ASSIGNMENT 10.2

NAME: M.Ramsai

ENROLLMENT ID: 2503A51L53

TASK 1:

->Write python program as shown below.
->Use an AI assistant to review and suggest corrections

> Users > ramsa > OneDrive > Documents > ai assistant lab > AIACLAB10.1.py > ...

```python
# Corrected factorial calculation function
def calcFact(n):
    result = 1
    # Factorial should multiply from 1 to n (inclusive)
    for i in range(1, n + 1):
        result = result * i
    return result

def main():
    num = 5
    FACT = calcFact(num)
    print("The factorial of", num, "is", FACT)
    t = 10
    # Compare FACT to t, not just 10
    if FACT > t:
        print("BIG Number")
    else:
        print("small number")

main()

# Comments:
# - The range in calcFact should be range(1, n+1) to include n.
# - Removed unused variable x.
# - In the if statement, compare FACT to t for clarity.
# - Improved print statements for
```

OUTPUT:

```
/ai assistant lab/AIACLAB10.1.py"
The factorial of 5 is 120
BIG Number
```

TASK 2

->Write the Python code for Fibonacci as shown below and execute.
• Ask AI to improve variable names, add comments, and apply PEP8 formatting (cleaned up).
• Students evaluate which suggestions improve readability most. one.

```python
# Fibonacci series generator with improved variable names and inline comments

def fibonacci_series(n_terms):
    """Generate Fibonacci series up to n_terms elements."""
    first = 0
    second = 1
    series = [first, second]
    count = 2  # Already have two terms

    # Generate Fibonacci numbers until reaching n_terms
    while count < n_terms:
        next_term = first + second
        series.append(next_term)
        first = second
        second = next_term
        count += 1

    return series

def main():
    num_terms = 10
    fib_sequence = fibonacci_series(num_terms)
    print(f"Fibonacci series up to {num_terms} terms: {fib_sequence}")

main()
```

OUTPUT:

```
/ai assistant lab/AIACLAB10.1.py"
Fibonacci series up to 10 terms: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

TASK 3

Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).

• Incorporate manual docstring in code with NumPy Style

• Use AI assistance to generate a module-level docstring + individual function docstrings.

• Compare the AI-generated docstring with your manually written one.

Common Examples of Code Smells

• Long Function – A single function tries to do too many things.

• Duplicate Code – Copy-pasted logic in multiple places.

• Poor Naming – Variables or functions with confusing names (x1, foo, data123).

• Unused Variables – Declaring variables but never using them.

• Magic Numbers – Using unexplained constants (3.14159 instead of PI).

• Deep Nesting – Too many if/else levels, making code hard to read.

• Large Class – A single class handling too many responsibilities.

Why Detecting Code Smells is Important

• Makes code easier to read and maintain.

• Reduces chance of bugs in future updates.

• Helps in refactoring (improving structure without changing behavior).

• Encourages clean coding practices

CODE GENERATED:

```
"""
calculator_module.py

This module provides basic arithmetic operations: addition, subtraction, multiplication, and division.
Each function is documented using NumPy style docstrings for clarity and consistency.

Functions
---------
add(a, b)
    Returns the sum of two numbers.

subtract(a, b)
    Returns the difference between two numbers.

multiply(a, b)
    Returns the product of two numbers.

divide(a, b)
    Returns the quotient of two numbers. Raises ZeroDivisionError if b is zero.
"""
```

```python
def add(a, b):
    """
    Add two numbers.

    Parameters
    ----------
    a : float or int
        First number.
    b : float or int
        Second number.

    Returns
    -------
    float or int
        The sum of a and b.
    """
    return a + b
```

```python
def subtract(a, b):
    """
    Subtract two numbers.

    Parameters
    ----------
    a : float or int
        First number.
    b : float or int
        Second number.

    Returns
    -------
    float or int
        The difference of a and b.
    """
    return a - b
```

```
def multiply(a, b):
    """
    Multiply two numbers.

    Parameters
    ----------
    a : float or int
        First number.
    b : float or int
        Second number.

    Returns
    -------
    float or int
        The product of a and b.
    """
    return a * b
```

```python
def divide(a, b):
    """
    Divide two numbers.

    Parameters
    ----------
    a : float or int
        Numerator.
    b : float or int
        Denominator.

    Returns
    -------
    float
        The quotient of a and b.

    Raises
    ------
    ZeroDivisionError
        If b is zero.
    """
    if b == 0:
        raise ZeroDivisionError("Cannot divide by zero.")
    return a / b
```