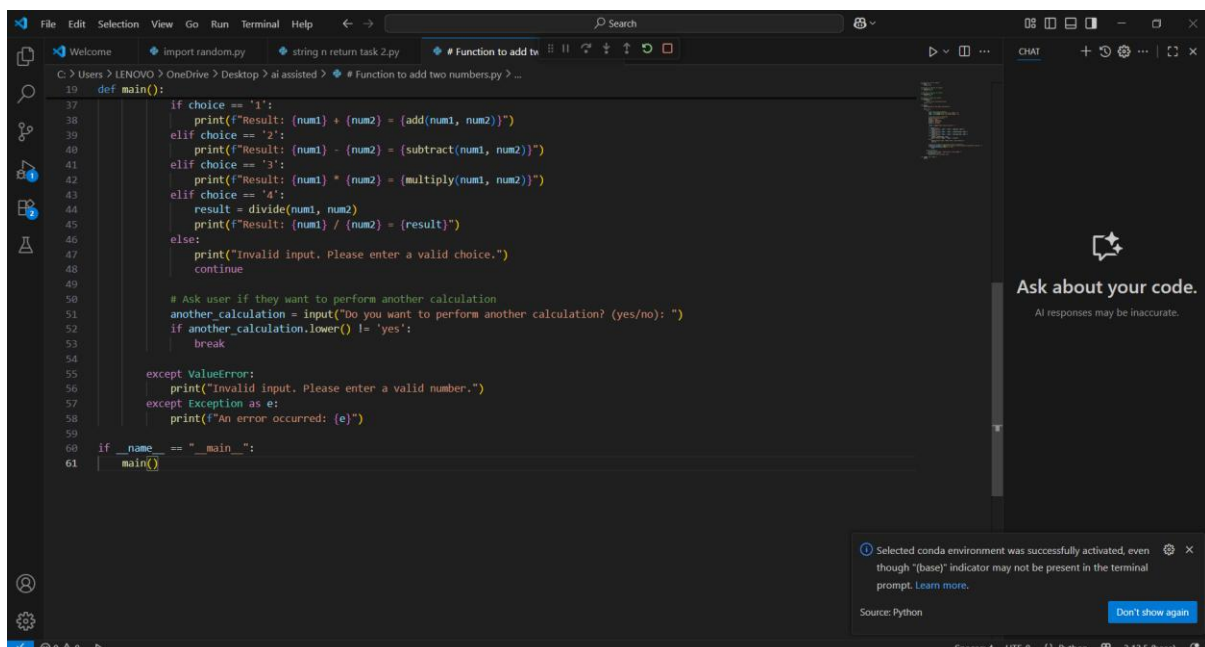
Prompt: Ask Google Gemini to generate a Python program that implements a simple calculator using functions (add, subtract, multiply, divide). Then, ask Gemini to explain how the code works.

Code generated:



The screenshot shows a code editor with a Python file named "Function to add two numbers.py". The code defines four functions: add, subtract, multiply, and divide. The divide function includes a check for division by zero. A main function is also defined, which prompts the user for two numbers and an operation choice. The code is as follows:

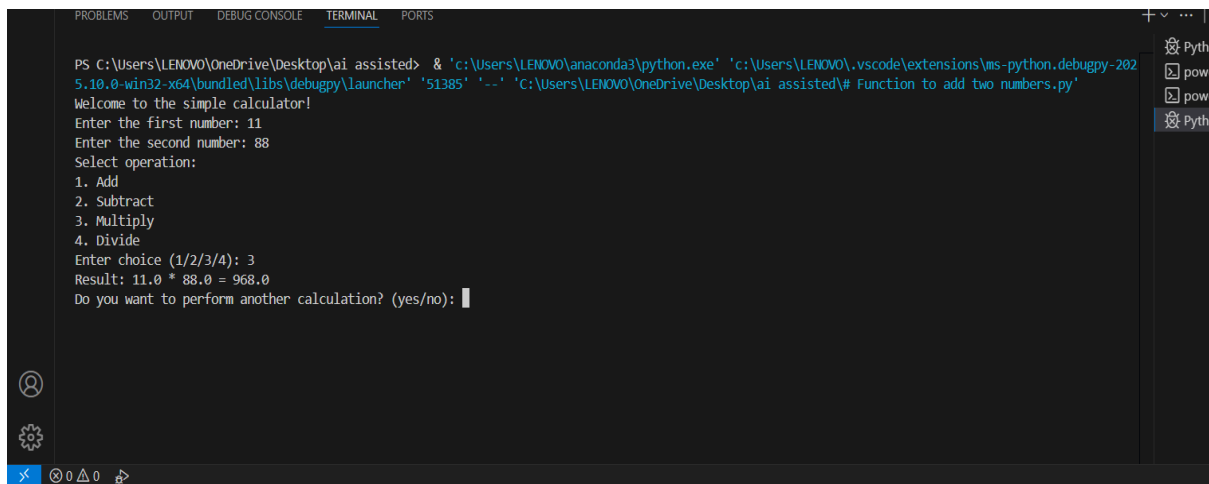
```
1 # Function to add two numbers
2 def add(x, y):
3     return x + y
4
5 # Function to subtract two numbers
6 def subtract(x, y):
7     return x - y
8
9 # Function to multiply two numbers
10 def multiply(x, y):
11     return x * y
12
13 # Function to divide two numbers
14 def divide(x, y):
15     if y == 0:
16         return "Error! Division by zero."
17     return x / y
18
19 def main():
20     print("Welcome to the simple calculator!")
21
22     while True:
23         try:
24             # Get user input for numbers
25             num1 = float(input("Enter the first number: "))
26             num2 = float(input("Enter the second number: "))
27
28             # Get user input for operation
29             print("Select operation:")
30             print("1. Add")
31             print("2. Subtract")
32             print("3. Multiply")
33             print("4. Divide")
34
35             choice = input("Enter choice (1/2/3/4): ")
36
37             if choice == '1':
```



The screenshot shows the continuation of the Python calculator program. The main function continues with logic to perform the selected operation based on the user's choice. It also includes error handling for invalid inputs and a loop to ask if the user wants to perform another calculation. The code is as follows:

```
38         print(f"Result: (num1) + (num2) = {add(num1, num2)}")
39     elif choice == '2':
40         print(f"Result: (num1) - (num2) = {subtract(num1, num2)}")
41     elif choice == '3':
42         print(f"Result: (num1) * (num2) = {multiply(num1, num2)}")
43     elif choice == '4':
44         result = divide(num1, num2)
45         print(f"Result: (num1) / (num2) = {result}")
46     else:
47         print("Invalid input. Please enter a valid choice.")
48         continue
49
50     # Ask user if they want to perform another calculation
51     another_calculation = input("Do you want to perform another calculation? (yes/no): ")
52     if another_calculation.lower() != 'yes':
53         break
54
55     except ValueError:
56         print("Invalid input. Please enter a valid number.")
57     except Exception as e:
58         print(f"An error occurred: {e}")
59
60 if __name__ == "__main__":
61     main()
```

Output:



```
PS C:\Users\LENOVO\OneDrive\Desktop\ai assisted> & 'c:\Users\LENOVO\anaconda3\python.exe' 'c:\Users\LENOVO\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher' '51385' '--' 'C:\Users\LENOVO\OneDrive\Desktop\ai assisted\# Function to add two numbers.py'
Welcome to the simple calculator!
Enter the first number: 11
Enter the second number: 88
Select operation:
1. Add
2. Subtract
3. Multiply
4. Divide
Enter choice (1/2/3/4): 3
Result: 11.0 * 88.0 = 968.0
Do you want to perform another calculation? (yes/no):
```

Observation:

1. Function Definitions

- These functions perform basic arithmetic:

->def add(x, y):

return x + y

-->def subtract(x, y):

return x - y

-->def multiply(x, y):

return x * y

-->def divide(x, y):

if y == 0:

-->return "Error: Division by zero"

return x / y

- Each function takes two numbers x and y, and returns the result.
- The divide() function includes error handling for division by zero.

2. The main() Function

- This is where user interaction happens:

def main():

```
print("Simple Calculator") print("Select operation:") print("1. Add")
print("2. Subtract") print("3. Multiply")
print("4. Divide")
```

- The program prints a menu of operations for the user.

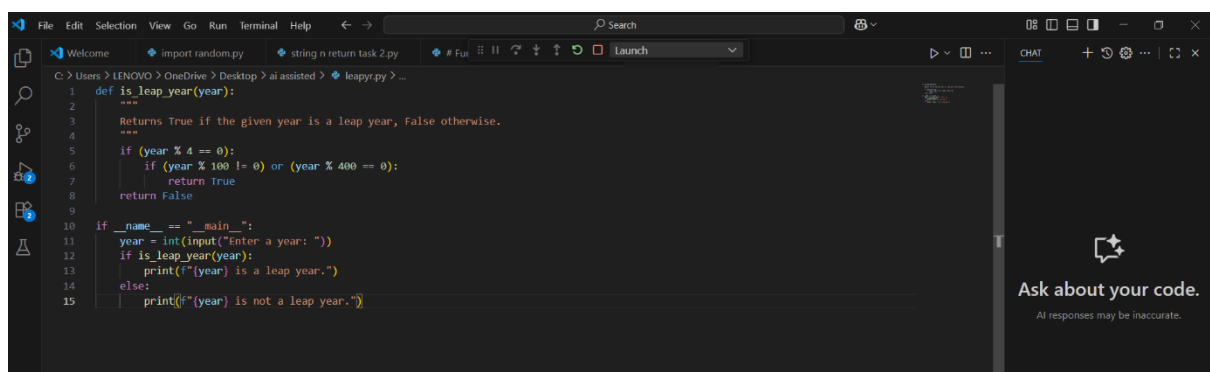
```
choice = input("Enter choice (1/2/3/4): ")
```

The user selects an operation (e.g., 3 for multiply).

Task#5

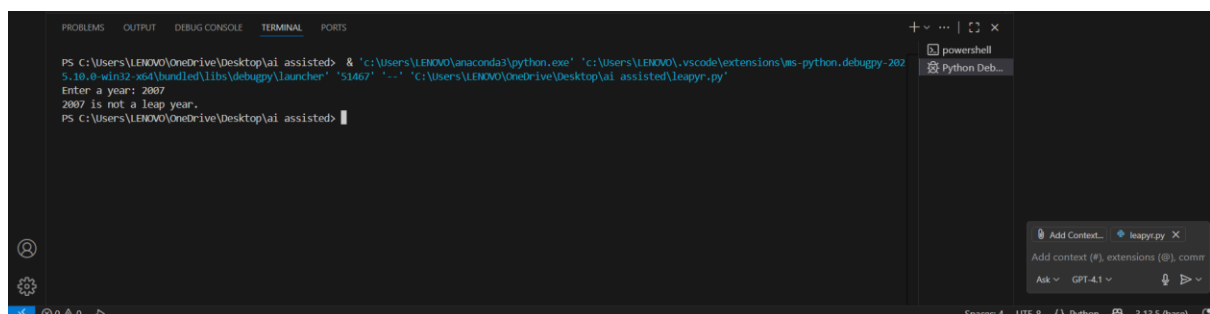
Prompt: Use Cursor AI to create a Python program that checks if a given year is a leap year or not. Try different prompt styles and see how Cursor modifies its code suggestions.

Code generated:



```
1 def is_leap_year(year):
2     """
3     Returns True if the given year is a leap year, False otherwise.
4     """
5     if (year % 4 == 0):
6         if (year % 100 != 0) or (year % 400 == 0):
7             return True
8     return False
9
10 if __name__ == "__main__":
11     year = int(input("Enter a year: "))
12     if is_leap_year(year):
13         print(f"{year} is a leap year.")
14     else:
15         print(f"{year} is not a leap year.")
```

Output:



```
PS C:\Users\LENOVO\OneDrive\Desktop\ai assisted> & 'c:\Users\LENOVO\anaconda3\python.exe' 'c:\Users\LENOVO\.vscode\extensions\ms-python.debugpy-202
5.10.0-win32-x64\lib\debugpy\launcher' '51467' '--' 'c:\Users\LENOVO\OneDrive\Desktop\ai assisted\leapyr.py'
Enter a year: 2007
2007 is not a leap year.
PS C:\Users\LENOVO\OneDrive\Desktop\ai assisted>
```

Observation:

1. Function Deflnition

- `def is_leap_year(year: int) -> bool:`

Returns True if the given year is a leap year, False otherwise. """

```
return year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)
```

- Purpose: Checks leap year condition.
- Type hinting:

- year: int → function expects an integer.

- → bool → function returns a boolean (True or False).
- Logic: Implements the leap year condition in one line.

2. Main Execution Block

```
if __name__ == "__main__":
```

```
    year = int(input("Enter a year: ")) if is_leap_year(year):
```

```
        print(f"{year} is a leap year.")
```

```
    else:
```

```
        print(f"{year} is not a leap year.")
```

- if __name__ == "__main__": Ensures this code only runs when the script is executed directly.
- input(...): Takes user input and converts it to an integer.

- Conditional check: Calls is_leap_year(year) to check and print the appropriate message.

Output from Terminal:

```
Enter a year: 2007
```

```
2007 is not a leap year.
```

- the output is correct.