SR UNIVERSITY

AI ASSISTED CODING

NAME: Meer Burhan Ali Hashmi

HTNO: 2503A51L44

Batch:20

Lab 10: Code Review and Quality: Using AI to improve code quality and readability

Lab Objectives:

- To understand the importance of code readability, maintainability, and quality.
- To explore how AI-assisted coding tools can review code and suggest improvements.
- To practice identifying code smells, redundant code, and poor naming conventions.
- To apply AI tools for refactoring and improving readability.
- To critically evaluate AI feedback and integrate it into real projects

Lab Outcomes (LOs):

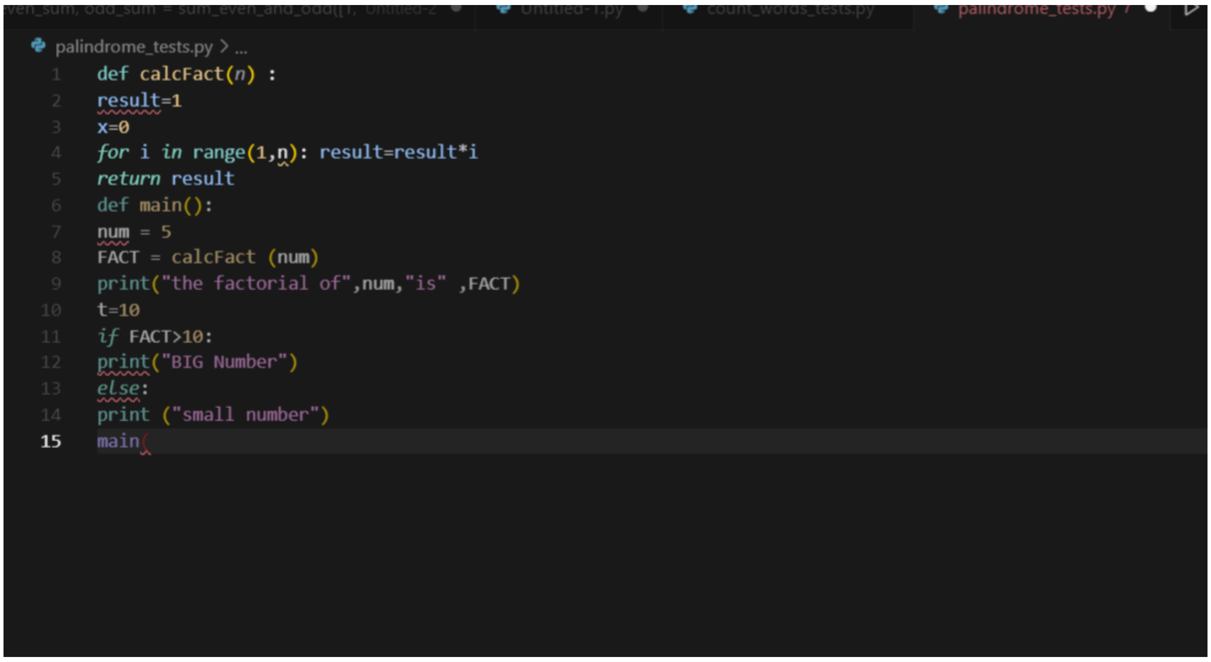
After completing this lab, students will be able to:

- Use AI-assisted tools (e.g., GitHub Copilot, Cursor AI) to review Python code.
- Identify and correct syntax issues, code smells, and inefficient logic.
- Improve readability by applying consistent formatting, naming, and comments.
- Refactor code with AI suggestions while ensuring functionality is preserved.
- Apply best practices for writing clean, maintainable, and professional code.

Task Description#1 AI-Assisted Code Review (Basic Errors):

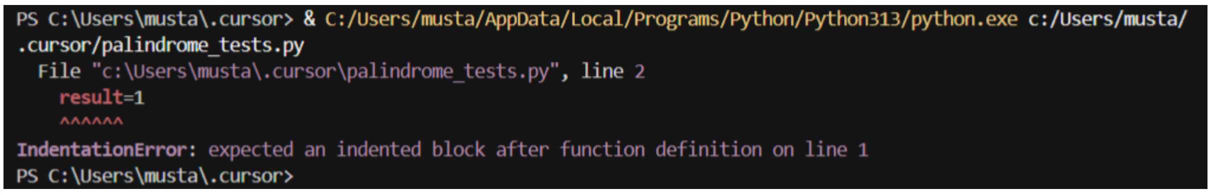
- Write python program as shown below.
- Use an AI assistant to review and suggest corrections.

Given program to ai which was manually written:



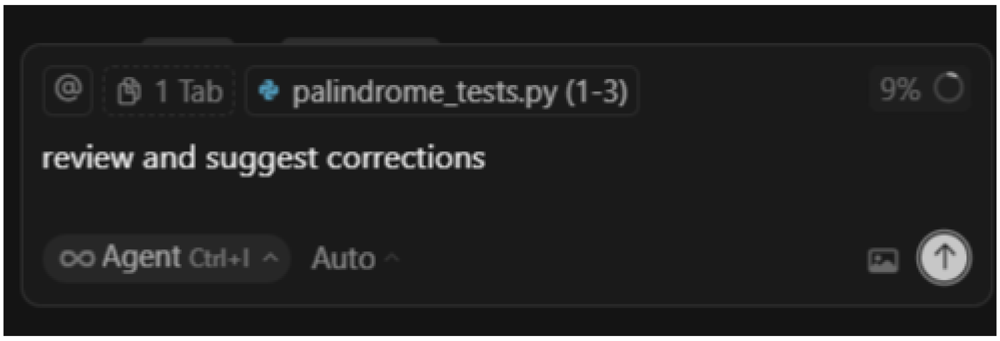
```
def calcFact(n) :
result=1
x=0
for i in range(1,n): result=result*i
return result
def main():
num = 5
FACT = calcFact (num)
print("the factorial of",num,"is" ,FACT)
t=10
if FACT>10:
print("BIG Number")
else:
print ("small number")
main(
```

Output:



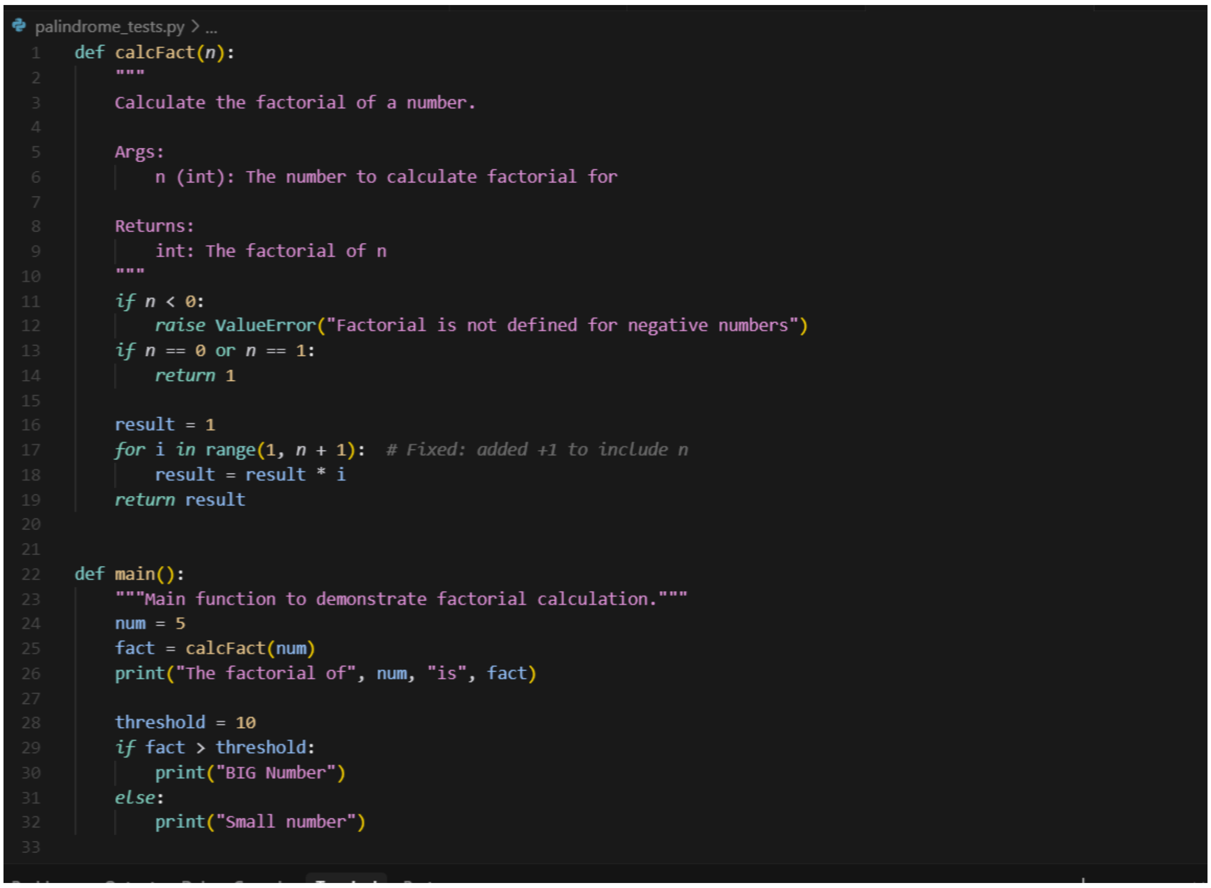
```
PS C:\Users\musta\.cursor> & C:/Users/musta/AppData/Local/Programs/Python/Python313/python.exe c:/Users/musta/
.cursor/palindrome_tests.py
  File "c:\Users\musta\.cursor\palindrome_tests.py", line 2
    result=1
    ^^^^^^
IndentationError: expected an indented block after function definition on line 1
PS C:\Users\musta\.cursor>
```

Given prompt to AI:



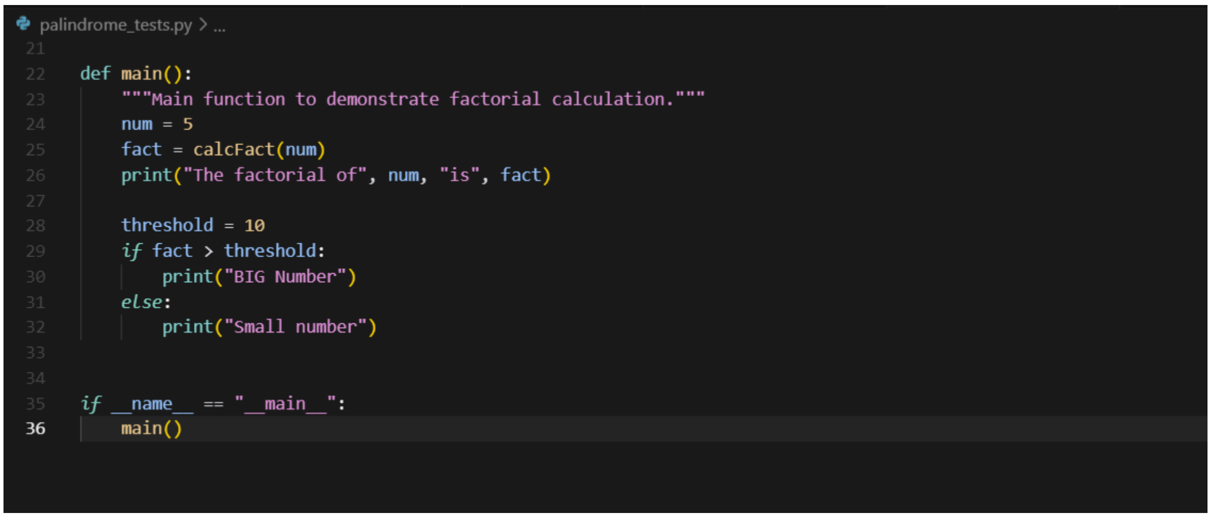
review and suggest corrections

Given program by AI:



palindrome_tests.py > ...

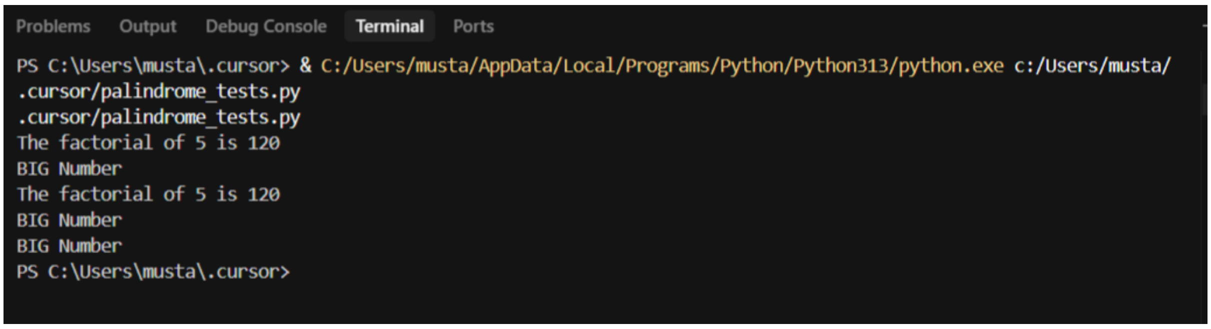
```python
def calcFact(n):
    """
    Calculate the factorial of a number.

    Args:
        n (int): The number to calculate factorial for

    Returns:
        int: The factorial of n
    """
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    if n == 0 or n == 1:
        return 1

    result = 1
    for i in range(1, n + 1):  # Fixed: added +1 to include n
        result = result * i
    return result


def main():
    """Main function to demonstrate factorial calculation."""
    num = 5
    fact = calcFact(num)
    print("The factorial of", num, "is", fact)

    threshold = 10
    if fact > threshold:
        print("BIG Number")
    else:
        print("Small number")

```



palindrome_tests.py > ...

```python

def main():
    """Main function to demonstrate factorial calculation."""
    num = 5
    fact = calcFact(num)
    print("The factorial of", num, "is", fact)

    threshold = 10
    if fact > threshold:
        print("BIG Number")
    else:
        print("Small number")


if __name__ == "__main__":
    main()
```

Output:



```
PS C:\Users\musta\.cursor> & C:/Users/musta/AppData/Local/Programs/Python/Python313/python.exe c:/Users/musta/
.cursor/palindrome_tests.py
.cursor/palindrome_tests.py
The factorial of 5 is 120
BIG Number
The factorial of 5 is 120
BIG Number
BIG Number
PS C:\Users\musta\.cursor>
```

Observation:

AI corrected the loop range to include n for proper factorial calculation.

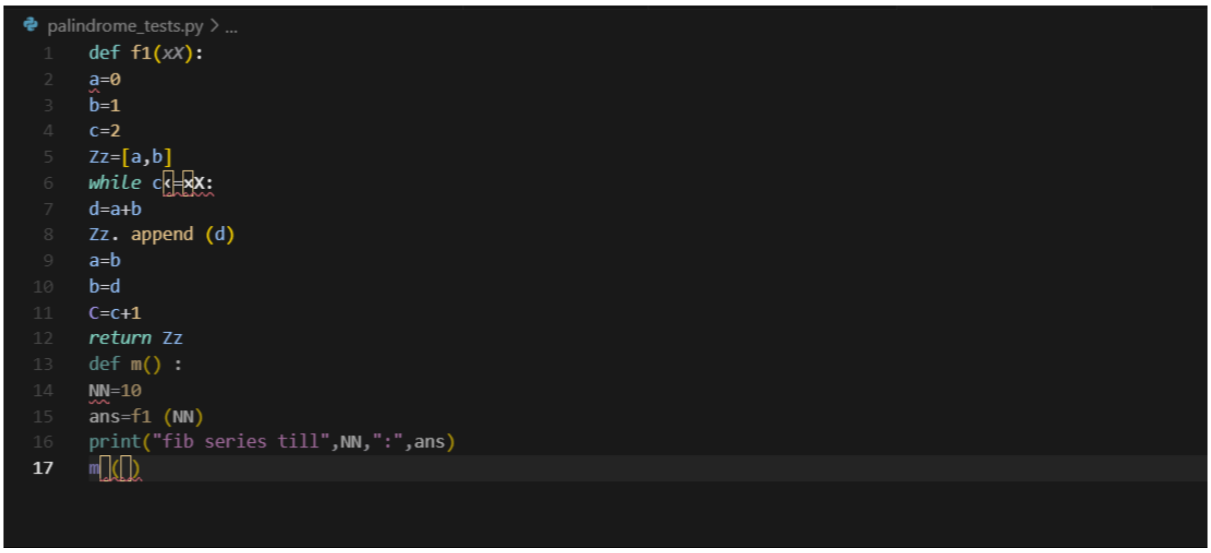
It fixed indentation and removed the unused variable.

It also corrected the missing parenthesis in the main() call so the program runs correctly.

Task Description#2 Automatic Inline Comments:

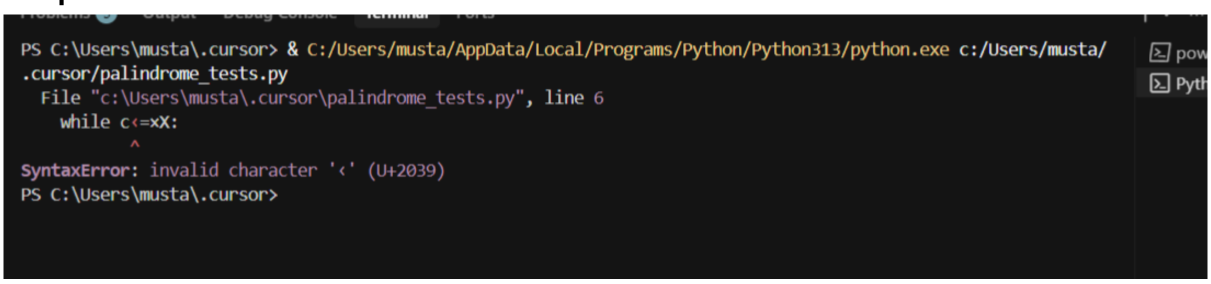
- Write the Python code for Fibonacci as shown below and execute.
- Ask AI to improve variable names, add comments, and apply PEP8 formatting (cleaned up).
- Students evaluate which suggestions improve readability most. one

Given manually written program to AI:



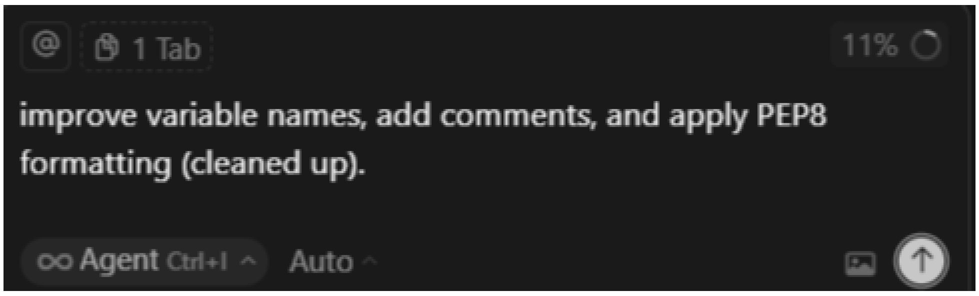
```
def f1(xX):
a=0
b=1
c=2
Zz=[a,b]
while c<=xX:
d=a+b
Zz. append (d)
a=b
b=d
C=c+1
return Zz
def m() :
NN=10
ans=f1 (NN)
print("fib series till",NN,":",ans)
m()
```

Output:



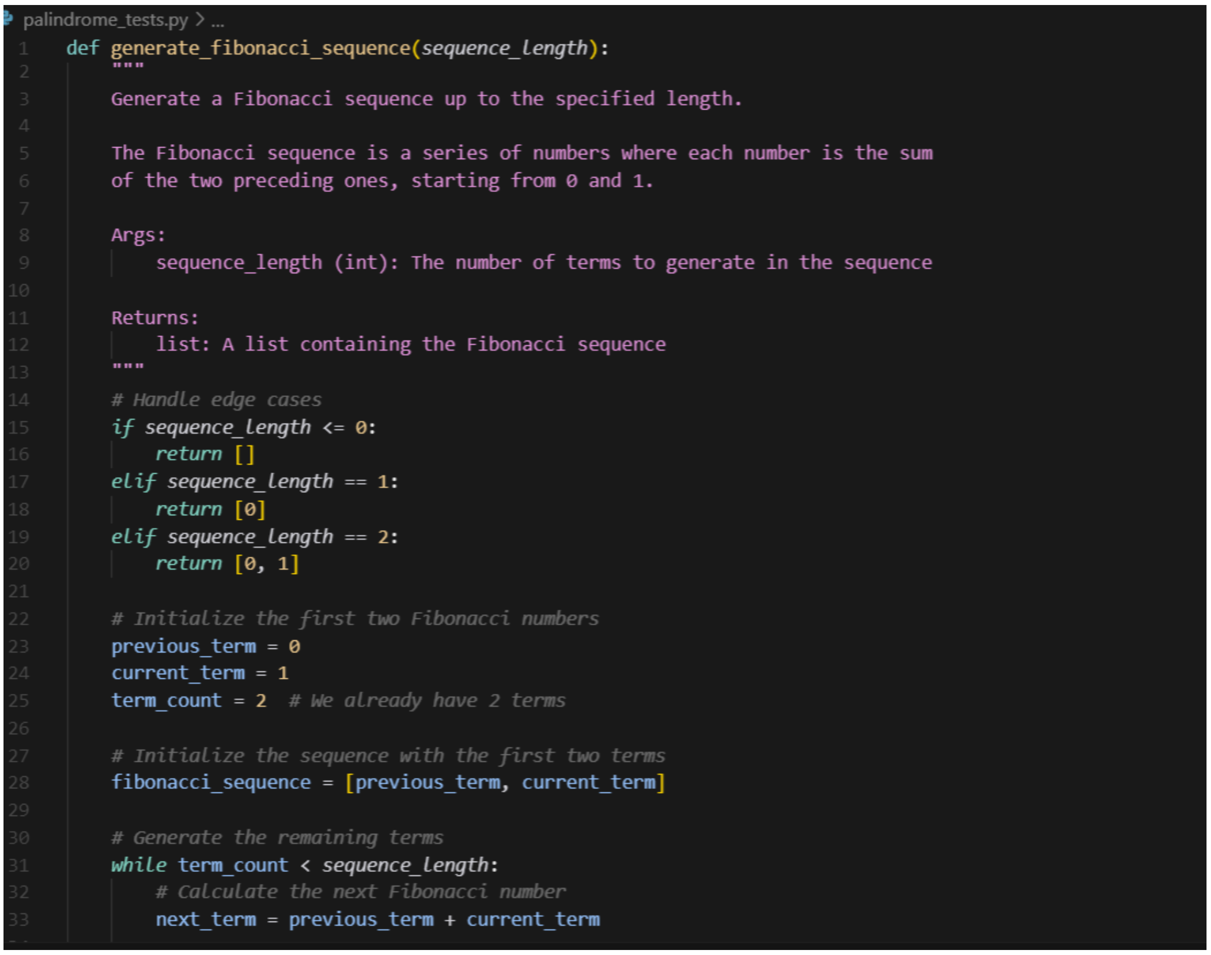
```
PS C:\Users\musta\.cursor> & C:/Users/musta/AppData/Local/Programs/Python/Python313/python.exe c:/Users/musta/
.cursor/palindrome_tests.py
  File "c:\Users\musta\.cursor\palindrome_tests.py", line 6
    while c‹=xX:
           ^
SyntaxError: invalid character '‹' (U+2039)
PS C:\Users\musta\.cursor>
```

Given prompt:

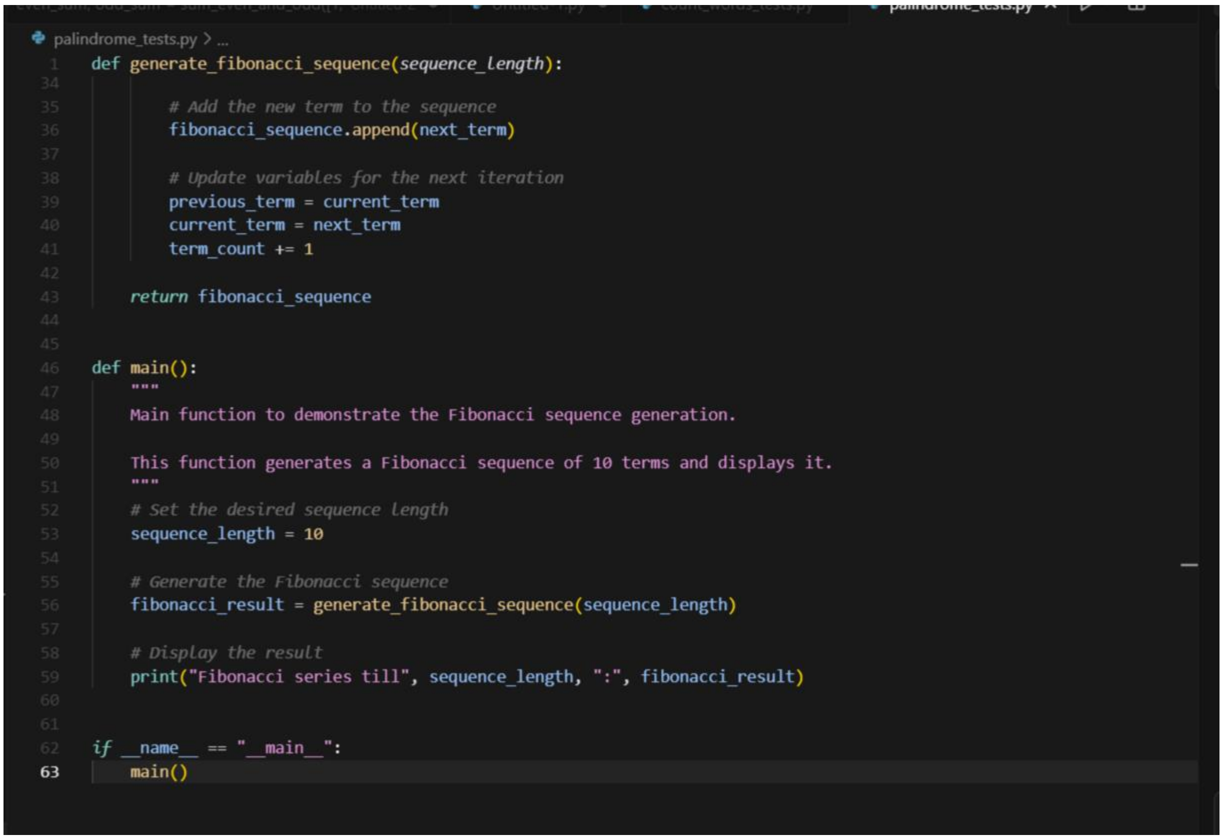


improve variable names, add comments, and apply PEP8 formatting (cleaned up).

Given program by AI:

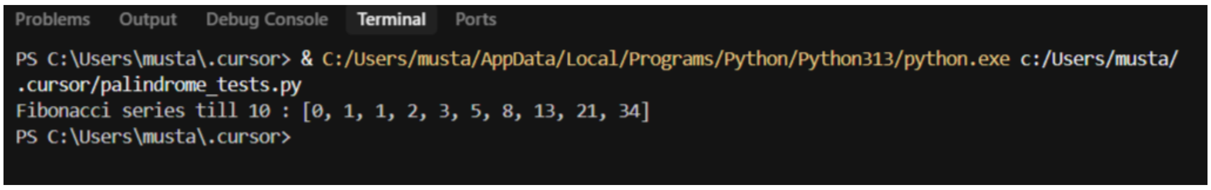


```python
def generate_fibonacci_sequence(sequence_length):
    """
    Generate a Fibonacci sequence up to the specified length.

    The Fibonacci sequence is a series of numbers where each number is the sum
    of the two preceding ones, starting from 0 and 1.

    Args:
        sequence_length (int): The number of terms to generate in the sequence

    Returns:
        list: A list containing the Fibonacci sequence
    """
    # Handle edge cases
    if sequence_length <= 0:
        return []
    elif sequence_length == 1:
        return [0]
    elif sequence_length == 2:
        return [0, 1]

    # Initialize the first two Fibonacci numbers
    previous_term = 0
    current_term = 1
    term_count = 2  # We already have 2 terms

    # Initialize the sequence with the first two terms
    fibonacci_sequence = [previous_term, current_term]

    # Generate the remaining terms
    while term_count < sequence_length:
        # Calculate the next Fibonacci number
        next_term = previous_term + current_term
```



```python
def generate_fibonacci_sequence(sequence_length):

        # Add the new term to the sequence
        fibonacci_sequence.append(next_term)

        # Update variables for the next iteration
        previous_term = current_term
        current_term = next_term
        term_count += 1

    return fibonacci_sequence


def main():
    """
    Main function to demonstrate the Fibonacci sequence generation.

    This function generates a Fibonacci sequence of 10 terms and displays it.
    """
    # Set the desired sequence length
    sequence_length = 10

    # Generate the Fibonacci sequence
    fibonacci_result = generate_fibonacci_sequence(sequence_length)

    # Display the result
    print("Fibonacci series till", sequence_length, ":", fibonacci_result)


if __name__ == "__main__":
    main()
```

Output:



```
PS C:\Users\musta\.cursor> & C:/Users/musta/AppData/Local/Programs/Python/Python313/python.exe c:/Users/musta/
.cursor/palindrome_tests.py
Fibonacci series till 10 : [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
PS C:\Users\musta\.cursor>
```

Observation:

AI improved the variable names to make the code more readable and meaningful.

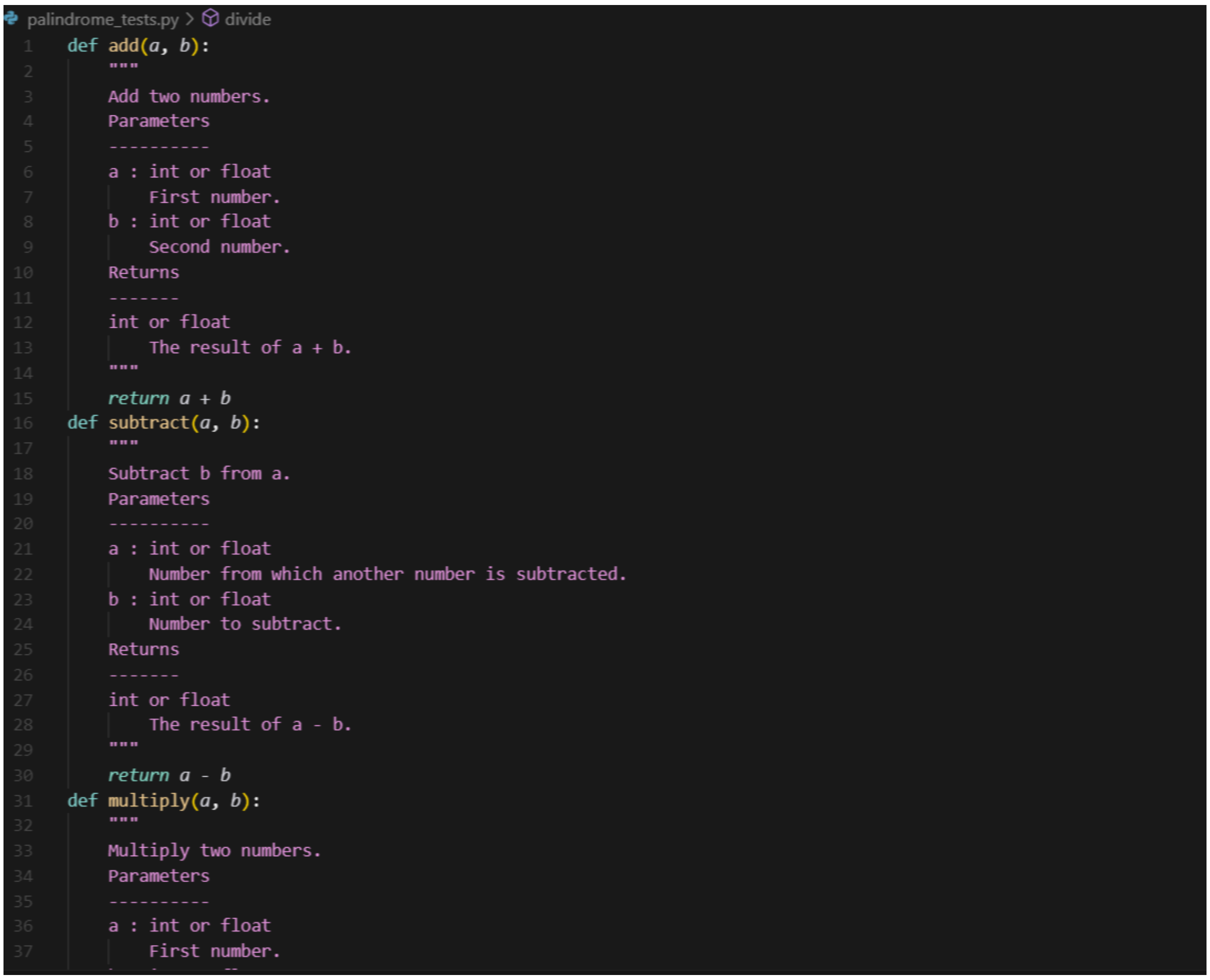
It applied PEP8 formatting with proper indentation and spacing.

It also added comments and a docstring to clearly explain the function and its purpose.

Task Description#3

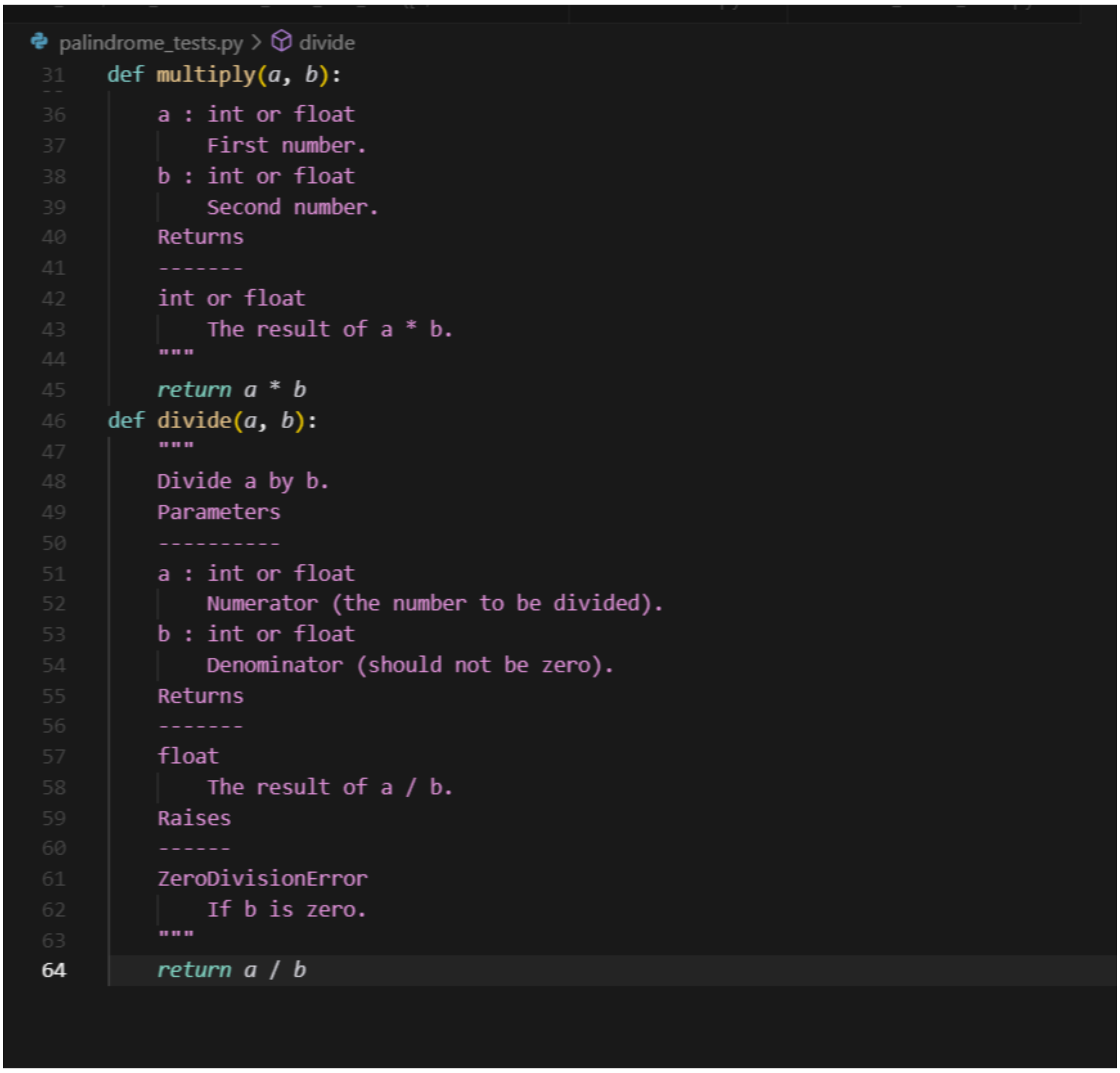
- Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).
- Incorporate manual **docstring** in code with NumPy Style
- Use AI assistance to generate a module-level docstring + individual function docstrings.
- Compare the AI-generated docstring with your manually written one.

Given program with multiple functions and manual docstring in code:



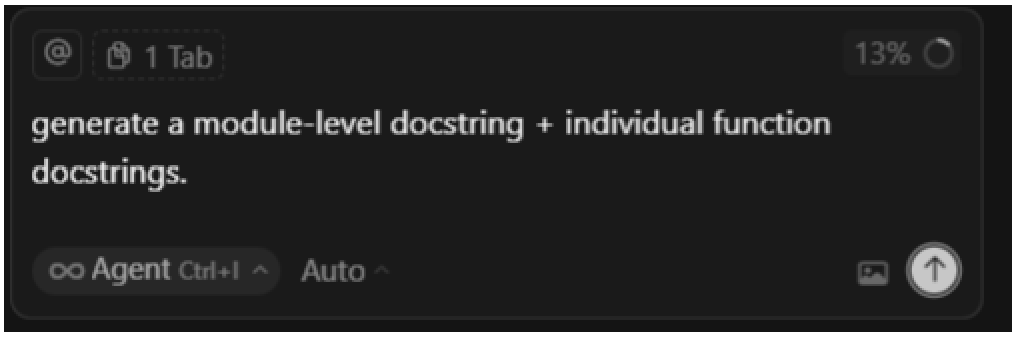
palindrome_tests.py > divide

```python
def add(a, b):
    """
    Add two numbers.
    Parameters
    ----------
    a : int or float
        First number.
    b : int or float
        Second number.
    Returns
    -------
    int or float
        The result of a + b.
    """
    return a + b
def subtract(a, b):
    """
    Subtract b from a.
    Parameters
    ----------
    a : int or float
        Number from which another number is subtracted.
    b : int or float
        Number to subtract.
    Returns
    -------
    int or float
        The result of a - b.
    """
    return a - b
def multiply(a, b):
    """
    Multiply two numbers.
    Parameters
    ----------
    a : int or float
        First number.
```


palindrome_tests.py > divide

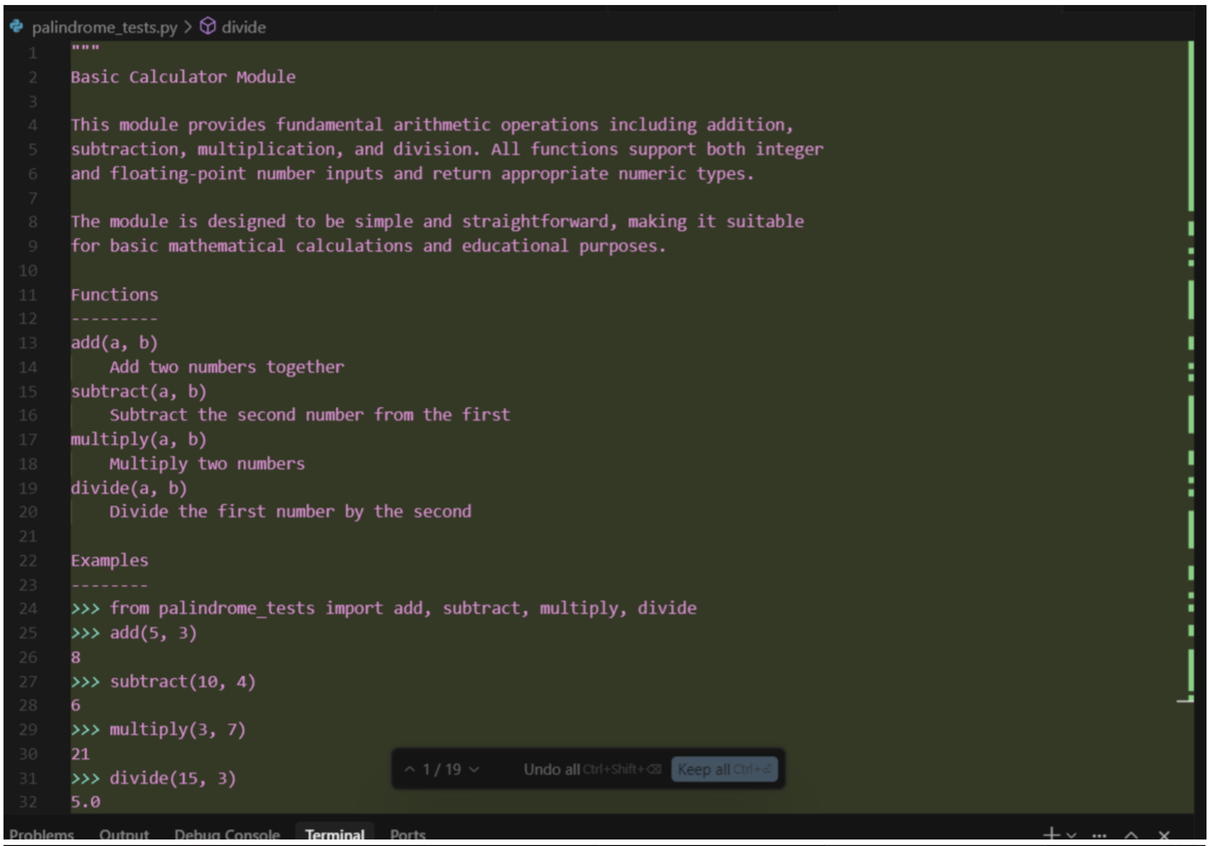
```
def multiply(a, b):
    a : int or float
        First number.
    b : int or float
        Second number.
    Returns
    -------
    int or float
        The result of a * b.
    """
    return a * b
def divide(a, b):
    """
    Divide a by b.
    Parameters
    ----------
    a : int or float
        Numerator (the number to be divided).
    b : int or float
        Denominator (should not be zero).
    Returns
    -------
    float
        The result of a / b.
    Raises
    ------
    ZeroDivisionError
        If b is zero.
    """
    return a / b
```

Now, the given prompt to the AI:

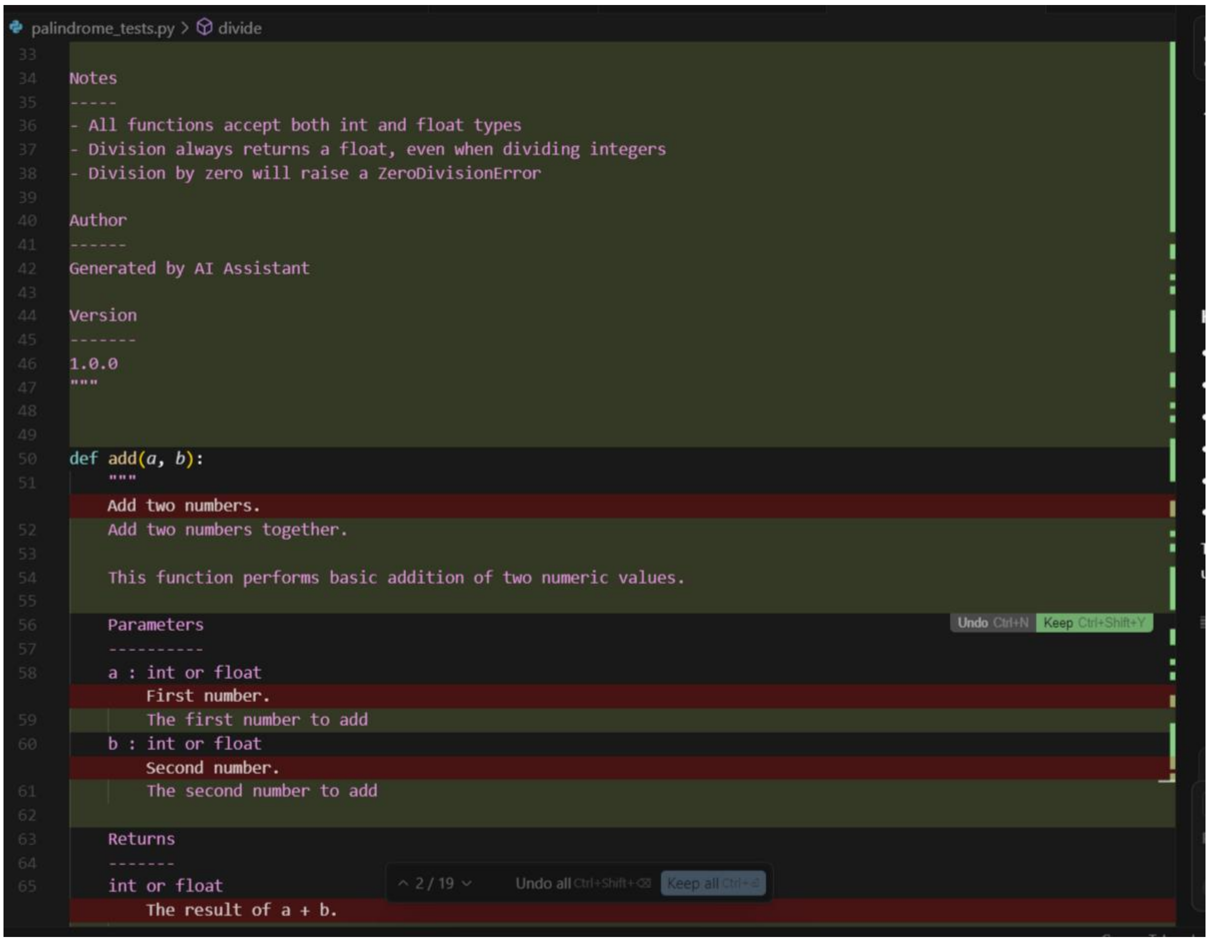


generate a module-level docstring + individual function docstrings.

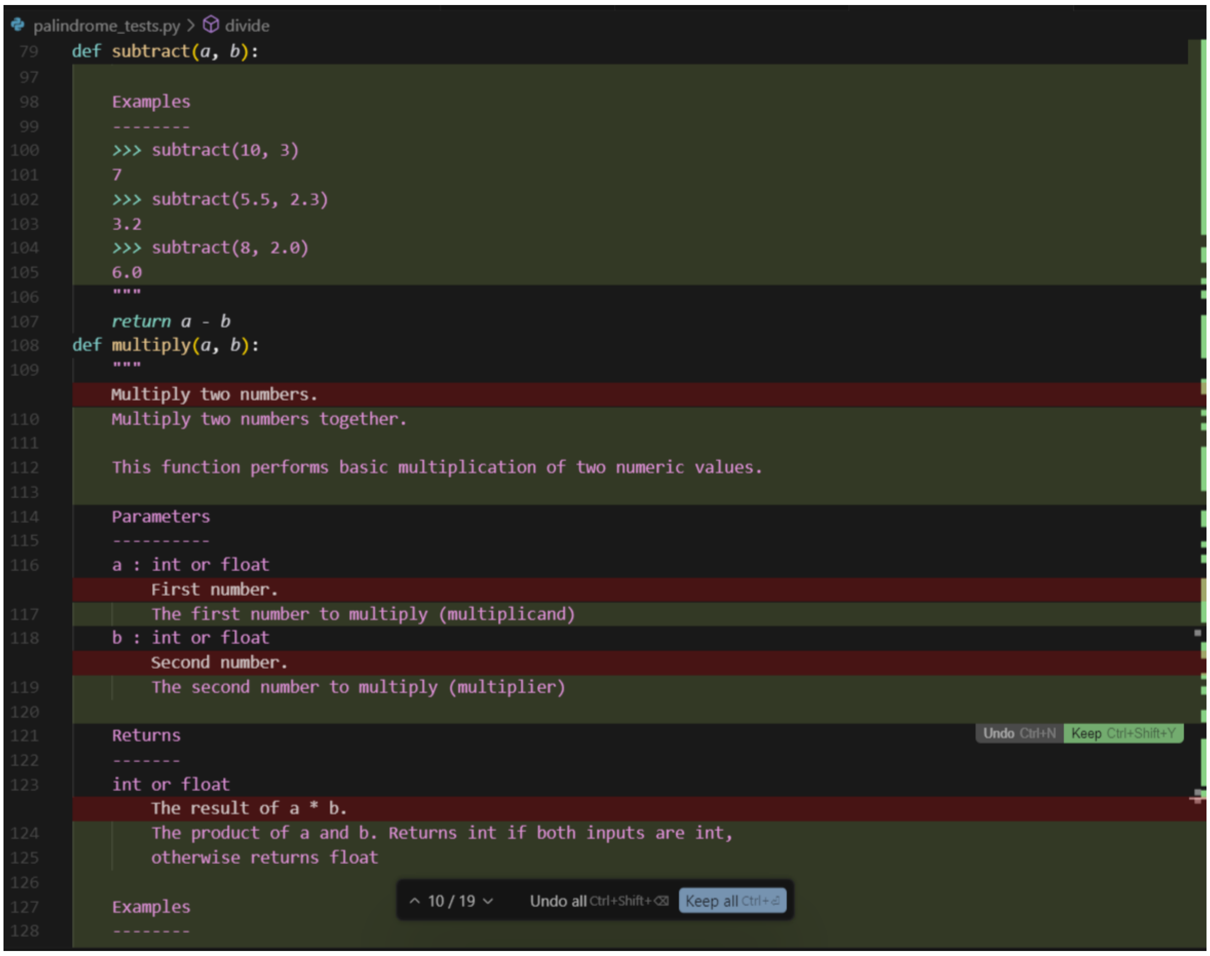
Given code by AI:



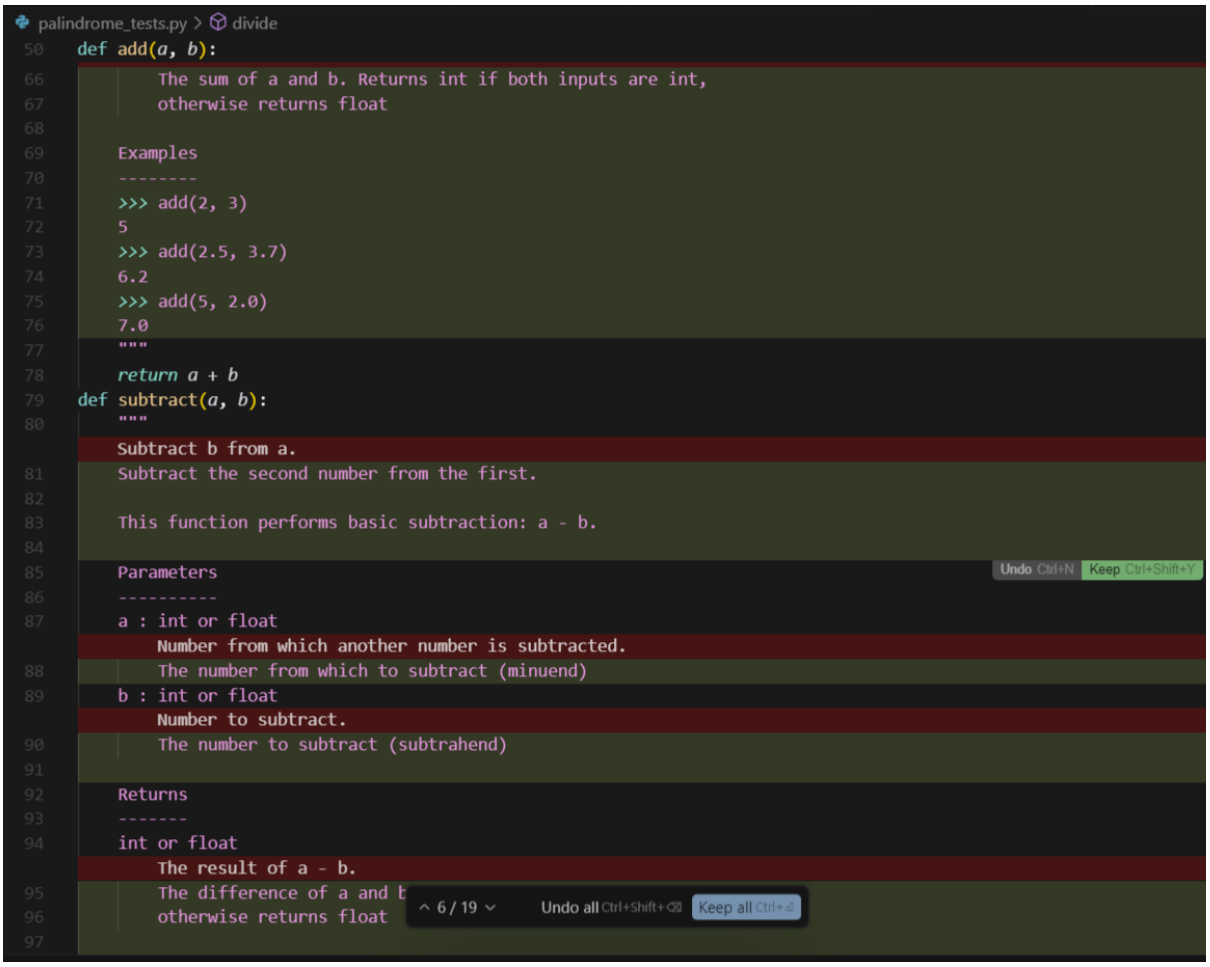
```
"""
Basic Calculator Module

This module provides fundamental arithmetic operations including addition,
subtraction, multiplication, and division. All functions support both integer
and floating-point number inputs and return appropriate numeric types.

The module is designed to be simple and straightforward, making it suitable
for basic mathematical calculations and educational purposes.

Functions
---------
add(a, b)
    Add two numbers together
subtract(a, b)
    Subtract the second number from the first
multiply(a, b)
    Multiply two numbers
divide(a, b)
    Divide the first number by the second

Examples
--------
>>> from palindrome_tests import add, subtract, multiply, divide
>>> add(5, 3)
8
>>> subtract(10, 4)
6
>>> multiply(3, 7)
21
>>> divide(15, 3)
5.0
```



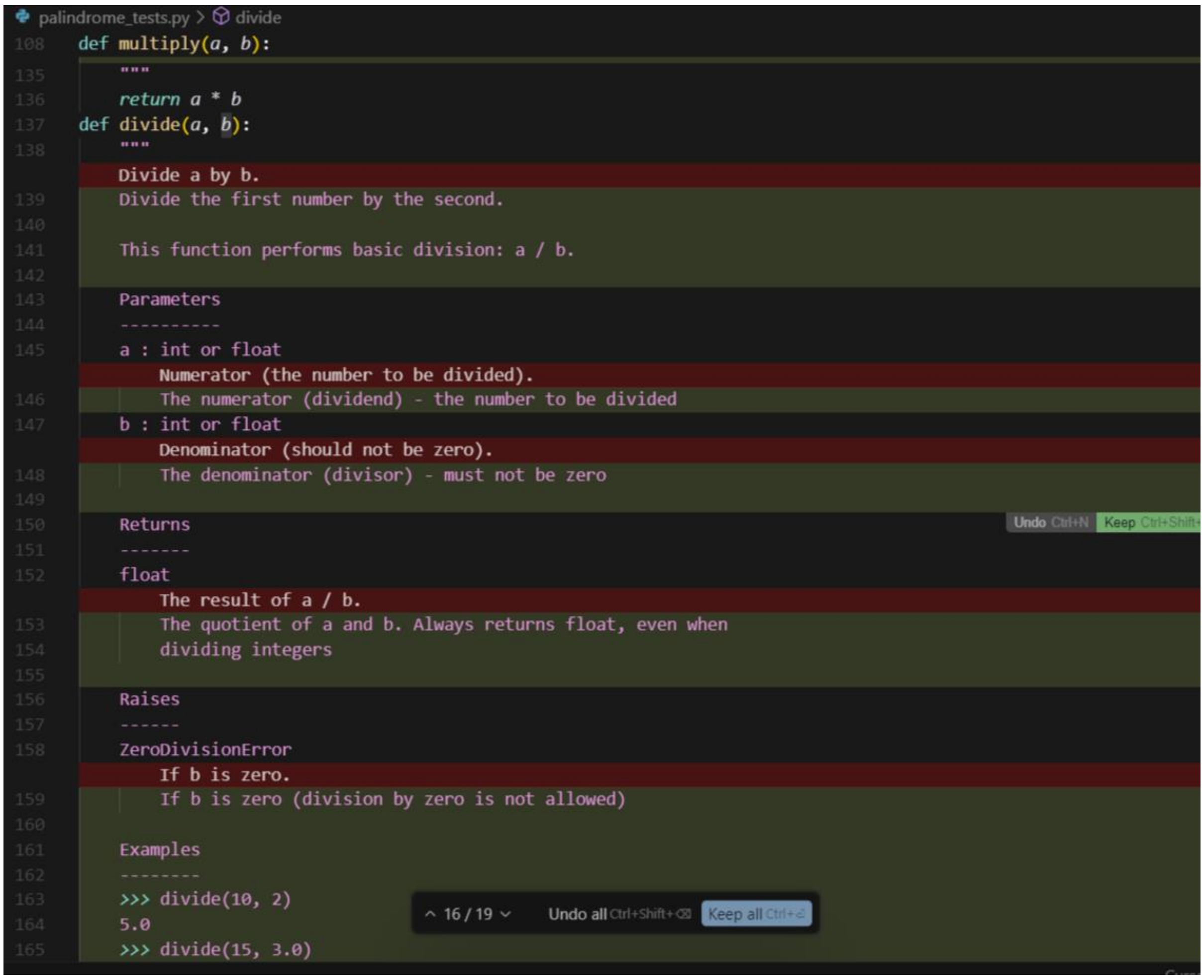
```

Notes
-----
- All functions accept both int and float types
- Division always returns a float, even when dividing integers
- Division by zero will raise a ZeroDivisionError

Author
------
Generated by AI Assistant

Version
-------
1.0.0
"""


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

    This function performs basic addition of two numeric values.

    Parameters
    ----------
    a : int or float
        First number.
        The first number to add
    b : int or float
        Second number.
        The second number to add

    Returns
    -------
    int or float
        The result of a + b.
```



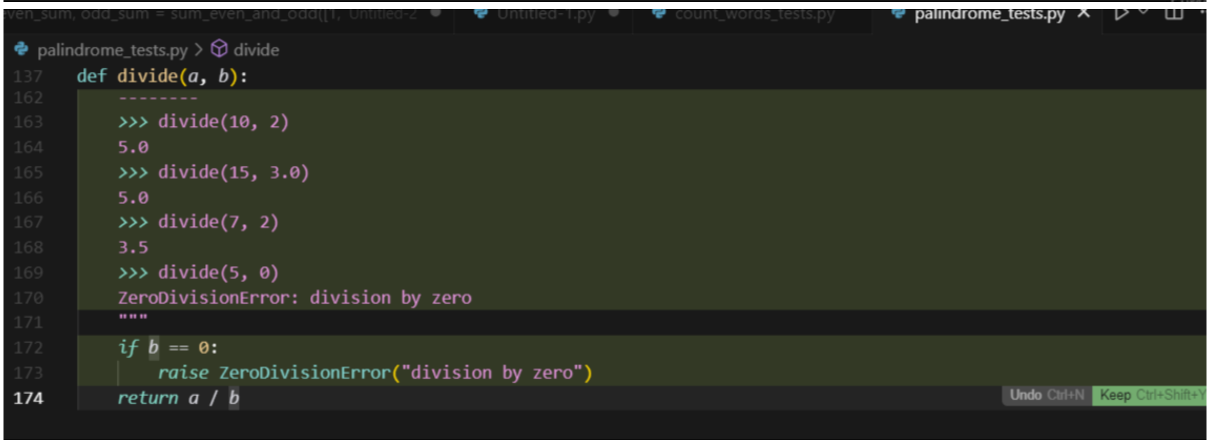
```
def subtract(a, b):

    Examples
    --------
    >>> subtract(10, 3)
    7
    >>> subtract(5.5, 2.3)
    3.2
    >>> subtract(8, 2.0)
    6.0
    """
    return a - b
def multiply(a, b):
    """
    Multiply two numbers.
    Multiply two numbers together.

    This function performs basic multiplication of two numeric values.

    Parameters
    ----------
    a : int or float
        First number.
        The first number to multiply (multiplicand)
    b : int or float
        Second number.
        The second number to multiply (multiplier)

    Returns
    -------
    int or float
        The result of a * b.
        The product of a and b. Returns int if both inputs are int,
        otherwise returns float

    Examples
    --------
```



```
def add(a, b):
        The sum of a and b. Returns int if both inputs are int,
        otherwise returns float

    Examples
    --------
    >>> add(2, 3)
    5
    >>> add(2.5, 3.7)
    6.2
    >>> add(5, 2.0)
    7.0
    """
    return a + b
def subtract(a, b):
    """
    Subtract b from a.
    Subtract the second number from the first.

    This function performs basic subtraction: a - b.

    Parameters
    ----------
    a : int or float
        Number from which another number is subtracted.
        The number from which to subtract (minuend)
    b : int or float
        Number to subtract.
        The number to subtract (subtrahend)

    Returns
    -------
    int or float
        The result of a - b.
        The difference of a and b
        otherwise returns float
```



```python
def multiply(a, b):
    """
    return a * b
def divide(a, b):
    """
    Divide a by b.
    Divide the first number by the second.

    This function performs basic division: a / b.

    Parameters
    ----------
    a : int or float
        Numerator (the number to be divided).
        The numerator (dividend) - the number to be divided
    b : int or float
        Denominator (should not be zero).
        The denominator (divisor) - must not be zero

    Returns
    -------
    float
        The result of a / b.
        The quotient of a and b. Always returns float, even when
        dividing integers

    Raises
    ------
    ZeroDivisionError
        If b is zero.
        If b is zero (division by zero is not allowed)

    Examples
    --------
    >>> divide(10, 2)
    5.0
    >>> divide(15, 3.0)
```



```python
def divide(a, b):
    --------
    >>> divide(10, 2)
    5.0
    >>> divide(15, 3.0)
    5.0
    >>> divide(7, 2)
    3.5
    >>> divide(5, 0)
    ZeroDivisionError: division by zero
    """
    if b == 0:
        raise ZeroDivisionError("division by zero")
    return a / b
```

Comparison:

AI-Generated Docstring:

- Very formal and polished.
- Uses full sentences like *“This module provides basic arithmetic operations…”*.
- Adds sections like *Functions* with descriptions.
- Looks professional but can feel a bit too “perfect” or lengthy.

Manually Written Docstring:

- Shorter and to the point.
- Uses simple wording like *“Add two numbers”*, *“Subtract b from a”*.
- Easier to read and less formal.
- More in line with how a student would normally explain code.