ASSIGNMENT

Name: G.Sriram

Roll Number: 2303A51191

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.

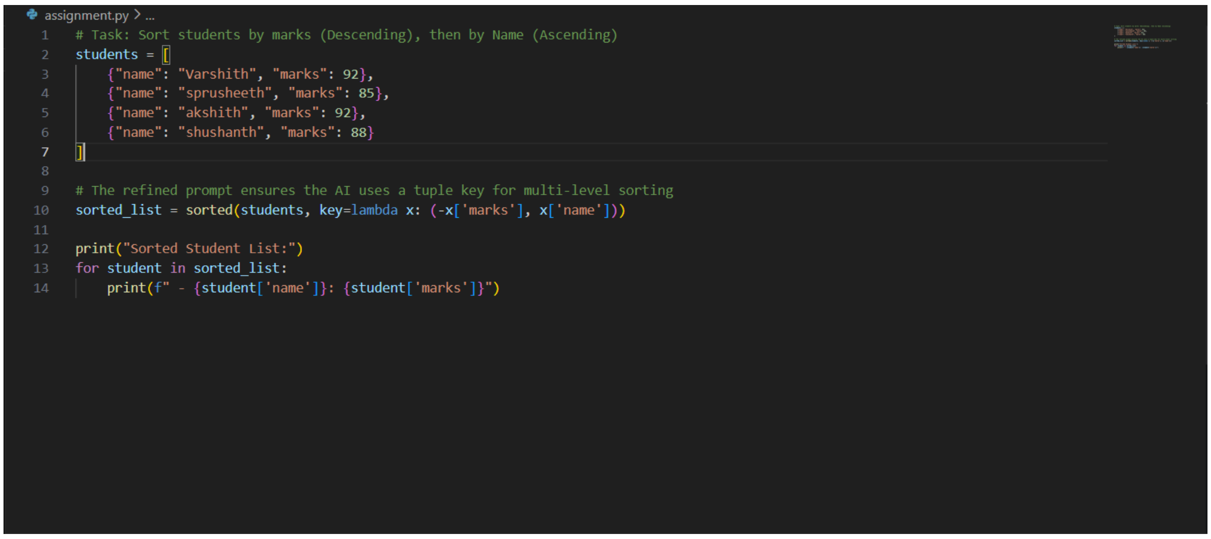
Expected Output-1

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

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.

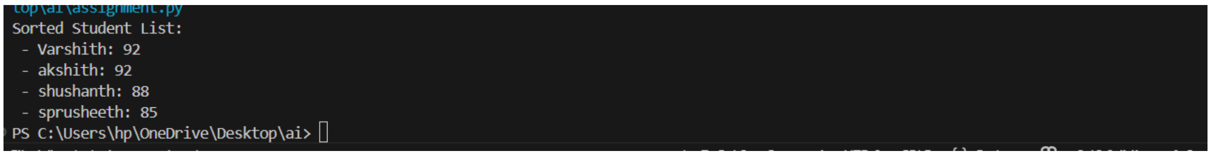


assignment.py > ...

```python
# Task: Sort students by marks (Descending), then by Name (Ascending)
students = [
    {"name": "Varshith", "marks": 92},
    {"name": "sprusheeth", "marks": 85},
    {"name": "akshith", "marks": 92},
    {"name": "shushanth", "marks": 88}
]

# The refined prompt ensures the AI uses a tuple key for multi-level sorting
sorted_list = sorted(students, key=lambda x: (-x['marks'], x['name']))

print("Sorted Student List:")
for student in sorted_list:
    print(f" - {student['name']}: {student['marks']}")
```

Expected Output-2

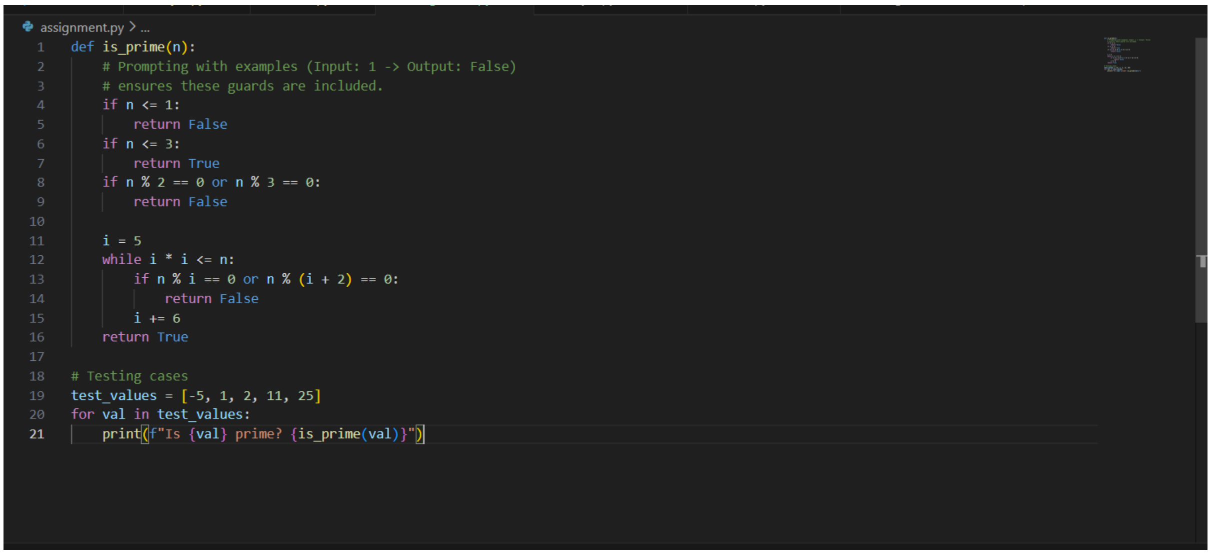
• AI-generated sorting function evolves from ambiguous logic to an accurate and efficient implementation.



```
top\ai\assignment.py
Sorted Student List:
 - Varshith: 92
 - akshith: 92
 - shushanth: 88
 - sprusheeth: 85
PS C:\Users\hp\OneDrive\Desktop\ai> 
```

Task Description-3

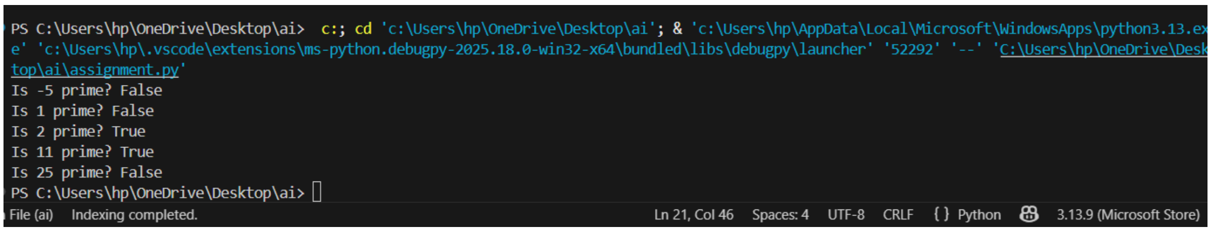
• Few-Shot Prompting for Prime Number Validation: Provide multiple input-output examples for a function that checks whether a number is prime. Observe how few-shot prompting improves correctness.



```python
def is_prime(n):
    # Prompting with examples (Input: 1 -> Output: False)
    # ensures these guards are included.
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False

    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True

# Testing cases
test_values = [-5, 1, 2, 11, 25]
for val in test_values:
    print(f"Is {val} prime? {is_prime(val)}")
```

Expected Output-3

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

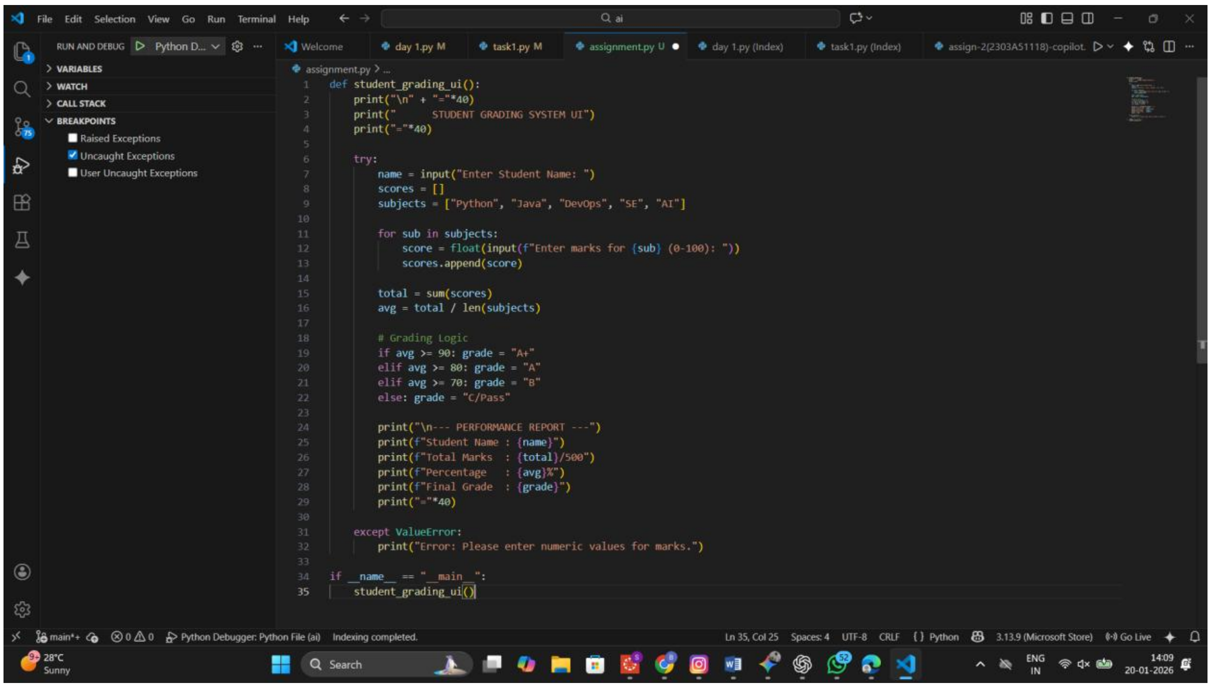


```
PS C:\Users\hp\OneDrive\Desktop\ai>  c:; cd 'c:\Users\hp\OneDrive\Desktop\ai'; & 'c:\Users\hp\AppData\Local\Microsoft\WindowsApps\python3.13.ex
e' 'c:\Users\hp\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '52292' '--' 'C:\Users\hp\OneDrive\Desk
top\ai\assignment.py'
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

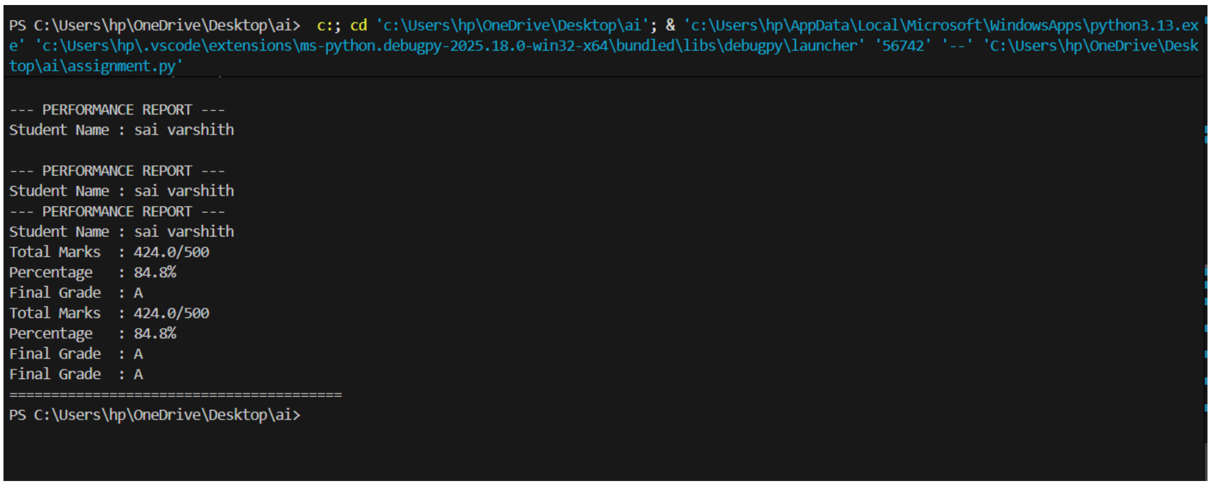
input.



```python
def student_grading_ui():
    print("\n" + "="*40)
    print("      STUDENT GRADING SYSTEM UI")
    print("="*40)

    try:
        name = input("Enter Student Name: ")
        scores = []
        subjects = ["Python", "Java", "DevOps", "SE", "AI"]

        for sub in subjects:
            score = float(input(f"Enter marks for {sub} (0-100): "))
            scores.append(score)

        total = sum(scores)
        avg = total / len(subjects)

        # Grading Logic
        if avg >= 90: grade = "A+"
        elif avg >= 80: grade = "A"
        elif avg >= 70: grade = "B"
        else: grade = "C/Pass"

        print("\n--- PERFORMANCE REPORT ---")
        print(f"Student Name : {name}")
        print(f"Total Marks  : {total}/500")
        print(f"Percentage   : {avg}%")
        print(f"Final Grade  : {grade}")
        print("="*40)

    except ValueError:
        print("Error: Please enter numeric values for marks.")

if __name__ == "__main__":
    student_grading_ui()
```

Expected Output-4

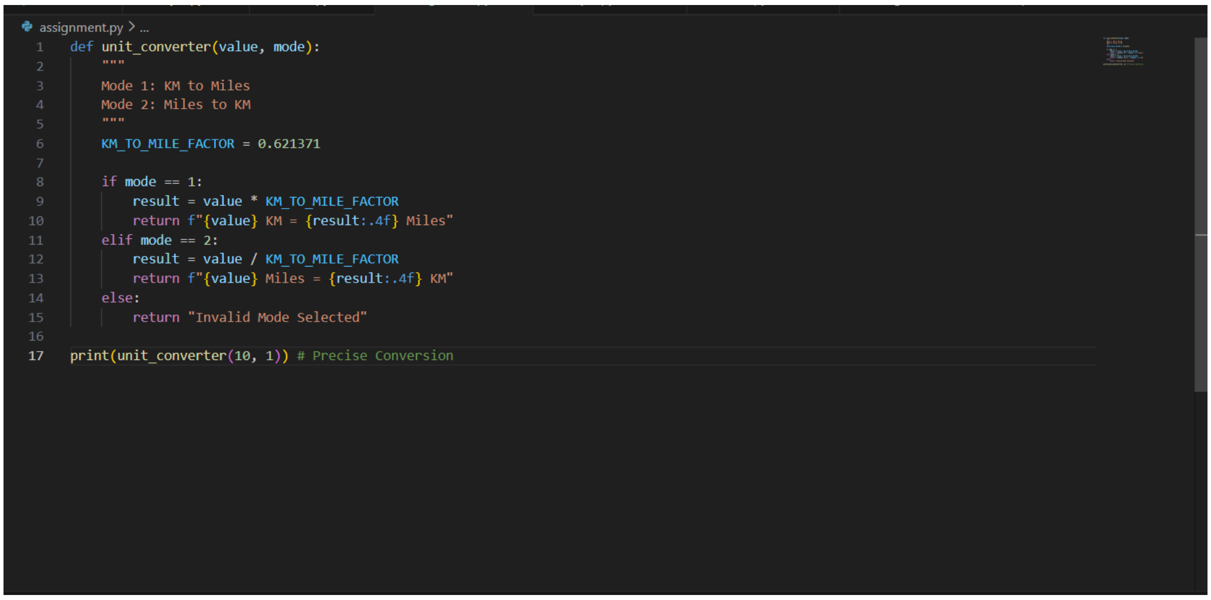
• Well-structured UI code with accurate calculations and clear output display.



```
PS C:\Users\hp\OneDrive\Desktop\ai>  c:; cd 'c:\Users\hp\OneDrive\Desktop\ai'; & 'c:\Users\hp\AppData\Local\Microsoft\WindowsApps\python3.13.ex
e' 'c:\Users\hp\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '56742' '--' 'C:\Users\hp\OneDrive\Desk
top\ai\assignment.py'

--- PERFORMANCE REPORT ---
Student Name : sai varshith

--- PERFORMANCE REPORT ---
Student Name : sai varshith
--- PERFORMANCE REPORT ---
Student Name : sai varshith
Total Marks  : 424.0/500
Percentage   : 84.8%
Final Grade  : A
Total Marks  : 424.0/500
Percentage   : 84.8%
Final Grade  : A
Final Grade  : A
========================================
PS C:\Users\hp\OneDrive\Desktop\ai>
```

Task Description-5

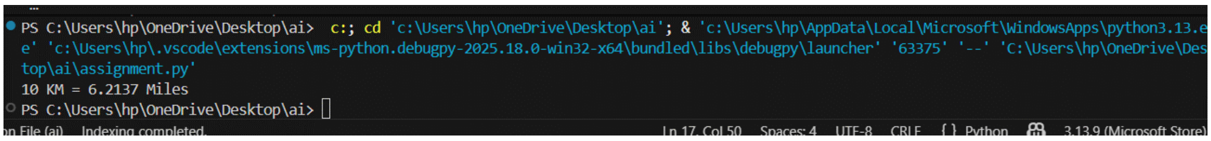
• Analyzing Prompt Specificity in Unit Conversion Functions: Improving a Unit Conversion Function (Kilometers to Miles and Miles to Kilometers) Using Clear Instructions.



```
assignment.py > ...
def unit_converter(value, mode):
    """
    Mode 1: KM to Miles
    Mode 2: Miles to KM
    """
    KM_TO_MILE_FACTOR = 0.621371

    if mode == 1:
        result = value * KM_TO_MILE_FACTOR
        return f"{value} KM = {result:.4f} Miles"
    elif mode == 2:
        result = value / KM_TO_MILE_FACTOR
        return f"{value} Miles = {result:.4f} KM"
    else:
        return "Invalid Mode Selected"

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> 
```

n File (ai)   Indexing completed.   Ln 17, Col 50   Spaces: 4   UTF-8   CRLF   {} Python   3.13.9 (Microsoft Store)