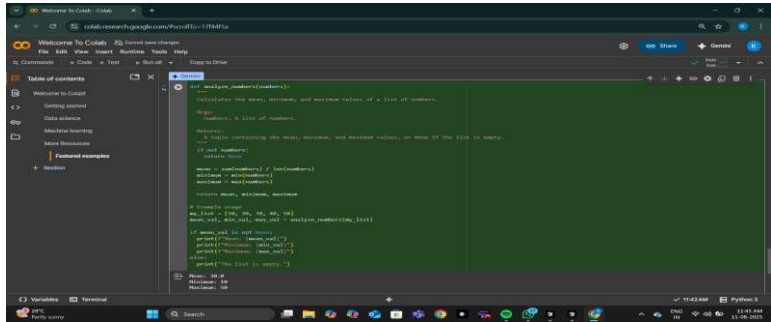


SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		Assignment Type: Lab	Academic Year:2025- 2026
Course Coordinator Name		Venkataramana Veeramsetty	
Instructor(s) Name		Dr. V. Venkataramana (Co-Ordinator)	
		Dr. T. Sampath Kumar	
		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	24CS002 PC215	Course Title	AI Assisted Coding
Year/Sem	II/I	Regulation	R24
Date and Day of Assignment	Week1 - Monday	Time(s)	
Duration	2 Hours	Applicable to Batches	24CSBTB01 To 24CSBTB39
Assignment Number: 2.1(Present assignment number)/24(Total number of assignments)			
Q . N o .	Question		Exp ecte d Tim e to com

		plet e	
1	<p>Lab 2: Exploring Additional AI Coding Tools – Gemini (Colab) and Cursor AI</p> <p>Lab Objectives:</p> <ul style="list-style-type: none"> • To explore and evaluate the functionality of Google Gemini for AI-assisted coding within Google Colab. • To understand and use Cursor AI for code generation, explanation, and refactoring. • To compare outputs and usability between Gemini, GitHub Copilot, and Cursor AI. • To perform code optimization and documentation using AI tools. <p>Lab Outcomes (LOs):</p> <p>After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> • Generate Python code using Google Gemini in Google Colab. • Analyze the effectiveness of code explanations and suggestions by Gemini. • Set up and use Cursor AI for AI-powered coding assistance. • Evaluate and refactor code using Cursor AI features. • Compare AI tool behavior and code quality across different platforms. 		Week 1 - Monday
	<p>Task Description #1</p> <ul style="list-style-type: none"> • Use Google Gemini in Colab to write a Python function that reads a list of numbers and calculates the mean, minimum, and maximum values. <p>Expected Output #1</p> <ul style="list-style-type: none"> • Functional code with correct output and screenshot.  <p>Task Description #2</p> <ul style="list-style-type: none"> • Compare Gemini and Copilot outputs for a Python function that checks whether a number is an Armstrong number. Document the steps, prompts, and 		

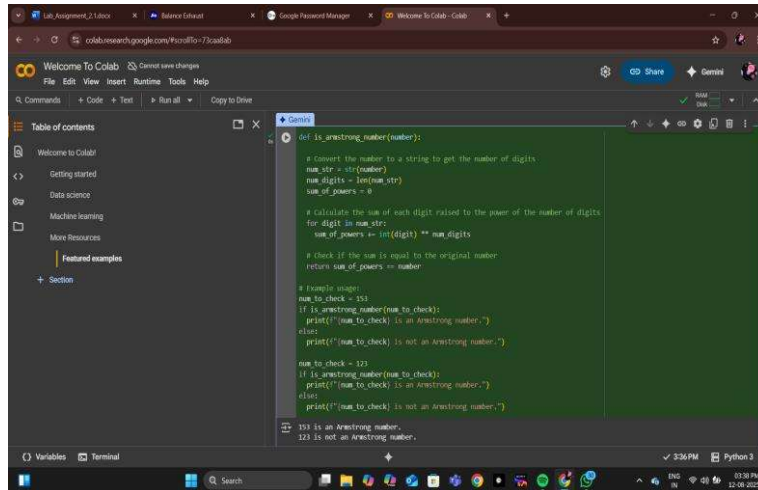
outputs.

Expected Output #2

We use the same prompt for both models to ensure fairness:

Prompt: "Python function that checks whether a number is an Armstrong number."

Gemini output:

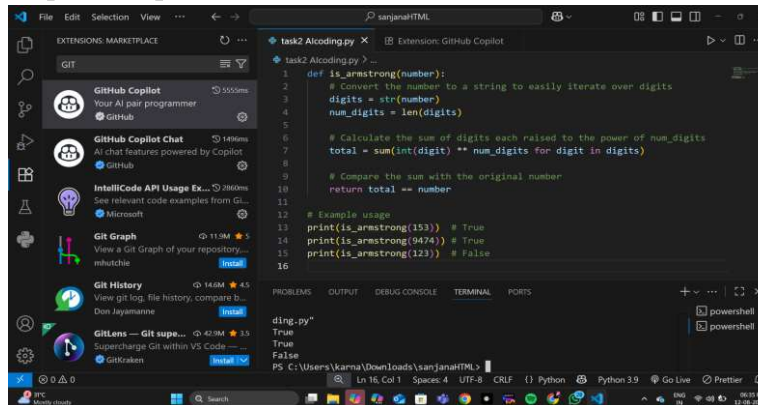


The screenshot shows a Google Colab notebook with a Python function `def is_armstrong_number(number):`. The function converts the number to a string, calculates the sum of each digit raised to the power of the number of digits, and checks if the sum equals the original number. It includes example usage for 153 and 123.

```
def is_armstrong_number(number):  
    # Convert the number to a string to get the number of digits  
    num_str = str(number)  
    num_digits = len(num_str)  
    sum_of_powers = 0  
  
    # Calculate the sum of each digit raised to the power of the number of digits  
    for digit in num_str:  
        sum_of_powers += int(digit) ** num_digits  
  
    # Check if the sum is equal to the original number  
    return sum_of_powers == number  
  
# Example usage:  
num_to_check = 153  
if is_armstrong_number(num_to_check):  
    print(f"{num_to_check} is an Armstrong number.")  
else:  
    print(f"{num_to_check} is not an Armstrong number.")  
  
num_to_check = 123  
if is_armstrong_number(num_to_check):  
    print(f"{num_to_check} is an Armstrong number.")  
else:  
    print(f"{num_to_check} is not an Armstrong number.")
```

The output shows: 153 is an Armstrong number. 123 is not an Armstrong number.

Copilot output:



The screenshot shows a Visual Studio Code editor with a Python function `def is_armstrong(number):`. The function converts the number to a string, calculates the sum of each digit raised to the power of the number of digits, and checks if the sum equals the original number. It includes example usage for 153, 9474, and 123.

```
def is_armstrong(number):  
    # Convert the number to a string to easily iterate over digits  
    digits = str(number)  
    num_digits = len(digits)  
  
    # Calculate the sum of digits each raised to the power of num_digits  
    total = sum(int(digit) ** num_digits for digit in digits)  
  
    # Compare the sum with the original number  
    return total == number  
  
# Example usage  
print(is_armstrong(153)) # True  
print(is_armstrong(9474)) # True  
print(is_armstrong(123)) # False
```

Task Description #3

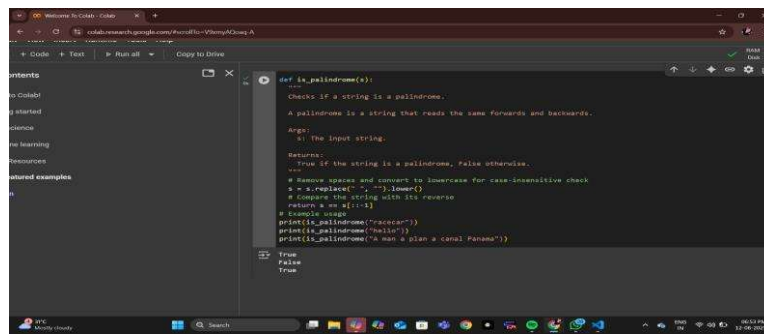
- Ask Gemini to explain a Python function (e.g., `is_prime(n)` or `is_palindrome(s)`) line by line.
- Choose either a prime-checking or palindrome-checking function and document the explanation provided by Gemini.

Expected Output #3

- Detailed explanation with the code snippet and

Gemini's response.

GEMINI:

A screenshot of a Google Colab notebook interface. The code editor shows a Python function `def is_palindrome(s):` with a docstring explaining its purpose: "Checks if a string is a palindrome. A palindrome is a string that reads the same forwards and backwards." The function takes `s` as the input string and returns a boolean. The code includes comments for removing spaces and converting to lowercase, and a comparison of the string with its reverse. Test cases are shown at the bottom: `print(is_palindrome("racecar"))` returns `True`, `print(is_palindrome("hello"))` returns `False`, and `print(is_palindrome("A man a plan a canal Panama"))` returns `True`.

```
def is_palindrome(s):
    """
    Checks if a string is a palindrome.
    A palindrome is a string that reads the same forwards and backwards.

    Args:
        s: The input string.

    Returns:
        True if the string is a palindrome, False otherwise.
    """
    # Remove spaces and convert to lowercase for case-insensitive check
    s = s.replace(" ", "").lower()
    # Compare the string with its reverse
    return s == s[::-1]

# Example usage
print(is_palindrome("racecar"))
print(is_palindrome("hello"))
print(is_palindrome("A man a plan a canal Panama"))
```

```
def is_palindrome(s):
```

This part defines the function `is_palindrome` that takes one argument `s` (the string to check). The triple quotes enclose a docstring, which explains what the function does, its arguments, and what it returns.

```
s = s.replace(" ", "").lower()
```

This line prepares the input string `s` for the palindrome check.

`s.replace(" ", "")`: This removes all spaces from the string. This is important so that phrases like "A man a plan a canal Panama" are treated correctly as palindromes.

`.lower()`: This converts the entire string to lowercase. This ensures that the check is case-insensitive, so "Racecar" is also considered a palindrome.

```
# Compare the string with its reverse
return s == s[::-1]
```

This is the core of the palindrome check.

`s[::-1]`: This is a Python slicing trick that creates a reversed copy of the string `s`.

`s == s[::-1]`: This compares the modified string `s` (with no spaces and in lowercase) with its reversed version. If they are the same, the function returns `True`, indicating that the original string was a

palindrome. Otherwise, it returns False.

```
# Example usage
print(is_palindrome("racecar"))
print(is_palindrome("hello"))
print(is_palindrome("A man a plan a canal Panama"))
```

These lines show how to use the `is_palindrome` function and print the results for a few example strings.

Task Description #4

- Install and configure Cursor AI. Use it to generate a Python function (e.g., sum of the first N natural numbers) and test its output.
- Optionally, compare Cursor AI's generated code with Gemini's output.

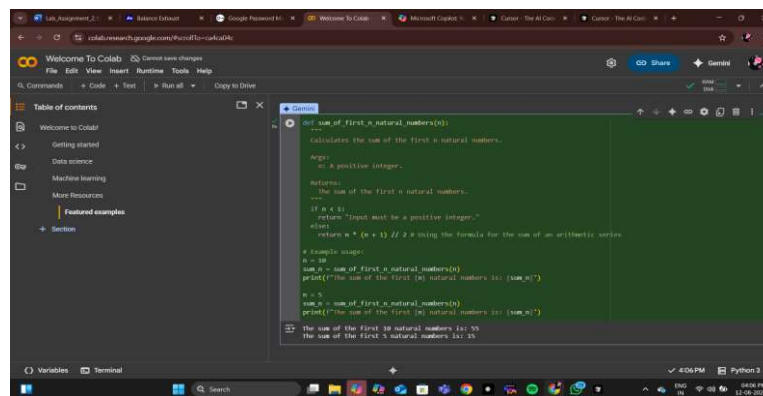
Expected Output #4

- Screenshots of Cursor AI setup, prompts used, and generated code with output.
-

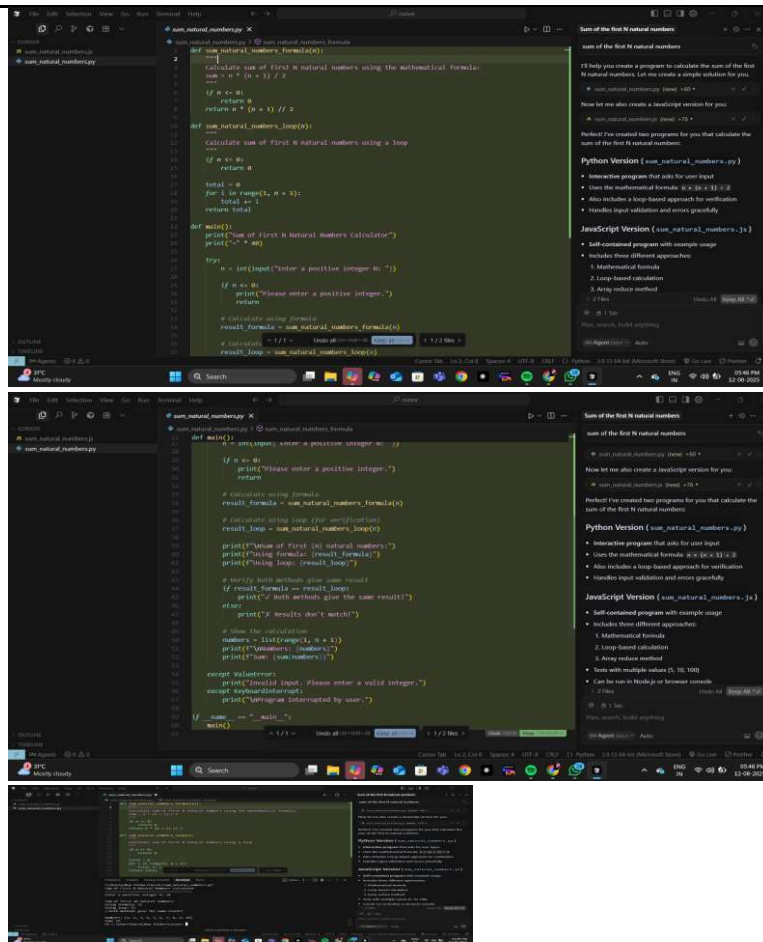
PROMPT USED:

- "Write a Python function that returns the sum of the first N natural numbers."

GEMINI:



CURSOR AI:



Task Description #5

- Students need to write a Python program to calculate the sum of odd numbers and even numbers in a given tuple.
- Refactor the code to improve logic and readability.

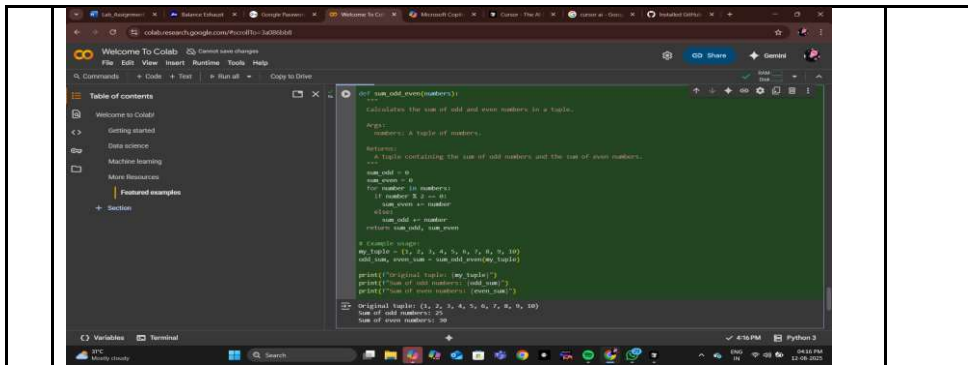
Expected Output #5

- Student-written refactored code with explanations and output screenshots.

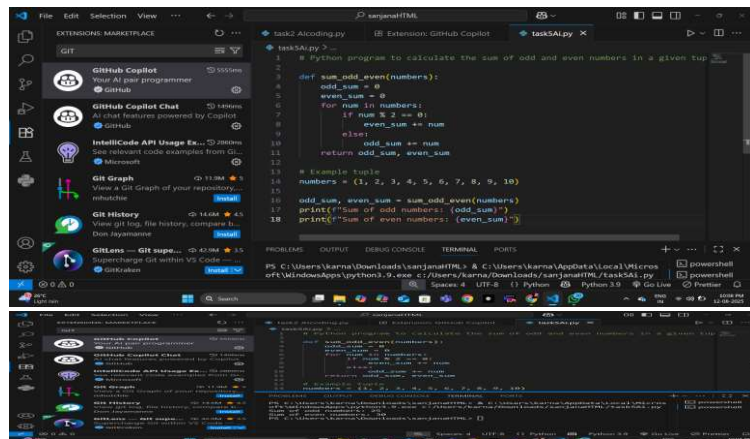
PROMPT:

"Write a Python function that takes a tuple of numbers and returns the sum of even and odd numbers separately. Refactor the code to improve readability and logic."

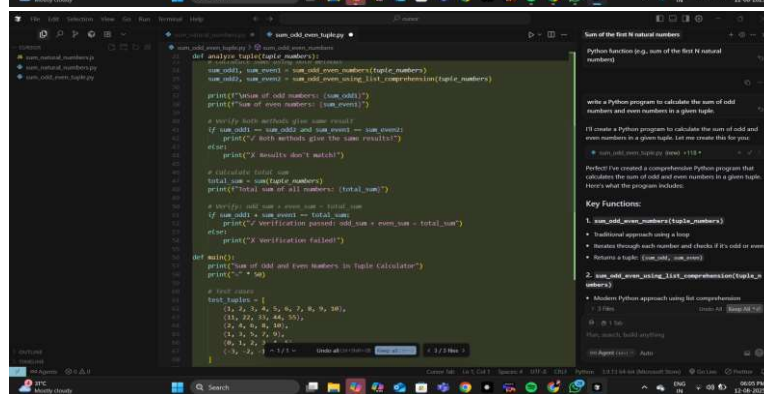
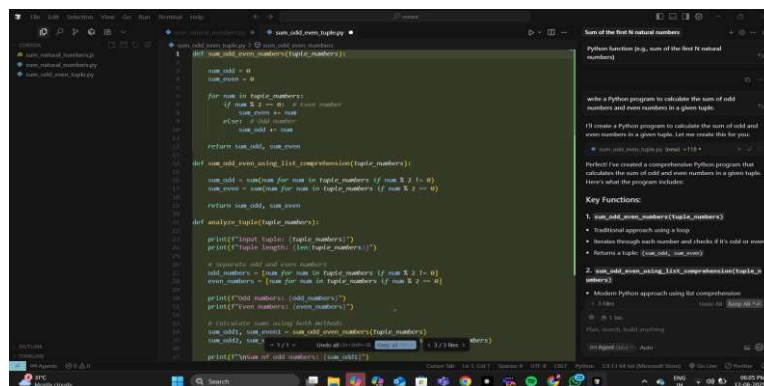
GEMINI:

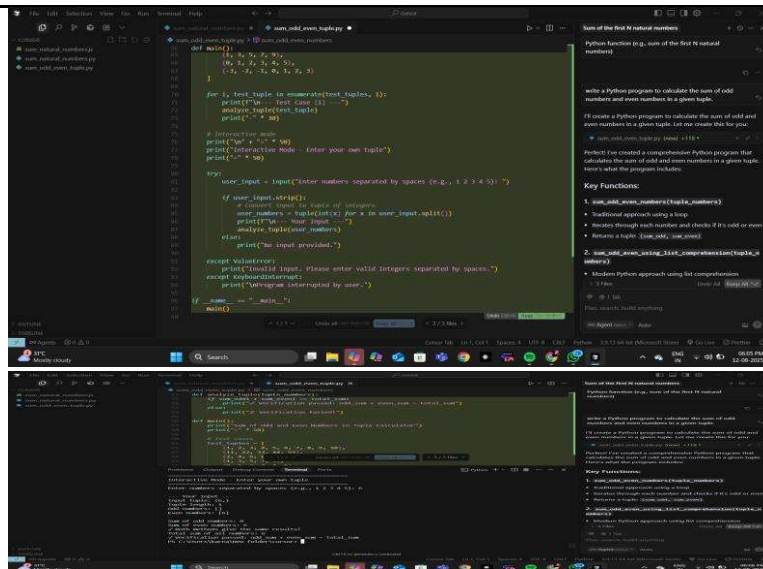


COPILOT:



CURSOR AI:





CODE EXPLANATION:

numbers = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

- This line defines a **tuple** named **numbers** that contains integers from 1 to 10.
- Tuples are ordered and immutable collections in Python.

even_sum = 0

odd_sum = 0

- These two lines initialize variables to store the **sum of even** and **sum of odd** numbers.
- Both start at 0.

for num in numbers:

- This line starts a **loop** that goes through each element (num) in the numbers tuple.

if num % 2 == 0:

- Checks if the number is **even**.
- % is the **modulus operator** — it returns the remainder.
- If **num % 2 == 0**, the number is divisible by 2 → it's even.

even_sum += num

- If the number is even, it gets **added to** even_sum.

else:

- If the number is **not even**, it must be **odd**.

odd_sum += num

- Adds the odd number to odd_sum.

print("Sum of even numbers:", even_sum)

print("Sum of odd numbers:", odd_sum)

- These lines **print the final results** after the loop finishes.
- You'll see the total sum of even and odd numbers separately.

Note:

- Students must submit a single Word document including:
 - Prompts used for AI tools
 - Copilot/Gemini/Cursor outputs
 - Code explanations
 - Screenshots of outputs and environments

Evaluation Criteria:

Criteria	Max Marks
Successful Use of Gemini in Colab (Task#1 & #2)	1.0
Code Explanation Accuracy (Gemini) (Task#3)	0.5
Cursor AI Setup and Usage (Task#4)	0.5
Refactoring and Improvement Analysis (Task#5)	0.5
Total	2.5 Marks