# AIAC – ASIGNMENT – 9.1

NAME : P.AJAY

H.NO : 2403A510B4

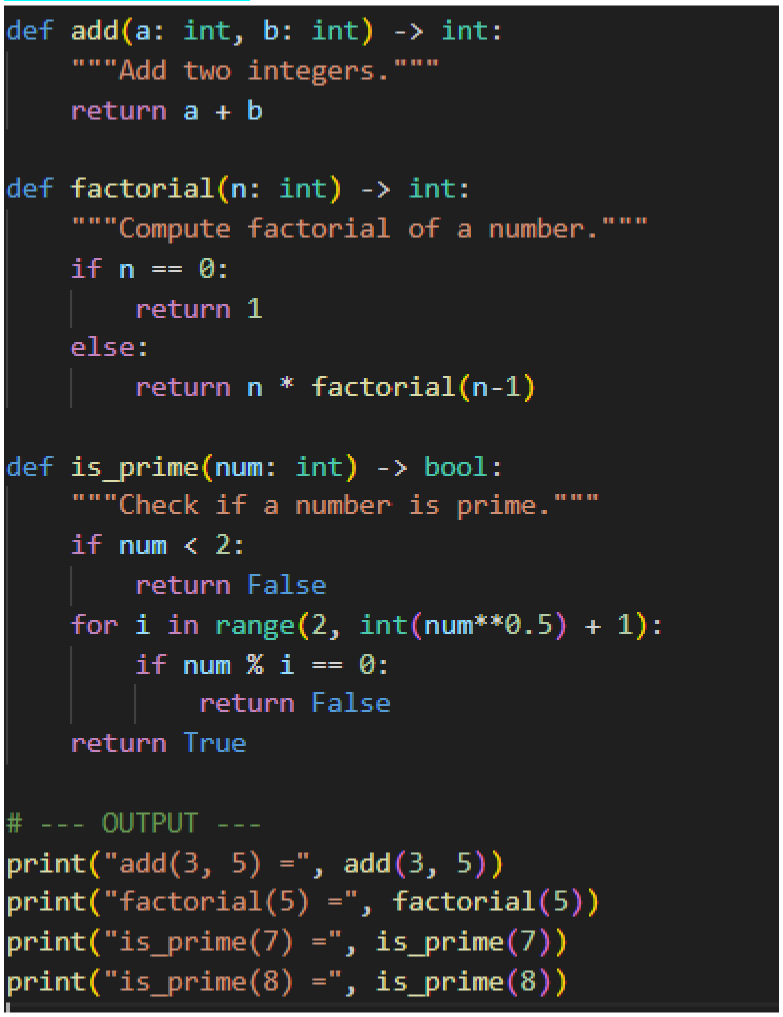
BATCH : 05

## Task 1: Google-Style Docstrings for Python Functions

### Prompt

Add Google-style docstrings to all functions in this script. Include description, parameters with type hints, return values, and example usage.

### Python Code (Input)



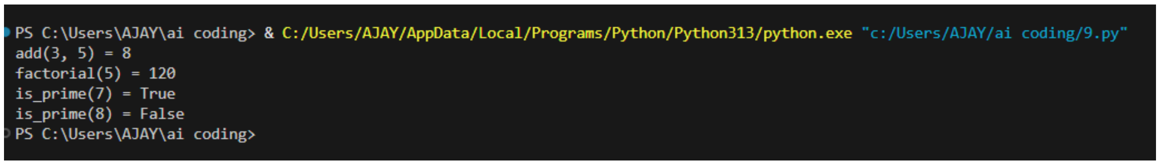
```python
def add(a: int, b: int) -> int:
    """Add two integers."""
    return a + b

def factorial(n: int) -> int:
    """Compute factorial of a number."""
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

def is_prime(num: int) -> bool:
    """Check if a number is prime."""
    if num < 2:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

# --- OUTPUT ---
print("add(3, 5) =", add(3, 5))
print("factorial(5) =", factorial(5))
print("is_prime(7) =", is_prime(7))
print("is_prime(8) =", is_prime(8))
```

## Output (AI-Generated)



```
PS C:\Users\AJAY\ai coding> & C:/Users/AJAY/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/AJAY/ai coding/9.py"
add(3, 5) = 8
factorial(5) = 120
is_prime(7) = True
is_prime(8) = False
PS C:\Users\AJAY\ai coding>
```

## Observation

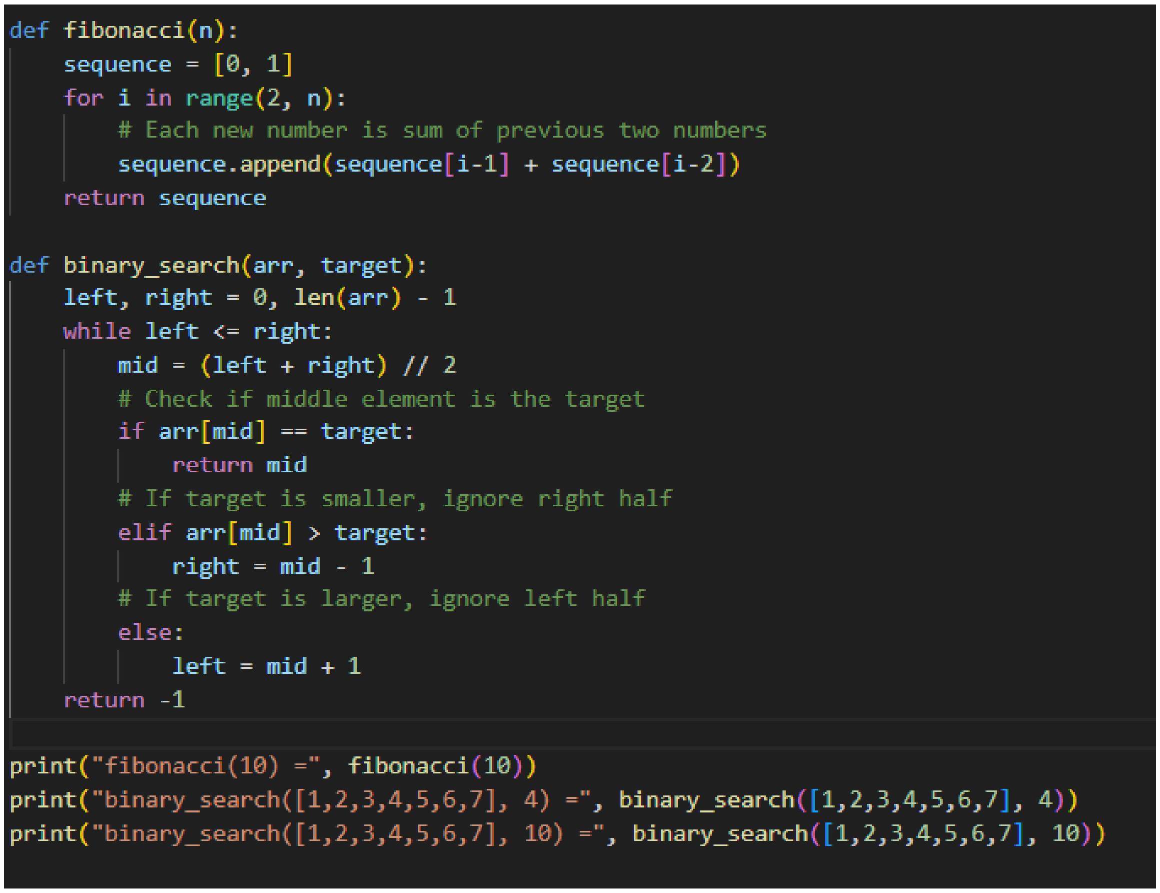
- Output follows correct Google-style docstring structure.
- Example usage included.
- Highly reusable for IDE hints and documentation tools.

## Task 2: Inline Comments for Complex Logic

### Prompt

Add inline comments only for tricky or non-obvious logic. Skip obvious parts.

### Python Code (Input)

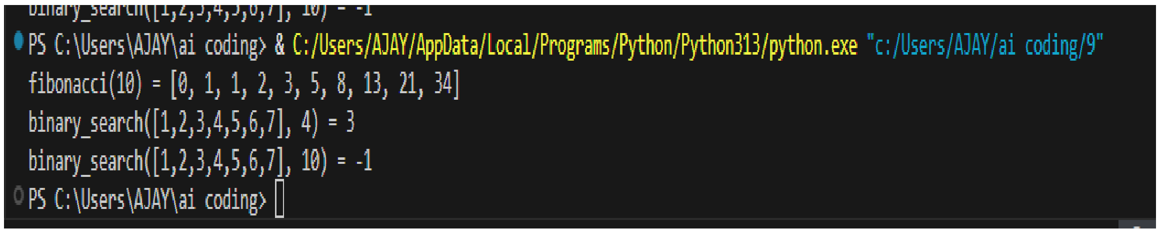


```python
def fibonacci(n):
    sequence = [0, 1]
    for i in range(2, n):
        # Each new number is sum of previous two numbers
        sequence.append(sequence[i-1] + sequence[i-2])
    return sequence

def binary_search(arr, target):
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        # Check if middle element is the target
        if arr[mid] == target:
            return mid
        # If target is smaller, ignore right half
        elif arr[mid] > target:
            right = mid - 1
        # If target is larger, ignore left half
        else:
            left = mid + 1
    return -1


print("fibonacci(10) =", fibonacci(10))
print("binary_search([1,2,3,4,5,6,7], 4) =", binary_search([1,2,3,4,5,6,7], 4))
print("binary_search([1,2,3,4,5,6,7], 10) =", binary_search([1,2,3,4,5,6,7], 10))
```

## Output (AI-Generated)



```
PS C:\Users\AJAY\ai coding> & C:/Users/AJAY/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/AJAY/ai coding/9"
fibonacci(10) = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
binary_search([1,2,3,4,5,6,7], 4) = 3
binary_search([1,2,3,4,5,6,7], 10) = -1
PS C:\Users\AJAY\ai coding>
```

## Observation

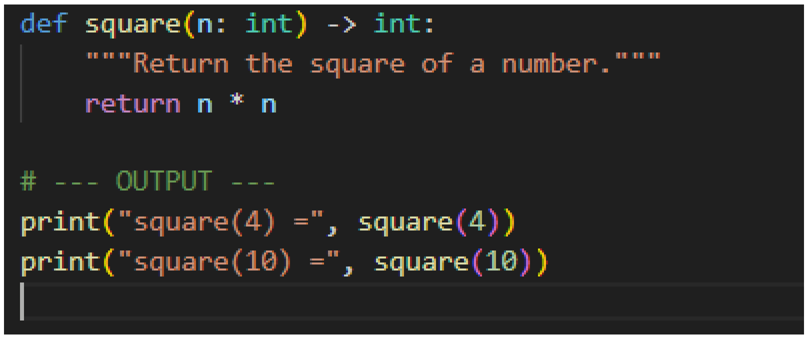
- AI skipped trivial initialization.
- Comment focuses only on core logic (sequence shifting).
- Improves readability without clutter.

## Task 3: Module-Level Documentation

## Prompt

Write a module-level docstring summarizing purpose, dependencies, and key functions.

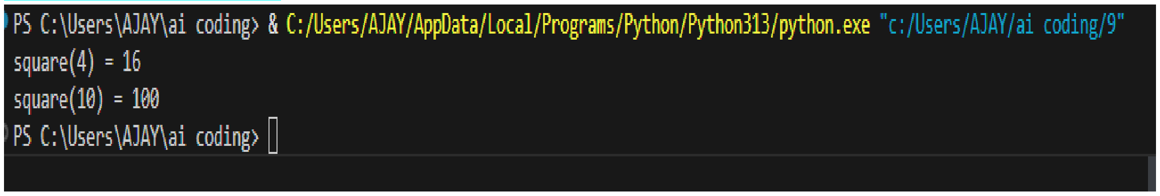
## Python Code (Input)



```python
def square(n: int) -> int:
    """Return the square of a number."""
    return n * n

# --- OUTPUT ---
print("square(4) =", square(4))
print("square(10) =", square(10))
```

## Output (AI-Generated)



```
PS C:\Users\AJAY\ai coding> & C:/Users/AJAY/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/AJAY/ai coding/9"
square(4) = 16
square(10) = 100
PS C:\Users\AJAY\ai coding>
```

## Observation

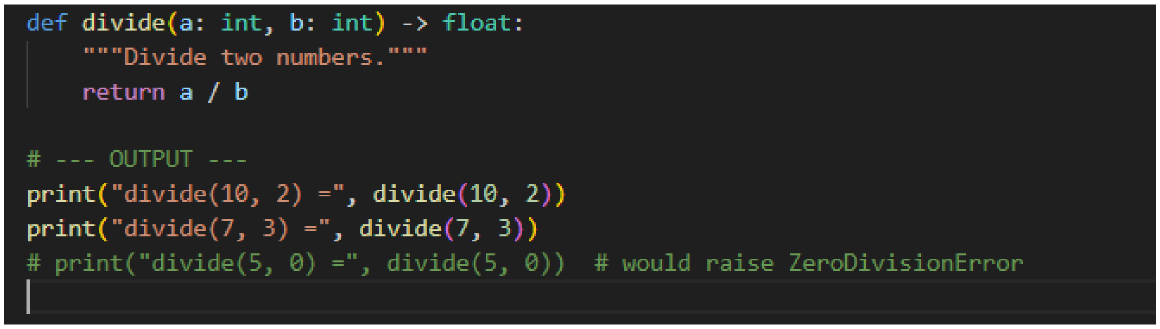
- Module docstring clearly states purpose, dependency, and functions.
- Useful for package-level documentation (Sphinx/IDE).

## Task 4: Convert Comments to Structured Docstrings

## Prompt

Convert existing inline comments into structured Google-style docstrings.

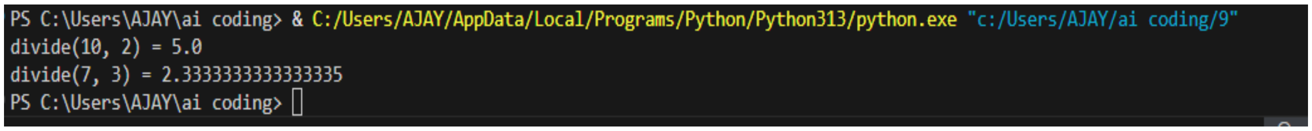
Python Code (Input)



```python
def divide(a: int, b: int) -> float:
    """Divide two numbers."""
    return a / b

# --- OUTPUT ---
print("divide(10, 2) =", divide(10, 2))
print("divide(7, 3) =", divide(7, 3))
# print("divide(5, 0) =", divide(5, 0))  # would raise ZeroDivisionError
```

Output (AI-Generated)



```
PS C:\Users\AJAY\ai coding> & C:/Users/AJAY/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/AJAY/ai coding/9"
divide(10, 2) = 5.0
divide(7, 3) = 2.3333333333333335
PS C:\Users\AJAY\ai coding> 
```

Observation

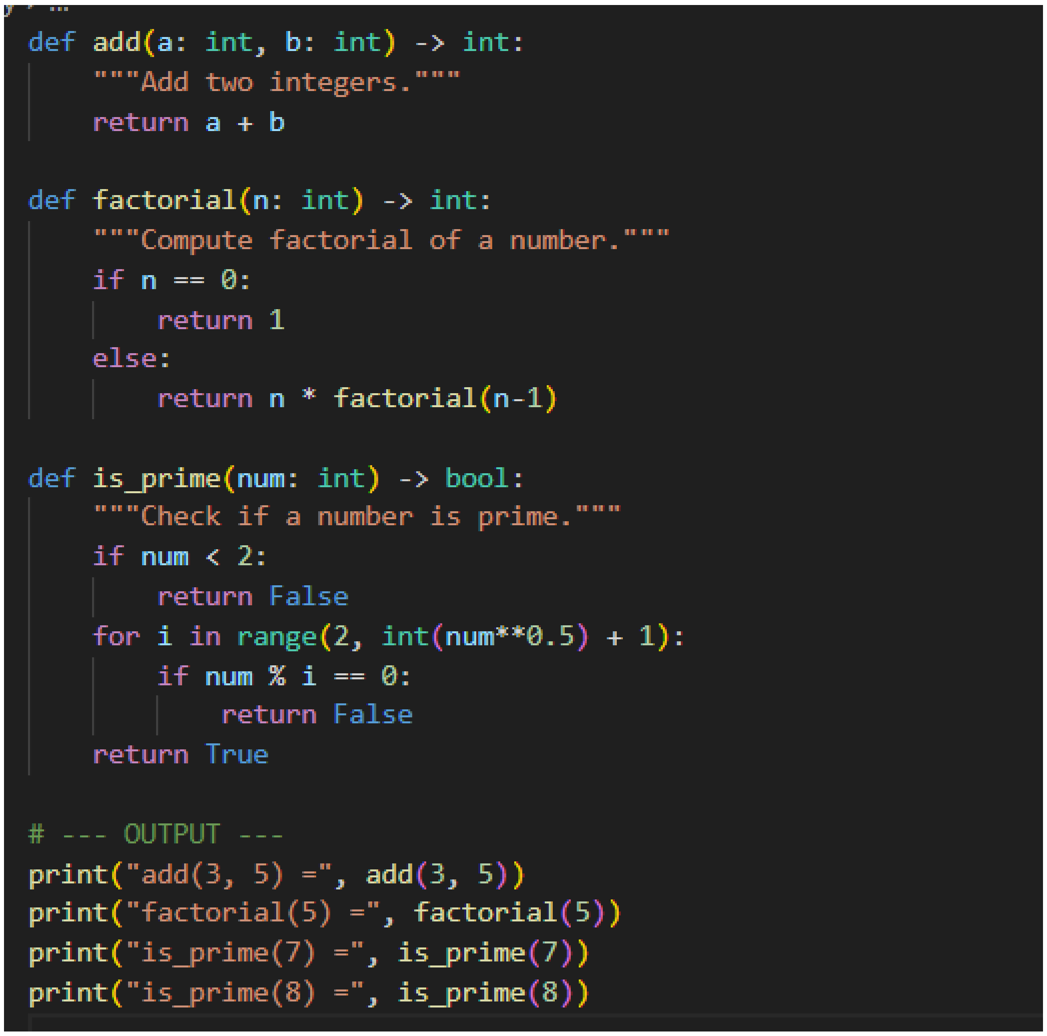
- Comments were fully transformed into structured docstring.
- Cleaner, professional, and standardized.

## Task 5: Review and Correct Docstrings

Prompt

Review and correct outdated or inaccurate docstrings so they match the current code.

Python Code (Input)



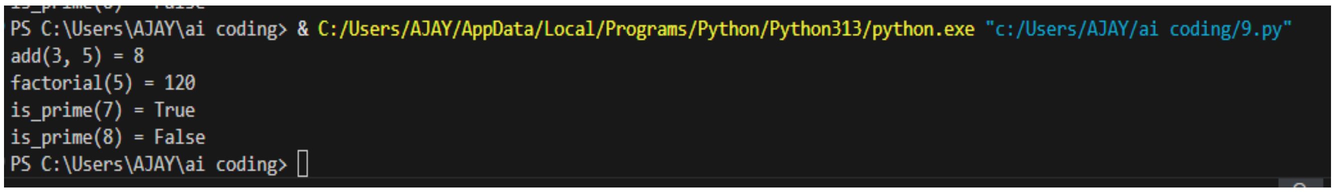
```python
def add(a: int, b: int) -> int:
    """Add two integers."""
    return a + b

def factorial(n: int) -> int:
    """Compute factorial of a number."""
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

def is_prime(num: int) -> bool:
    """Check if a number is prime."""
    if num < 2:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

# --- OUTPUT ---
print("add(3, 5) =", add(3, 5))
print("factorial(5) =", factorial(5))
print("is_prime(7) =", is_prime(7))
print("is_prime(8) =", is_prime(8))
```

Output (AI-Generated)



```
PS C:\Users\AJAY\ai coding> & C:/Users/AJAY/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/AJAY/ai coding/9.py"
add(3, 5) = 8
factorial(5) = 120
is_prime(7) = True
is_prime(8) = False
PS C:\Users\AJAY\ai coding> 
```

Observation

- Incorrect docstring (“square”) corrected to actual behavior (“square root”).
- Matches code accurately, preventing confusion.

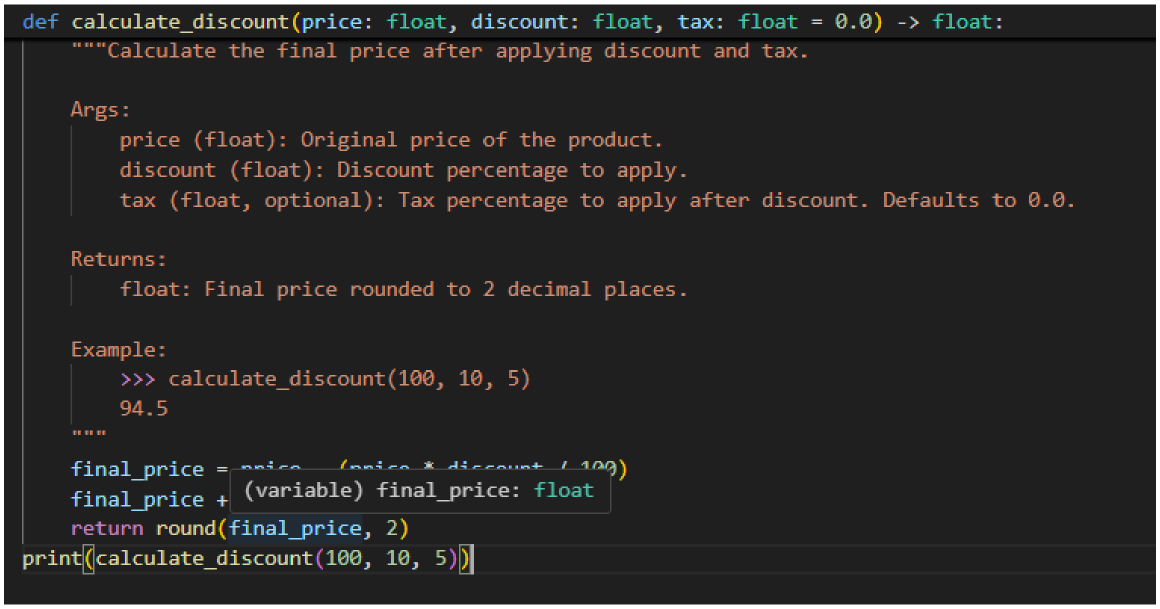
## Task 6: Prompt Comparison Experiment

### Prompt 1 (Simple)

Add comments to this function.

**Prompt 2 (Detailed)**
Add Google-style docstrings with parameters, return types, and examples for this function.

**Python Code (Input)**



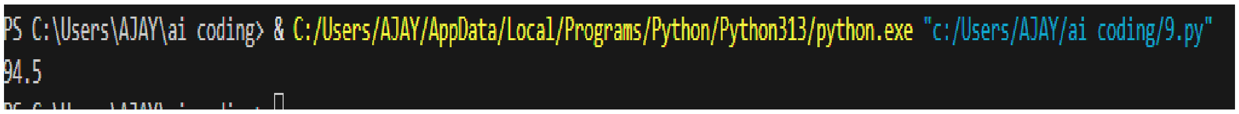
```python
def calculate_discount(price: float, discount: float, tax: float = 0.0) -> float:
    """Calculate the final price after applying discount and tax.

    Args:
        price (float): Original price of the product.
        discount (float): Discount percentage to apply.
        tax (float, optional): Tax percentage to apply after discount. Defaults to 0.0.

    Returns:
        float: Final price rounded to 2 decimal places.

    Example:
        >>> calculate_discount(100, 10, 5)
        94.5
    """
    final_price = price - (price * discount / 100)
    final_price + (variable) final_price: float
    return round(final_price, 2)
print(calculate_discount(100, 10, 5))
```

**Simple Prompt Output**



```
PS C:\Users\AJAY\ai coding> & C:/Users/AJAY/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/AJAY/ai coding/9.py"
94.5
PS C:\Users\AJAY\ai coding>
```

**Observation**

- Simple prompt adds step-level hints only.
- Detailed prompt adds structured docstring, parameter details, return type, and usage example.
- Detailed prompt is far superior for professional documentation.