AI Assisted Coding

Name : B.Thanmai

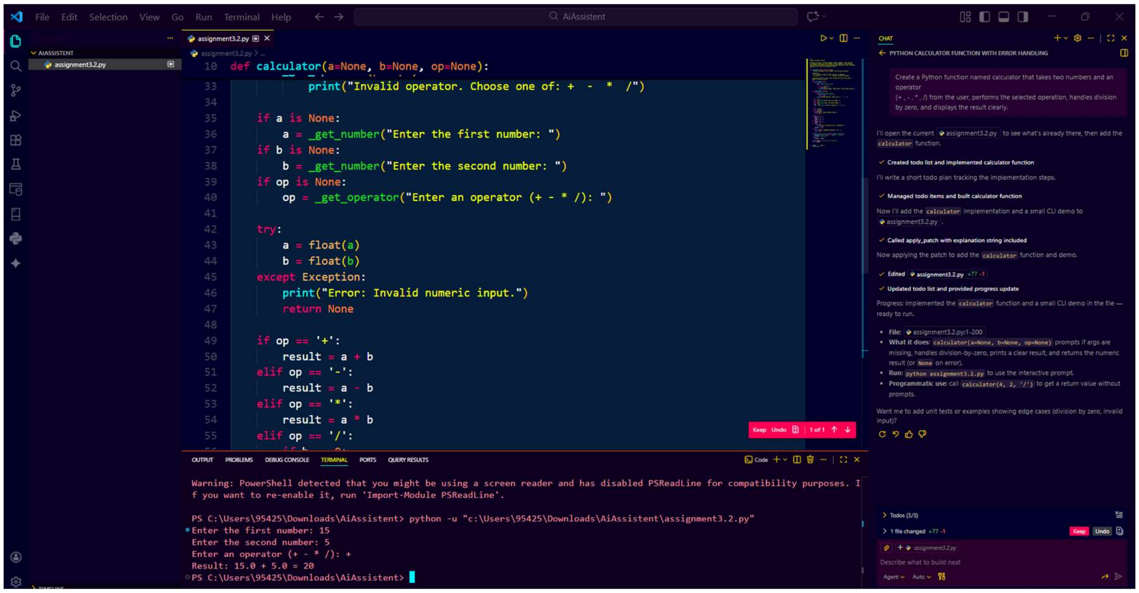
Hall : 2303a52126

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.



```python
def calculator(a=None, b=None, op=None):
            print("Invalid operator. Choose one of: +  -  *  /")

    if a is None:
        a = _get_number("Enter the first number: ")
    if b is None:
        b = _get_number("Enter the second number: ")
    if op is None:
        op = _get_operator("Enter an operator (+ - * /): ")

    try:
        a = float(a)
        b = float(b)
    except Exception:
        print("Error: Invalid numeric input.")
        return None

    if op == '+':
        result = a + b
    elif op == '-':
        result = a - b
    elif op == '*':
        result = a * b
    elif op == '/':
```

```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. I
f you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\95425\Downloads\AiAssistent> python -u "c:\Users\95425\Downloads\AiAssistent\assignment3.2.py"
Enter the first number: 15
Enter the second number: 5
Enter an operator (+ - * /): +
Result: 15.0 + 5.0 = 20
PS C:\Users\95425\Downloads\AiAssistent>
```

PYTHON CALCULATOR FUNCTION WITH ERROR HANDLING

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.

I'll open the current assignment3.2.py to see what's already there, then add the `calculator` function.

✓ **Created todo list and implemented calculator function**

I'll write a short todo plan tracking the implementation steps.

✓ **Managed todo items and built calculator function**

Now I'll add the `calculator` implementation and a small CLI demo to assignment3.2.py .

✓ **Called apply_patch with explanation string included**

Now applying the patch to add the `calculator` function and demo.

✓ Edited assignment3.2.py +77 -1

✓ **Updated todo list and provided progress update**

Progress: Implemented the `calculator` function and a small CLI demo in the file — ready to run.

- **File:** assignment3.2.py:1-200
- **What it does:** `calculator(a=None, b=None, op=None)` prompts if args are missing, handles division-by-zero, prints a clear result, and returns the numeric result (or `None` on error).
- **Run:** `python assignment3.2.py` to use the interactive prompt.
- **Programmatic use:** call `calculator(4, 2, '/')` to get a return value without prompts.

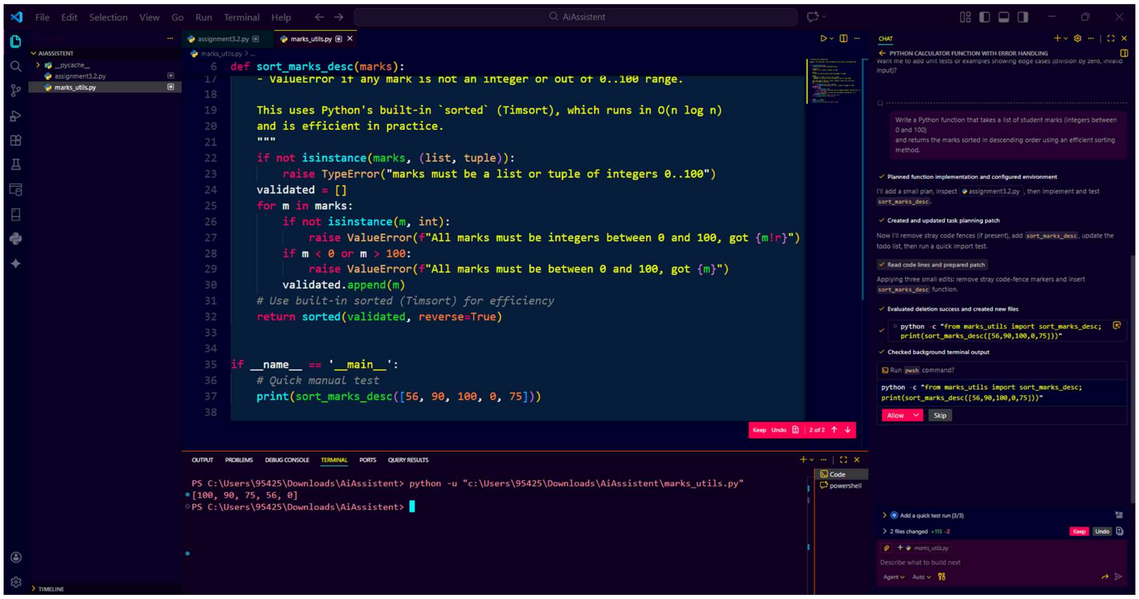
Want me to add unit tests or examples showing edge cases (division by zero, invalid input)?

Todos (3/3)

1 file changed +77 -1

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.



```python
def sort_marks_desc(marks):
    - ValueError if any mark is not an integer or out of 0..100 range.

    This uses Python's built-in `sorted` (Timsort), which runs in O(n log n)
    and is efficient in practice.
    """
    if not isinstance(marks, (list, tuple)):
        raise TypeError("marks must be a list or tuple of integers 0..100")
    validated = []
    for m in marks:
        if not isinstance(m, int):
            raise ValueError(f"All marks must be integers between 0 and 100, got {m!r}")
        if m < 0 or m > 100:
            raise ValueError(f"All marks must be between 0 and 100, got {m}")
        validated.append(m)
    # Use built-in sorted (Timsort) for efficiency
    return sorted(validated, reverse=True)


if __name__ == '__main__':
    # Quick manual test
    print(sort_marks_desc([56, 90, 100, 0, 75]))
```

```
PS C:\Users\95425\Downloads\AiAssistent> python -u "c:\Users\95425\Downloads\AiAssistent\marks_utils.py"
[100, 90, 75, 56, 0]
PS C:\Users\95425\Downloads\AiAssistent>
```

PYTHON CALCULATOR FUNCTION WITH ERROR HANDLING

Want me to add unit tests or examples showing edge cases (division by zero, invalid input)?

Write a Python function that takes a list of student marks (integers between 0 and 100)
and returns the marks sorted in descending order using an efficient sorting method.

✓ Planned function implementation and configured environment

I'll add a small plan, inspect assignment3.2.py, then implement and test `sort_marks_desc`.

✓ Created and updated task planning patch

Now I'll remove stray code fences (if present), add `sort_marks_desc`, update the todo list, then run a quick import test.

✓ Read code lines and prepared patch

Applying three small edits: remove stray code-fence markers and insert `sort_marks_desc` function.

✓ Evaluated deletion success and created new files

```
python -c "from marks_utils import sort_marks_desc; print(sort_marks_desc([56,90,100,0,75]))"
```

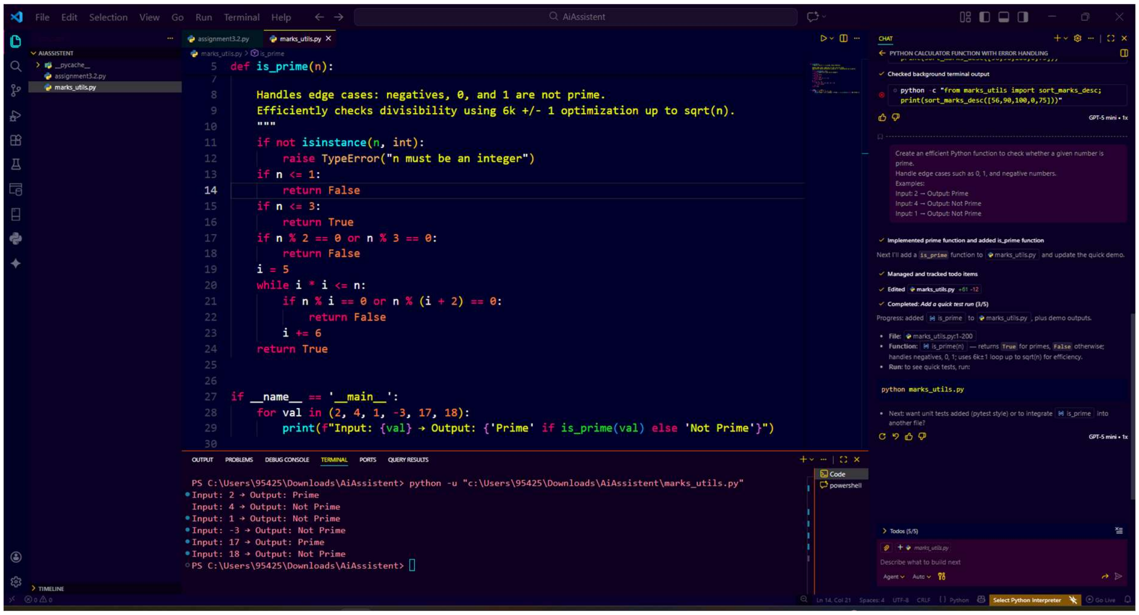
✓ Checked background terminal output

Run `pwsh` command?

```
python -c "from marks_utils import sort_marks_desc; print(sort_marks_desc([56,90,100,0,75]))"
```

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):

    Handles edge cases: negatives, 0, and 1 are not prime.
    Efficiently checks divisibility using 6k +/- 1 optimization up to sqrt(n).
    """
    if not isinstance(n, int):
        raise TypeError("n must be an integer")
    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


if __name__ == '__main__':
    for val in (2, 4, 1, -3, 17, 18):
        print(f"Input: {val} → Output: {'Prime' if is_prime(val) else 'Not Prime'}")
```

```
PS C:\Users\95425\Downloads\AiAssistent> python -u "c:\Users\95425\Downloads\AiAssistent\marks_utils.py"
Input: 2 → Output: Prime
Input: 4 → Output: Not Prime
Input: 1 → Output: Not Prime
Input: -3 → Output: Not Prime
Input: 17 → Output: Prime
Input: 18 → Output: Not Prime
PS C:\Users\95425\Downloads\AiAssistent>
```

PYTHON CALCULATOR FUNCTION WITH ERROR HANDLING

✓ Checked background terminal output

```
python -c "from marks_utils import sort_marks_desc; print(sort_marks_desc([56,90,100,0,75]))"
```

GPT-5 mini • 1x

Create an efficient Python function to check whether a given number is prime.
Handle edge cases such as 0, 1, and negative numbers.
Examples:
Input: 2 → Output: Prime
Input: 4 → Output: Not Prime
Input: 1 → Output: Not Prime

✓ Implemented prime function and added is_prime function

Next I'll add a `is_prime` function to marks_utils.py and update the quick demo.

✓ Managed and tracked todo items

✓ Edited marks_utils.py +61 -12

✓ Completed: *Add a quick test run* (3/5)

Progress: added is_prime to marks_utils.py, plus demo outputs.

- **File:** marks_utils.py:1-200
- **Function:** is_prime(n) — returns `True` for primes, `False` otherwise; handles negatives, 0, 1; uses 6k±1 loop up to sqrt(n) for efficiency.
- **Run:** to see quick tests, run:

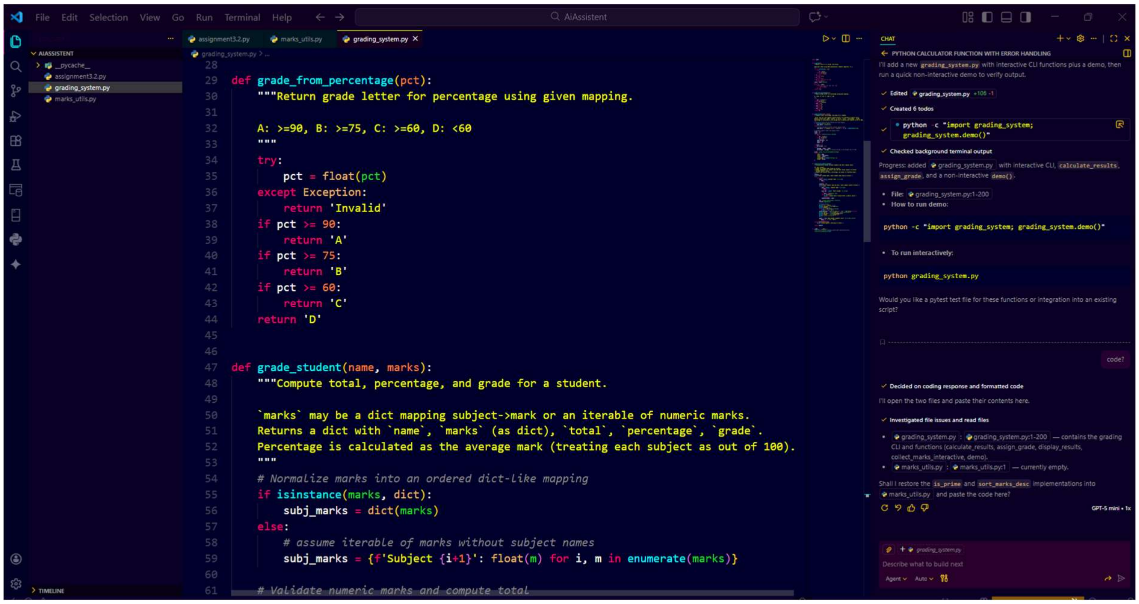
```
python marks_utils.py
```

- Next: want unit tests added (pytest style) or to integrate is_prime into another file?

GPT-5 mini • 1x

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 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)}

```

PYTHON CALCULATOR FUNCTION WITH ERROR HANDLING

I'll add a new `grading_system.py` with interactive CLI functions plus a demo, then run a quick non-interactive demo to verify output.

✓ Edited grading_system.py +106 -1

✓ Created 6 todos

```
python -c "import grading_system;
grading_system.demo()"
```

✓ Checked background terminal output

Progress: added grading_system.py with interactive CLI, `calculate_results`, `assign_grade`, and a non-interactive `demo()`.

- **File:** grading_system.py:1-200
- **How to run demo:**

```
python -c "import grading_system; grading_system.demo()"
```

- **To run interactively:**

```
python grading_system.py
```

Would you like a pytest test file for these functions or integration into an existing script?

code?

✓ Decided on coding response and formatted code

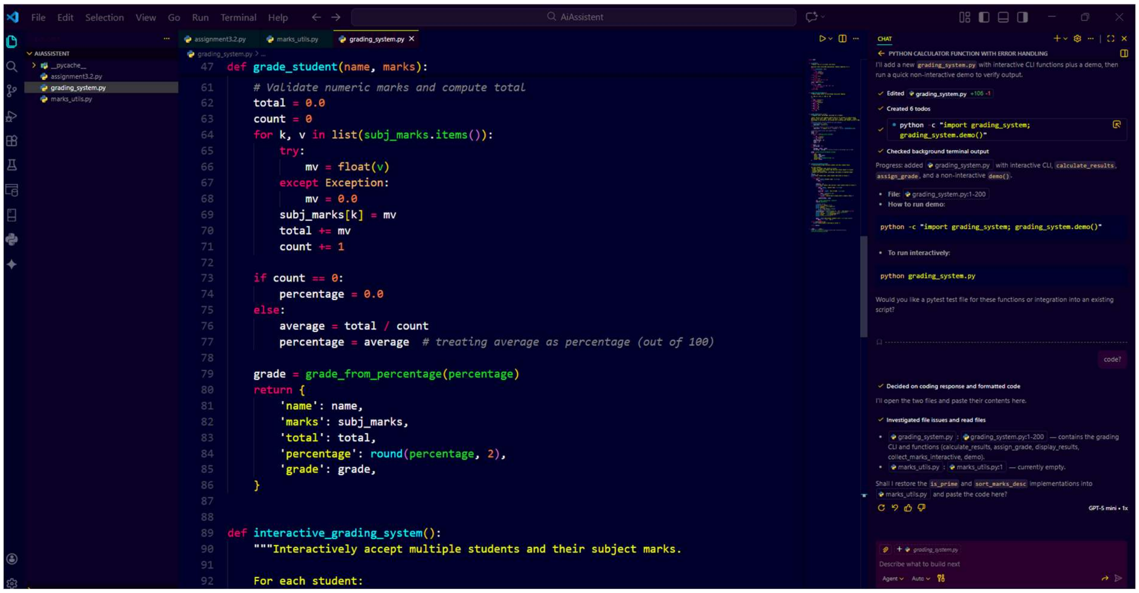
I'll open the two files and paste their contents here.

✓ Investigated file issues and read files

- grading_system.py : grading_system.py:1-200 — contains the grading CLI and functions (calculate_results, assign_grade, display_results, collect_marks_interactive, demo).
- marks_utils.py : marks_utils.py:1 — currently empty.

Shall I restore the `is_prime` and `sort_marks_desc` implementations into marks_utils.py and paste the code here?

GPT-5 mini • 1x



```python
    # 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 CALCULATOR FUNCTION WITH ERROR HANDLING

I'll add a new `grading_system.py` with interactive CLI functions plus a demo, then run a quick non-interactive demo to verify output.

✓ Edited grading_system.py +106 -1

✓ Created 6 todos

python -c "import grading_system; grading_system.demo()"

✓ Checked background terminal output

Progress: added grading_system.py with interactive CLI, `calculate_results`, `assign_grade`, and a non-interactive `demo()`.

- **File:** grading_system.py:1-200
- **How to run demo:**

python -c "import grading_system; grading_system.demo()"

- **To run interactively:**

python grading_system.py

Would you like a pytest test file for these functions or integration into an existing script?

code?

✓ Decided on coding response and formatted code

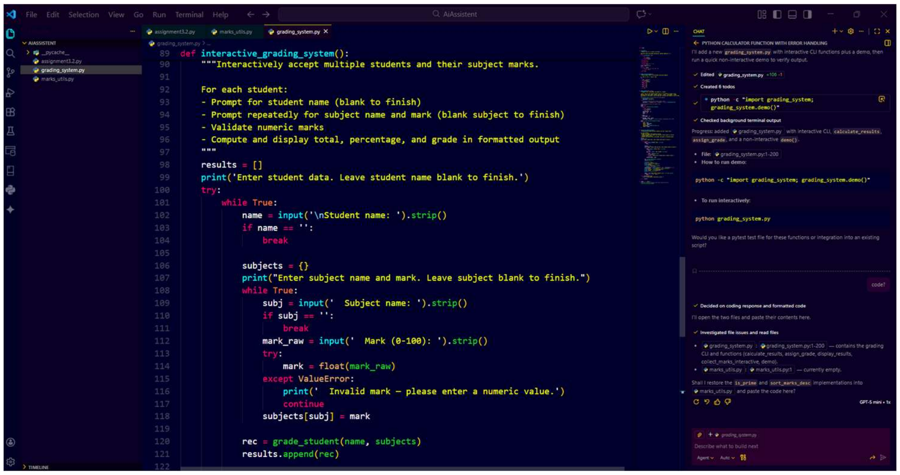
I'll open the two files and paste their contents here.

✓ Investigated file issues and read files

- grading_system.py : grading_system.py:1-200 — contains the grading CLI and functions (calculate_results, assign_grade, display_results, collect_marks_interactive, demo).
- marks_utils.py : marks_utils.py:1 — currently empty.

Shall I restore the `is_prime` and `sort_marks_desc` implementations into marks_utils.py and paste the code here?

GPT-5 mini • 1x



```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 CALCULATOR FUNCTION WITH ERROR HANDLING

I'll add a new `grading_system.py` with interactive CLI functions plus a demo, then run a quick non-interactive demo to verify output.

✓ Edited grading_system.py +106 -1

✓ Created 6 todos

✓
```
python -c "import grading_system;
grading_system.demo()"
```

✓ Checked background terminal output

Progress: added grading_system.py with interactive CLI, `calculate_results`, `assign_grade`, and a non-interactive `demo()`.

- File: grading_system.py:1-200
- How to run demo:

```
python -c "import grading_system; grading_system.demo()"
```

- To run interactively:

```
python grading_system.py
```

Would you like a pytest test file for these functions or integration into an existing script?

code?

✓ Decided on coding response and formatted code

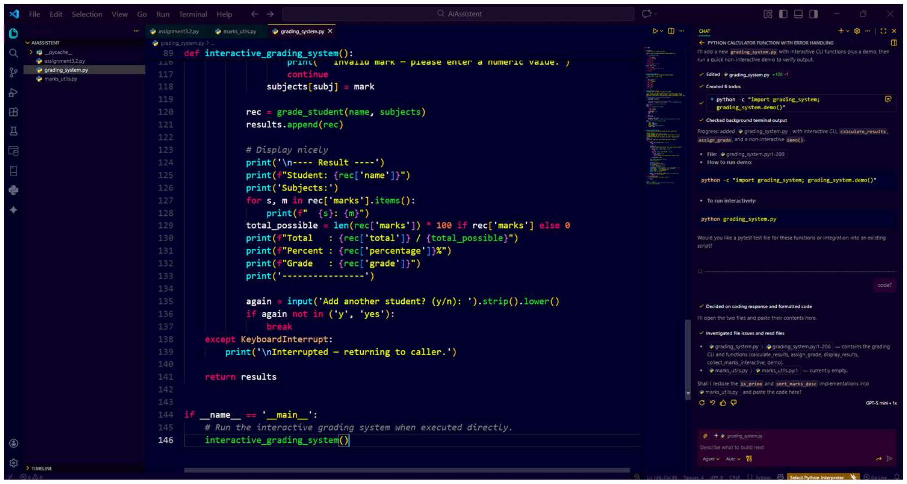
I'll open the two files and paste their contents here.

✓ Investigated file issues and read files

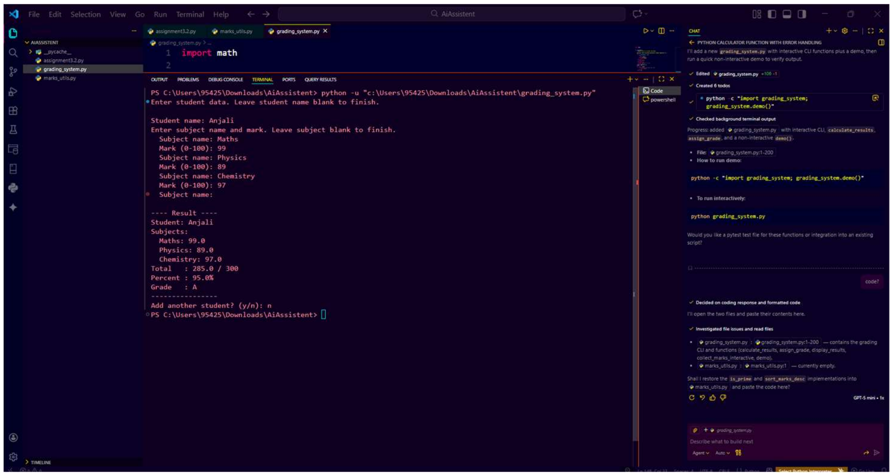
- grading_system.py : grading_system.py:1-200 — contains the grading CLI and functions (calculate_results, assign_grade, display_results, collect_marks_interactive, demo).
- marks_utils.py : marks_utils.py:1 — currently empty.

Shall I restore the `is_prime` and `sort_marks_desc` implementations into marks_utils.py and paste the code here?

GPT-5 mini • 1x
grading_system.py
Describe what to build next
Agent
Auto


File Edit Selection View Go Run Terminal Help
AiAssistent
AIASSISTENT
TIMELINE
CHAT
Select Python Interpreter

```python
def interactive_grading_system():
                    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()
```

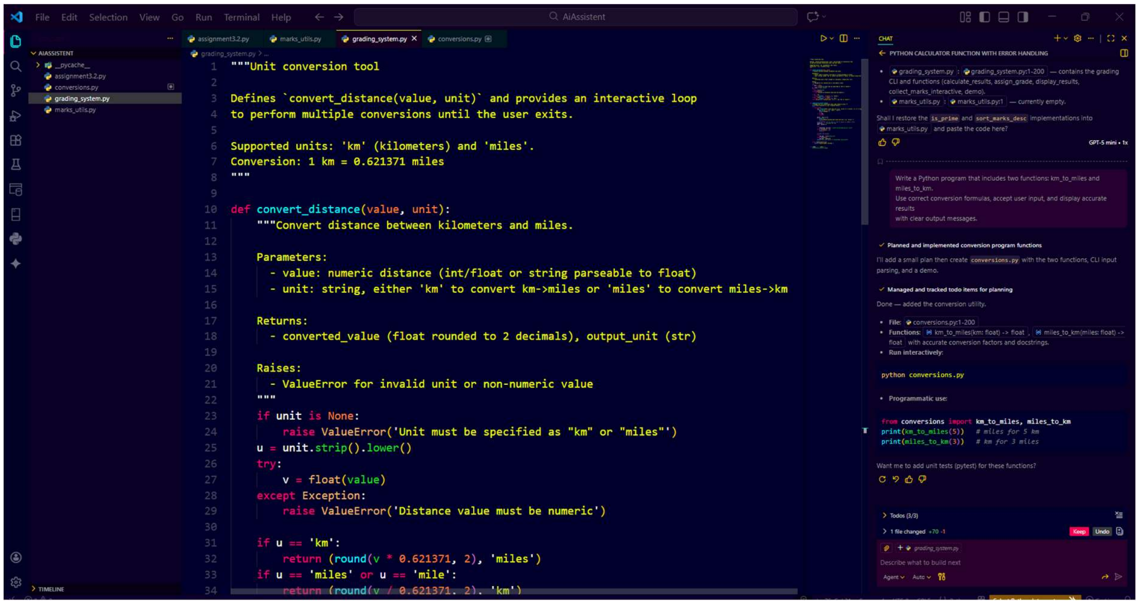

File Edit Selection View Go Run Terminal Help
AiAssistent
OUTPUT PROBLEMS DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS
Code
powershell
CHAT
Select Python Interpreter

```python
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.



```python
"""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')
```

CHAT

PYTHON CALCULATOR FUNCTION WITH ERROR HANDLING

- grading_system.py : grading_system.py:1-200 — contains the grading CLI and functions (calculate_results, assign_grade, display_results, collect_marks_interactive, demo).
- marks_utils.py : marks_utils.py:1 — currently empty.

Shall I restore the `is_prime` and `sort_marks_desc` implementations into marks_utils.py and paste the code here?

GPT-5 mini • 1x

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**

I'll add a small plan then create `conversions.py` with the two functions, CLI input parsing, and a demo.

✓ **Managed and tracked todo items for planning**

Done — added the conversion utility.

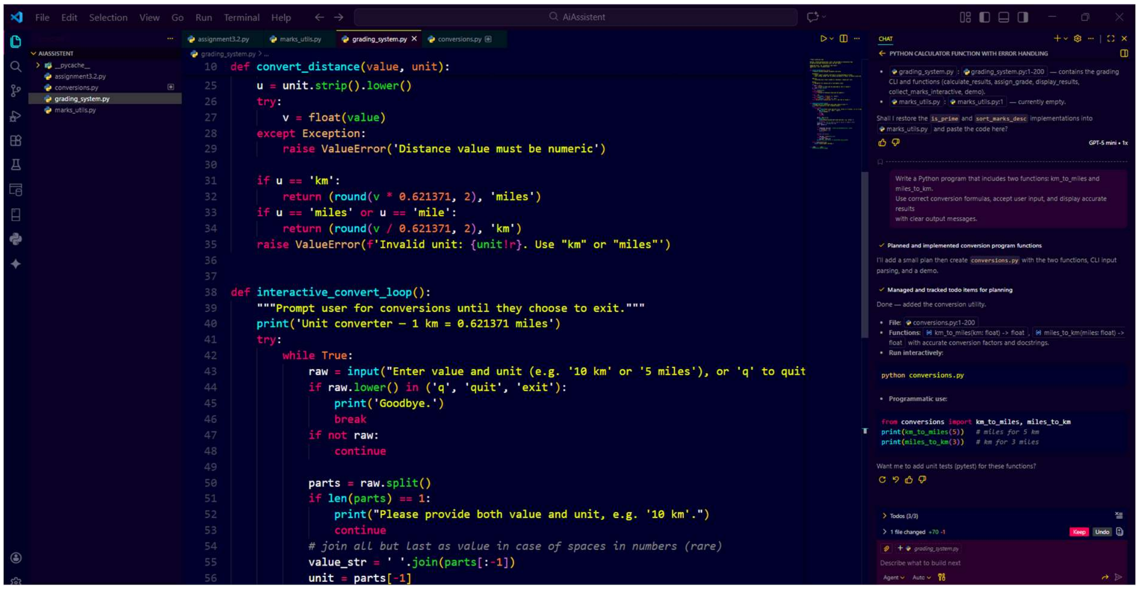
- **File:** conversions.py:1-200
- **Functions:** km_to_miles(km: float) -> float , miles_to_km(miles: float) -> float with accurate conversion factors and docstrings.
- **Run interactively:**

```
python conversions.py
```

- **Programmatic use:**

```python
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?



```python
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]
```

CHAT

PYTHON CALCULATOR FUNCTION WITH ERROR HANDLING

- grading_system.py : grading_system.py:1-200 — contains the grading CLI and functions (calculate_results, assign_grade, display_results, collect_marks_interactive, demo).
- marks_utils.py : marks_utils.py:1 — currently empty.

Shall I restore the is_prime and sort_marks_desc implementations into marks_utils.py and paste the code here?

GPT-5 mini • 1x

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

I'll add a small plan then create conversions.py with the two functions, CLI input parsing, and a demo.

✓ Managed and tracked todo items for planning

Done — added the conversion utility.

- File: conversions.py:1-200
- Functions: km_to_miles(km: float) -> float , miles_to_km(miles: float) -> float with accurate conversion factors and docstrings.
- Run interactively:

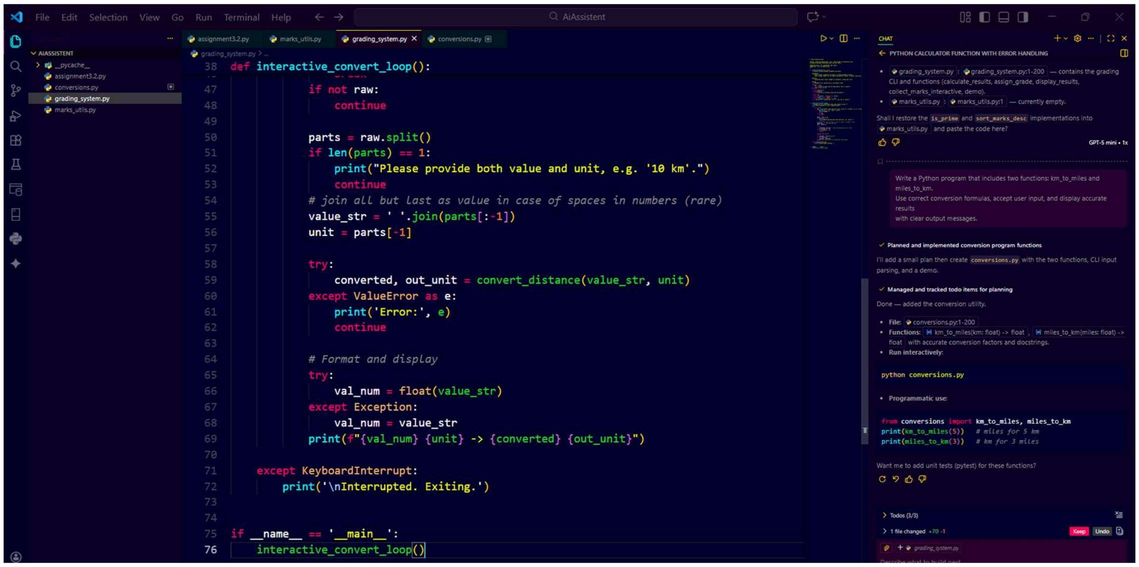
python conversions.py

- 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?

Todos (3/3)
1 file changed +70 -1
Keep
Undo
grading_system.py
Describe what to build next
Agent
Auto



```python
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()
```

PYTHON CALCULATOR FUNCTION WITH ERROR HANDLING

- grading_system.py : grading_system.py:1-200 — contains the grading CLI and functions (calculate_results, assign_grade, display_results, collect_marks_interactive, demo).
- marks_utils.py : marks_utils.py:1 — currently empty.

Shall I restore the `is_prime` and `sort_marks_desc` implementations into marks_utils.py and paste the code here?

GPT-5 mini • 1x

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**

I'll add a small plan then create `conversions.py` with the two functions, CLI input parsing, and a demo.

✓ **Managed and tracked todo items for planning**

Done — added the conversion utility.

- **File:** conversions.py:1-200
- **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:**

```python
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?

Todos (3/3)

1 file changed +70 -1