

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		Assignment Type: Lab	Academic Year:2025-2026
Course Coordinator Name		Venkataramana Veeramsetty	
Instructor(s) Name		Dr. V. Venkataramana (Co-ordinator)	
		Dr. T. Sampath Kumar	
		Dr. Pramoda Patro	
		Dr. Brij Kishor Tiwari	
		Dr.J.Ravichander	
		Dr. Mohammand Ali Shaik	
		Dr. Anirodh Kumar	
		Mr. S.Naresh Kumar	
		Dr. RAJESH VELPULA	
		Mr. Kundhan Kumar	
		Ms. Ch.Rajitha	
		Mr. M Prakash	
		Mr. B.Raju	
		Intern 1 (Dharma teja)	
		Intern 2 (Sai Prasad)	
		Intern 3 (Sowmya)	
		NS_2 (Mounika)	
Course Code	24CS002PC215	Course Title	AI Assisted Coding
Year/Sem	II/I	Regulation	R24
Date and Day of Assignment	Week5 - Monday	Time(s)	
Duration	2 Hours	Applicable to Batches	
AssignmentNumber: 9.1(Present assignment number)/24(Total number of assignments)			
Q.No.	Question		Expected Time to complete
1	Lab 9 – Code Review and Quality: Using AI to improve code quality and readability Lab Objectives <ul style="list-style-type: none"> ● Inline comments 		Week5 - Monday

	<ul style="list-style-type: none">• Docstrings• Auto-documentation tools• AI-assisted summarization <p>Task Description #1 (AI-Assisted Bug Detection)</p> <p>Scenario: A junior developer wrote the following Python function to calculate factorials:</p> <pre>def factorial(n): result = 1 for i in range(1, n): result = result * i return result</pre> <ul style="list-style-type: none">• Run the code and test it with factorial(5) (expected output = 120).• Use AI (prompting) to review this code and identify the bug.• Ask AI to suggest corrections and rewrite the code.• Compare AI's corrected code with your own fix. <p><u>Code:</u></p>	
--	---	--

Q Commands | + Code + Text | ▶ Run all ▼

[16]
✓ 0s

```
def factorial(n):  
    result = 1  
    for i in range(1, n):  
        result = result * i  
    return result  
print(factorial(5))
```

↩ 24

Manual Code:

Q Commands | + Code + Text | ▶ Run all ▼

[4]
✓ 29s

```
def factorial(n):  
    result = 1  
    for i in range(1, n + 1):  
        result = result * i  
    return result  
  
# user input  
n = int(input("Enter a non-negative integer: "))  
  
if n < 0:  
    print("Factorial is not defined for negative numbers.")  
else:  
    print(f"Factorial of {n} is {factorial(n)}")
```

↩ Enter a non-negative integer: 5
Factorial of 5 is 120

Prompt:

Generate the code and identify the bug and suggest corrections.

AI-Generated Code:

CO

Untitled21.ipynb

☆

☁

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all ▼

☰

🔍

⏪

🔑

📁

[7]

✓ 2s

▶

```
def factorial(n):  
    """  
    Calculates the factorial of a non-negative integer.  
  
    Args:  
        n: An integer for which the factorial is to be calculated.  
  
    Returns:  
        The factorial of n, or a message if n is negative.  
    """  
    if n < 0:  
        return "Factorial is not defined for negative numbers"  
    result = 1  
    for i in range(1, n + 1):  
        result *= i  
    return result  
  
# Get input from the user  
try:  
    num = int(input("Enter a non-negative integer to calculate its factorial: "))  
    print(f"The factorial of {num} is {factorial(num)}")  
except ValueError:  
    print("Invalid input. Please enter an integer.")
```

↵

Enter a non-negative integer to calculate its factorial: 5
The factorial of 5 is 120

Comparison AI and manual code:

- 1. Both use for i in range(1, n+1).
- 2. Manual code prints an error message; AI code raises a ValueError.
- 3. Manual code is beginner-friendly; AI code is more descriptive with comments.

Task Description #2 (Improving Readability & Documentation)

Scenario: The following code works but is poorly written:

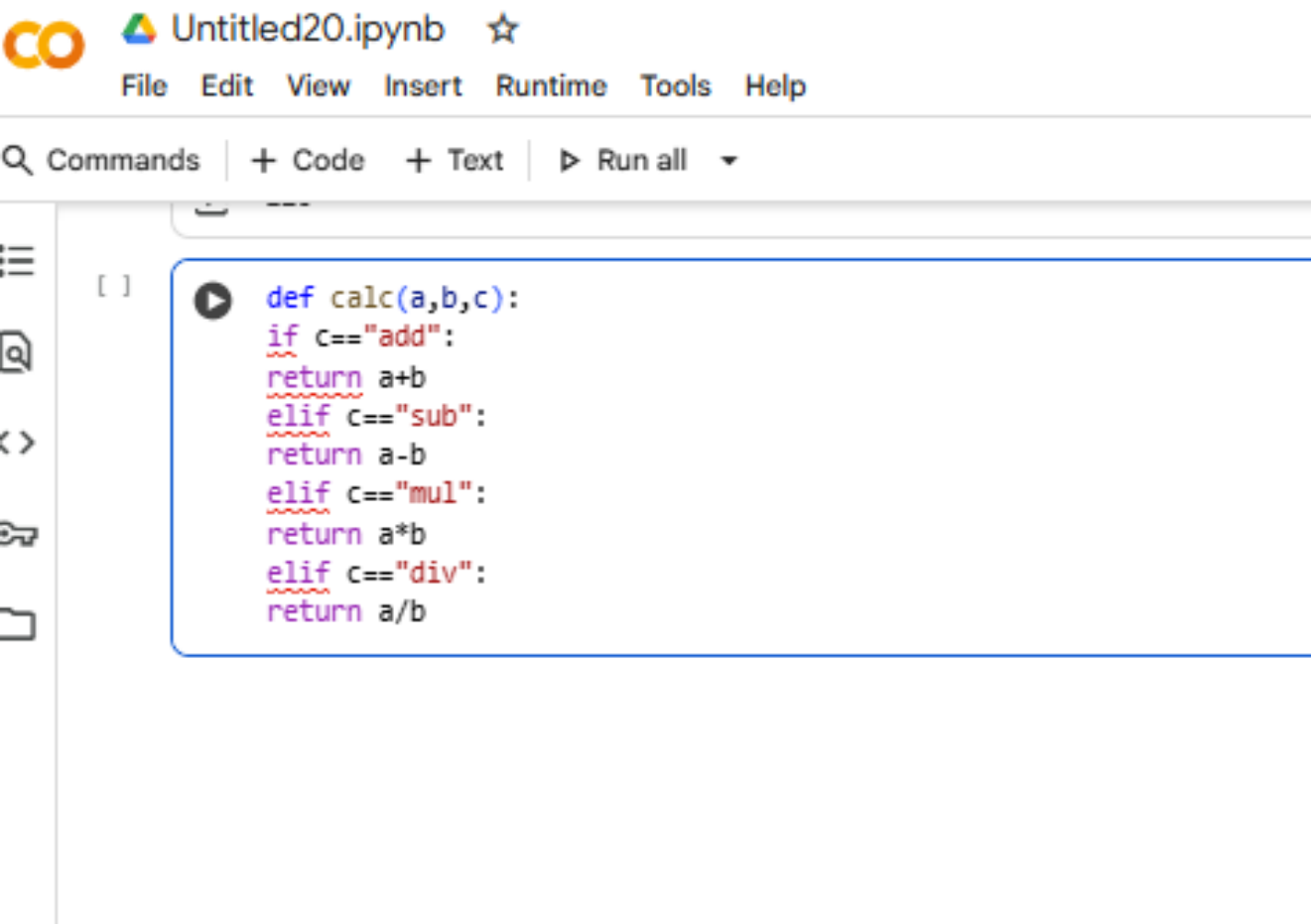
```
def calc(a,b,c):  
    if c=="add":  
        return a+b  
    elif c=="sub":  
        return a-b  
    elif c=="mul":  
        return a*b  
    elif c=="div":  
        return a/b
```

- Use AI to review this code for readability, naming, and documentation issues.
- Prompt AI to rewrite the code with:
 - Clear function & variable names.
 - Proper docstrings.
 - Exception handling for division by zero.
- Compare the before-and-after versions to evaluate AI's contribution.

Prompt:

Rewrite the code with clear function and variable names and use proper docstrings and exception handling for division by zero.

Before Versions:



```
def calc(a,b,c):  
    if c=="add":  
        return a+b  
    elif c=="sub":  
        return a-b  
    elif c=="mul":  
        return a*b  
    elif c=="div":  
        return a/b
```

After AI versions:

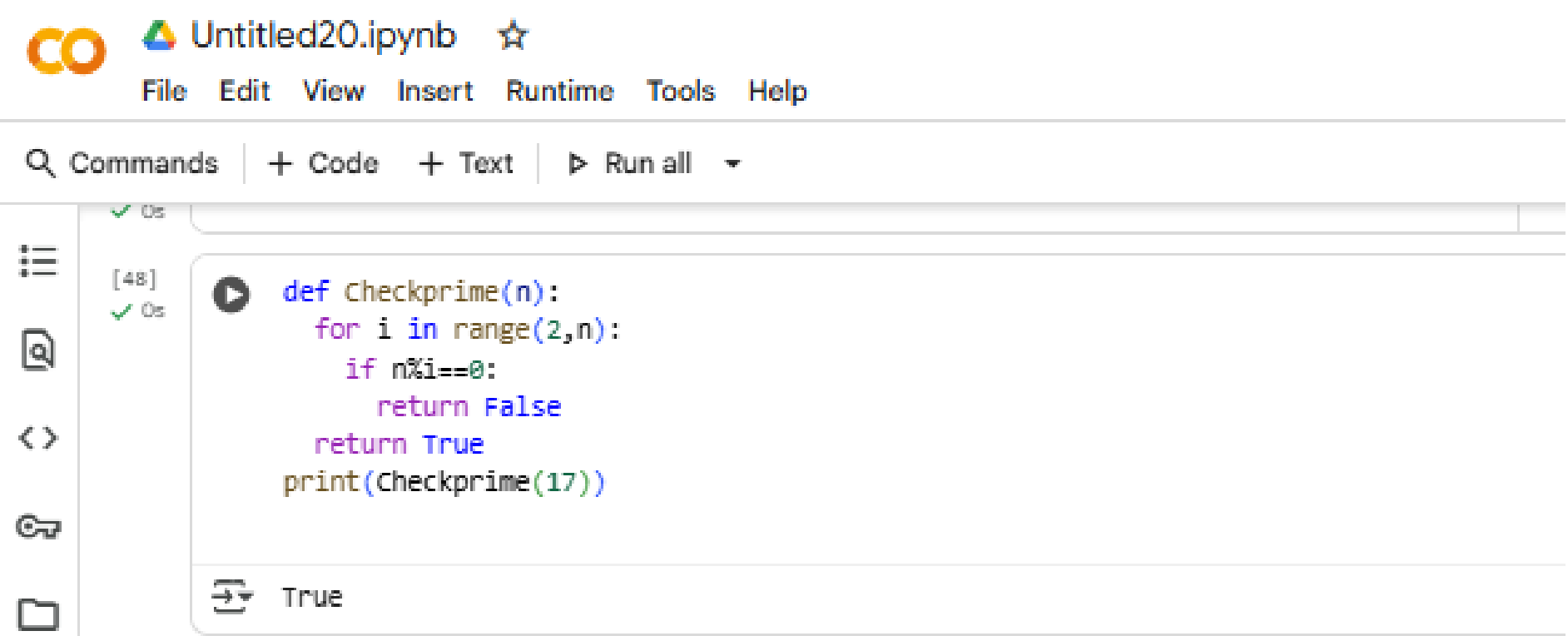

```
def Checkprime(n):
    for i in range(2,n):
        if n%i==0:
            return False
    return True
```

- Run this code and verify correctness.
- Use AI to perform a code quality review for PEP8 compliance.
- Prompt AI to return a refactored version with proper indentation, spacing, and naming conventions.
- Discuss how automated AI review can save time in large-scale projects.

Prompt:

Convert the following code to give a refactored version with proper indentation, spacing, and naming conventions.

Code:



Refactored version:

CO Untitled21.ipynb ☆
File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all ▼

[11]

```
def Checkprime(n):  
    """  
    Checks if a non-negative integer is a prime number.  
  
    Args:  
        n: An integer to check for primality.  
  
    Returns:  
        True if n is a prime number, False otherwise.  
    """  
    if n <= 1:  
        return False  
    for i in range(2, int(n**0.5) + 1):  
        if n % i == 0:  
            return False  
    return True  
  
# Get input from the user with validation  
while True:  
    try:  
        num = int(input("Enter a non-negative integer to check if it's prime: "))  
        if num < 0:  
            print("Please enter a non-negative integer.")  
        else:  
            break  
    except ValueError:  
        print("Invalid input. Please enter an integer.")  
  
if Checkprime(num):  
    print(f"{num} is a prime number.")  
else:  
    print(f"{num} is not a prime number.")
```

Enter a non-negative integer to check if it's prime: 17
17 is a prime number.

How AI Review Saves Time in Large-Scale Projects:

- 1.Consistency: Enforces uniform coding standards across teams.
- 2.Reduced Manual Effort: AI handles trivial fixes, humans focus on logic/design.
- 3.Scalability: Efficiently reviews hundreds of pull requests daily.
- 4.KnowledgeSharing: Helps junior developers learn best practices.

Task Description #4 (AI as a Code Reviewer in Real Projects)

Scenario: You are part of a GitHub project. A teammate submits this pull request:

```
def processData(d):  
    return [x*2 for x in d if x%2==0]
```

- Review this function manually for readability, reusability, and edge cases.
- Use AI to generate a code review comment, focusing on:
 - Naming conventions.

- Input validation (e.g., what if d is not a list?).
- Adding type hints.
- Modify the function based on AI's suggestions.
- Write a short reflection: *Would you trust AI as a standalone reviewer, or only as a support tool? Why?*

Prompt:

Review the following Python function for readability, naming conventions, reusability.

AI Modified Function:

Untitled21.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
[14]: from typing import List

def process_data(numbers: List[int]) -> List[int]:
    """
    Return a list of doubled values for even integers in the input list.

    Parameters
    -----
    numbers : List[int]
        A list of integers.

    Returns
    -----
    List[int]
        A new list containing doubled values of the even integers.

    Raises
    -----
    ValueError
        If input is not a list of integers or contains non-integer elements.
    """
    if not isinstance(numbers, list):
        raise ValueError("Input must be a list of integers.")
    if not all(isinstance(x, int) for x in numbers):
        raise ValueError("All elements in the list must be integers.")
    return [x * 2 for x in numbers if x % 2 == 0]

if __name__ == "__main__":
    try:
        user_input = input("Enter a list of integers separated by spaces: ")
        # Convert the input string to a list of integers
        nums = [int(x) for x in user_input.split()]
        print("Input:", nums)
        print("Processed:", process_data(nums))
    except ValueError as e:
        print("Error:", e)
    except Exception as e:
        print("An unexpected error occurred:", e)
```

Enter a list of integers separated by spaces: 1 2 3 4
Input: [1, 2, 3, 4]
Processed: [4, 8]

Code Review:

- 1.Naming Conventions: The function name processData should be renamed to process_data.
- 2.InputValidation: The function should check that the input is a list of integers, and show a clear error if it is not.
- 3.Type hints: Add type hints so it's clear what the function takes and returns.

	<p><u>Short reflection:</u></p> <p>I would use AI mainly as a support tool, not as a standalone reviewer. AI is very helpful for catching common issues like naming, formatting, missing type hints, and simple bugs, which saves a lot of time in large projects.</p>	
--	---	--