

## **Assignment-3.2**

**Name :Kashireddy Tejaswini**

**Ht.no : 2303A51425**

**Batch : 21**

---

### **TaskDescription-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.

#### **Prompt used-1:**

write a python function named calculator that takes three parameters: two numbers and an operator (as a string: '+', '-', '\*', '/').

#### **Code-1:**

```

gmmnt 3.2.py > ...
#write a python function named calculator that takes three parameters: two numbers and an operator (as a string)
def calculator(num1, num2, operator):
    if operator == '+':
        return num1 + num2
    elif operator == '-':
        return num1 - num2
    elif operator == '*':
        return num1 * num2
    elif operator == '/':
        if num2 != 0:
            return num1 / num2
        else:
            return "Error: Division by zero"
    else:
        return "Error: Invalid operator"
# Example usage:
user_input1=float(input("Enter first number: "))
user_input2=float(input("Enter second number: "))
user_operator=input("Enter an operator (+, -, *, /): ")
result=calculator(user_input1,user_input2,user_operator)
print("The result is:", result)

```

## Output-1:

```

(base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding> & C:/Users/WINDOWS/
rs/WINDOWS/OneDrive/Desktop/Teja 3_2/AI Assit coding/Assignment-3.2.py"
Enter first number: 5
Enter second number: 10
Enter an operator (+, -, *, /): +
The result is: 15.0
(base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding>

```

## Promptused-2:

write a python function named calculator that takes three parameters: two numbers and

an operator (as a string: '+', '-', '\*', '/').

requirements:

Take 2 numbers and operator as input Handle

division by zero

Handle invalid operator inputs

example input:10,5,"+"output:15

## Code-2:

```
#write a python function named calculator that takes three parameters: two numbers and an operator (as a string)
#requirements:
#Take 2 numbers and operator as input
#Handle division by zero
#Handle invalid operator inputs
#example input:10,5,"+"output:15
def calculator(num1, num2, operator):
    if operator == '+':
        return num1 + num2
    elif operator == '-':
        return num1 - num2
    elif operator == '*':
        return num1 * num2
    elif operator == '/':
        if num2 != 0:
            return num1 / num2
        else:
            return "Error: Division by zero"
    else:
        return "Error: Invalid operator"
# Example usage:
user_input1=float(input("Enter first number: "))
user_input2=float(input("Enter second number: "))
user_operator=input("Enter an operator (+, -, *, /): ")
result=calculator(user_input1,user_input2,user_operator)
print("The result is:", result)
```

## Output-2:

```
Enter first number: 20
Enter second number: 16
Enter an operator (+, -, *, /): -
Enter an operator (+, -, *, /): -
The result is: 4.0
(base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding> █
```

## Comparison of 2 codes:

<b>Aspect</b>	<b>Initial Calculator Version</b>	<b>Improved Calculator Version</b>
<b>Input Handling</b>	Assumes numeric inputs are already valid	Converts inputs to float and validates them
<b>Error Handling</b>	Handles division by zero and invalid operator	Handles invalid numbers, division by zero, and invalid operators
<b>Code Structure</b>	Basic conditional logic inside function	Cleaner structure with explicit input validation
<b>Readability</b>	Understandable but less descriptive	Improved readability with comments and clear variable names
<b>Robustness</b>	May fail for non-numeric inputs	More robust and user-friendly
<b>User Feedback</b>	Limited error messages	Clear and informative error messages
<b>Suitability</b>	Suitable for simple calculations	Suitable for real-world and user-facing programs

## **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.

### **Code-1:**

```

#write a python program to sort student marks.
def sort_student_marks(marks):
    try:
        marks_list = [float(mark) for mark in marks]
        marks_list.sort()
        return marks_list
    except ValueError:
        return "Error: All marks should be numbers."
# Example usage:
user_marks = input("Enter student marks separated by spaces: ").split()
sorted_marks = sort_student_marks(user_marks)
print("Sorted student marks:", sorted_marks)

```

## Output-1:

```

PS D:\AI assist> python -u "d:\AI assist\code3.2.py"
Enter student marks separated by spaces: 78 85 92 60
Sorted student marks: [60.0, 78.0, 85.0, 92.0]
PS D:\AI assist> python -u "d:\AI assist\code3.2.py"
Enter student marks separated by spaces: 90
Sorted student marks: [90.0]
PS D:\AI assist> python -u "d:\AI assist\code3.2.py"
Enter student marks separated by spaces:
Sorted student marks: []
PS D:\AI assist>

```

## Code-2:

```

#write a python function that takes a list of student marks as input and sorts them in ascending order.
#constraints:
#Marks are integers between 0 and 100.
#Handle duplicate marks correctly.
#Use an efficient sorting algorithm.
#Display the sorted list clearly.
#Example input: [88, 92, 75, 88, 95]
#Example output: [75,88,88,92,95]
def sort_student_marks(marks):
    try:
        marks_list = [int(mark) for mark in marks if 0 <= int(mark) <= 100]
        marks_list.sort()
        return marks_list
    except ValueError:
        return "Error: All marks should be numeric values."
user_marks=input("Enter student marks separated by spaces: ").split()
sorted_marks=sort_student_marks(user_marks)
print("Sorted student marks:", sorted_marks)

```

## Output-2:

```
coding/task.py"
Enter student marks separated by spaces: 65 72 68
Sorted student marks: [65, 68, 72]
(base) PS C:\Users\WINDOWS\Desktop\Teja_3_2\AI Assit coding>
```

### **Justification:**

Refining the prompt by specifying sorting order and input constraints improved the accuracy and structure of the AI-generated sorting logic.

The refined prompt guided the AI to handle numeric conversion, duplicates, and empty inputs more reliably.

This shows that clearer prompts lead to more efficient and predictable AI-generated programs.

### **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.

### **Prompt used:**

Write a python function that checks whether a given number is prime or not. example

input: 2 output: prime number

input: 7 output: prime number

input: 1 output: not a prime number

### **Code:**

```

#Write a python function that checks whether a given number is prime or not. example input: 2 output: prime number
#input:7 output: prime number input:1output:not a prime number
def is_prime(number):
    try:
        num = int(number)
        if num <= 1:
            return "Not a prime number"
        for i in range(2, int(num**0.5) + 1):
            if num % i == 0:
                return "Not a prime number"
        return "Prime number"
    except ValueError:
        return "Error: Input should be an integer."
# Example usage:
user_input = input("Enter a number to check if it's prime: ")
result = is_prime(user_input)
print(result)

```

## Output:

```

base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding> &
nter a number to check if it's prime: 2
rime number
base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding> &
oding/task.py"
nter a number to check if it's prime: 7
rime number
base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding> &
oding/task.py"
nter a number to check if it's prime: 1
ot a prime number
base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding> █

```

## Justification:

Few-shot prompting improves the prime-checking logic by showing the AI clear examples of both prime and non-prime numbers.

These examples guide the AI to correctly handle edge cases such as 0, 1, and negative numbers, which are often missed in zero-shot prompting.

As a result, the generated function is more accurate, efficient, and reliable in validating prime numbers.

## **TaskDescription-4**

- Prompt-GuidedUIDesignforStudentGradingSystem:Create a user interface for a student grading system that calculates total marks, percentage, and grade based on user input.

### **PromptUsed:**

Design a simple and user-friendly student grading system UI Requirements:

1. Create input fields to enter marks for multiple subjects.
2. Calculate total marks and percentage based on the entered values.
3. Assign grades based on percentage (e.g., A, B, C, Fail).
4. Display total marks, percentage, and grade clearly on the screen.
5. Use a clean and well-structured layout.
6. Validate inputs to ensure only numeric values are accepted.
7. Use HTML and CSS for the user interface and JavaScript for calculations. Create all files in one file

### **Code:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Grading System</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
            padding: 20px;
            background-color: #f4f4f4;
        }
        .container {
            max-width: 500px;
            margin: auto;
            background: white;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
        input[type="number"] {
            width: 100%;
            padding: 10px;
            margin: 10px 0;
            box-sizing: border-box;
        }
        button {
            padding: 10px 15px;
            background-color: #28a745;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }
        button:hover {
            background-color: #218838;
        }
        .result {
            margin-top: 20px;
        }
    </style>
</head>
<body>
```

```

<body>
    <div class="container">
        <h2>Student Grading System</h2>
        <label for="marks">Enter Marks for 5 Subjects:</label>
        <input type="number" id="subject1" placeholder="Subject 1" min="0" max="100">
        <input type="number" id="subject2" placeholder="Subject 2" min="0" max="100">
        <input type="number" id="subject3" placeholder="Subject 3" min="0" max="100">
        <input type="number" id="subject4" placeholder="Subject 4" min="0" max="100">
        <input type="number" id="subject5" placeholder="Subject 5" min="0" max="100">
        <button onclick="calculateGrade()">Calculate Grade</button>
        <div class="result" id="result"></div>
    </div>

    <script>
        function calculateGrade() {
            let marks = [];
            for (let i = 1; i <= 5; i++) {
                let mark = parseFloat(document.getElementById('subject' + i).value);
                if (isNaN(mark) || mark < 0 || mark > 100) {
                    alert('Please enter valid marks between 0 and 100 for all subjects.');
                    return;
                }
                marks.push(mark);
            }

            let totalMarks = marks.reduce((a, b) => a + b, 0);
            let percentage = (totalMarks / 500) * 100;
            let grade;

            if (percentage >= 90) {
                grade = 'A';
            } else if (percentage >= 80) {
                grade = 'B';
            } else if (percentage >= 70) {
                grade = 'C';
            } else if (percentage >= 60) {
                grade = 'D';
            } else {
                grade = 'Fail';
            }

            document.getElementById('result').innerHTML =
```

```

```

```
            document.getElementById('result').innerHTML =
                <p>Total Marks: ${totalMarks} / 500</p>
                <p>Percentage: ${percentage.toFixed(2)}%</p>
                <p>Grade: ${grade}</p>
```;
        </script>
    </body>
</html>

```

**Output:**

## Student Grading System

Enter Marks for 5 Subjects:

**Calculate Grade**

Total Marks: 420 / 500

Percentage: 84.00%

Grade: B

# Student Grading System

Enter Marks for 5 Subjects:

**Calculate Grade**

Total Marks: 125 / 500

Percentage: 25.00%

Grade: Fail

## Justification:

The prompt-guided UI design ensures accurate calculation of total marks, percentage, and grade based on user input.

Clear input validation improves reliability by handling invalid and missing values effectively.

A structured UI layout enhances readability and provides clear output for students and evaluators.

## TaskDescription-5

- AnalyzingPromptSpecificityinUnitConversionFunctions:ImprovingaUnitConversionFunction(KilometerstoMilesandMilestoKilometers)Using Clear Instructions.

### Promptused-1:

WriteaPythonfunctiontoconvertkilometerstomilesandmilestokilometers. Take input from the user and display the result.

### Code-1:

```
#WriteaPythonfunctiontoconvertkilometerstomilesandmilestokilometers.  
#Take input from the user and display the result.  
def convert_distance(value, unit):  
    try:  
        distance=float(value)  
        if unit.lower()=='km':  
            miles=distance*0.621371  
            return f"{distance} kilometers is equal to {miles} miles."  
        elif unit.lower()=='miles':  
            kilometers=distance/0.621371  
            return f"{distance} miles is equal to {kilometers} kilometers."  
        else:  
            return "Error: Invalid unit. Please use 'km' or 'miles'."  
    except ValueError:  
        return "Error: Please enter a valid number for distance."  
user_value=input("Enter the distance value: ")  
user_unit=input("Enter the unit (km/miles): ")  
result=convert_distance(user_value,user_unit)  
print(result)
```

### Output-1:

```
(base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding> C
(base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding> c
(base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding> &
coding/task.py"
Enter the distance value: 1
Enter the unit (km/miles): miles
1.0 miles is equal to 1.6093444978925633 kilometers.
(base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding> &
coding/task.py"
Enter the distance value: 1
Enter the unit (km/miles): km
1.0 kilometers is equal to 0.621371 miles.
(base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding> &
coding/task.py"
Enter the distance value: -3
Enter the unit (km/miles): miles
-3.0 miles is equal to -4.82803349367769 kilometers.
(base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding> ^
```

**Prompt used-2:**

Write a Python program using functions to convert:

- Kilometers to miles
- Miles to kilometers

Requirements:

- Take numeric input from the user.
- Allow the user to choose the conversion type.
- Use correct conversion formulas.
- Validate input and handle invalid values.
- Display clear and formatted output.

**Code-2:**

```

#Write a Python program using functions to convert:
#Kilometers to miles
#Miles to kilometers Requirements:
#Take numeric input from the user.
#Allow the user to choose the conversion type.
#Use correct conversion formulas.
#Validate input and handle invalid values.
#Display clear and formatted output.
def convert_distance(value, unit):
    try:
        distance = float(value)
        if unit.lower() == 'km':
            miles = distance * 0.621371
            return f"{distance} kilometers is equal to {miles:.2f} miles."
        elif unit.lower() == 'miles':
            km = distance / 0.621371
            return f"{distance} miles is equal to {km:.2f} kilometers."
        else:
            return "Error: Invalid unit. Please use 'km' or 'miles'."
    except ValueError:
        return "Error: Input should be a numeric value."
# Example usage:
user_value = input("Enter the distance values: ")
user_unit = input("Enter the unit ('km' for kilometers, 'miles' for miles): ")
result = convert_distance(user_value, user_unit)
print(result)

```

## Output-2:

```

(base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding> &
coding/task.py"
Enter the distance values: 1
Enter the unit ('km' for kilometers, 'miles' for miles): miles
1.0 miles is equal to 1.61 kilometers.
(base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding> &
coding/task.py"
Enter the distance values: 0
Enter the unit ('km' for kilometers, 'miles' for miles): km
0.0 kilometers is equal to 0.00 miles.
(base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding>

```

## Justification:

In the first prompt, the unit conversion logic produces correct results but lacks strict validation, allowing negative distance values.

With a more detailed prompt, the program improves by adding clearer output formatting and better handling of invalid inputs.

Overall, increased prompt specificity leads to more accurate, robust, and user-friendly unit conversion code.