

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
ProgramName: B. Tech		Assignment Type: Lab	AcademicYear: 2025-2026
CourseCoordinatorName		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)	
CourseCode	24CS002PC215	CourseTitle	AI Assisted Coding
Year/Sem	II/I	Regulation	R24
Date and Day of Assignment	Week2 - Wednesday	Time(s)	
Duration	2 Hours	Applicable to Batches	
AssignmentNumber: 2.3(Present assignment number)/24(Total number of assignments)			
Q.No.	Question	Expected Time to complete	
1	<p>Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques</p> <p><b>Lab Objectives:</b></p> <ul style="list-style-type: none"> <li>To explore and apply different levels of prompt examples in AI-assisted code generation.</li> <li>To understand how zero-shot, one-shot, and few-shot prompting affect AI output quality.</li> <li>To evaluate the impact of context richness and example quantity on AI performance.</li> </ul>	Week2 - Wednesday	

- To build awareness of prompt strategy effectiveness for different problem types.

### Lab Outcomes (LOs):

After completing this lab, students will be able to:

- Use zero-shot prompting to instruct AI with minimal context.
- Use one-shot prompting with a single example to guide AI code generation.
- Apply few-shot prompting using multiple examples to improve AI responses.
- Compare AI outputs across the three prompting strategies.

### Task Description#1

- Zero-shot: Prompt AI to write a function that checks whether a given year is a leap year.

### Expected Output#1

- AI-generated function with no examples provided

Prompt:

Create a Python program that takes a input from the user and checks whether a given year is a leap year

Code:

```
# Get input from the user
try:
    year_input = int(input("Enter a year to check if it's a leap year: "))
    if is_leap_year(year_input):
        print(f"{year_input} is a leap year.")
    else:
        print(f"{year_input} is not a leap year.")
except ValueError:
    print("Invalid input. Please enter a valid year as a number.")
```

out put:

The screenshot shows a Jupyter Notebook environment. The code cell contains the Python code for checking a leap year. The output cell shows the result of running the code: "Enter a year to check if it's a leap year: 1956" followed by "1956 is a leap year.". On the right, the Gemini chat window is open, displaying the prompt and the AI's response. The AI's response includes a confirmation of the task and a list of suggestions for further exploration, such as modifying the function to handle different date formats, adding more error handling, and testing with different years.

### Task Description#2

- One-shot: Give one input-output example to guide AI in writing a function that converts centimeters to inches.

### Expected Output#2

- Function with correct conversion logic

Prompt:

Write a Python function that takes a input from the user and convert cm to inches that

takes a length in centimeters as input and returns the equivalent length in inches."

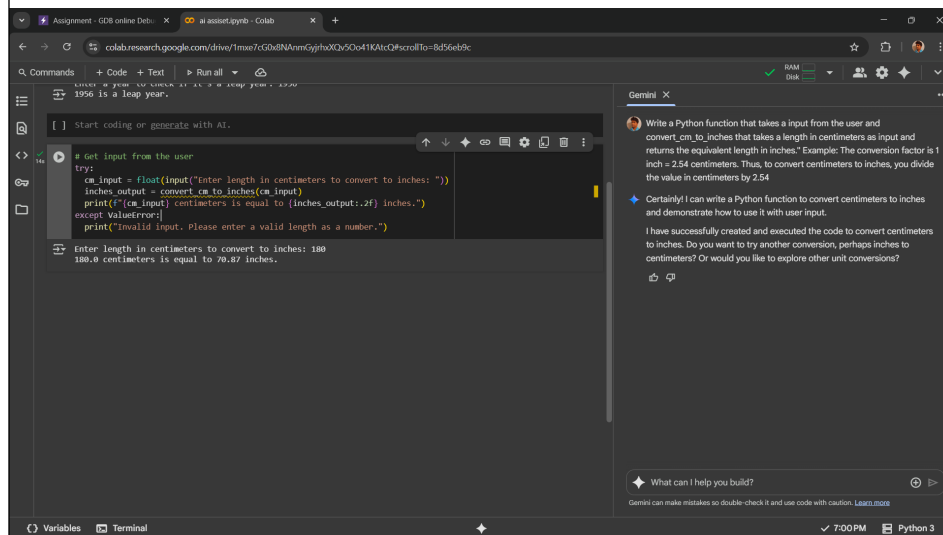
Example:

The conversion factor is 1 inch = 2.54 centimeters. Thus, to convert centimeters to inches, you divide the value in centimeters by 2.54

Code:

```
# Get input from the user
try:
    cm_input = float(input("Enter length in centimeters to convert to inches: "))
    inches_output = convert_cm_to_inches(cm_input)
    print(f"{cm_input} centimeters is equal to {inches_output:.2f} inches.")
except ValueError:
    print("Invalid input. Please enter a valid length as a number.")
```

out put:



### Task Description#3

- Few-shot: Provide 2–3 examples to generate a function that formats full names as “Last, First”.

### Expected Output#3

- Well-structured function respecting the examples

Prompt:

create a python function that formats full names as "Last, First".

Example 1

input: enter a name: Lokesh kamatham

output: last name = kamatham

First name =lokesh

Example 2

input: enter a name: Rithvik Gona

output: last name = Gona

First name = Rithvik

Example 3

input: enter a name: Nikhil thatikonda

output: Last name = thatikonda

First name = Nikhil

Code:

```
def format_name(full_name):  
    """  
    Formats a full name as "Last, First".  
  
    Args:  
        full_name: The full name string (e.g., "Lokesh kamatham").  
  
    Returns:  
        A string with the name formatted as "Last, First".  
    """  
    names = full_name.split()  
    if len(names) >= 2:  
        last_name = names[-1]  
        first_name = " ".join(names[:-1])  
        return f"Last name = {last_name.capitalize()}\nFirst name = {first_name.capitalize()}"  
    else:  
        return "Invalid name format. Please provide a full name with at least a first and last name."  
  
# Example usage:  
name1 = "Lokesh kamatham"  
formatted_name1 = format_name(name1)  
print(formatted_name1)  
  
name2 = "Rithvik Gona"  
formatted_name2 = format_name(name2)  
print(formatted_name2)  
  
name3 = "Nikhil thatikonda"  
formatted_name3 = format_name(name3)  
print(formatted_name3)
```

output:

The screenshot shows a Google Colab notebook with the following content:

```
def format_name(full_name):  
    """  
    Formats a full name as "Last, First".  
  
    Args:  
        full_name: The full name string (e.g., "Lokesh kamatham").  
  
    Returns:  
        A string with the name formatted as "Last, First".  
    """  
    names = full_name.split()  
    if len(names) >= 2:  
        last_name = names[-1]  
        first_name = " ".join(names[:-1])  
        return f"Last name = {last_name.capitalize()}\nFirst name = {first_name.capitalize()}"  
    else:  
        return "Invalid name format. Please provide a full name with at least a first and last name."  
  
# Example usage:  
name1 = "Lokesh kamatham"  
formatted_name1 = format_name(name1)  
print(formatted_name1)  
  
name2 = "Rithvik Gona"  
formatted_name2 = format_name(name2)  
print(formatted_name2)  
  
name3 = "Nikhil thatikonda"  
formatted_name3 = format_name(name3)  
print(formatted_name3)
```

The output of the code is displayed in the terminal:

```
Last name = Kamatham  
First name = Lokesh  
Last name = Gona  
First name = Rithvik  
Last name = Thatikonda  
First name = Nikhil
```

#### Task Description#4

- Compare zero-shot and few-shot prompts for writing a function that counts the number of vowels in a string.

#### Expected Output#4

- Functional output and comparative reflection

#### Prompt:

1. Write a Python function that takes a string as input and returns the number of vowels (a, e, i, o, u) in the string.

2. Write a Python function that counts the number of vowels (a, e, i, o, u) in a string.

#### # Example 1

Input: "hello world"

Output: 3

#### # Example 2

Input: "Python"

Output: 1

#### # Example 3

Input: "AI is amazing"

Output: 6

#### Code:

1. def count\_vowels(input\_string):

"""

Counts the number of vowels (a, e, i, o, u) in a string, case-insensitive.

#### Args:

input\_string: The string to count vowels from.

#### Returns:

The number of vowels in the string.

"""

vowels = "aeiou"

count = 0

for char in input\_string.lower():

if char in vowels:

count += 1

return count

#### # Example usage

text = "Hello World"

vowel\_count = count\_vowels(text)

print(f"The number of vowels in '{text}' is: {vowel\_count}")

text\_2 = "This is another example."

vowel\_count\_2 = count\_vowels(text\_2)

print(f"The number of vowels in '{text\_2}' is: {vowel\_count\_2}")

2. def count\_vowels(input\_string):

"""

Counts the number of vowels (a, e, i, o, u) in a string, case-insensitive.

#### Args:

input\_string: The string to check.

#### Returns:

The number of vowels in the string.

"""

vowels = "aeiou"

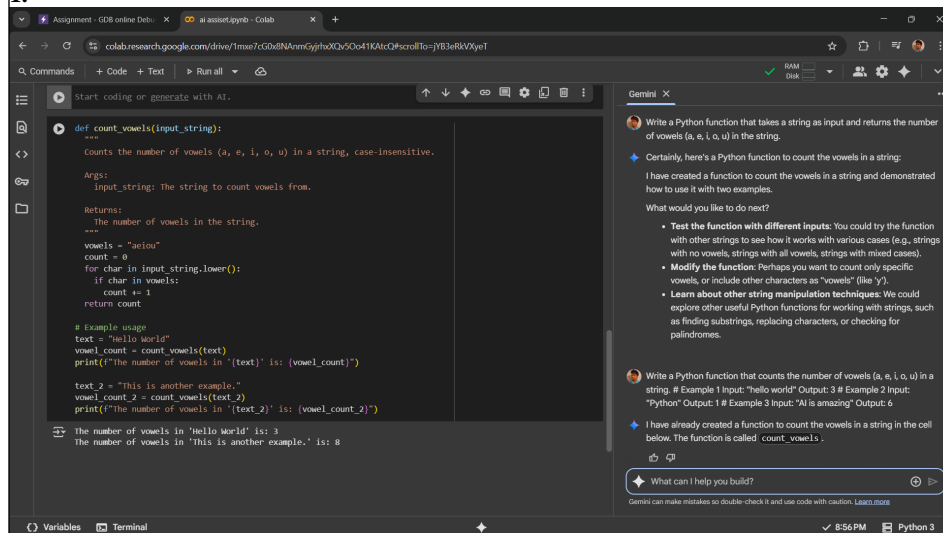
```
vowel_count = 0
for char in input_string.lower():
    if char in vowels:
        vowel_count += 1
return vowel_count

# Example 1
input_string_1 = "hello world"
vowel_count_1 = count_vowels(input_string_1)
print(f"Input: '{input_string_1}'\nOutput: {vowel_count_1}")

# Example 2
input_string_2 = "Python"
vowel_count_2 = count_vowels(input_string_2)
print(f"Input: '{input_string_2}'\nOutput: {vowel_count_2}")

# Example 3
input_string_3 = "AI is amazing"
vowel_count_3 = count_vowels(input_string_3)
print(f"Input: '{input_string_3}'\nOutput: {vowel_count_3}")
```

Output:  
1.



The screenshot shows a Google Colab notebook with a Python function `count_vowels` and its execution output. The function takes a string as input and returns the number of vowels (a, e, i, o, u) in a case-insensitive manner. The output shows the function being called with three different inputs: "hello world", "Python", and "AI is amazing", resulting in 3, 2, and 3 vowels respectively.

```
def count_vowels(input_string):
    """
    Counts the number of vowels (a, e, i, o, u) in a string, case-insensitive.

    Args:
        input_string: The string to count vowels from.

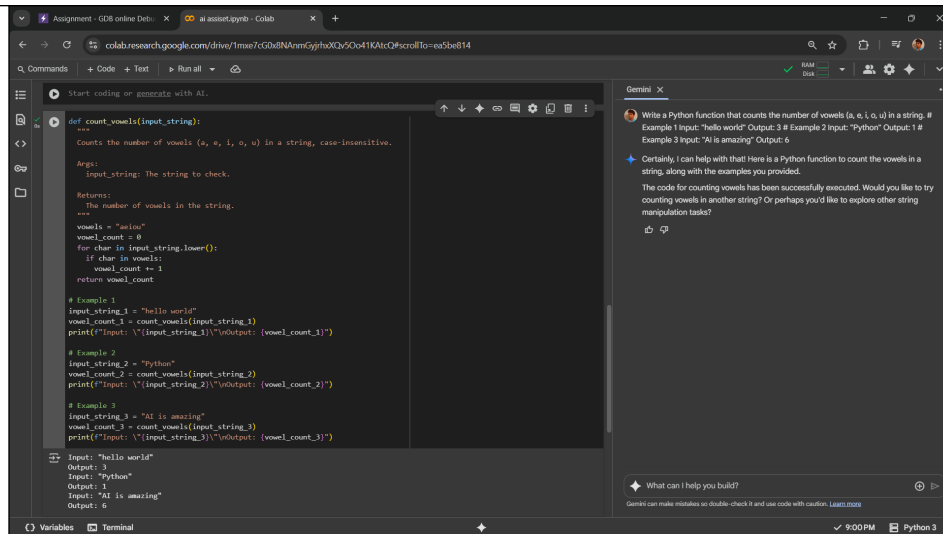
    Returns:
        The number of vowels in the string.
    """
    vowels = "aeiou"
    count = 0
    for char in input_string.lower():
        if char in vowels:
            count += 1
    return count

# Example usage
text = "hello world"
vowel_count = count_vowels(text)
print(f"The number of vowels in '{text}' is: {vowel_count}")

text_2 = "This is another example."
vowel_count_2 = count_vowels(text_2)
print(f"The number of vowels in '{text_2}' is: {vowel_count_2}")
```

The number of vowels in 'hello world' is: 3  
The number of vowels in 'This is another example.' is: 8

2.



### Task Description#5

- Use few-shot prompting to generate a function that reads a .txt file and returns the number of lines.

### Expected Output#5

- Working file-processing function with AI-guided logic

### Prompt:

generate a Python function that reads a .txt file and returns the number of lines

#### Example 1

Input: A text file named example.txt with the following content:

Line 1

Line 2

Line 3

Expected Output:3

#### Example 2

Input: A non-existent text file named nonexistent.txt

Expected Output:

0 (with an appropriate error message indicating the file was not found)

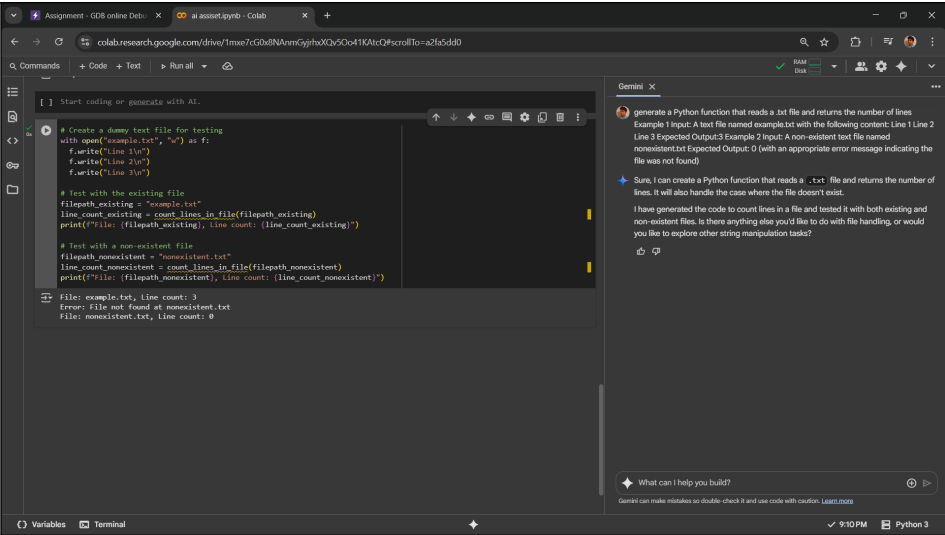
### Code:

```
# Create a dummy text file for testing
with open("example.txt", "w") as f:
    f.write("Line 1\n")
    f.write("Line 2\n")
    f.write("Line 3\n")

# Test with the existing file
filepath_existing = "example.txt"
line_count_existing = count_lines_in_file(filepath_existing)
print(f"File: {filepath_existing}, Line count: {line_count_existing}")

# Test with a non-existent file
filepath_nonexistent = "nonexistent.txt"
line_count_nonexistent = count_lines_in_file(filepath_nonexistent)
print(f"File: {filepath_nonexistent}, Line count: {line_count_nonexistent}")
```

### Output:



**Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots**

**Evaluation Criteria:**

Criteria	Max Marks
Zero Shot (Task #1)	0.5
One Shot (Task#2)	0.5
Few Shot (Task#3 & Task #5)	1.0
Comparison (Task#4)	0.5
Total	2.5 Marks