

Assignment-3.2

Name:M Varshith Reddy

HTNO:2303A52087

Batch:40

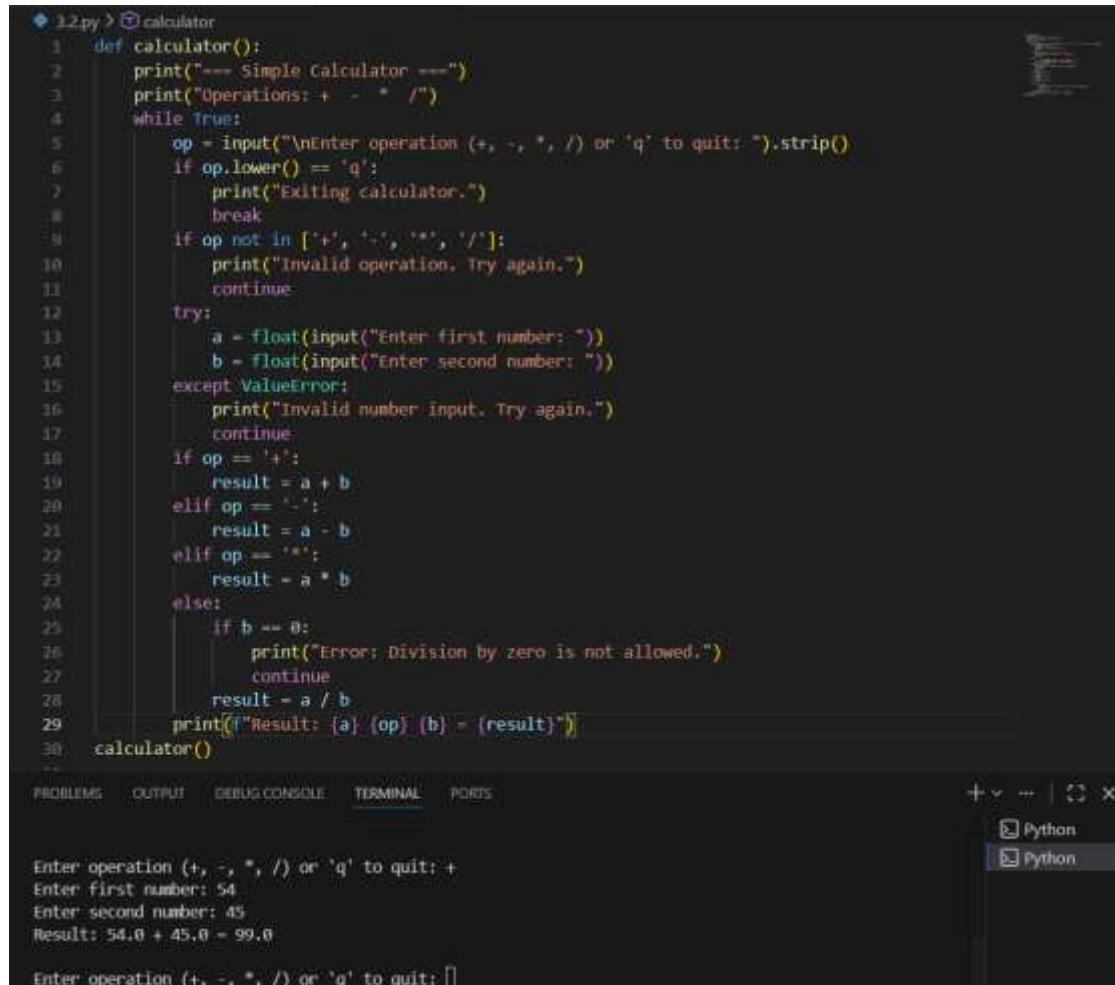
Task Description-1

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:



```
❶ 3.2.py > @calculator
 1 def calculator():
 2     print("== Simple Calculator ==")
 3     print("Operations: + - * /")
 4     while True:
 5         op = input("\nEnter operation (+, -, *, /) or 'q' to quit: ").strip()
 6         if op.lower() == 'q':
 7             print("Exiting calculator.")
 8             break
 9         if op not in ['+', '-', '*', '/']:
10             print("invalid operation. try again.")
11             continue
12         try:
13             a = float(input("Enter first number: "))
14             b = float(input("Enter second number: "))
15         except ValueError:
16             print("Invalid number input. Try again.")
17             continue
18         if op == '+':
19             result = a + b
20         elif op == '-':
21             result = a - b
22         elif op == '*':
23             result = a * b
24         else:
25             if b == 0:
26                 print("Error: Division by zero is not allowed.")
27                 continue
28             result = a / b
29         print(f"Result: {a} {op} {b} = {result}")
30     calculator()
```

The screenshot shows a Python code editor with the file '3.2.py' open. The code defines a 'calculator()' function that prints a welcome message and a list of operations. It then enters a loop where it prompts the user for an operation and two numbers. If the operation is not one of the four supported ones, it prints an error message and continues. If the user enters 'q', it exits the loop. Otherwise, it performs the calculation and prints the result. The code editor has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and WORDS. The TERMINAL tab is active, showing the execution of the script and its output. The output shows the user entering '+', then 54 and 45, and seeing the result 99.0. The user then types another line starting with 'Enter operation'.

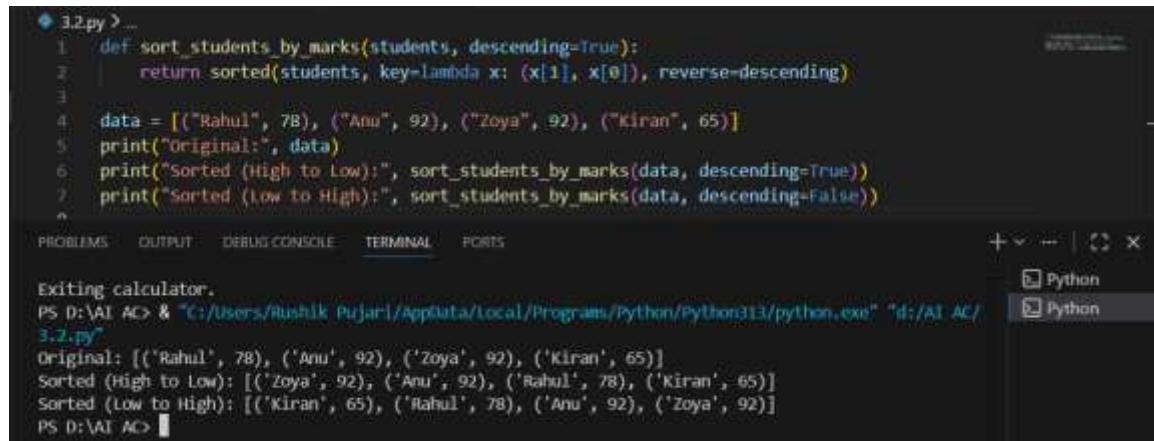
Task Description-2

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.

Code:



```
3.2.py > ...
1 def sort_students_by_marks(students, descending=True):
2     return sorted(students, key=lambda x: (x[1], x[0]), reverse=descending)
3
4 data = [("Rahul", 78), ("Anu", 92), ("Zoya", 92), ("Kiran", 65)]
5 print("Original:", data)
6 print("Sorted (High to Low):", sort_students_by_marks(data, descending=True))
7 print("Sorted (Low to High):", sort_students_by_marks(data, descending=False))
8

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ✎ ⌂ ×
Python Python
Exiting calculator.
PS D:\AI\AC> & "C:/Users/Rushik_Pujari/AppData/Local/Programs/Python/Python313/python.exe" "d:/AI/AC/3.2.py"
Original: [('Rahul', 78), ('Anu', 92), ('Zoya', 92), ('Kiran', 65)]
Sorted (High to Low): [('Zoya', 92), ('Anu', 92), ('Rahul', 78), ('Kiran', 65)]
Sorted (Low to High): [('Kiran', 65), ('Rahul', 78), ('Anu', 92), ('Zoya', 92)]
PS D:\AI\AC>
```

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.

Code:

```
◆ 3.2.py > ...
1 def is_prime(n):
2     if n <= 1:
3         return False
4     if n <= 3:
5         return True
6     if n % 2 == 0 or n % 3 == 0:
7         return False
8
9     i = 5
10    while i * i <= n:
11        if n % i == 0 or n % (i + 2) == 0:
12            return False
13        i += 6
14    return True
15
16 tests = [0, 1, 2, 3, 4, 5, 9, 11, 49, 97]
17 for t in tests:
18     print(t, "->", is_prime(t))
19
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

3.2.py*
0 -> False
1 -> False
2 -> True
3 -> True
4 -> False
5 -> True
9 -> False
11 -> True
49 -> False
97 -> True
PS D:\AI AC> []

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.

Code:

```
1  #> calculate_grade
2  def calculate_grade(percentage):
3      if percentage >= 90:
4          return "A"
5      elif percentage >= 80:
6          return "B"
7      elif percentage >= 70:
8          return "C"
9      elif percentage >= 60:
10         return "D"
11     elif percentage >= 50:
12         return "E"
13     else:
14         return "F"
15
16 def student_grading_system():
17     print("---- Student Grading System ----")
18     try:
19         n = int(input("Enter number of subjects: "))
20         if n <= 0:
21             print("Number of subjects must be positive.")
22             return
23     except ValueError:
24         print("Invalid input. Enter an integer.")
25         return
26     marks = []
27     for i in range(1, n + 1):
28         try:
29             m = float(input("Enter marks for subject {} (0-100): ".format(i)))
30             if not (0 <= m <= 100):
31                 print("Marks must be between 0 and 100.")
32                 return
33             marks.append(m)
34         except ValueError:
35             print("Invalid marks input.")
36             return
37     total = sum(marks)
38     percentage = total / n
39     grade = calculate_grade(percentage)
40     print("\n---- Result ----")
41     print("Total Marks: ({total:.2f}) / ({n}*100)")
42     print("Percentage: ({percentage:.2f}%)")
43     print("Grade: ({grade})")
44     student_grading_system()
45
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + - X
PS D:\AI\AC> & "C:/Users/Rushik_Patel/AppData/Local/Programs/Python/Python311/python.exe" "d:/AI/AC/3.2.py"
Enter number of subjects: 5
Invalid input. Enter an integer.
PS D:\AI\AC> 5
PS D:\AI\AC> & "C:/Users/Rushik_Patel/AppData/Local/Programs/Python/Python311/python.exe" "d:/AI/AC/3.2.py"
---- Student Grading System ---
Enter number of subjects: 5
Enter marks for subject 1 (0-100): 79
Enter marks for subject 2 (0-100): 57
Enter marks for subject 3 (0-100): 84
Enter marks for subject 4 (0-100): 97
Enter marks for subject 5 (0-100): 64
---- Result ---
Total Marks: 381.00 / 500
Percentage: 76.20%
Grade: B
```

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.

Code:

The screenshot shows a Jupyter Notebook interface with a code cell and its corresponding output.

```
❶ 32.py > ⚒ unit_converter
  1 def km_to_miles(km):
  2     return km * 0.62137
  3 def miles_to_km(miles):
  4     return miles * 1.60934
  5 def unit_converter():
  6     print("==> Unit Converter (Km <-> Miles) ==>")
  7     print("1) Kilometers to Miles")
  8     print("2) Miles to Kilometers")
  9     print("3) Exit")
 10    while True:
 11        choice = input("\nChoose an option (1/2/3): ").strip()
 12        if choice == "3":
 13            print("Exiting converter...")
 14            break
 15        if choice not in ["1", "2"]:
 16            print("Invalid option, Try again.")
 17            continue
 18        try:
 19            value = float(input("Enter value: "))
 20            if value < 0:
 21                print("Distance cannot be negative.")
 22                continue
 23            except ValueError:
 24                print("Invalid numeric input.")
 25                continue
 26            if choice == "1":
 27                print(f"[Value] Km = {km_to_miles(value)} Miles")
 28            else:
 29                print(f"[Value] Miles = {miles_to_km(value)} Km")
 30        unit_converter()
 31
```

TERMINAL OUTPUT CONSOLE DEBUG

```
+ - | ×
Python Python
+ × - | ×
Choose an option (1/2/3): 1
Enter value: 15
15.0 miles = 24.100660 km

Choose an option (1/2/3): 2
Enter value: 67
67.0 km = 41.631857 miles

Choose an option (1/2/3): 3
Exiting converter,
PS D:\AI AC> [ ]
```