

2303A51200

Batch 04

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING																		
Program Name: B. Tech		Assignment Type: Lab																		
Course Coordinator Name		Dr. Rishabh Mittal																		
Instructor(s) Name		<table border="1"><tr><td>Mr. S Naresh Kumar</td></tr><tr><td>Ms. B. Swathi</td></tr><tr><td>Dr. Sasanko Shekhar Gantayat</td></tr><tr><td>Mr. Md Sallauddin</td></tr><tr><td>Dr. Mathivanan</td></tr><tr><td>Mr. Y Srikanth</td></tr><tr><td>Ms. N Shilpa</td></tr><tr><td>Dr. Rishabh Mittal (Coordinator)</td></tr><tr><td>Dr. R. Prashant Kumar</td></tr><tr><td>Mr. Ankushavali MD</td></tr><tr><td>Mr. B Viswanath</td></tr><tr><td>Ms. Sujitha Reddy</td></tr><tr><td>Ms. A. Anitha</td></tr><tr><td>Ms. M.Madhuri</td></tr><tr><td>Ms. Katherashala Swetha</td></tr><tr><td>Ms. Velpula sumalatha</td></tr><tr><td>Mr. Bingi Raju</td></tr></table>		Mr. S Naresh Kumar	Ms. B. Swathi	Dr. Sasanko Shekhar Gantayat	Mr. Md Sallauddin	Dr. Mathivanan	Mr. Y Srikanth	Ms. N Shilpa	Dr. Rishabh Mittal (Coordinator)	Dr. R. Prashant Kumar	Mr. Ankushavali MD	Mr. B Viswanath	Ms. Sujitha Reddy	Ms. A. Anitha	Ms. M.Madhuri	Ms. Katherashala Swetha	Ms. Velpula sumalatha	Mr. Bingi Raju
Mr. S Naresh Kumar																				
Ms. B. Swathi																				
Dr. Sasanko Shekhar Gantayat																				
Mr. Md Sallauddin																				
Dr. Mathivanan																				
Mr. Y Srikanth																				
Ms. N Shilpa																				
Dr. Rishabh Mittal (Coordinator)																				
Dr. R. Prashant Kumar																				
Mr. Ankushavali MD																				
Mr. B Viswanath																				
Ms. Sujitha Reddy																				
Ms. A. Anitha																				
Ms. M.Madhuri																				
Ms. Katherashala Swetha																				
Ms. Velpula sumalatha																				
Mr. Bingi Raju																				
CourseCode	23CS002PC304	Course Title	AI Assisted Coding																	
Year/Sem	III/II	Regulation	R23																	
Date and Day of Assignment	Week1 – Tuesday	Time(s)	23CSBTB01 To 23CSBTB52																	
Duration	2 Hours	Applicable to Batches	All batches																	
Assignment Number:1.3(Present assignment number)/24(Total number of assignments)																				
Q.No.	Question		Expected Time to																	

		<i>complete</i>
1	<p>Lab 2: Exploring Additional AI Coding Tools beyond Copilot – 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. 	Week1 - Monday
	<p>Task 1: Cleaning Sensor Data</p> <p>❖ Scenario: ❖ You are cleaning IoT sensor data where negative values are invalid.</p> <p>❖ Task: Use Gemini in Colab to generate a function that filters out all negative numbers from a list.</p> <p>❖ Expected Output: ➤ Before/after list ➤ Screenshot of Colab execution</p> <p>❖ OUTPUT</p> <p>❖ PROMPT:</p> <p>❖ Write a python function to remove negative numbers from the list</p> <p>❖</p>	

```

▶ def remove_negative_numbers_simple(input_list):
    filtered_list = []
    for num in input_list:
        if num >= 0:
            filtered_list.append(num)
    return filtered_list

my_list = [1, -2, 3, -4, 5, 0, -10]
filtered_list = remove_negative_numbers_simple(my_list)
print(f"Original list: {my_list}")
print(f"List without negative numbers: {filtered_list}")

... Original list: [1, -2, 3, -4, 5, 0, -10]
List without negative numbers: [1, 3, 5, 0]

```

❖ Explanation:

- ❖ The function loops through each number in the input list.
- ❖ It checks if the number is 0 or positive.
- ❖ Only non-negative numbers are added to a new list.
- ❖ Finally, it returns the filtered list without negative values.

Task 2: String Character Analysis

❖ Scenario:

You are building a text-analysis feature.

❖ Task:

Use Gemini to generate a Python function that counts vowels, consonants, and digits in a string.

❖ Expected Output:

- Working function
- Sample inputs and outputs

Output:

Prompt

Generate a python function that counts vowels constants and numbers in a string

```

# Define a sample text
text = "This is a simple example of text analysis."

# Split the text into words using space as a delimiter
words = text.split()

# Count the number of words
word_count = len(words)

print(f"Original text: {text}")
print(f"Number of words: {word_count}")

```

Output for code

... Original text: This is a simple example of text analysis.
Number of words: 8

Explanation:

The function goes through each character in the given string.

If the character is a letter, it checks whether it is a vowel or a consonant.

If the character is a digit, it increases the digit count.

Finally, it returns the counts of vowels, consonants, and digits.

Task 3: Palindrome Check – Tool Comparison

❖ Scenario:

You must decide which AI tool is clearer for string logic.

❖ Task:

Generate a palindrome-checking function using Gemini and Copilot, then compare the results.

❖ Expected Output:

- Side-by-side code comparison
- Observations on clarity and structure

Output

Prompt

Generating palindrome code in git hub copilot , gemi

ni and comparison

```
"""
#generate python code for palidrome checking
def is_palindrome(s):
    """Check if a string is a palindrome."""
    cleaned_str = ''.join(c.lower() for c in s if c.isalnum())
    return cleaned_str == cleaned_str[::-1]

# Example usage
if __name__ == "__main__":
    test_strings = ["A man, a plan, a canal: Panama", "racecar", "hello", "No 'x' in Nixon"]
    for s in test_strings:
        result = is_palindrome(s)
        print(f"'{s}' is a palindrome: {result}")

```

```
PS C:\Ai_coding> & C:/Users/mukes/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Ai_coding/day_2.py
"A man, a plan, a canal: Panama" is a palindrome: True
"racecar" is a palindrome: True
"hello" is a palindrome: False
"No 'x' in Nixon" is a palindrome: True
PS C:\Ai_coding>
```

```
def is_palindrome(text):
    cleaned_text = "".join(char.lower() for char in text if char.isalnum())
    return cleaned_text == cleaned_text[::-1]

print(f"'madam' is a palindrome: {is_palindrome('madam')}")
print(f"'racecar' is a palindrome: {is_palindrome('racecar')}")
print(f"'hello' is a palindrome: {is_palindrome('hello')}")
print(f"'Python' is a palindrome: {is_palindrome('Python')}")
```

Output for code

```
'madam' is a palindrome: True
'racecar' is a palindrome: True
'hello' is a palindrome: False
'Python' is a palindrome: False
```

Eplanation:

Step-by-step comparison of both palindrome codes:

Function Definition:

Both codes define a function named `is_palindrome` that takes a string as input.

Text Cleaning:

In both codes, the input string is converted to lowercase and all non-alphanumeric characters are removed to ensure accurate palindrome

	<p>checking.</p> <p>Palindrome Logic: Each code compares the cleaned string with its reverse using string slicing (<code>[::-1]</code>) to determine if it is a palindrome.</p> <p>Return Value: Both functions return True if the string is a palindrome and False otherwise.</p> <p>Testing the Function: The first code directly calls the function with <code>print()</code> statements, while the second code tests multiple strings inside an if <code>__name__ == "__main__"</code> block.</p> <p>Code Structure: The first code is simple and straightforward, whereas the second code is better structured, includes a docstring, and is more suitable for larger programs.</p> <hr/> <p>Task 4: Code Explanation Using AI</p> <ul style="list-style-type: none"> ❖ Scenario: You are reviewing unfamiliar code written by another developer. ❖ Task: Ask Gemini to explain a Python function (prime check OR palindrome check) line by line. ❖ Expected Output: <ul style="list-style-type: none"> ➢ Code snippet ➢ AI explanation ➢ Student comments on understanding <p>Output</p> <p><i>prompt</i></p> <p>generate a python code for prime number explain line by line</p> <pre># A list of numbers that might be processed in unfamiliar code my_list = [1, -2, 3, -4, 5, 0, -10] # Filter the list to keep only positive numbers (including zero in this case) # This helps to understand data transformation logic filtered_list = [x for x in my_list if x >= 0] print(f"Original list: {my_list}") print(f"Filtered list (non-negative numbers): {filtered_list}")</pre>	
--	--	--

	<p>Output for code</p> <hr/> <pre>Original list: [1, -2, 3, -4, 5, 0, -10] Filtered list (non-negative numbers): [1, 3, 5, 0]</pre> <hr/> <p>Explanation:</p> <p>Step-by-step execution explanation for <code>is_prime(number)</code> function:</p> <p>Let's trace <code>is_prime(7):</code></p> <ol style="list-style-type: none"> 1. <code>def is_prime(number):</code>: The function <code>is_prime</code> is called with <code>number = 7</code>. 2. <code>if number <= 1:</code> ($7 \leq 1$) is <code>False</code>. So, this condition is skipped. 3. <code>for i in range(2, int(number**0.5) + 1):</code>: The loop starts. <code>number**0.5</code> (square root of 7) is approximately <code>2.64</code>. <code>int(2.64)</code> is <code>2</code>. So the range is <code>range(2, 2 + 1)</code>, which means <code>range(2, 3)</code>. The loop will run for <code>i = 2</code>. 4. <code>i = 2</code> (first iteration of the loop): <ul style="list-style-type: none"> o <code>if number % i == 0:</code> ($7 \% 2$) (7 divided by 2) is <code>1</code> (remainder is 1). <code>1 == 0</code> is <code>False</code>. So, this condition is skipped. 5. The loop finishes because <code>i</code> reached the end of the range. 6. <code>return True</code>: Since no divisor was found, the function returns <code>True</code>. <p>Thus, <code>is_prime(7)</code> correctly returns <code>True</code>.</p> <hr/> <p>Let's trace <code>is_prime(10):</code></p> <ol style="list-style-type: none"> 1. <code>def is_prime(number):</code>: The function <code>is_prime</code> is called with <code>number = 10</code>. 2. <code>if number <= 1:</code> ($10 \leq 1$) is <code>False</code>. So, this condition is skipped. 3. <code>for i in range(2, int(number**0.5) + 1):</code>: The loop starts. <code>number**0.5</code> (square root of 10) is approximately <code>3.16</code>. <code>int(3.16)</code> is <code>3</code>. So the range is <code>range(2, 3 + 1)</code>, which means <code>range(2, 4)</code>. The loop will run for <code>i = 2, 3</code>. 4. <code>i = 2</code> (first iteration of the loop): <ul style="list-style-type: none"> o <code>if number % i == 0:</code> ($10 \% 2$) (10 divided by 2) is <code>0</code> (remainder is 0). <code>0 == 0</code> is <code>True</code>. So, this condition is met. o <code>return False</code>: The function immediately returns <code>False</code> because a divisor (2) was found. <p>Thus, <code>is_prime(10)</code> correctly returns <code>False</code>.</p> <hr/> <p>Note: Report should be submitted as a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots.</p>	
--	--	--