

AI ASSISTANT CODING

ASSIGNMENT-3.2

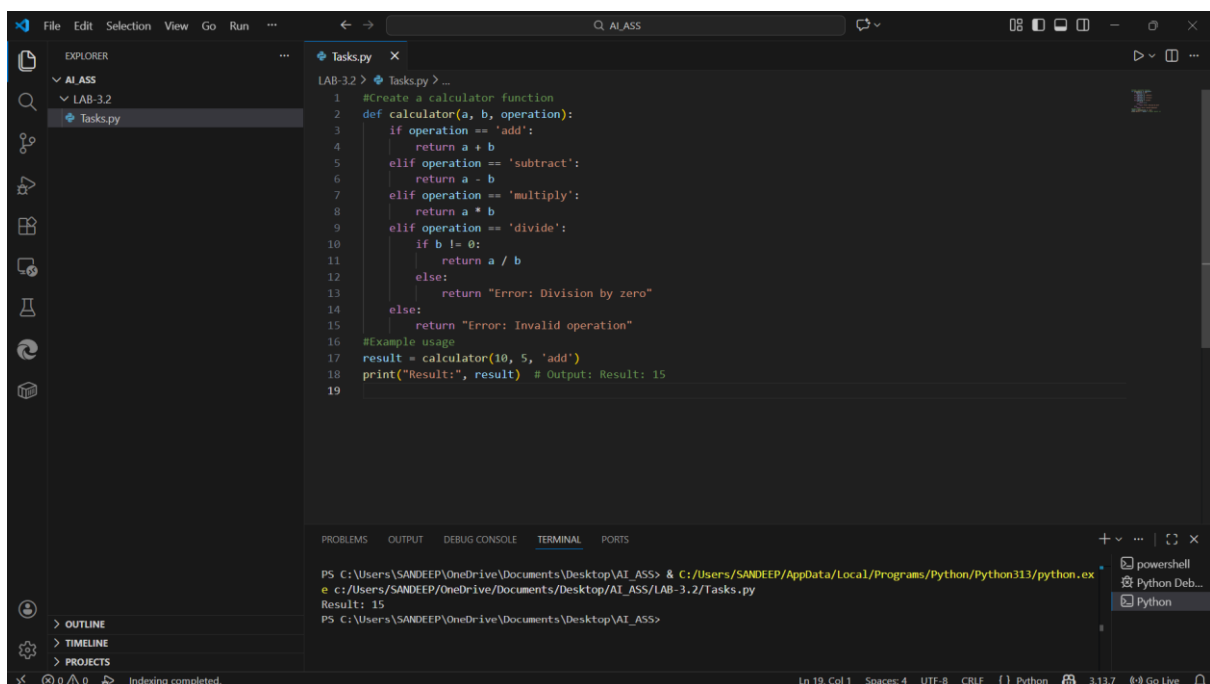
Task Description-1

- Progressive Prompting for Calculator Design: Ask the AI to design a simple calculator program by initially providing only the function name. Gradually enhance the prompt by adding comments and usage examples.

Expected Output-1

- Comparison showing improvement in AI-generated calculator logic and structure.

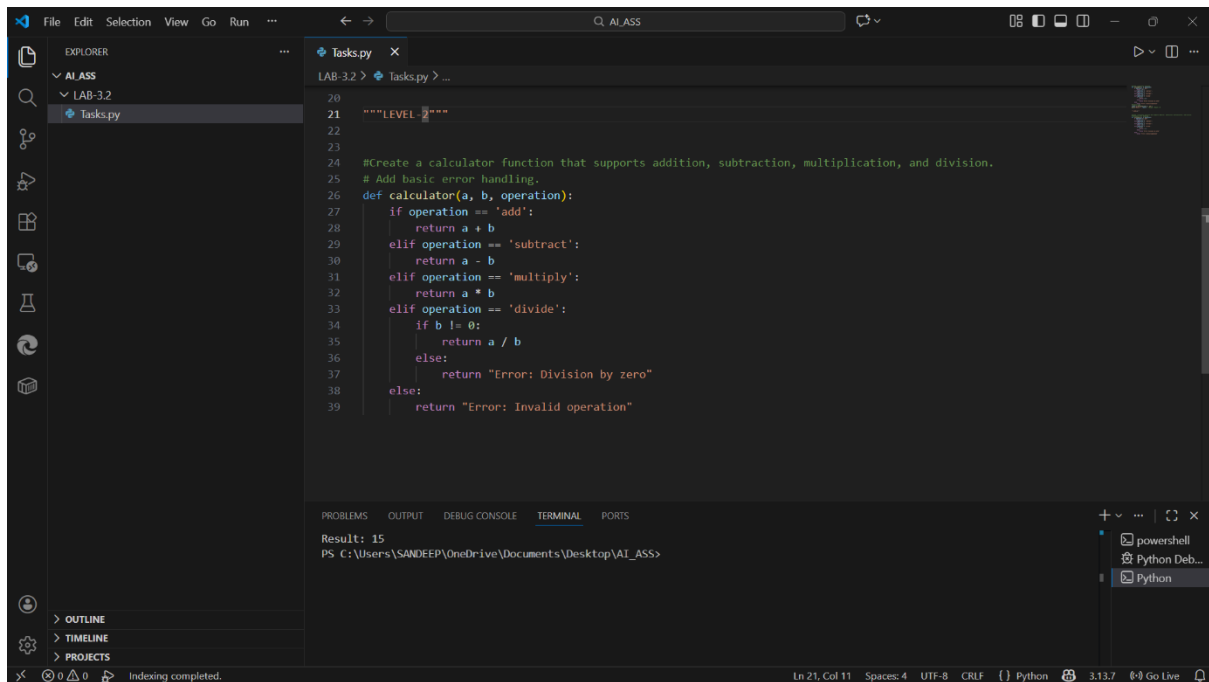
CODE:



```
1 #Create a calculator function
2 def calculator(a, b, operation):
3     if operation == 'add':
4         return a + b
5     elif operation == 'subtract':
6         return a - b
7     elif operation == 'multiply':
8         return a * b
9     elif operation == 'divide':
10        if b != 0:
11            return a / b
12        else:
13            return "Error: Division by zero"
14    else:
15        return "Error: Invalid operation"
16 #Example usage
17 result = calculator(10, 5, 'add')
18 print("Result:", result) # Output: Result: 15
19
```

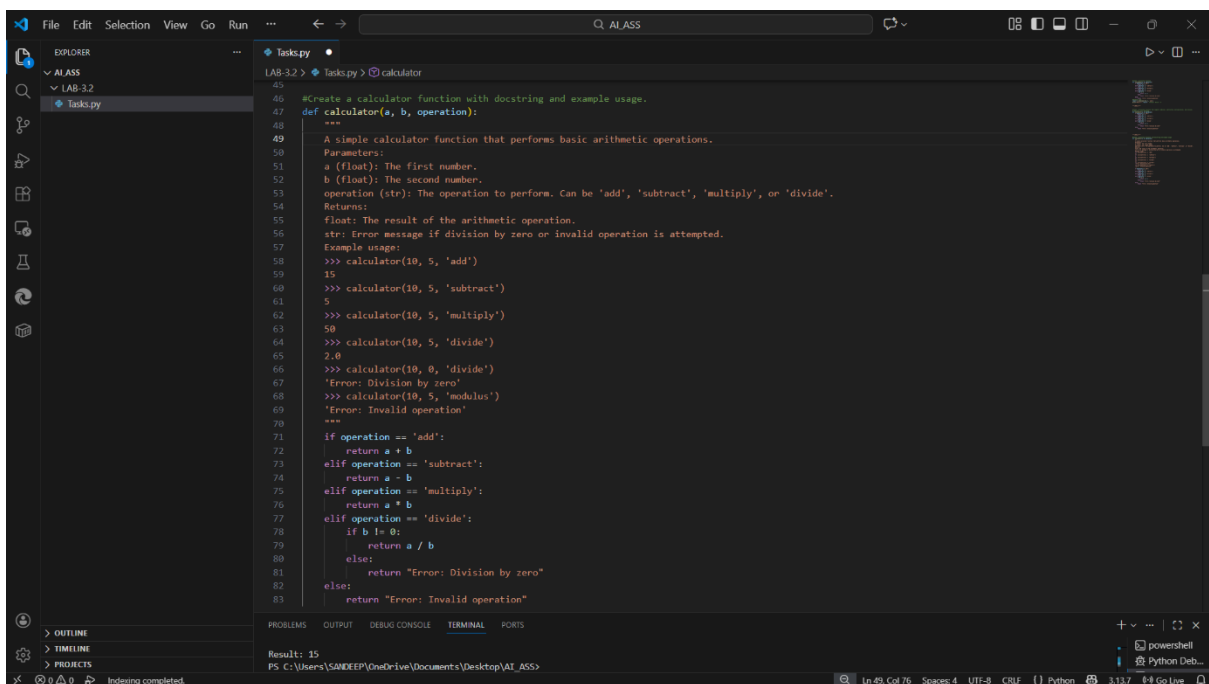
Terminal Output:

```
PS C:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS> & C:\Users\SANDEEP\AppData\Local\Programs\Python\Python313\python.exe
e c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS\LAB-3.2\Tasks.py
Result: 15
PS C:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS>
```



```
20
21 """LEVEL 2"""
22
23
24 #Create a calculator function that supports addition, subtraction, multiplication, and division.
25 # Add basic error handling.
26 def calculator(a, b, operation):
27     if operation == 'add':
28         return a + b
29     elif operation == 'subtract':
30         return a - b
31     elif operation == 'multiply':
32         return a * b
33     elif operation == 'divide':
34         if b != 0:
35             return a / b
36         else:
37             return "Error: Division by zero"
38     else:
39         return "Error: Invalid operation"
```

Result: 15
PS C:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS>



```
46 #Create a calculator function with docstring and example usage.
47 def calculator(a, b, operation):
48     """
49     A simple calculator function that performs basic arithmetic operations.
50     Parameters:
51     a (float): The first number.
52     b (float): The second number.
53     operation (str): The operation to perform. Can be 'add', 'subtract', 'multiply', or 'divide'.
54     Returns:
55     float: The result of the arithmetic operation.
56     str: Error message if division by zero or invalid operation is attempted.
57     Example usage:
58     >>> calculator(10, 5, 'add')
59     15
60     >>> calculator(10, 5, 'subtract')
61     5
62     >>> calculator(10, 5, 'multiply')
63     50
64     >>> calculator(10, 5, 'divide')
65     2.0
66     >>> calculator(10, 0, 'divide')
67     'Error: Division by zero'
68     >>> calculator(10, 5, 'modulus')
69     'Error: Invalid operation'
70     """
71     if operation == 'add':
72         return a + b
73     elif operation == 'subtract':
74         return a - b
75     elif operation == 'multiply':
76         return a * b
77     elif operation == 'divide':
78         if b != 0:
79             return a / b
80         else:
81             return "Error: Division by zero"
82     else:
83         return "Error: Invalid operation"
```

Result: 15
PS C:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS>

Task Description-2

- Refining Prompts for Sorting Logic: Start with a vague prompt for sorting student marks, then refine it to clearly specify sorting order and constraints.

Expected Output-2

- AI-generated sorting function evolves from ambiguous logic to an accurate and efficient implementation.

The screenshot shows a VS Code editor with a file named `Tasks.py` open. The code defines a function `sort_student_marks` that takes a list of marks and returns a sorted list. The example usage shows a list of marks `[88, 92, 79, 85, 95]` being sorted to `[79, 85, 88, 92, 95]`. The terminal output shows the execution of the script, which prints the sorted marks.

```
1 #Write a function to sort student marks.
2 def sort_student_marks(marks):
3     return sorted(marks)
4 # Example usage:
5 student_marks = [88, 92, 79, 85, 95]
6 sorted_marks = sort_student_marks(student_marks)
7 print("Sorted student marks:", sorted_marks)
```

```
PS C:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS> cd "c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS"; & 'c:\Users\SANDEEP\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\SANDEEP\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '53631' '--' 'c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS\LAB-3.2\Tasks.py'
Sorted student marks: [79, 85, 88, 92, 95]
```

The screenshot shows a VS Code editor with a file named `Tasks.py` open. The code defines a function `sort_student_marks_descending` that takes a list of marks and returns a sorted list in descending order. The example usage shows a list of marks `[85, 92, 78, 90, 88]` being sorted to `[92, 90, 88, 85, 78]`. The terminal output shows the execution of the script, which prints the sorted marks in descending order.

```
1 #Write a function to sort student marks in descending order.
2 #Do not use built-in sort functions.
3 def sort_student_marks_descending(marks):
4     n = len(marks)
5     # Using bubble sort algorithm to sort the marks in descending order
6     for i in range(n):
7         for j in range(0, n-i-1):
8             if marks[j] < marks[j+1]:
9                 # Swap if the element found is less than the next element
10                marks[j], marks[j+1] = marks[j+1], marks[j]
11     return marks
12 # Example usage
13 student_marks = [85, 92, 78, 90, 88]
14 sorted_marks = sort_student_marks_descending(student_marks)
15 print("Sorted student marks in descending order:", sorted_marks)
```

```
PS C:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS> cd "c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS"; & 'c:\Users\SANDEEP\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\SANDEEP\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '62315' '--' 'c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS\LAB-3.2\Tasks.py'
Sorted student marks in descending order: [92, 90, 88, 85, 78]
```

Task Description-3

- Few-Shot Prompting for Prime Number Validation: Provide multiple input-output examples for a function that checks whether a number is prime. Observe how few-shot prompting improves correctness.

Expected Output-3

Improved prime-checking function with better edge-case handling.

```
1 #Check if a number is prime.
2 # Input: 2 + Output: Prime
3 # Input: 4 + Output: Not Prime
4 # Input: 7 + Output: Prime
5 # Input: 1 + Output: Not Prime
6
7 def is_prime(num):
8     """
9     Checks if a number is prime.
10    Parameters:
11    num (int): The number to check.
12    Returns:
13    str: 'Prime' if the number is prime, 'Not Prime' otherwise.
14    Example usage:
15    >>> is_prime(2)
16    'Prime'
17    >>> is_prime(4)
18    'Not Prime'
19    >>> is_prime(7)
20    'Prime'
21    >>> is_prime(1)
22    'Not Prime'
23    """
24    if num <= 1:
25        return "Not Prime"
26    for i in range(2, int(num**0.5) + 1):
27        if num % i == 0:
28            return "Not Prime"
29    return "Prime"
30
31 # Example usage
32 print(is_prime(2)) # Output: Prime
33 print(is_prime(4)) # Output: Not Prime
34 print(is_prime(7)) # Output: Prime
35 print(is_prime(1)) # Output: Not Prime
```

PS C:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS> cd 'c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS'; & 'c:\Users\SANDEEP\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS\Task.py'

Prime
Not Prime
Prime
Not Prime

Task Description-4

- Prompt-Guided UI Design for Student Grading System: Create a user interface for a student grading system that calculates total marks, percentage, and grade based on user input.

Expected Output-4

- Well-structured UI code with accurate calculations and clear output display.

```
1 # Create a Python UI program that accepts student marks,
2 # calculates total, percentage, and assigns grade.
3 def student_grading():
4     marks = []
5     for i in range(5):
6         mark = float(input(f"Enter marks for subject {i+1}: "))
7         marks.append(mark)
8     total = sum(marks)
9     percentage = (total / 500) * 100
10    if percentage >= 90:
11        grade = 'A'
12    elif percentage >= 80:
13        grade = 'B'
14    elif percentage >= 70:
15        grade = 'C'
16    elif percentage >= 60:
17        grade = 'D'
18    else:
19        grade = 'F'
20    print(f"Total Marks: {total}")
21    print(f"Percentage: {percentage:.2f}%")
22    print(f"Grade: {grade}")
23    student_grading()
24
```

PS C:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS> cd 'c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS'; & 'c:\Users\SANDEEP\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS\Task.py'

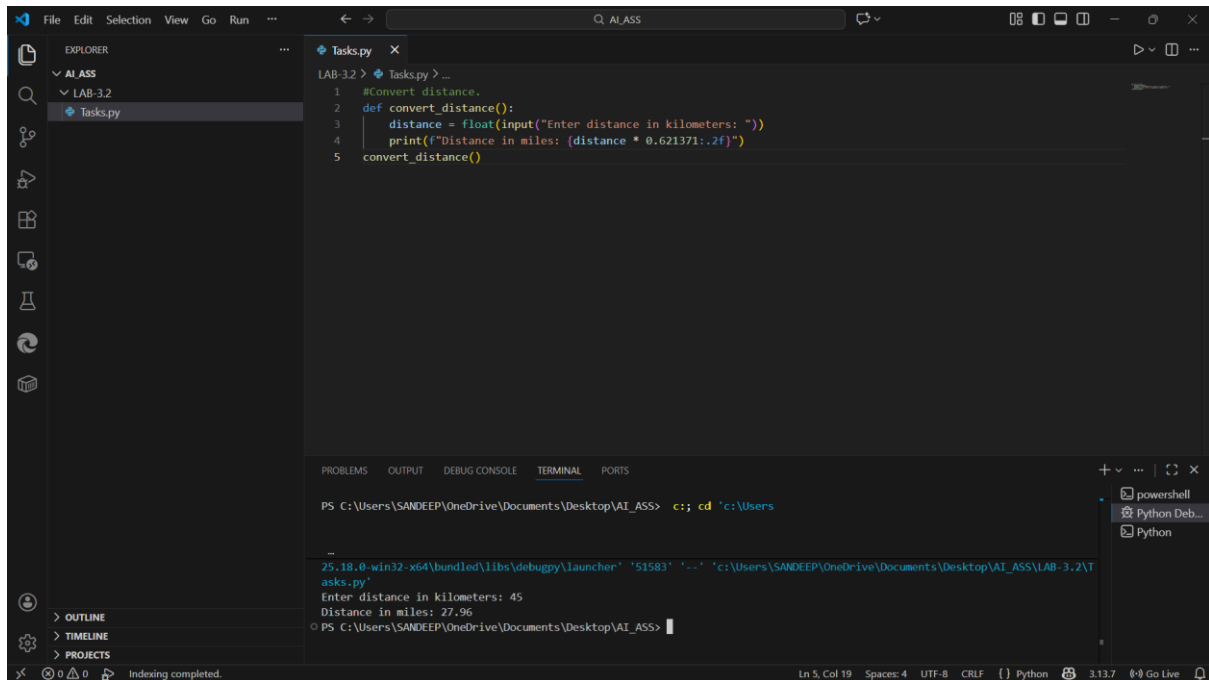
Enter marks for subject 1: 90
Enter marks for subject 2: 80
Enter marks for subject 3: 70
Enter marks for subject 4: 60
Enter marks for subject 5: 50
Total Marks: 350.0
Percentage: 70.00%
Grade: C

Task Description-5

- Analyzing Prompt Specificity in Unit Conversion Functions: Improving a Unit Conversion Function (Kilometers to Miles and Miles to Kilometers) Using Clear Instructions.

Expected Output-5

- Analysis of code quality and accuracy differences across multiple prompt variations.



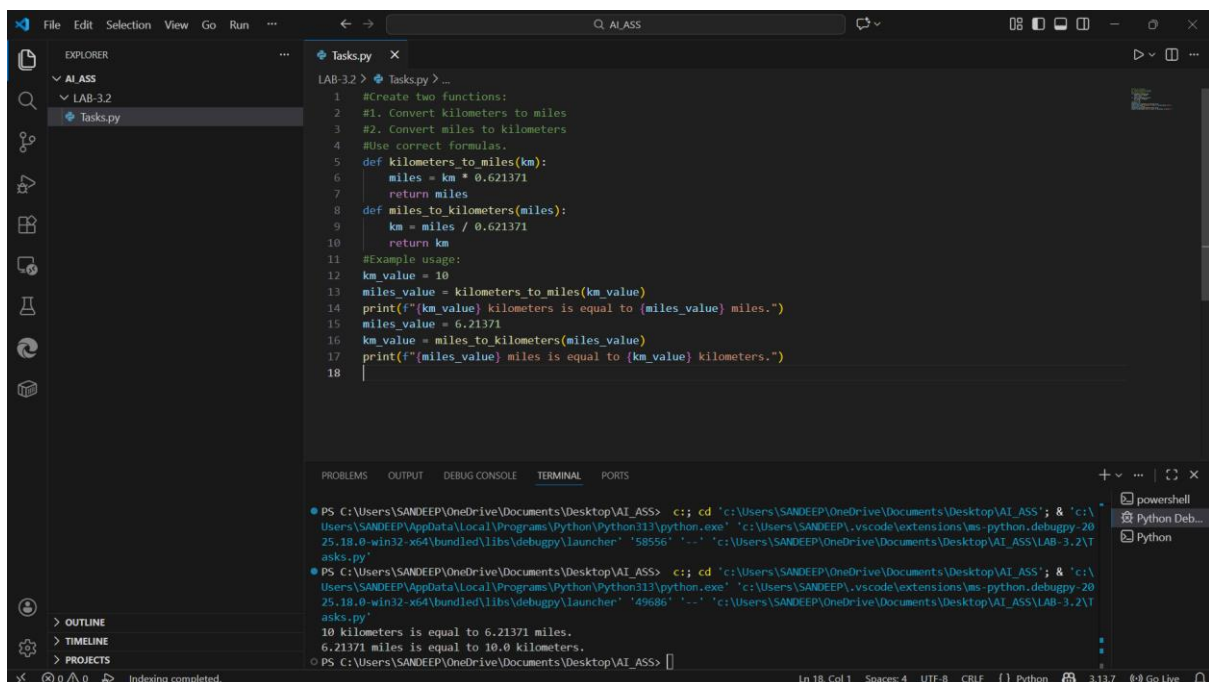
The screenshot shows a Visual Studio Code editor with a file explorer on the left showing a project named 'AI_ASS' containing a file 'Tasks.py'. The main editor window displays the following Python code:

```
1 #Convert distance.
2 def convert_distance():
3     distance = float(input("Enter distance in kilometers: "))
4     print(f"Distance in miles: {distance * 0.621371:.2f}")
5 convert_distance()
```

The bottom panel shows the terminal output:

```
PS C:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS> c:\> cd 'c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS\LAB-3.2\Tasks.py'
25.18.0-win32-x64\bundled\libs\debugpy\launcher '51583' '--' 'c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS\LAB-3.2\Tasks.py'
Enter distance in kilometers: 45
Distance in miles: 27.96
PS C:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS>
```

Kilometers to Miles and Miles to Kilometers:



The screenshot shows a Visual Studio Code editor with a file explorer on the left showing a project named 'AI_ASS' containing a file 'Tasks.py'. The main editor window displays the following Python code:

```
1 #Create two functions:
2 #1. Convert kilometers to miles
3 #2. Convert miles to kilometers
4 #Use correct formulas.
5 def kilometers_to_miles(km):
6     miles = km * 0.621371
7     return miles
8 def miles_to_kilometers(miles):
9     km = miles / 0.621371
10    return km
11 #Example usage:
12 km_value = 10
13 miles_value = kilometers_to_miles(km_value)
14 print(f"{km_value} kilometers is equal to {miles_value} miles.")
15 miles_value = 6.21371
16 km_value = miles_to_kilometers(miles_value)
17 print(f"{miles_value} miles is equal to {km_value} kilometers.")
18
```

The bottom panel shows the terminal output:

```
PS C:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS> c:\> cd 'c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS' & 'c:\Users\SANDEEP\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\SANDEEP\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58556' '--' 'c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS\LAB-3.2\Tasks.py'
PS C:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS> c:\> cd 'c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS' & 'c:\Users\SANDEEP\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\SANDEEP\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '49696' '--' 'c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS\LAB-3.2\Tasks.py'
10 kilometers is equal to 6.21371 miles.
6.21371 miles is equal to 10.0 kilometers.
PS C:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS>
```