Task 1: Function Summary Generation

```
def calculate_area(radius):
    return 3.14 * radius * radius

def is_even(num):
    return num % 2 == 0

def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n-1)
```

AI Response (used as docstrings):

```
def calculate_area(radius):
    """Calculates the area of a circle given its radius."""
    return 3.14 * radius * radius

def is_even(num):
    """Determines if a given number is even."""
    return num % 2 == 0

def factorial(n):
    """Computes the factorial of a non-negative integer."""
    if n == 0:
        return 1
    return n * factorial(n-1)
```

Task 2: Logical Explanation for Conditions and Loops

Original code without comments:

```
numbers = [1,2,3,4,5,6,7,8,9,10]
evens = []
for num in numbers:
    if num % 2 == 0:
```

```
        evens.append(num)
    else:
        print("odd")
print(evens)
```

Prompt:

```
Explain logic behind conditions and loops in this code. Add inline comments only for
decision logic and loop behavior.
```

Output with explanations:

```
numbers = [1,2,3,4,5,6,7,8,9,10]
evens = []
for num in numbers:  # Loop iterates through each number in the list to check parity
    if num % 2 == 0:  # Condition checks if number is divisible by 2 (even)
        evens.append(num)  # Collect even numbers in list
    else:  # Handles odd numbers
        print("odd")  # Output for odd cases
print(evens)  # Display final list of evens
```

Task 3: File-Level Overview

Sample file content (math_utils.py original):

```
def add(a,b):
    return a+b

def subtract(a,b):
    return a-b

def multiply(a,b):
    return a*b
```

Prompt:

```
Generate high-level overview for this entire Python file. Summarize purpose and
functionality. Place at top.
```

AI generated overview added:

```
"""
Math Utilities Module
This file provides basic arithmetic operations including addition, subtraction, and
multiplication for numerical values.
Useful for simple calculations in math-heavy scripts.
"""

def add(a,b):
    return a+b

def subtract(a,b):
    return a-b

def multiply(a,b):
    return a*b
```

Task 4: Refine Existing Documentation

Original code with poor comments:

```
# this does something
def divide(a, b):
    if b != 0: # check
        return a / b
    else:
        print("error") # bad

x = 10
for i in range(5): # loop
    x += i
# end
```

Prompt:

```
Refine these comments for clarity and consistency. Keep meaning same, improve wording.
```

Refined output:

```
# Division function with zero-check
def divide(a, b):
```

```
    """Divides a by b, handles division by zero."""
    if b != 0:  # Ensures denominator is not zero to avoid errors
        return a / b
    else:
        print("Cannot divide by zero")  # Error message for invalid input

x = 10
for i in range(5):  # Accumulates sum of first 5 numbers into x
    x += i
print(x)  # Outputs the computed sum
```

Task 5: Prompt Detail Impact Study

Same function:

```
def max_in_list(lst):
    max_val = lst[^1_0]
    for item in lst:
        if item > max_val:
            max_val = item
    return max_val
```

Brief Prompt:

```
Add docstring to this function.
```

Brief Output:

```
def max_in_list(lst):
    """Finds max in list."""
    max_val = lst[^1_0]
    for item in lst:
        if item > max_val:
            max_val = item
    return max_val
```

Detailed Prompt:

```
Write a comprehensive docstring for this function. Include: purpose, parameters (type and desc), returns (type and desc), logic explanation without code. Use Google style if possible. Function finds maximum value in a list using iteration.

def max_in_list(lst):
    [code]
```

Detailed Output:

```
def max_in_list(lst):
    """
    Finds the maximum value in a non-empty list of numbers.

    Args:
        lst (list): A non-empty list of comparable numbers.

    Returns:
        int or float: The maximum value in the list.

    Raises:
        IndexError: If the list is empty.

    Logic: Initializes max with first element, iterates through list, updates max if larger value found.
    """
    max_val = lst[^1_0]
    for item in lst:
        if item > max_val:
            max_val = item
    return max_val
```

Comparison Table

Aspect	Brief Prompt Output	Detailed Prompt Output
Completeness	Basic purpose only	Full params, returns, raises, logic
Clarity	Vague "finds max"	Precise descriptions, types
Accuracy	Correct but minimal	Accurate, adds error handling note
Length	1 line	Multi-line comprehensive