

ASSIGNMENT

Name: B.Sai Tejaswi

Roll Number: 2303A51184

Batch - 03

AI Assisted Coding

20-01-2026

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.

```
assignment.py > ...
1  # =====
2  # TASK 1: PROGRESSIVE PROMPTING (CALCULATOR)
3  # =====
4
5  # --- VERSION A: INITIAL PROMPT (Zero Context) ---
6  # Prompt: "def calculator():"
7  def calculator_basic():
8      # The AI often just returns a simple hardcoded example
9      a = 10
10     b = 5
11     print("Basic Add:", a + b)
12
13
14 # --- VERSION B: ENHANCED PROMPT (With Context & Logic) ---
15 # Prompt: "Create a calculator class with methods for add, subtract, multiply,
16 # and divide. Include comments and handle division by zero."
17
18 class ProCalculator:
19     """
20     A robust calculator class developed through
21     progressive prompt engineering.
22     """
23
24     def add(self, a, b):
25         return a + b
26
27     def subtract(self, a, b):
28         return a - b
29
```

Expected Output-1

- Comparison showing improvement in AI-generated calculator logic and structure.

```
PS C:\Users\hp\OneDrive\Desktop\ai> c::; cd 'c:\Users\hp\OneDrive\Desktop\ai'; & 'c:\Users\hp\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\hp\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '50181' '--' 'C:\Users\hp\OneDrive\Desktop\ai\assignment.py'
--- Running Task 1: Calculator ---
Basic Add: 15
Enhanced Add: 30
Enhanced Divide: Error: Cannot divide by zero!
-----
PS C:\Users\hp\OneDrive\Desktop\ai> 
```

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.

```

assignment.py > ...
1 # Task: Sort students by marks (Descending), then by Name (Ascending)
2 students = [
3     [{"name": "Teju", "marks": 92}],
4     [{"name": "priya", "marks": 85}],
5     [{"name": "harshini", "marks": 92}],
6     [{"name": "sharanya", "marks": 88}]
7 ]
8
9 # The refined prompt ensures the AI uses a tuple key for multi-level sorting
10 sorted_list = sorted(students, key=lambda x: (-x['marks'], x['name']))
11
12 print("Sorted Student List:")
13 for student in sorted_list:
14     print(f" - {student['name']}: {student['marks']}")

```

Expected Output-2

- AI-generated sorting function evolves from ambiguous logic to an accurate and efficient

implementation.

```

PS C:\Users\hp\OneDrive\Desktop\ai> c::; cd 'c:\Users\hp\OneDrive\Desktop\ai'; & 'c:\Users\hp\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\hp\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '63670' '--' 'c:\Users\hp\OneDrive\Desktop\ai\assignment.py'
Sorted Student List:
- Teju: 92
- harshini: 92
- sharanya: 88
- priya: 85
PS C:\Users\hp\OneDrive\Desktop\ai>

```

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.

```
assignment.py > ...
1 def is_prime(n):
2     # Prompting with examples (Input: 1 -> Output: False)
3     # ensures these guards are included.
4     if n <= 1:
5         return False
6     if n <= 3:
7         return True
8     if n % 2 == 0 or n % 3 == 0:
9         return False
10
11     i = 5
12     while i * i <= n:
13         if n % i == 0 or n % (i + 2) == 0:
14             return False
15         i += 6
16     return True
17
18 # Testing cases
19 test_values = [-5, 1, 2, 11, 25]
20 for val in test_values:
21     print(f"Is {val} prime? {is_prime(val)}")
```

Expected Output-3

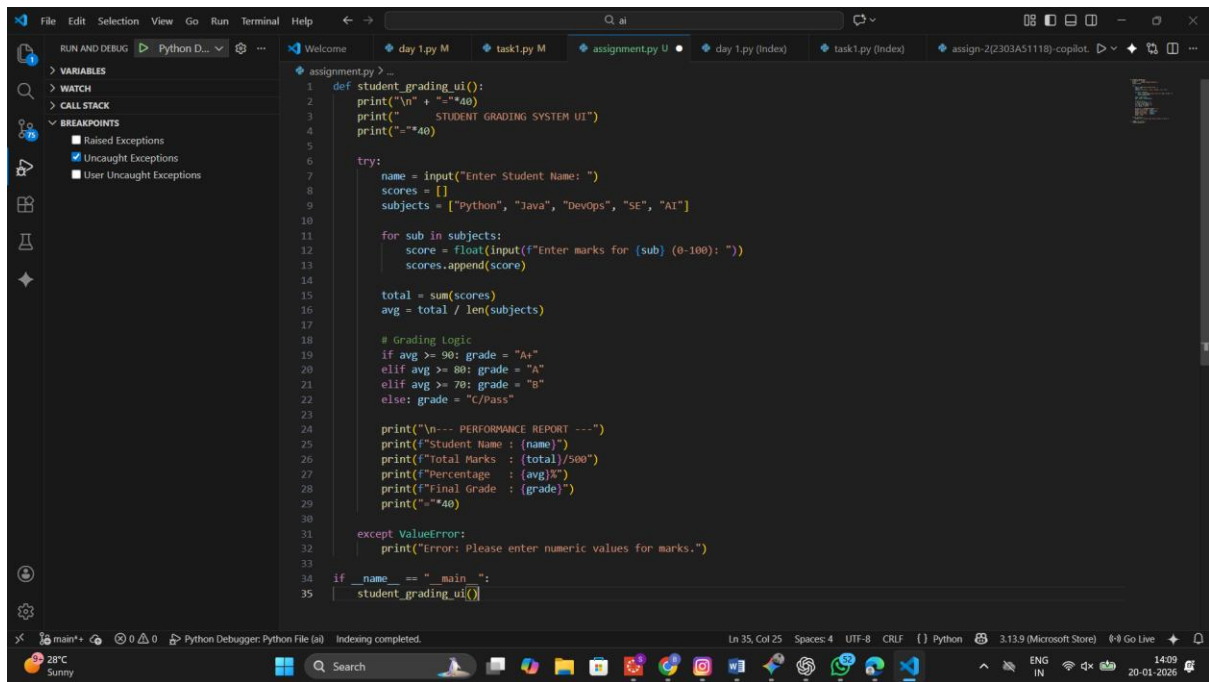
- Improved prime-checking function with better edge-case handling.

```
PS C:\Users\hp\OneDrive\Desktop\ai> c::; cd 'c:\Users\hp\OneDrive\Desktop\ai'; & 'c:\Users\hp\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\hp\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '52292' '--' 'c:\Users\hp\OneDrive\Desktop\ai\assignment.py'
Is -5 prime? False
Is 1 prime? False
Is 2 prime? True
Is 11 prime? True
Is 25 prime? False
PS C:\Users\hp\OneDrive\Desktop\ai>
```

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



```
1 def student_grading_ui():
2     print("\n" + "="*40)
3     print("      STUDENT GRADING SYSTEM UI")
4     print("="*40)
5
6     try:
7         name = input("Enter Student Name: ")
8         scores = []
9         subjects = ["Python", "Java", "DevOps", "SE", "AI"]
10
11         for sub in subjects:
12             score = float(input(f"Enter marks for {sub} (0-100): "))
13             scores.append(score)
14
15         total = sum(scores)
16         avg = total / len(subjects)
17
18         # Grading Logic
19         if avg >= 90: grade = "A+"
20         elif avg >= 80: grade = "A"
21         elif avg >= 70: grade = "B"
22         else: grade = "C/Pass"
23
24         print("\n--- PERFORMANCE REPORT ---")
25         print(f"Student Name : {name}")
26         print(f"Total Marks : {total}/500")
27         print(f"Percentage : {avg}%")
28         print(f"Final Grade : {grade}")
29         print("="*40)
30
31     except ValueError:
32         print("Error: Please enter numeric values for marks.")
33
34 if __name__ == "__main__":
35     student_grading_ui()
```

input.

Expected Output-4

- Well-structured UI code with accurate calculations and clear output display.



```
=====
      STUDENT GRADING SYSTEM UI
=====
Enter Student Name: Sai Tejaswi
Enter marks for Python (0-100): 99
Enter marks for Java (0-100): 97
Enter marks for DevOps (0-100): 96
Enter marks for SE (0-100): 99
Enter marks for AI (0-100): 97

--- PERFORMANCE REPORT ---
Student Name : Sai Tejaswi
Total Marks : 488.0/500
Percentage : 97.6%
Final Grade : A+
```

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.

```
assignment.py > ...
1 def unit_converter(value, mode):
2     """
3     Mode 1: KM to Miles
4     Mode 2: Miles to KM
5     """
6     KM_TO_MILE_FACTOR = 0.621371
7
8     if mode == 1:
9         result = value * KM_TO_MILE_FACTOR
10        return f"{value} KM = {result:.4f} Miles"
11    elif mode == 2:
12        result = value / KM_TO_MILE_FACTOR
13        return f"{value} Miles = {result:.4f} KM"
14    else:
15        return "Invalid Mode Selected"
16
17 print(unit_converter(10, 1)) # Precise Conversion
```

Expected Output-5

- Analysis of code quality and accuracy differences across multiple prompt variations.

```
PS C:\Users\hp\OneDrive\Desktop\ai> c:: cd 'c:\Users\hp\OneDrive\Desktop\ai'; & 'c:\Users\hp\AppData\Local\Microsoft\WindowsApps\python3.13.e
e' 'c:\Users\hp\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '63375' '--' 'C:\Users\hp\OneDrive\Des
top\ai\assignment.py'
10 KM = 6.2137 Miles
PS C:\Users\hp\OneDrive\Desktop\ai> 
```