

AI Assisted Coding

Name : A.Anjali

Hall : 2303a52385

Batch : 33

Lab 3: Prompt Engineering – Improving Prompts and Context Management

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.

The screenshot displays an AI-assisted coding interface. The main editor shows a Python file named `assignment3.2.py` with the following code:

```
10 def calculator(a=None, b=None, op=None):
33     print("Invalid operator. Choose one of: + - * /")
34
35     if a is None:
36         a = _get_number("Enter the first number: ")
37     if b is None:
38         b = _get_number("Enter the second number: ")
39     if op is None:
40         op = _get_operator("Enter an operator (+ - * /): ")
41
42     try:
43         a = float(a)
44         b = float(b)
45     except Exception:
46         print("Error: Invalid numeric input.")
47         return None
48
49     if op == '+':
50         result = a + b
51     elif op == '-':
52         result = a - b
53     elif op == '*':
54         result = a * b
55     elif op == '/':
```

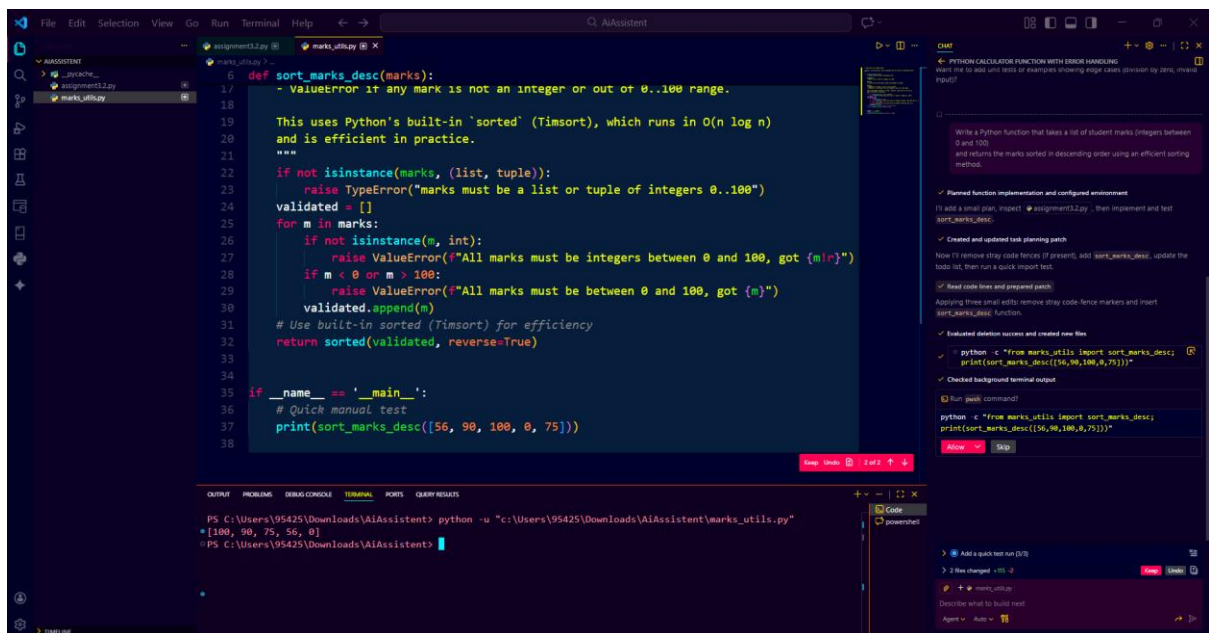
The terminal window at the bottom shows the execution of the program:

```
PS C:\Users\95425\Downloads\AiAssistant> python -u "c:\Users\95425\Downloads\AiAssistant\assignment3.2.py"
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadline for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadline'.
Enter the first number: 15
Enter the second number: 5
Enter an operator (+ - * /): +
Result: 15.0 + 5.0 = 20
PS C:\Users\95425\Downloads\AiAssistant>
```

The right sidebar shows a chat window with the AI assistant. The prompt is: "Create a Python function named calculator that takes two numbers and an operator (+, -, *, /) from the user, performs the selected operation, handles division by zero, and displays the result clearly." The assistant's response includes a todo list and the implementation of the `calculator` function.

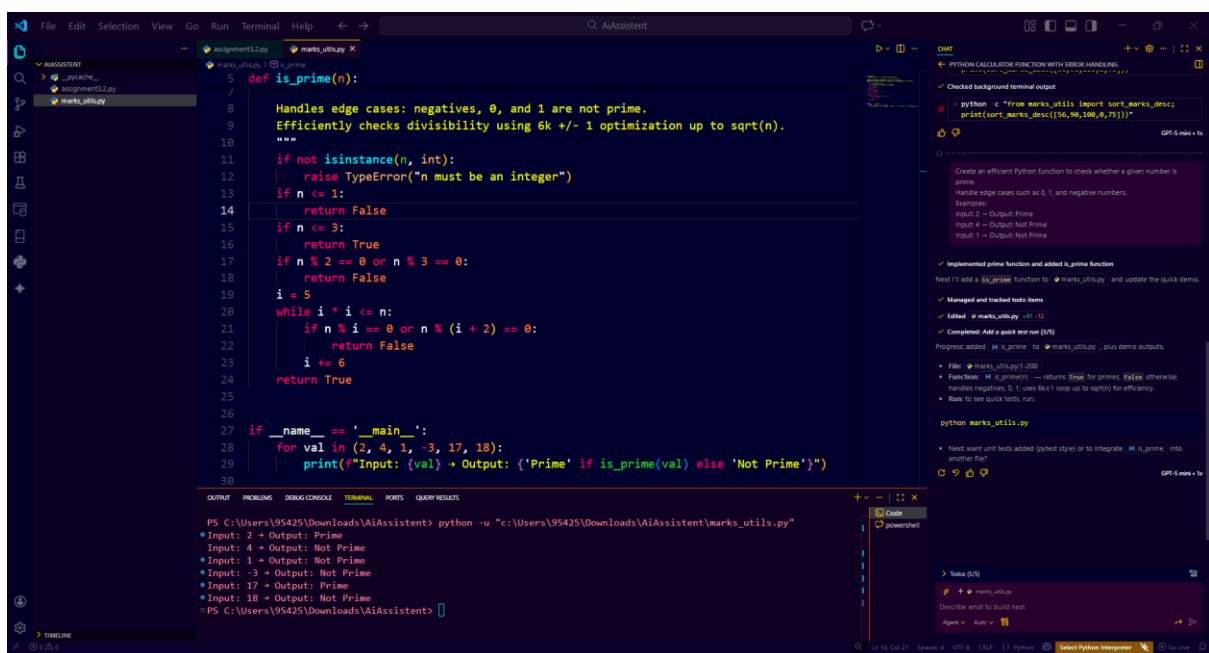
Task Description-2

- Refining Prompts for Sorting Logic: Start with a vague prompt for sorting student marks, then refine it to clearly specify sorting order and constraints.



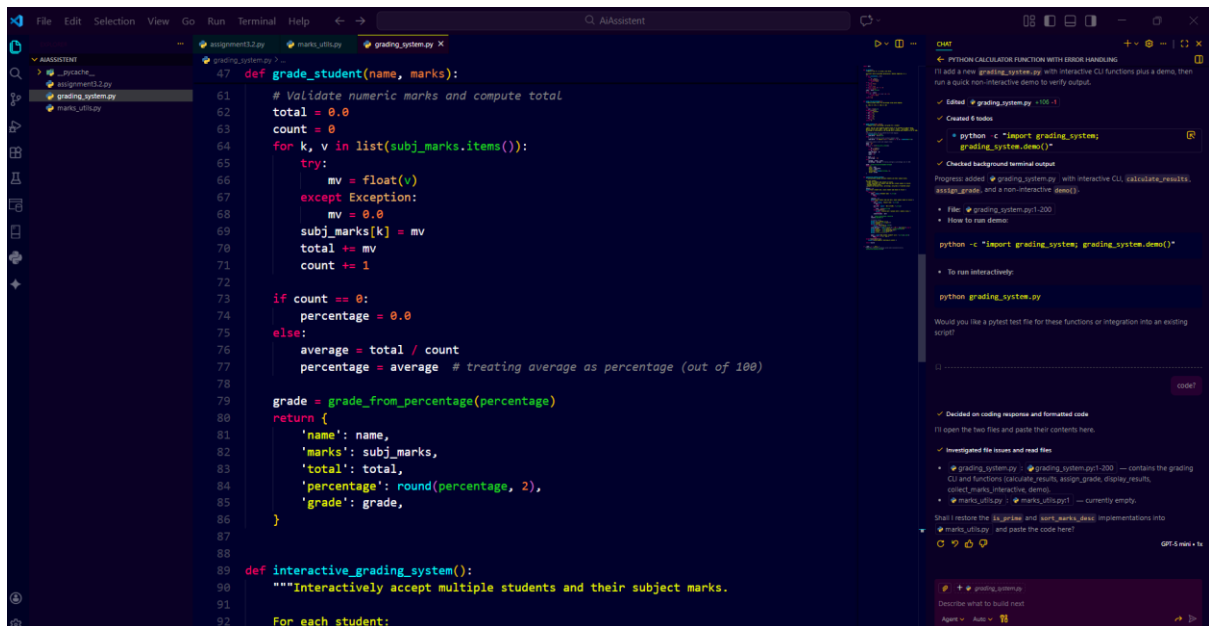
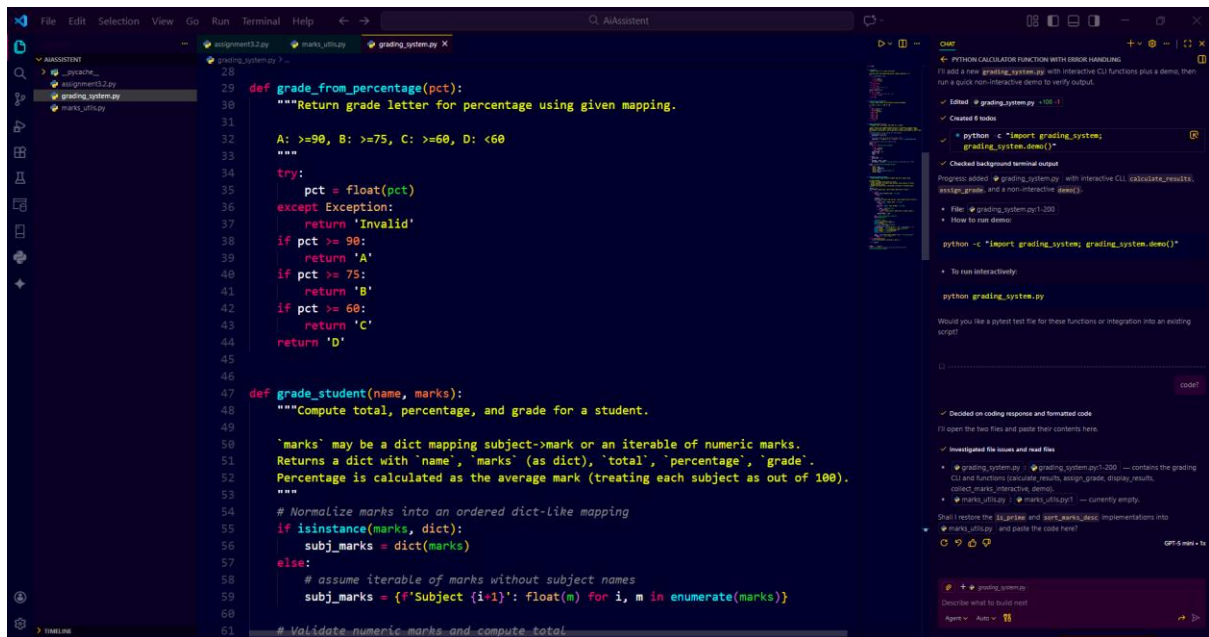
Task Description-3

- 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.



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.



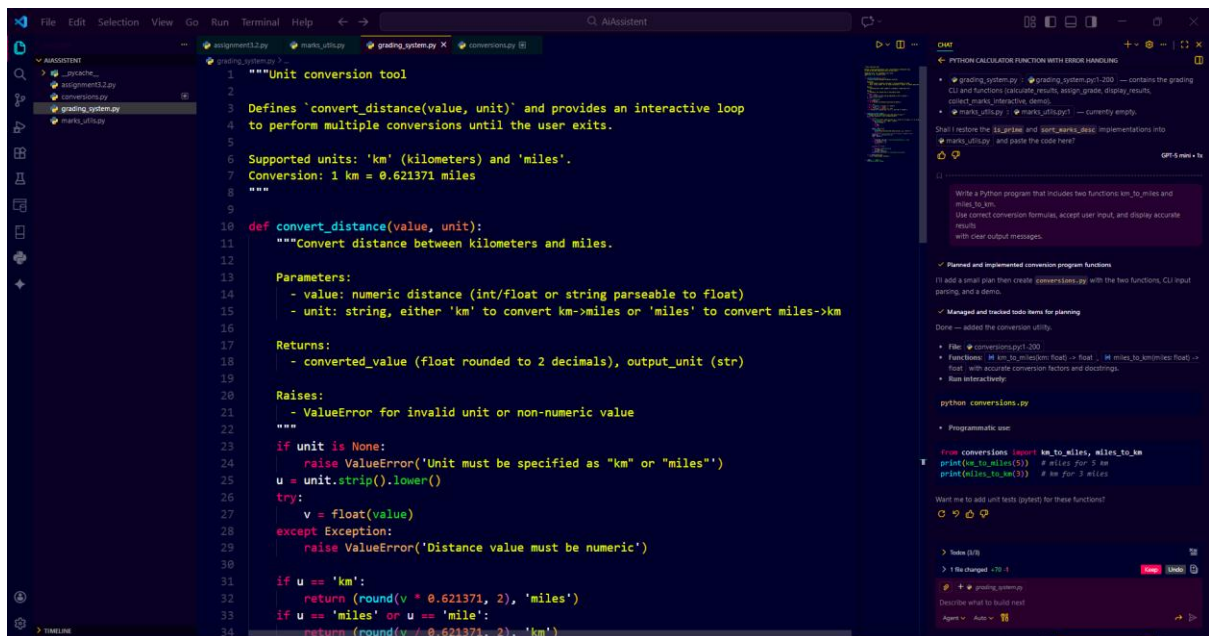
```
File Edit Selection View Go Run Terminal Help
assignment2.py marks_utils.py grading_system.py X
90 def interactive_grading_system():
91     """Interactively accept multiple students and their subject marks.
92
93     For each student:
94     - Prompt for student name (blank to finish)
95     - Prompt repeatedly for subject name and mark (blank subject to finish)
96     - Validate numeric marks
97     - Compute and display total, percentage, and grade in formatted output
98     """
99     results = []
100     print('Enter student data. Leave student name blank to finish.')
101     try:
102         while True:
103             name = input('\nStudent name: ').strip()
104             if name == '':
105                 break
106
107             subjects = {}
108             print('Enter subject name and mark. Leave subject blank to finish.')
109             while True:
110                 subj = input(' Subject name: ').strip()
111                 if subj == '':
112                     break
113
114                 mark_raw = input(' Mark (0-100): ').strip()
115                 try:
116                     mark = float(mark_raw)
117                 except ValueError:
118                     print(' Invalid mark - please enter a numeric value.')
119                     continue
120                 subjects[subj] = mark
121
122             rec = grade_student(name, subjects)
123             results.append(rec)
124
125     except KeyboardInterrupt:
126         print('\nInterrupted - returning to caller.')
127     return results
128
129 if __name__ == '__main__':
130     # Run the interactive grading system when executed directly.
131     interactive_grading_system()
```

```
File Edit Selection View Go Run Terminal Help
assignment2.py marks_utils.py grading_system.py X
125     print('\n--- Result ----')
126     print(f'Student: {rec["name"]}')
127     print('Subjects:')
128     for s, m in rec['marks'].items():
129         print(f'    {s}: {m}')
130     total_possible = len(rec['marks']) * 100 if rec['marks'] else 0
131     print(f'Total : {rec["total"]} / {total_possible}')
132     print(f'Percent : {rec["percentage"]} %')
133     print(f'Grade : {rec["grade"]}')
134     print('-----')
135
136     again = input('Add another student? (y/n): ').strip().lower()
137     if again not in ('y', 'yes'):
138         break
139
140 except KeyboardInterrupt:
141     print('\nInterrupted - returning to caller.')
142
143 return results
144
145 if __name__ == '__main__':
146     # Run the interactive grading system when executed directly.
147     interactive_grading_system()
```

```
File Edit Selection View Go Run Terminal Help
assignment2.py marks_utils.py grading_system.py X
1 import math
2
3 OUTPUT MODULES DEBUG CONSOLE OUTPUTS PLOTS QUIT RESULTS
PS C:\Users\Y5425\Downloads\AIAssistant> python -u "C:\Users\Y5425\Downloads\AIAssistant\grading_system.py"
Enter student data. Leave student name blank to finish.
Student name: Anjali
Enter subject name and mark. Leave subject blank to finish.
Subject name: Maths
Mark (0-100): 98
Subject name: Physics
Mark (0-100): 89
Subject name: Chemistry
Mark (0-100): 97
Subject name:
--- Result ----
Student: Anjali
Subjects:
Maths: 98.0
Physics: 89.0
Chemistry: 97.0
Total : 285.0 / 300
Percent : 95.0%
Grade : A
Add another student? (y/n): n
PS C:\Users\Y5425\Downloads\AIAssistant>
```

Task Description-5

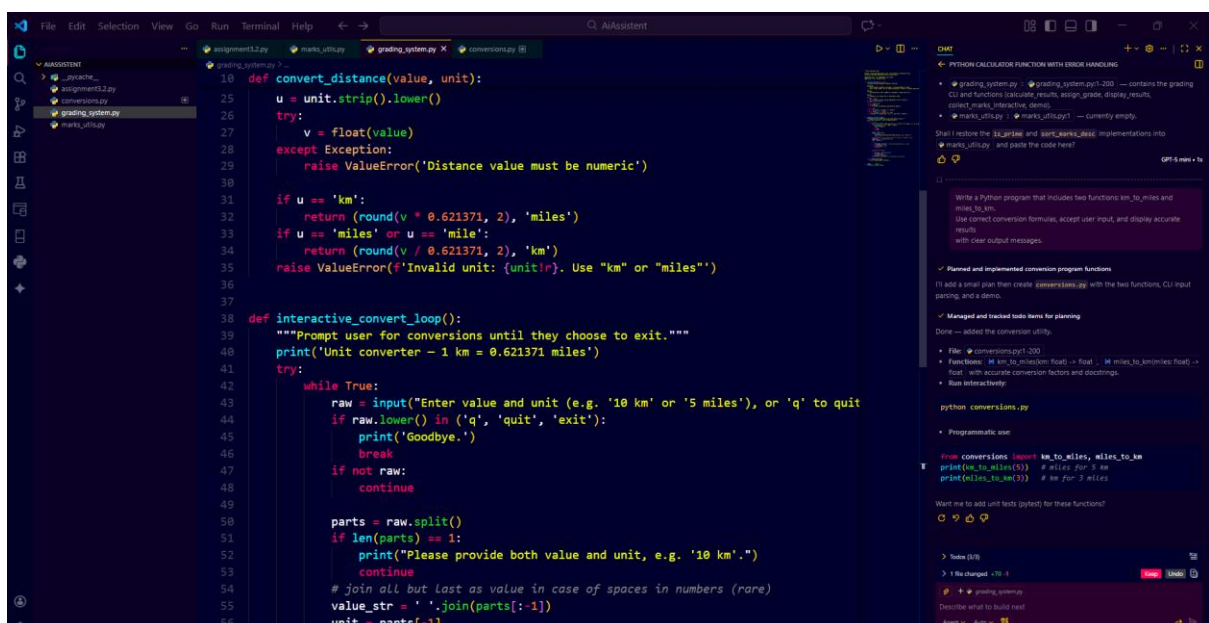
- Analyzing Prompt Specificity in Unit Conversion Functions: Improving a Unit Conversion Function (Kilometers to Miles and Miles to Kilometers) Using Clear Instructions.



```
1 """Unit conversion tool
2
3 Defines 'convert_distance(value, unit)' and provides an interactive loop
4 to perform multiple conversions until the user exits.
5
6 Supported units: 'km' (kilometers) and 'miles'.
7 Conversion: 1 km = 0.621371 miles
8 """
9
10 def convert_distance(value, unit):
11     """Convert distance between kilometers and miles.
12
13     Parameters:
14     - value: numeric distance (int/float or string parseable to float)
15     - unit: string, either 'km' to convert km->miles or 'miles' to convert miles->km
16
17     Returns:
18     - converted_value (float rounded to 2 decimals), output_unit (str)
19
20     Raises:
21     - ValueError for invalid unit or non-numeric value
22     """
23     if unit is None:
24         raise ValueError('Unit must be specified as "km" or "miles"')
25     u = unit.strip().lower()
26     try:
27         v = float(value)
28     except Exception:
29         raise ValueError('Distance value must be numeric')
30
31     if u == 'km':
32         return (round(v * 0.621371, 2), 'miles')
33     if u == 'miles' or u == 'mile':
34         return (round(v / 0.621371, 2), 'km')
```

AIAssistant suggestions:

- Write a Python program that includes two functions `km_to_miles` and `miles_to_km`. Use correct conversion formulas, accept user input, and display accurate results with clear output messages.
- Planned and implemented conversion program functions
- Managed and tracked todo items for planning
- Done — added the conversion utility
- File: `conversions.py`
- Functions: `km_to_miles(km: float) -> float`, `miles_to_km(miles: float) -> float` with accurate conversion factors and docstrings.
- Run interactively
- python conversions.py
- Programmatic use
- from conversions import km_to_miles, miles_to_km
- print(km_to_miles(5)) # miles for 5 km
- print(miles_to_km(3)) # km for 3 miles
- Want me to add unit tests (pytest) for these functions?
- Take (10)
- 1 file changed +10 -1
- 4 # grading strategy
- Describe what to build next
- Again -> Auto ->



```
10 def convert_distance(value, unit):
11     u = unit.strip().lower()
12     try:
13         v = float(value)
14     except Exception:
15         raise ValueError('Distance value must be numeric')
16
17     if u == 'km':
18         return (round(v * 0.621371, 2), 'miles')
19     if u == 'miles' or u == 'mile':
20         return (round(v / 0.621371, 2), 'km')
21     raise ValueError(f'Invalid unit: {unit}. Use "km" or "miles"')
22
23 def interactive_convert_loop():
24     """Prompt user for conversions until they choose to exit."""
25     print('Unit converter - 1 km = 0.621371 miles')
26     try:
27         while True:
28             raw = input("Enter value and unit (e.g. '10 km' or '5 miles'), or 'q' to quit\n")
29             if raw.lower() in ('q', 'quit', 'exit'):
30                 print('Goodbye.')
31                 break
32             if not raw:
33                 continue
34
35             parts = raw.split()
36             if len(parts) == 1:
37                 print("Please provide both value and unit, e.g. '10 km'.")
38                 continue
39             # join all but last as value in case of spaces in numbers (rare)
40             value_str = ' '.join(parts[:-1])
41             unit = parts[-1]
```

AIAssistant suggestions:

- Write a Python program that includes two functions `km_to_miles` and `miles_to_km`. Use correct conversion formulas, accept user input, and display accurate results with clear output messages.
- Planned and implemented conversion program functions
- Managed and tracked todo items for planning
- Done — added the conversion utility
- File: `conversions.py`
- Functions: `km_to_miles(km: float) -> float`, `miles_to_km(miles: float) -> float` with accurate conversion factors and docstrings.
- Run interactively
- python conversions.py
- Programmatic use
- from conversions import km_to_miles, miles_to_km
- print(km_to_miles(5)) # miles for 5 km
- print(miles_to_km(3)) # km for 3 miles
- Want me to add unit tests (pytest) for these functions?
- Take (10)
- 1 file changed +10 -1
- 4 # grading strategy
- Describe what to build next
- Again -> Auto ->

