

# AI ASSISTANT CODING ASSIGNMENT 3.2

---

NAME : N. THANAY

ROLL NO : 2303A51424

BATCH NO : 21

---

## 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.**
- 

## Output :

---

```
def calculator(a, b, operation):
    if operation == 'add':
        return a + b
    elif operation == 'subtract':
        return a - b
    elif operation == 'multiply':
        return a * b
    elif operation == 'divide':
        if b != 0:
            return a / b
        else:
            return "Error: Division by zero"
    else:
        return "Error: Invalid operation"
```

```
# give it as a scientific calculator also with this included operations
def scientific_calculator(a, b=None, operation=None):
    import math

    if operation in ['add', 'subtract', 'multiply', 'divide']:
        return calculator(a, b, operation)
    elif operation == 'sqrt':
        return math.sqrt(a)
    elif operation == 'power':
        return math.pow(a, b)
    elif operation == 'log':
        if a > 0:
            return math.log(a)
        else:
            return "Error: Logarithm of non-positive number"
    elif operation == 'sin':
        return math.sin(a)
    elif operation == 'cos':
        return math.cos(a)
    elif operation == 'tan':
        return math.tan(a)
    else:
        return "Error: Invalid operation"
```

```
#it should also perform normal calculator operations
# Example usage:
print(scientific_calculator(10, 5, 'add'))      # Output: 15
print(scientific_calculator(10, 5, 'subtract')) # Output: 5
print(scientific_calculator(10, 5, 'multiply')) # Output: 50

#it should also perform far more complex scientific calculations
print(scientific_calculator(16, operation='sqrt')) # Output: 4.0
print(scientific_calculator(2, 3, 'power'))        # Output:      8.0
print(scientific_calculator(10, operation='log'))   # Output: 2.302585092994046
```

```
#give a code which should also perform for n number of inputs for all the operations
def n_input_calculator(*args, operation):
    if operation == 'add':
        return sum(args)
    elif operation == 'subtract':
        result = args[0]
        for num in args[1:]:
            result -= num
        return result
    elif operation == 'multiply':
        result = 1
        for num in args:
            result *= num
        return result
    elif operation == 'divide':
        result = args[0]
        for num in args[1:]:
            if num != 0:
                result /= num
            else:
                return "Error: Division by zero"
        return result
    else:
        return "Error: Invalid operation"
```

---

## EXPLANATION :

---

```
#explain the code with comments
# The function n_input_calculator takes a variable number of arguments (*args) and an operation
# It performs the specified operation on all the input numbers
# For addition, it sums all the numbers
# For subtraction, it subtracts each subsequent number from the first
# For multiplication, it multiplies all the numbers together
# For division, it divides the first number by each subsequent number, checking for division by zero
```

---

### 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.**
- 

## OUTPUT:

---

```
#generate a code for sorting student marks.
def sort_student_marks(marks):
    return sorted(marks)
# Example usage:
marks = [88, 92, 79, 85, 95]
sorted_marks = sort_student_marks(marks)
print("Sorted student marks:", sorted_marks)
# Output: Sorted student marks: [79, 85, 88, 92, 95]
```

```

# use insertion sort algorithm to sort the marks
def insertion_sort_student_marks(marks):
    for i in range(1, len(marks)):
        key = marks[i]
        j = i - 1
        while j >= 0 and key < marks[j]:
            marks[j + 1] = marks[j]
            j -= 1
        marks[j + 1] = key
    return marks
# Example usage:
marks = [88, 92, 79, 85, 95]
sorted_marks = insertion_sort_student_marks(marks)
print("Sorted student marks using insertion sort:", sorted_marks)
# Output: Sorted student marks using insertion sort: [79, 85, 88, 92, 95]

```

```

#also grade the students based on their marks , also students have 6 subjects
def grade_students(marks):
    grades = []
    for mark in marks:
        average = sum(mark) / len(mark)
        if average >= 90:
            grades.append('A')
        elif average >= 80:
            grades.append('B')
        elif average >= 70:
            grades.append('C')
        elif average >= 60:
            grades.append('D')
        else:
            grades.append('F')
    return grades
# Example usage:
student_marks = [[88, 92, 79, 85, 95, 90], [70, 75, 80, 65, 60, 72], [50, 55, 60, 58, 52, 49]]
student_grades = grade_students(student_marks)
print("Student grades:", student_grades)
# Output: Student grades: ['A', 'C', 'F']

```

```

# The function grade_students takes a list of lists, where each inner list contains marks for 6 subjects
# It calculates the average marks for each student and assigns a grade based on the average
# Finally, it returns a list of grades corresponding to each student

```

---

