AI Assisted Coding

Name :K.Srija

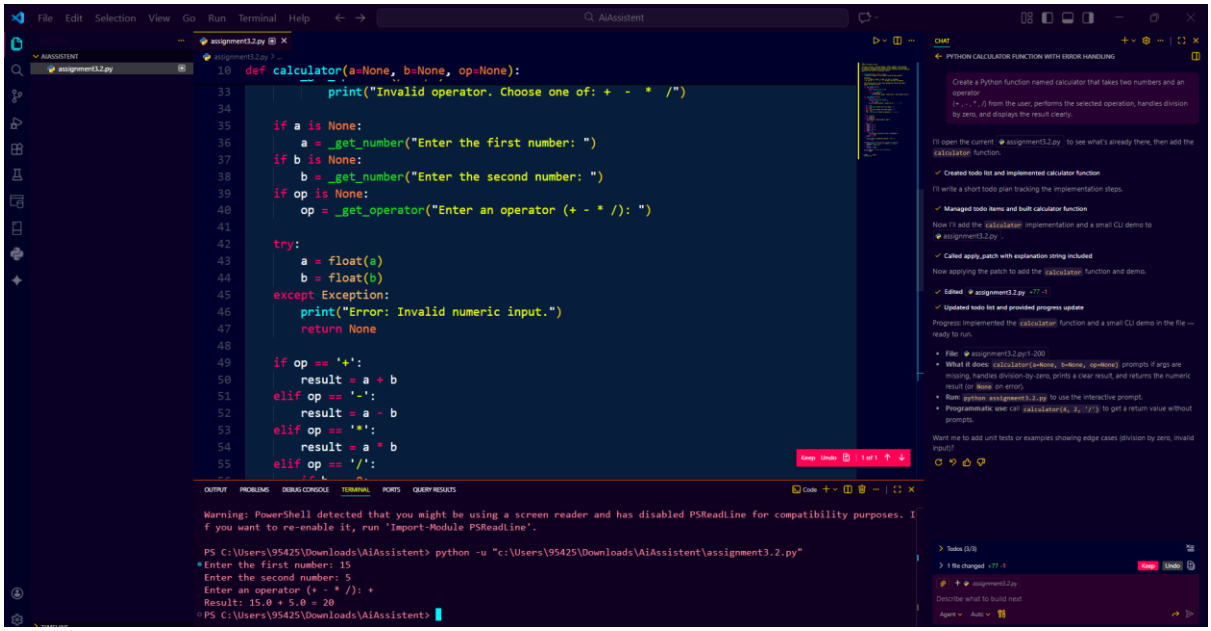
Hall : 2303a52189

Batch : 34

Lab 3: Prompt Engineering – Improving Prompts and Context Management

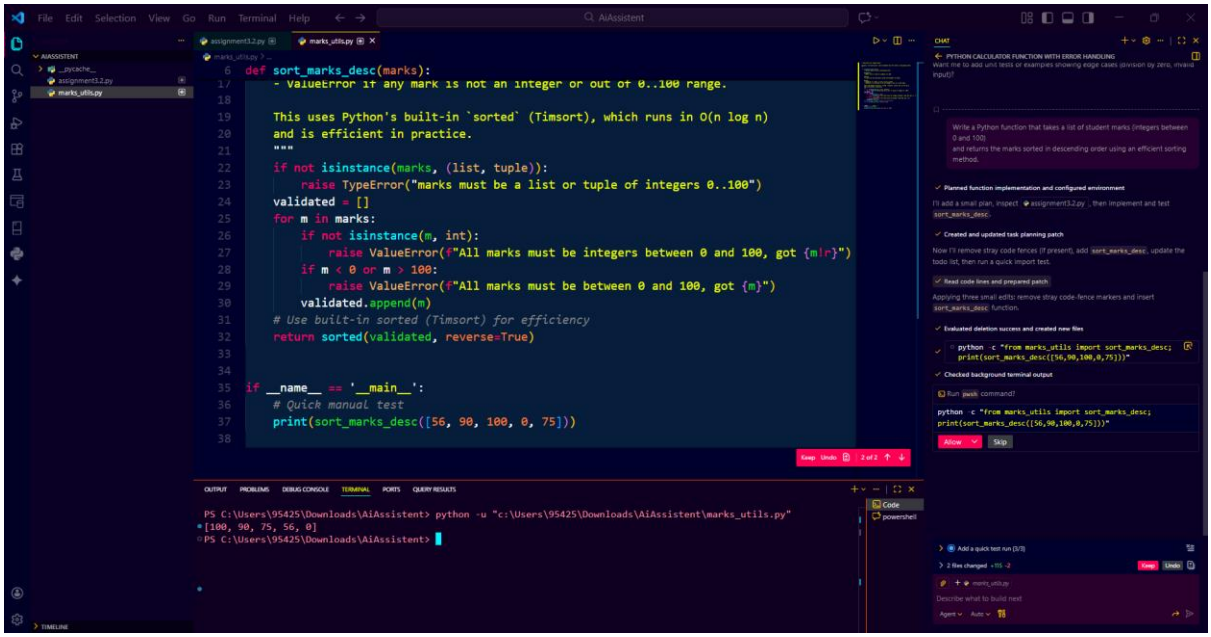
Task Description-1

• 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.



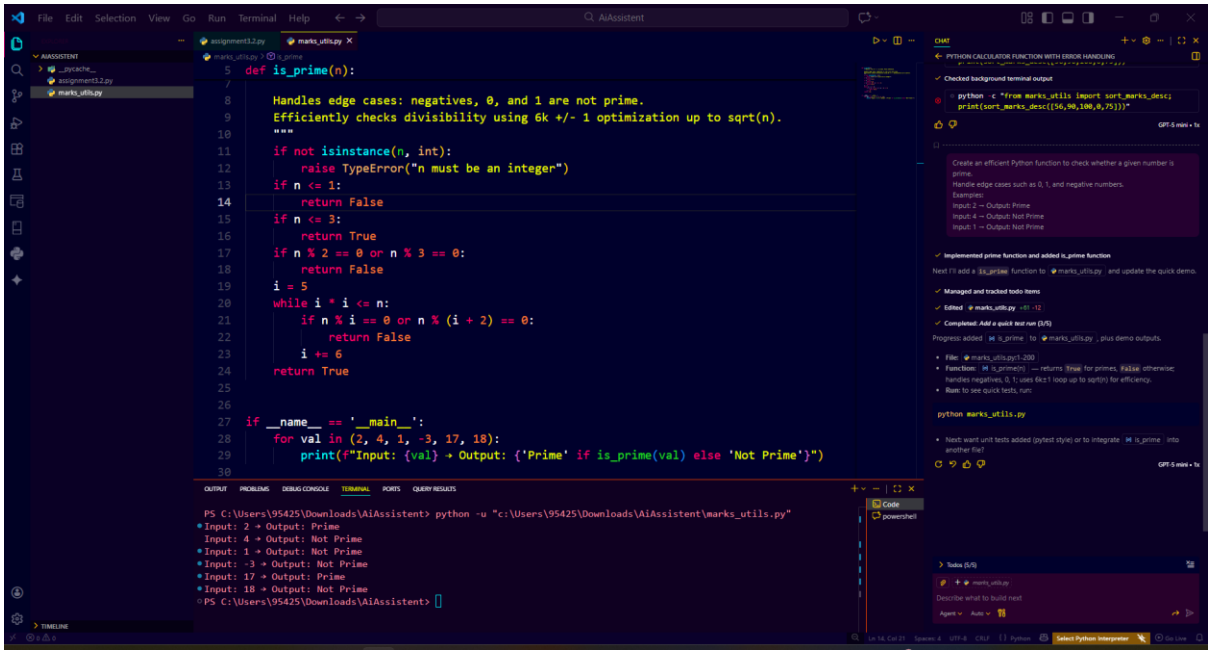
Task Description-2

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



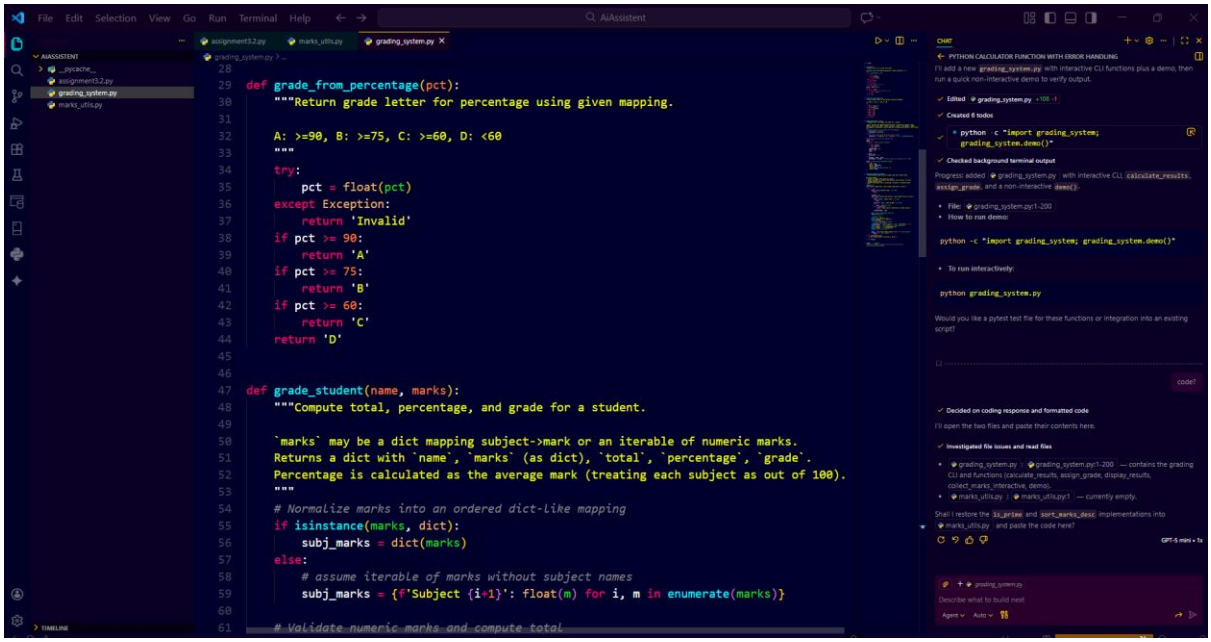
Task Description-3

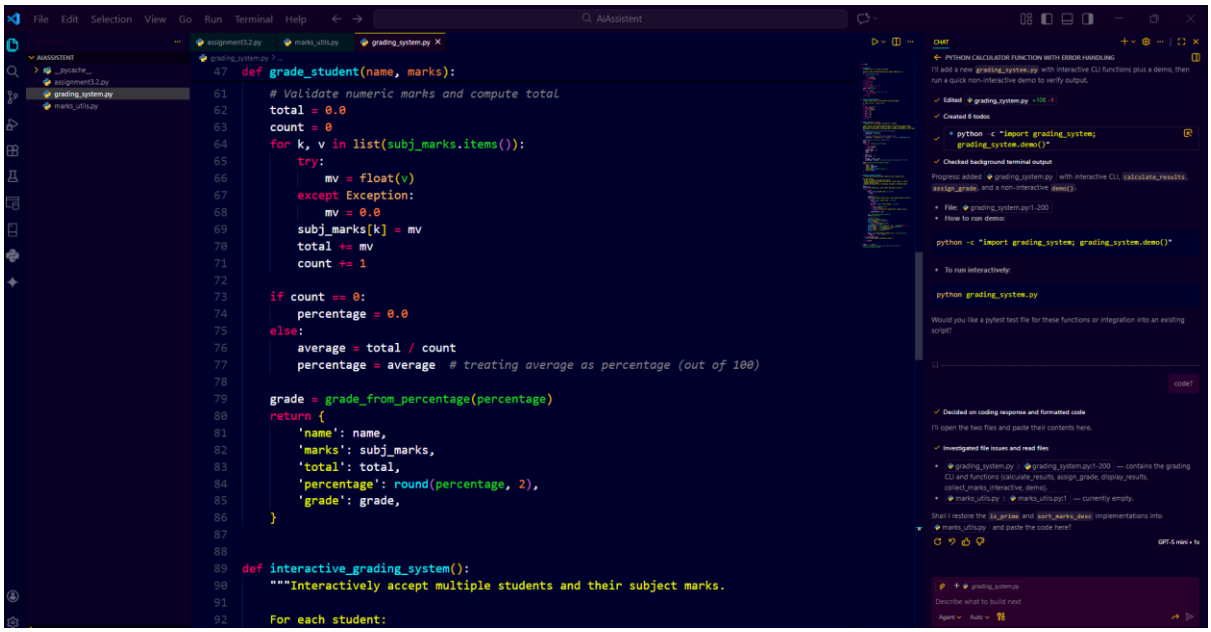
• Few-Shot Prompting for Prime Number Validation: Provide multiple input-output examples for a function that checks whether a number is prime. Observe how few-shot prompting improves correctness.

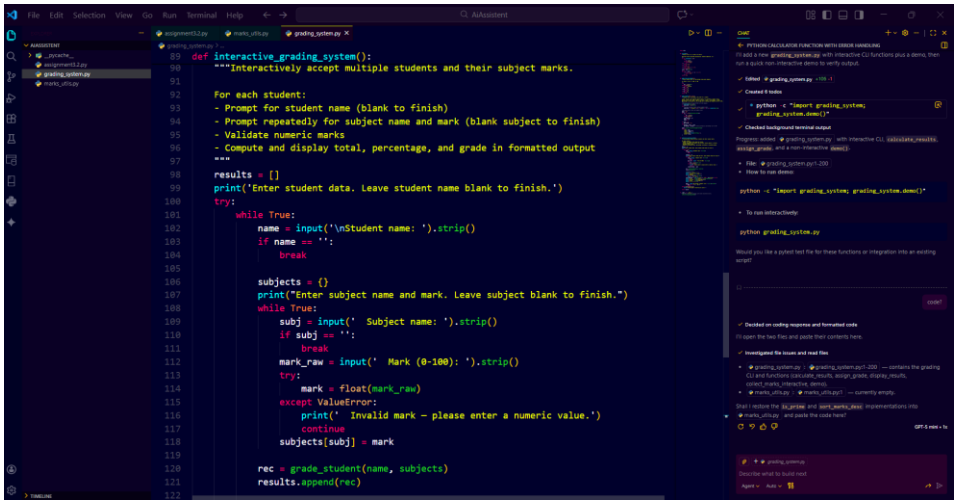


Task Description-4

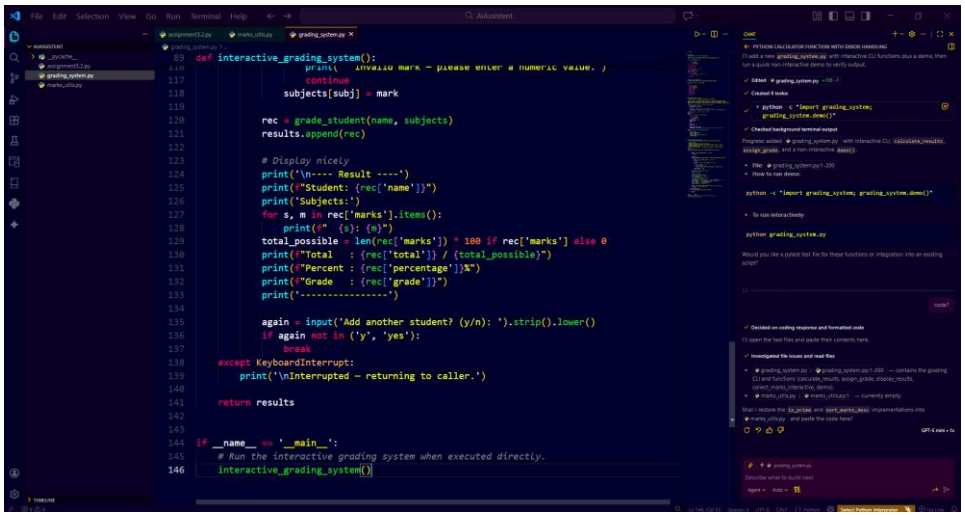
• Prompt-Guided UI Design for Student Grading System: Create a user interface for a student grading system that calculates total marks, percentage, and grade based on user input.


def grade_from_percentage(pct):
"""Return grade letter for percentage using given mapping.
A: >=90, B: >=75, C: >=60, D: <60
"""
try:
pct = float(pct)
except Exception:
return 'Invalid'
if pct >= 90:
return 'A'
if pct >= 75:
return 'B'
if pct >= 60:
return 'C'
return 'D'
def grade_student(name, marks):
"""Compute total, percentage, and grade for a student.
`marks` may be a dict mapping subject->mark or an iterable of numeric marks.
Returns a dict with `name`, `marks` (as dict), `total`, `percentage`, `grade`.
Percentage is calculated as the average mark (treating each subject as out of 100).
"""
# Normalize marks into an ordered dict-like mapping
if isinstance(marks, dict):
subj_marks = dict(marks)
else:
# assume iterable of marks without subject names
subj_marks = {f'Subject {i+1}': float(m) for i, m in enumerate(marks)}
# Validate numeric marks and compute total

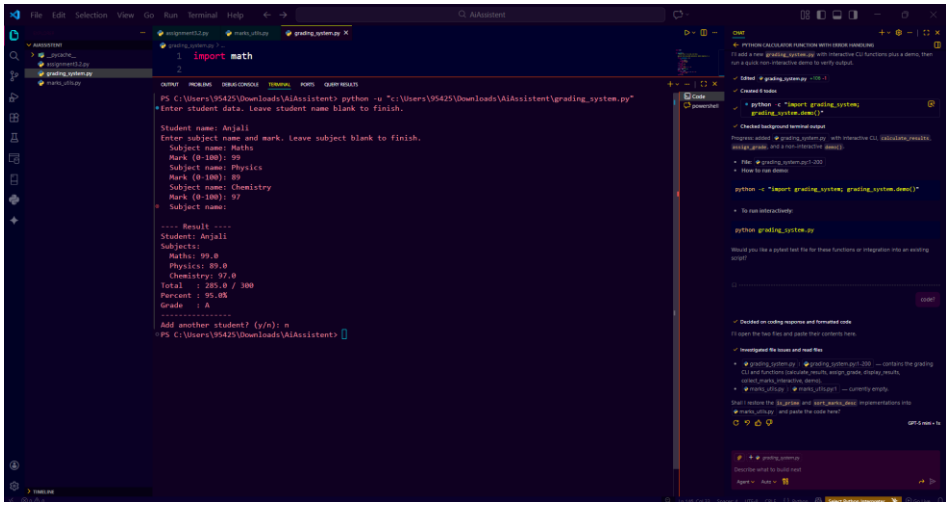

def grade_student(name, marks):
# Validate numeric marks and compute total
total = 0.0
count = 0
for k, v in list(subj_marks.items()):
try:
mv = float(v)
except Exception:
mv = 0.0
subj_marks[k] = mv
total += mv
count += 1
if count == 0:
percentage = 0.0
else:
average = total / count
percentage = average # treating average as percentage (out of 100)
grade = grade_from_percentage(percentage)
return {
'name': name,
'marks': subj_marks,
'total': total,
'percentage': round(percentage, 2),
'grade': grade,
}
def interactive_grading_system():
"""Interactively accept multiple students and their subject marks.
For each student:



```python
def interactive_grading_system():
    """Interactively accept multiple students and their subject marks.

    For each student:
    - Prompt for student name (blank to finish)
    - Prompt repeatedly for subject name and mark (blank subject to finish)
    - Validate numeric marks
    - Compute and display total, percentage, and grade in formatted output
    """
    results = []
    print('Enter student data. Leave student name blank to finish.')
    try:
        while True:
            name = input('\nStudent name: ').strip()
            if name == '':
                break

            subjects = {}
            print("Enter subject name and mark. Leave subject blank to finish.")
            while True:
                subj = input('  Subject name: ').strip()
                if subj == '':
                    break
                mark_raw = input('  Mark (0-100): ').strip()
                try:
                    mark = float(mark_raw)
                except ValueError:
                    print('  Invalid mark — please enter a numeric value.')
                    continue
                subjects[subj] = mark

            rec = grade_student(name, subjects)
            results.append(rec)
```



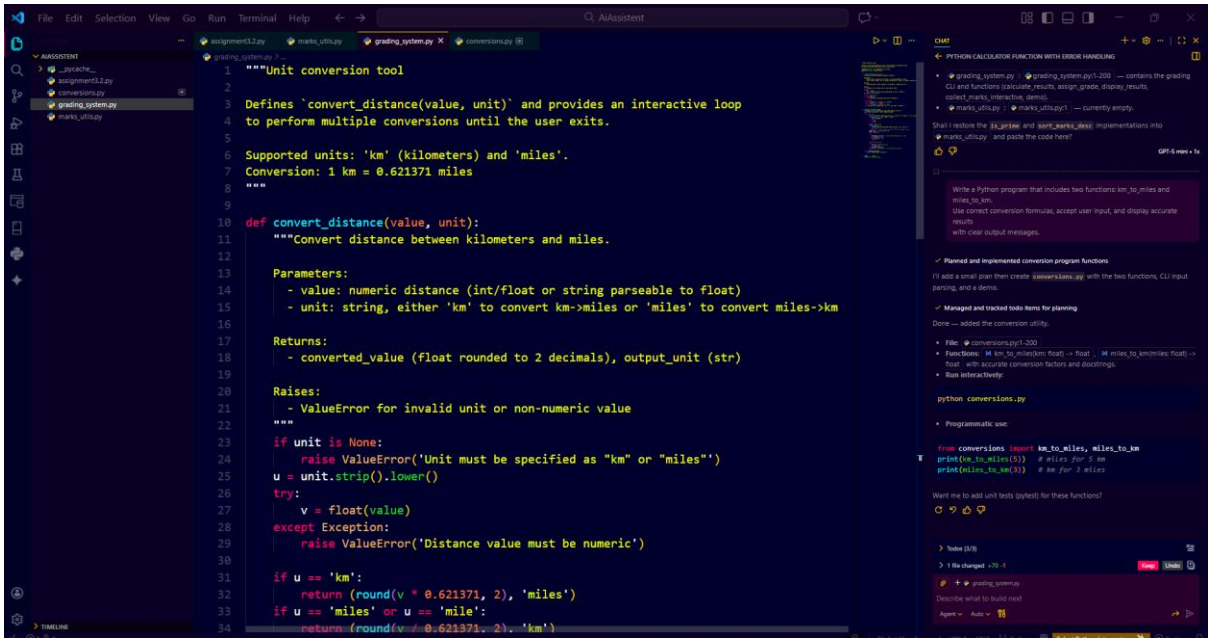
```python
def interactive_grading_system():
                    print('  Invalid mark — please enter a numeric value.')
                    continue
                subjects[subj] = mark

            rec = grade_student(name, subjects)
            results.append(rec)

            # Display nicely
            print('\n---- Result ----')
            print(f"Student: {rec['name']}")
            print('Subjects:')
            for s, m in rec['marks'].items():
                print(f"  {s}: {m}")
            total_possible = len(rec['marks']) * 100 if rec['marks'] else 0
            print(f"Total   : {rec['total']} / {total_possible}")
            print(f"Percent : {rec['percentage']}%")
            print(f"Grade   : {rec['grade']}")
            print('----------------')

            again = input('Add another student? (y/n): ').strip().lower()
            if again not in ('y', 'yes'):
                break
    except KeyboardInterrupt:
        print('\nInterrupted — returning to caller.')

    return results


if __name__ == '__main__':
    # Run the interactive grading system when executed directly.
    interactive_grading_system()
```



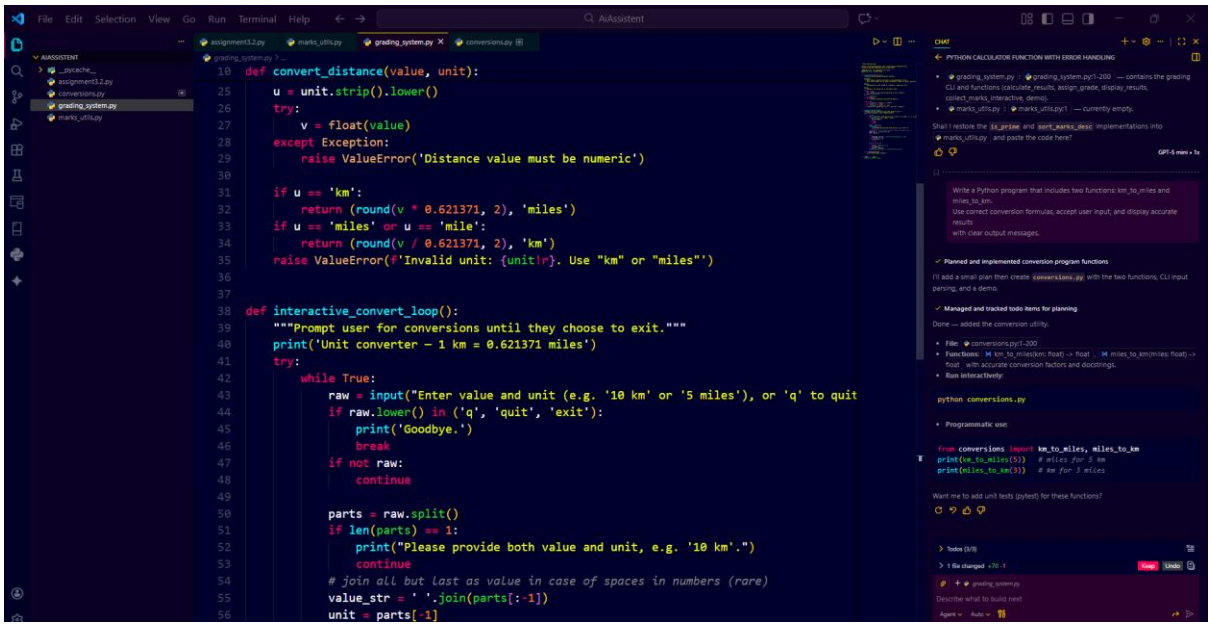
```
import math

PS C:\Users\95425\Downloads\AiAssistent> python -u "c:\Users\95425\Downloads\AiAssistent\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): 99
  Subject name: Physics
  Mark (0-100): 89
  Subject name: Chemistry
  Mark (0-100): 97
  Subject name:

---- Result ----
Student: Anjali
Subjects:
  Maths: 99.0
  Physics: 89.0
  Chemistry: 97.0
Total   : 285.0 / 300
Percent : 95.0%
Grade   : A
----------------
Add another student? (y/n): n
PS C:\Users\95425\Downloads\AiAssistent>
```

Task Description-5

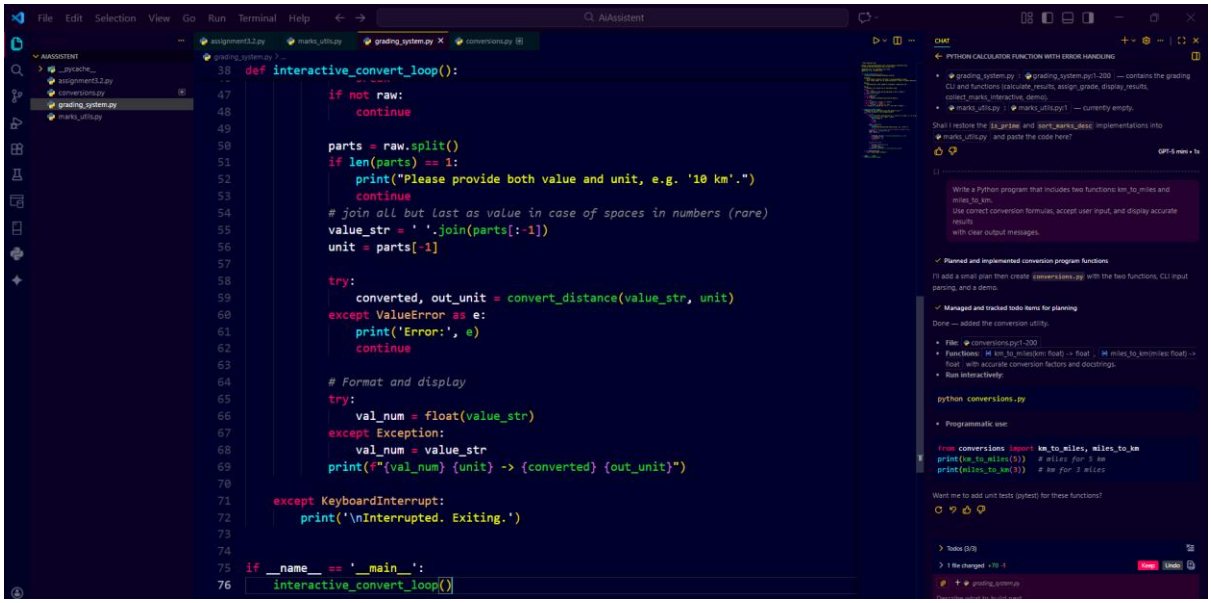
• Analyzing Prompt Specificity in Unit Conversion Functions: Improving a Unit Conversion Function (Kilometers to Miles and Miles to Kilometers) Using Clear Instructions.



```
"""Unit conversion tool

Defines `convert_distance(value, unit)` and provides an interactive loop
to perform multiple conversions until the user exits.

Supported units: 'km' (kilometers) and 'miles'.
Conversion: 1 km = 0.621371 miles
"""

def convert_distance(value, unit):
    """Convert distance between kilometers and miles.

    Parameters:
      - value: numeric distance (int/float or string parseable to float)
      - unit: string, either 'km' to convert km->miles or 'miles' to convert miles->km

    Returns:
      - converted_value (float rounded to 2 decimals), output_unit (str)

    Raises:
      - ValueError for invalid unit or non-numeric value
    """
    if unit is None:
        raise ValueError('Unit must be specified as "km" or "miles"')
    u = unit.strip().lower()
    try:
        v = float(value)
    except Exception:
        raise ValueError('Distance value must be numeric')

    if u == 'km':
        return (round(v * 0.621371, 2), 'miles')
    if u == 'miles' or u == 'mile':
        return (round(v / 0.621371, 2), 'km')
```



```
def convert_distance(value, unit):
    u = unit.strip().lower()
    try:
        v = float(value)
    except Exception:
        raise ValueError('Distance value must be numeric')

    if u == 'km':
        return (round(v * 0.621371, 2), 'miles')
    if u == 'miles' or u == 'mile':
        return (round(v / 0.621371, 2), 'km')
    raise ValueError(f'Invalid unit: {unit!r}. Use "km" or "miles"')


def interactive_convert_loop():
    """Prompt user for conversions until they choose to exit."""
    print('Unit converter — 1 km = 0.621371 miles')
    try:
        while True:
            raw = input("Enter value and unit (e.g. '10 km' or '5 miles'), or 'q' to quit
            if raw.lower() in ('q', 'quit', 'exit'):
                print('Goodbye.')
                break
            if not raw:
                continue

            parts = raw.split()
            if len(parts) == 1:
                print("Please provide both value and unit, e.g. '10 km'.")
                continue
            # join all but last as value in case of spaces in numbers (rare)
            value_str = ' '.join(parts[:-1])
            unit = parts[-1]
```


def interactive_convert_loop():
if not raw:
continue
parts = raw.split()
if len(parts) == 1:
print("Please provide both value and unit, e.g. '10 km'.")
continue
# join all but last as value in case of spaces in numbers (rare)
value_str = ' '.join(parts[:-1])
unit = parts[-1]
try:
converted, out_unit = convert_distance(value_str, unit)
except ValueError as e:
print('Error:', e)
continue
# Format and display
try:
val_num = float(value_str)
except Exception:
val_num = value_str
print(f"{val_num} {unit} -> {converted} {out_unit}")
except KeyboardInterrupt:
print('\nInterrupted. Exiting.')
if __name__ == '__main__':
interactive_convert_loop()