

**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.

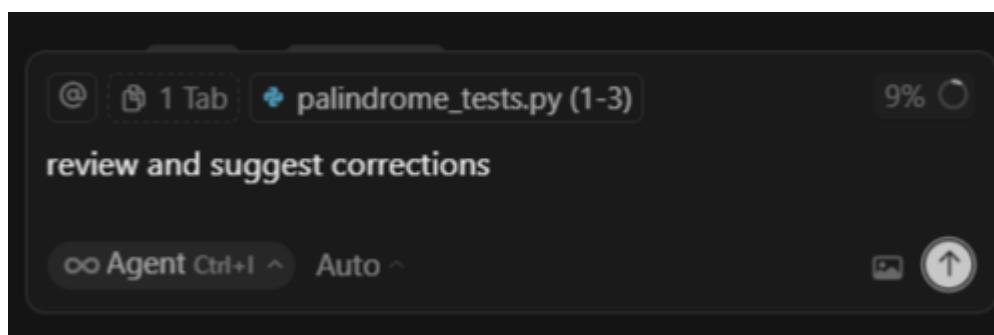
### Given program to ai which was manually written:

```
sum odd sum = sum even and odd() ... Untitled-1.py count_words_tests.py palindrome_tests.py
palindrome_tests.py > ...
1 def calcFact(n) :
2     result=1
3     x=0
4     for i in range(1,n): result=result*i
5     return result
6     def main():
7         num = 5
8         FACT = calcFact (num)
9         print("the factorial of",num,"is" ,FACT)
10        t=10
11        if FACT>10:
12            print("BIG Number")
13        else:
14            print ("small number")
15        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:

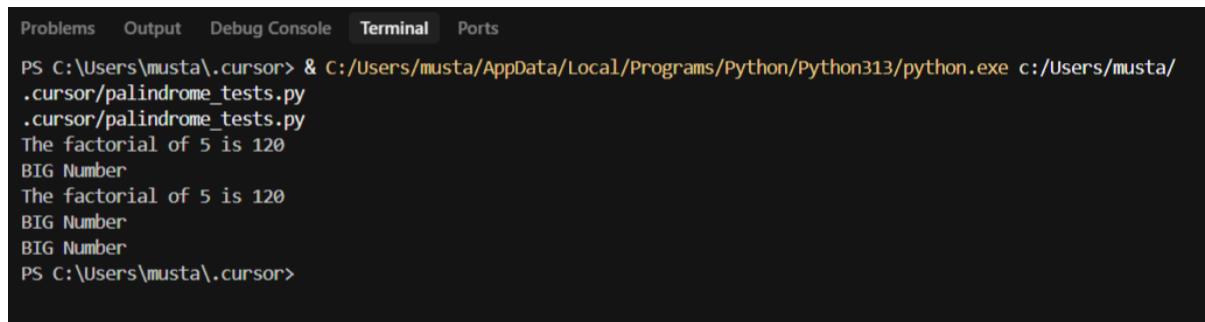


## Given program by AI:

```
.isFile("palindrome_tests.py") > ...
1  def calcFact(n):
2      """
3          Calculate the factorial of a number.
4
5      Args:
6          n (int): The number to calculate factorial for
7
8      Returns:
9          int: The factorial of n
10         """
11     if n < 0:
12         raise ValueError("Factorial is not defined for negative numbers")
13     if n == 0 or n == 1:
14         return 1
15
16     result = 1
17     for i in range(1, n + 1): # Fixed: added +1 to include n
18         result = result * i
19     return result
20
21
22 def main():
23     """Main function to demonstrate factorial calculation."""
24     num = 5
25     fact = calcFact(num)
26     print("The factorial of", num, "is", fact)
27
28     threshold = 10
29     if fact > threshold:
30         print("BIG Number")
31     else:
32         print("Small number")
33
```

```
.isFile("palindrome_tests.py") > ...
21
22 def main():
23     """Main function to demonstrate factorial calculation."""
24     num = 5
25     fact = calcFact(num)
26     print("The factorial of", num, "is", fact)
27
28     threshold = 10
29     if fact > threshold:
30         print("BIG Number")
31     else:
32         print("Small number")
33
34
35 if __name__ == "__main__":
36     main()
```

## Output:



```
Problems Output Debug Console Terminal Ports
PS C:\Users\musta\.cursor> & C:/Users/musta/AppData/Local/Programs/Python/Python313/python.exe c:/Users/musta/.cursor/palindrome_tests.py
.ccursor/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:

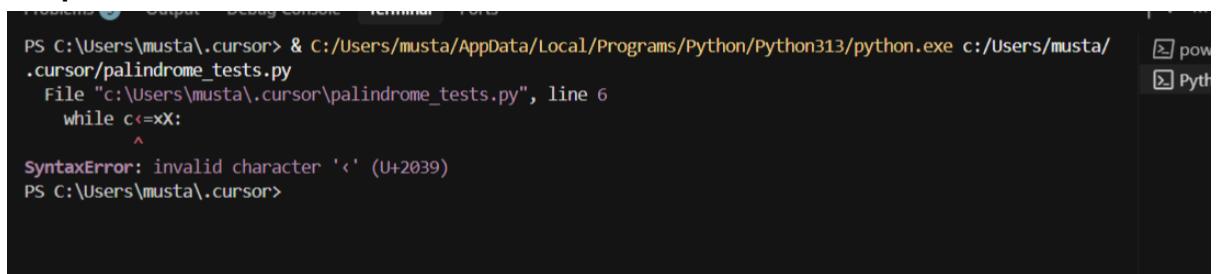
- Write the Python code for Fibonacci as shown below and execute.
- Ask AI to improve variable names, add comments, and apply PEP8 formatting (cleaned up).
- Students evaluate which suggestions improve readability most. one

## Given manually written program to AI:



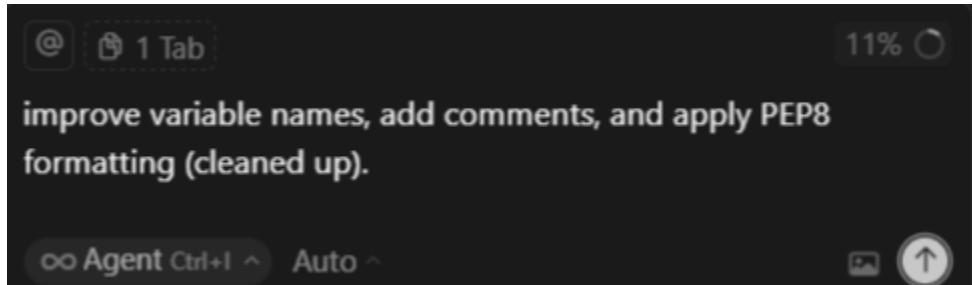
```
palindrome_tests.py > ...
1 def f1(xx):
2     a=0
3     b=1
4     c=2
5     zz=[a,b]
6     while c<xx:
7         d=a+b
8         zz.append (d)
9         a=b
10        b=d
11        c=c+1
12    return zz
13 def m() :
14     NN=10
15     ans=f1 (NN)
16     print("fib series till",NN,":",ans)
17     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:



## Given program by AI:

```
# palindrome_tests.py > ...
1  def generate_fibonacci_sequence(sequence_length):
2      """
3          Generate a Fibonacci sequence up to the specified length.
4
5          The Fibonacci sequence is a series of numbers where each number is the sum
6          of the two preceding ones, starting from 0 and 1.
7
8          Args:
9              sequence_length (int): The number of terms to generate in the sequence
10
11         Returns:
12             list: A list containing the Fibonacci sequence
13             """
14
15         # Handle edge cases
16         if sequence_length <= 0:
17             return []
18         elif sequence_length == 1:
19             return [0]
20         elif sequence_length == 2:
21             return [0, 1]
22
23         # Initialize the first two Fibonacci numbers
24         previous_term = 0
25         current_term = 1
26         term_count = 2 # We already have 2 terms
27
28         # Initialize the sequence with the first two terms
29         fibonacci_sequence = [previous_term, current_term]
29
30         # Generate the remaining terms
31         while term_count < sequence_length:
32             # Calculate the next Fibonacci number
33             next_term = previous_term + current_term
```

```
palindrome_tests.py > ...
1  def generate_fibonacci_sequence(sequence_length):
2
3      # Add the new term to the sequence
4      fibonacci_sequence.append(next_term)
5
6      # Update variables for the next iteration
7      previous_term = current_term
8      current_term = next_term
9      term_count += 1
10
11  return fibonacci_sequence
12
13
14
15
16  def main():
17      """
18          Main function to demonstrate the Fibonacci sequence generation.
19
20          This function generates a Fibonacci sequence of 10 terms and displays it.
21      """
22
23      # Set the desired sequence length
24      sequence_length = 10
25
26      # Generate the Fibonacci sequence
27      fibonacci_result = generate_fibonacci_sequence(sequence_length)
28
29      # Display the result
30      print("Fibonacci series till", sequence_length, ":", fibonacci_result)
31
32
33  if __name__ == "__main__":
34      main()
```

## Output:

```
Problems Output Debug Console Terminal Ports
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

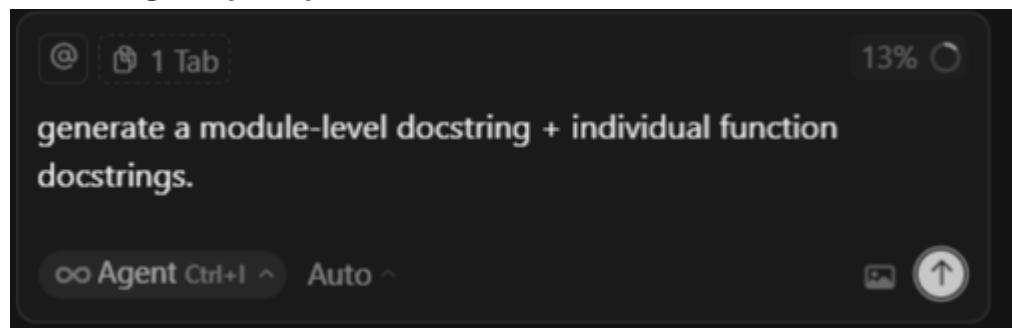
- Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).
- Incorporate manual **docstring** in code with NumPy Style
- Use AI assistance to generate a module-level docstring + individual function docstrings.
- Compare the AI-generated docstring with your manually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

### Given program with multiple functions and manual docstring in code:

```
palindrome_tests.py > divide
1 def add(a, b):
2     """
3         Add two numbers.
4         Parameters
5         -----
6         a : int or float
7             First number.
8         b : int or float
9             Second number.
10        Returns
11        -----
12        int or float
13            The result of a + b.
14        """
15        return a + b
16 def subtract(a, b):
17     """
18         Subtract b from a.
19         Parameters
20         -----
21         a : int or float
22             Number from which another number is subtracted.
23         b : int or float
24             Number to subtract.
25         Returns
26         -----
27         int or float
28             The result of a - b.
29        """
30        return a - b
31 def multiply(a, b):
32     """
33         Multiply two numbers.
34         Parameters
35         -----
36         a : int or float
37             First number.
```

```
palindrome_tests.py > divide
31 def multiply(a, b):
32     """
33     a : int or float
34         First number.
35     b : int or float
36         Second number.
37     Returns
38     -----
39     int or float
40         The result of a * b.
41     """
42     return a * b
43
44 def divide(a, b):
45     """
46     Divide a by b.
47     Parameters
48     -----
49     a : int or float
50         Numerator (the number to be divided).
51     b : int or float
52         Denominator (should not be zero).
53     Returns
54     -----
55     float
56         The result of a / b.
57     Raises
58     -----
59     ZeroDivisionError
60         If b is zero.
61     """
62     return a / b
```

Now, the given prompt to the AI:



## Given code by AI:

```
palindrome_tests.py > divide
1 """
2     Basic Calculator Module
3
4     This module provides fundamental arithmetic operations including addition,
5     subtraction, multiplication, and division. All functions support both integer
6     and floating-point number inputs and return appropriate numeric types.
7
8     The module is designed to be simple and straightforward, making it suitable
9     for basic mathematical calculations and educational purposes.
10
11    Functions
12    -----
13    add(a, b)
14        Add two numbers together
15    subtract(a, b)
16        Subtract the second number from the first
17    multiply(a, b)
18        Multiply two numbers
19    divide(a, b)
20        Divide the first number by the second
21
22    Examples
23    -----
24    >>> from palindrome_tests import add, subtract, multiply, divide
25    >>> add(5, 3)
26    8
27    >>> subtract(10, 4)
28    6
29    >>> multiply(3, 7)
30    21
31    >>> divide(15, 3)
32    5.0
```

```
palindrome_tests.py > divide
33
34    Notes
35    -----
36    - All functions accept both int and float types
37    - Division always returns a float, even when dividing integers
38    - Division by zero will raise a ZeroDivisionError
39
40    Author
41    -----
42    Generated by AI Assistant
43
44    Version
45    -----
46    1.0.0
47    """
48
49
50    def add(a, b):
51        """
52            Add two numbers.
53            Add two numbers together.
54
55            This function performs basic addition of two numeric values.
56
57            Parameters
58            -----
59            a : int or float
60                First number.
61                The first number to add
62            b : int or float
63                Second number.
64                The second number to add
65
66            Returns
67            -----
68            int or float
69                The result of a + b.
```

```
❖ palindrome_tests.py > ⌂ divide
79  def subtract(a, b):
80
81      Examples
82      -----
83      >>> subtract(10, 3)
84      7
85      >>> subtract(5.5, 2.3)
86      3.2
87      >>> subtract(8, 2.0)
88      6.0
89      """
90
91      return a - b
92  def multiply(a, b):
93      """
94
95      Multiply two numbers.
96      Multiply two numbers together.
97
98      This function performs basic multiplication of two numeric values.
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
```

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

-----

Undo Ctrl+N Keep Ctrl+Shift+Y

^ 10 / 19 ^ Undo all Ctrl+Shift+Q Keep all Ctrl+Shift+Y

```
palindrome_tests.py > divide
50 def add(a, b):
51     """The sum of a and b. Returns int if both inputs are int,
52     otherwise returns float
53
54     Examples
55     -----
56     >>> add(2, 3)
57     5
58     >>> add(2.5, 3.7)
59     6.2
60     >>> add(5, 2.0)
61     7.0
62     """
63     return a + b
64 def subtract(a, b):
65     """
66     Subtract b from a.
67     Subtract the second number from the first.
68
69     This function performs basic subtraction: a - b.
70
71     Parameters
72     -----
73     a : int or float
74         Number from which another number is subtracted.
75         The number from which to subtract (minuend)
76     b : int or float
77         Number to subtract.
78         The number to subtract (subtrahend)
79
80     Returns
81     -----
82     int or float
83         The result of a - b.
84         The difference of a and b
85         otherwise returns float
```

^ 6 / 19 ^

Undo all Ctrl+Shift+Q Keep all Ctrl+A

```

❷ palindrome_tests.py > ⌂ divide
108  def multiply(a, b):
109      """
110          return a * b
111      """
112  def divide(a, b):
113      """
114          Divide a by b.
115          Divide the first number by the second.
116
117          This function performs basic division: a / b.
118
119          Parameters
120          -----
121
122          a : int or float
123              Numerator (the number to be divided).
124              The numerator (dividend) - the number to be divided
125
126          b : int or float
127              Denominator (should not be zero).
128              The denominator (divisor) - must not be zero
129
130          Returns
131          -----
132
133          float
134              The result of a / b.
135              The quotient of a and b. Always returns float, even when
136              dividing integers
137
138          Raises
139          -----
140
141          ZeroDivisionError
142              If b is zero.
143              If b is zero (division by zero is not allowed)
144
145          Examples
146          -----
147
148          >>> divide(10, 2)
149          5.0
150
151          >>> divide(15, 3.0)
152
153
154
155
156
157
158
159
160
161
162
163
164
165
```

venv\_sum, odd\_sum = sum\_even\_and\_odd([1, 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100], [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99])

```

❷ palindrome_tests.py > ⌂ divide
137  def divide(a, b):
138      """
139          >>> divide(10, 2)
140          5.0
141          >>> divide(15, 3.0)
142          5.0
143          >>> divide(7, 2)
144          3.5
145          >>> divide(5, 0)
146          ZeroDivisionError: division by zero
147          """
148
149          if b == 0:
150              raise ZeroDivisionError("division by zero")
151
152          return a / b
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
```

## 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.