

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.

The screenshot shows a code editor with a Python script named `simple_calculator.py`. The code defines a function `simple_calculator(a, b, operation)` that performs basic arithmetic operations based on the `operation` parameter. It handles addition, subtraction, multiplication, division, and unknown operations. A comment at the end of the script provides a brief description. Below the code editor is a terminal window showing the execution of the script and its output for various operations.

```
C: > AI Assisted Coding Assessments > import hashlib.py > ...
1 def simple_calculator(a, b, operation):
2     if operation == 'add':
3         return a + b
4     elif operation == 'subtract':
5         return a - b
6     elif operation == 'multiply':
7         return a * b
8     elif operation == 'divide':
9         if b != 0:
10             return a / b
11         else:
12             return "Error: Division by zero"
13     else:
14         return "Error: Unknown operation"
15 # Example usage:
16 result = simple_calculator(10, 5, 'add')
17 result_subtract = simple_calculator(10, 5, 'subtract')
18 result_multiply = simple_calculator(10, 5, 'multiply')
19 result_divide = simple_calculator(10, 5, 'divide')
20 print("Addition Result:", result)
21 print("Subtraction Result:", result_subtract)
22 print("Multiplication Result:", result_multiply)
23 print("Division Result:", result_divide) # A simple calculator function that performs basic arithmetic operations

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Bhanu Prasad> & C:/Python314/python.exe "c:/AI Assisted Coding Assessments/import hashlib.py"
Addition Result: 15
Subtraction Result: 5
Multiplication Result: 50
Division Result: 2.0
PS C:\Users\Bhanu Prasad>
```

EXPLANATION:

In the above we created a function we the basic operations like addition ,subtraction,multiplication and division are performed.We need to enter the input and tand the operation which we need to perform

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.

```
C: > AI Assisted Coding Assessments > import hashlib.py > ...
1 def sort_students(students):
2     """
3         students: list of tuples (name, mark)
4         returns: sorted list by mark desc, then name asc
5     """
6     return sorted(students, key=lambda x: (-x[1], x[0]))
7 # Example usage:
8 students = [("Alice", 85), ("Bob", 90), ("Charlie", 85), ("David", 92)]
9 sorted_students = sort_students(students)
10 print(sorted_students)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
PS C:\Users\Bhanu Prasad> & C:/Python314/python.exe "c:/AI Assisted Coding Assessments/import hashlib.py"
[('David', 92), ('Bob', 90), ('Alice', 85), ('Charlie', 85)]
PS C:\Users\Bhanu Prasad>
```

EXPLANATION:

In the above code we are sorting the students marks based on the desending order.

In this I used a function called the sorted function which returs the values based on the desending order.

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.

```
C:/AI Assissted Coding Assessments > ➜ import hashlib.py > ...
1  def is_prime(n: int) -> bool:
2      if n <= 1:
3          return False
4      if n == 2:
5          return True
6      if n % 2 == 0:
7          return False
8
9      # Only check up to sqrt(n)
10     for i in range(3, int(n//2) + 1, 2):
11         if n % i == 0:
12             return False
13     return True
14 print(is_prime(29)) # Example usage
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Bhanu Prasad> & C:/Python314/python.exe "c:/AI Assissted Coding Assessments/import hashlib.py"
True
PS C:\Users\Bhanu Prasad>
```

EXPLANATION:

In the code we are checking the number whether the number is prime or not .

Based on the input the checks the number result the output in the Boolean format .

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.

```
C: > AI Assisted Coding Assessments > import hashlib.py > ...
1 def grading_student_marks_system(marks):
2     """
3         This function takes a list of student marks and returns a list of grades based on the following criteria:
4         - Marks >= 90: Grade 'A'
5         - Marks >= 80 and < 90: Grade 'B'
6         - Marks >= 70 and < 80: Grade 'C'
7         - Marks >= 60 and < 70: Grade 'D'
8         - Marks < 60: Grade 'F'
9
10    :param marks: List of integers representing student marks
11    :return: List of strings representing student grades
12    """
13    grades = []
14    for mark in marks:
15        if mark >= 90:
16            grades.append('A')
17        elif mark >= 80:
18            grades.append('B')
19        elif mark >= 70:
20            grades.append('C')
21        elif mark >= 60:
22            grades.append('D')
23        else:
24            grades.append('F')
25    return grades
26 total_marks = [95, 82, 67, 74, 58]
27 average_mark = sum(total_marks) / len(total_marks)
28 student_marks = [95, 82, 67, 74, 58]
29
30 student_grades = grading_student_marks_system(student_marks)
31 print(student_grades)
32 print(f"Average Mark: {average_mark}")
33 print(f"total_marks: {total_marks}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Bhanu Prasad> & C:/Python314/python.exe "c:/AI Assisted Coding Assessments/import hashlib.py"
['A', 'B', 'D', 'C', 'F']
Average Mark: 75.2
total_marks: [95, 82, 67, 74, 58]
PS C:\Users\Bhanu Prasad>
```

EXPLANATION:

In the code we are calculating the grades based on the marks. And displaying the total and the average marks of the students .

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.

```

C:/> AI Assissted Coding Assessments > import nasnilo.py > ...
1  def Unit_Convert(value, from_unit, to_unit):
2      """
3          Convert a value from one unit to another.
4
5          Parameters:
6              value (float): The numerical value to convert.
7              from_unit (str): The unit of the input value.
8              to_unit (str): The unit to convert the value to.
9
10         Returns:
11             float: The converted value.
12         """
13
14         # Define conversion factors
15         conversion_factors = {
16             'meters': 1.0,
17             'kilometers': 1000.0,
18             'centimeters': 0.01,
19             'millimeters': 0.001,
20             'miles': 1609.34,
21             'yards': 0.9144,
22             'feet': 0.3048,
23             'inches': 0.0254
24         }
25
26         # Check if units are valid
27         if from_unit not in conversion_factors or to_unit not in conversion_factors:
28             raise ValueError("Invalid unit provided.")
29
30         # Convert the value to meters first
31         value_in_meters = value * conversion_factors[from_unit]
32
33         # Convert from meters to the target unit
34         converted_value = value_in_meters / conversion_factors[to_unit]
35
36         return converted_value
37
38     # Example usage:
39     result = Unit_Convert(10, 'meters', 'kilometers')
40     print(f"10 meters is equal to {result} kilometers.")

```

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
PS C:\Users\Bhanu Prasad> & C:/Python314/python.exe "c:/AI Assissted Coding Assessments/import hashlib.py"
10 meters is equal to 0.01 kilometers.
PS C:\Users\Bhanu Prasad>

```

EXPLANATION:

In the code I declared a function which converts the units in to another units based on the user needs.