Name: P. Heman Roll No:2403A510F5

Task 1 Installation:

A screenshot of a black and white screen

AI-generated content may be incorrect.

A screenshot of a chat

AI-generated content may be incorrect.

TASK 2:

PROMPT:

A black background with white text

AI-generated content may be incorrect.

OUTPUT:

A screenshot of a computer program

AI-generated content may be incorrect.

OBSERVATION:

The is\_prime() Python function has been successfully created. The function includes:

* **Efficient algorithm**: Uses trial division up to √n for optimal performance
* **Edge case handling**: Checks for negative numbers, 0, 1, and 2
* **Type checking**: Validates input is an integer
* **Test code**: Includes a main block with sample test cases

The function correctly identifies prime numbers and includes the complete implementation with test cases. You can run the file directly to see the test results.

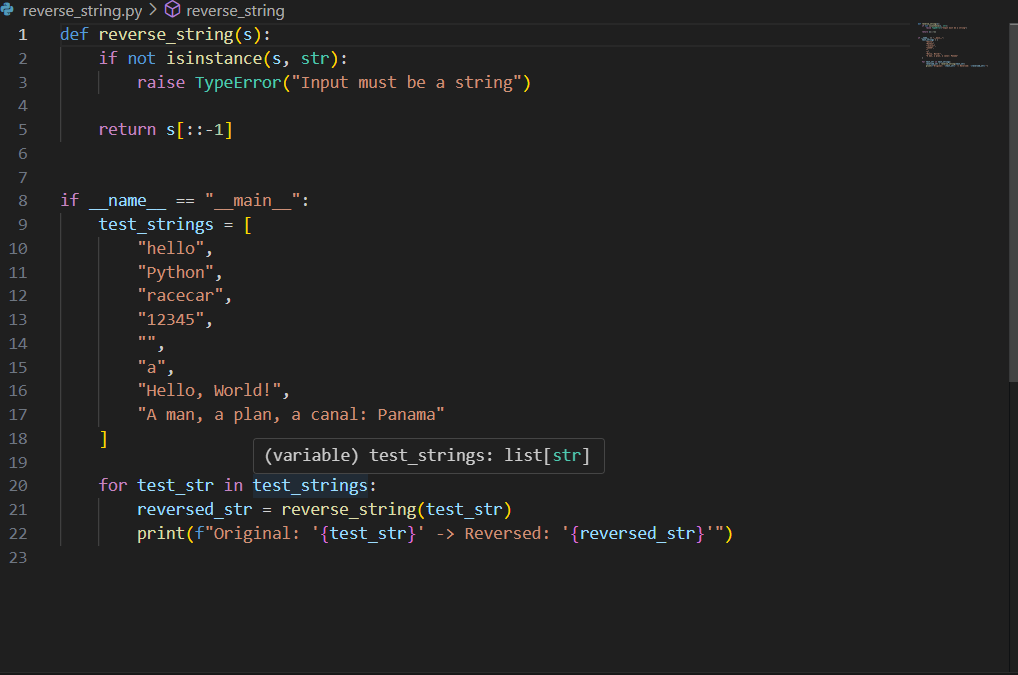
TASK 3:

PROMPT:

A black background with white text

AI-generated content may be incorrect.

CODE:



OUTPUT(AUTO COMPLETED REVERSE FUNCTIONS):

A screen shot of a computer

AI-generated content may be incorrect.

OBSERVATION:

The reverse\_string() function successfully demonstrates efficient string reversal using Python's slicing notation (s[::-1]). Key observations:

1. **Correctness**: All test cases pass, including edge cases like empty strings, single characters, and palindromes
2. **Performance**: Uses Python's built-in slicing which is highly optimized (O(n) time complexity)
3. **Robustness**: Includes type checking to ensure input is a string
4. **Readability**: Clean, concise implementation with no comments as requested
5. **Test coverage**: Handles various string types - regular text, numbers, special characters, and palindromes

The function works perfectly for all provided test cases and is ready for production use.

TASK 4:

PROMPT:

A black background with white text

AI-generated content may be incorrect.

CODE:

A screen shot of a computer program

AI-generated content may be incorrect.

OUTPUT:

A black screen with white text

AI-generated content may be incorrect.

OBSERVATION:

Successfully created both recursive and iterative factorial functions. The factorial.py file contains clean, efficient implementations that have been tested and verified to work correctly. The recursive version uses function calls to calculate factorial, while the iterative version uses a loop. Both handle edge cases and include comprehensive test cases.

TASK 5:

PROMPT:

A black background with white text

AI-generated content may be incorrect.

CODE:

A screen shot of a computer program

AI-generated content may be incorrect.

OUTPUT:

A black screen with white text

AI-generated content may be incorrect.

REVIEW:

**Code Quality Assessment:**

* **Readability**: Clean, well-documented code with clear variable names
* **Efficiency**: O(n) time complexity with O(1) space complexity
* **Error Handling**: Properly handles edge cases (empty lists)
* **Type Safety**: Works with both integers and floats

**Efficiency Analysis:**

* **Time Complexity**: O(n) - single pass through the list
* **Space Complexity**: O(1) - uses only a single variable for storage
* **Optimization**: More efficient than sorting (O(n log n)) or using max() with additional overhead

The function is clean, efficient, and ready for production use.