

# **COURSE: AI Assisted Coding**

**NAME : M.Harish**

**BATCH-34**

**HALLTICKET.NO : 2303A52387**

**AIAC-ASS - 3.2**

## **Lab 3:**

**Prompt Engineering – Improving Prompts and Context Management.**

### **Task Description – 1**

Progressive Prompting for Calculator Design

### **Objective:**

To observe how AI-generated code improves as prompt clarity and context increase.

### **AI Output – Level 1**

```
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
17 #Example usage
18 result = calculator(10, 5, 'add')
19 print("Result:", result) # Output: Result: 15
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
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>
```

## LEVEL-2:

```
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"
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

## LEVEL-3:

The screenshot shows a VS Code editor window with a file named 'Tasks.py' open. The code defines a function 'calculator(a, b, operation)' that performs basic arithmetic operations. The function includes a docstring with parameters, returns, and example usage. The example usage shows calls to the function for addition, subtraction, multiplication, and division, as well as error handling for division by zero and invalid operations. The terminal output shows the results of these calls: 15, 5, 50, 2.0, and error messages.

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

## Task -2:

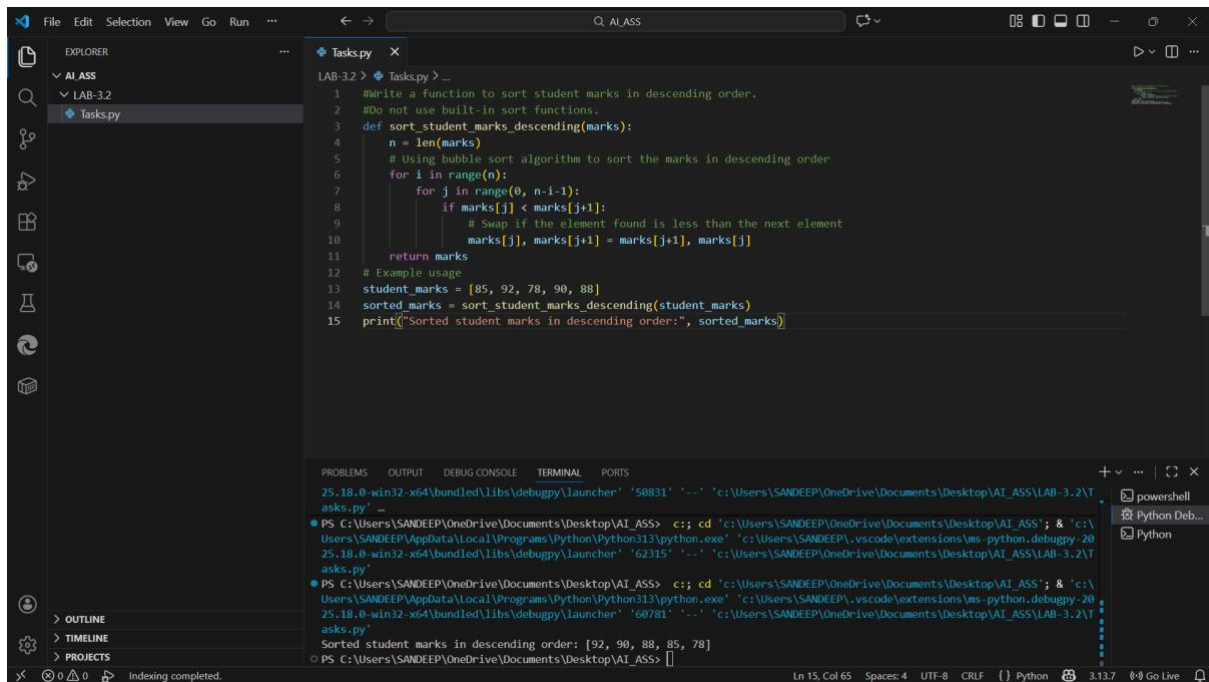
- Refining Prompts for Sorting Logic: Start with a vague prompt for sorting student marks, then refine it to clearly specify sorting order and constraints.

The screenshot shows a VS Code editor window with a file named 'Tasks.py' open. The code defines a function 'sort\_student\_marks(marks)' that sorts a list of student marks. The function includes a docstring with parameters, returns, and example usage. The example usage shows a list of marks, a call to the function, and the output of 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(f"Sorted student marks:", sorted_marks)
```

The terminal output shows the command to run the script and the output of the sorted marks: [79, 85, 88, 92, 95].

```
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\bundled\libs\debugpy\launcher' '53631' '-' 'c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS\LAB-3.2\Tasks.py'
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\bundled\libs\debugpy\launcher' '50831' '-' 'c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS\LAB-3.2\Tasks.py'
Sorted student marks: [79, 85, 88, 92, 95]
```



```
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)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

25.18.0-win32-x64\bundled\libs\debugpy\launcher '50831' '-' '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' '62315' '-' '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' '60781' '-' 'c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI\_ASS\LAB-3.2\Tasks.py'

Sorted student marks in descending order: [92, 90, 88, 85, 78]

PS C:\Users\SANDEEP\OneDrive\Documents\Desktop\AI\_ASS>

Ln 15, Col 65 Spaces: 4 UTF-8 CRLF {} Python 3.13.7 0-0 Go Live

## Task -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.

The screenshot shows the Visual Studio Code editor with a file named 'Task.py' open. The code is a Python function 'is\_prime(num)' that checks if a number is prime. It includes docstrings, comments, and example usage. The terminal at the bottom shows the command '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'' and the output of the script.

```
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 def is_prime(num):
7     """
8     Checks if a number is prime.
9     Parameters:
10     num (int): The number to check.
11     Returns:
12     str: 'Prime' if the number is prime, 'Not Prime' otherwise.
13     Example usage:
14     >>> is_prime(2)
15     'Prime'
16     >>> is_prime(4)
17     'Not Prime'
18     >>> is_prime(7)
19     'Prime'
20     >>> is_prime(1)
21     'Not Prime'
22     """
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
```

Terminal Output:

```
PS C:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS> c:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS\Task.py
Prime
Not Prime
Prime
Not Prime
```

## Task -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.

The screenshot shows the Visual Studio Code editor with a file named 'Task.py' open. The code is a Python function 'student\_grading()' that prompts the user to enter marks for five subjects, calculates the total marks, percentage, and assigns a grade based on the percentage. The terminal at the bottom shows the command to run the script and the output of the program.

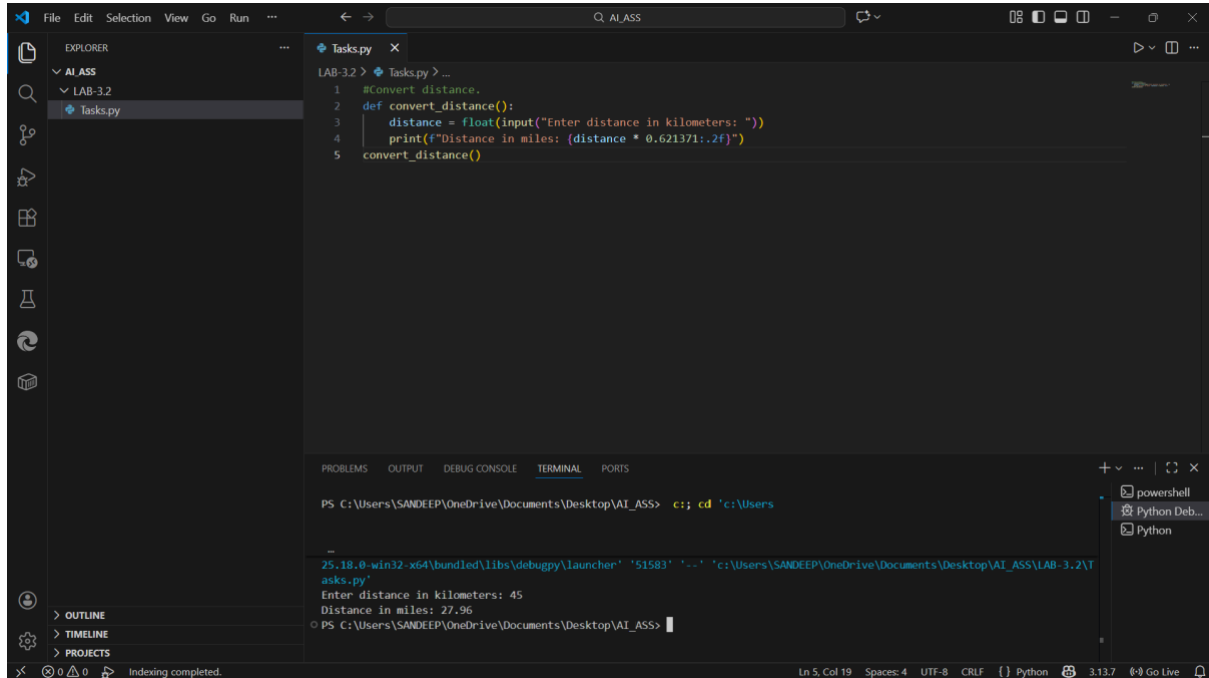
```
1 #Create a Python UI program that accepts student marks,
2 #calculates total, percentage, and assigns grade.
3 def student_grading():
4     marks = []
5     for i in range(5):
6         mark = float(input(f"Enter marks for subject {i+1}: "))
7         marks.append(mark)
8     total = sum(marks)
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
```

Terminal Output:

```
PS C:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS> 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.



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows a project named 'AI\_ASS' with a subfolder 'LAB-3.2' containing a file 'tasks.py'. The main editor window displays the code for 'tasks.py':

```
LAB-3.2 > tasks.py > ...
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()
```

Below the editor, the TERMINAL panel shows the command prompt output:

```
PS C:\Users\SANDEEP\OneDrive\Documents\Desktop\AI_ASS> cd 'c:\Users
...
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>
```

The status bar at the bottom indicates 'Ln 5, Col 19', 'Spaces: 4', 'UTF-8', 'CRLF', and the file is encoded in Python.

- (Kilometers to Miles and Miles to Kilometers)

