

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
<b>Program Name:</b> B. Tech		<b>Assignment Type:</b> Lab	
<b>Course Coordinator Name</b>		Dr. Rishabh Mittal	
		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	
<b>Course Code</b>	23CS002PC304	<b>Course Title</b>	AI Assisted Coding
<b>Year/Sem</b>	III/I	<b>Regulation</b>	R23
<b>Date and Day of Assignment</b>	Week 2 - Wednesday	<b>Time(s)</b>	23CSBTB01 To 23CSBTB52
<b>Duration</b>	2 Hours	<b>Applicable to Batches</b>	All batches
<b>Assignment Number:</b> 3.3(Present assignment number)/24(Total number of assignments)			

<b>Q.No.</b>	<b>Question</b>	<b>Expected Time to complete</b>
1	<p><b>Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques</b></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> <li>• To build awareness of prompt strategy effectiveness for different problem types</li> </ul>	Week2 - Wednesday

## 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 different prompting strategies

### Task 1: Zero-Shot Prompting – Leap Year Check

#### Scenario

Zero-shot prompting involves giving instructions without providing examples.

#### Task Description

Use zero-shot prompting to instruct an AI tool to generate a Python function that:

- Accepts a year as input
- Checks whether the given year is a leap year
- Returns an appropriate result

**Note:** No input-output examples should be provided in the prompt.

#### Expected Output

- AI-generated leap year checking function
- Correct logical conditions
- Sample input and output
- Screenshot of AI-generated response (if required)

The screenshot shows two separate AI-generated responses in a code editor, each with its own file tab and interface.

**Top Response (Gemini 2.5 Flash):**

```
def is_leap_year(year: int) -> bool:
    """
    Determines whether the given year is a leap year.

    Args:
        year: An integer representing the year.

    Returns:
        True if the year is a leap year, False otherwise.
    """
    return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)

# Sample input and output
sample_year = 2024
print(f"Is {sample_year} a leap year? {is_leap_year(sample_year)}")
sample_year_1 = 1900
print(f"Is {sample_year_1} a leap year? {is_leap_year(sample_year_1)}")
sample_year_2 = 2000
print(f"Is {sample_year_2} a leap year? {is_leap_year(sample_year_2)}")
sample_year_3 = 2020
print(f"Is {sample_year_3} a leap year? {is_leap_year(sample_year_3)}")
```

**Bottom Response (Gemini 2.5 Flash):**

```
def is_leap_year(year: int) -> bool:
    """
    Determines whether the given year is a leap year.

    Args:
        year: An integer representing the year.

    Returns:
        True if the year is a leap year, False otherwise.
    """
    return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)

# Sample input and output
sample_year = 2024
print(f"Is {sample_year} a leap year? {is_leap_year(sample_year)}")
sample_year_1 = 1900
print(f"Is {sample_year_1} a leap year? {is_leap_year(sample_year_1)}")
sample_year_2 = 2000
print(f"Is {sample_year_2} a leap year? {is_leap_year(sample_year_2)}")
sample_year_3 = 2020
print(f"Is {sample_year_3} a leap year? {is_leap_year(sample_year_3)})")
```

Both responses include a detailed explanatory note at the bottom, a "Accept and run" button, and a "What can I help you build?" input field.

# CODE EXPLANATION:

- A year is a leap year if:
  - It is divisible by 4 **and not** divisible by 100
  - OR divisible by 400
- Zero-shot prompting means **no examples were given in the prompt**

## Task 2: One-Shot Prompting – Centimeters to Inches Conversion

### Scenario

One-shot prompting guides AI using a single example.

### Task Description

Use one-shot prompting by providing one input-output example to generate a Python function that:

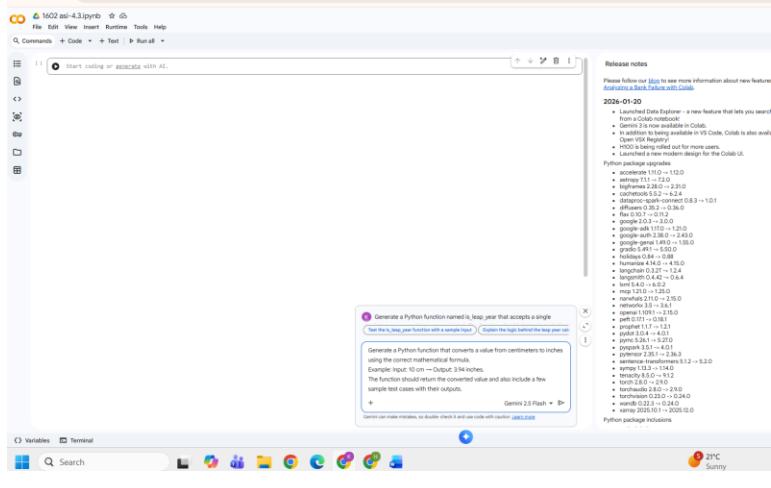
- Converts centimeters to inches
- Uses the correct mathematical formula

### Example provided in prompt:

Input: 10 cm → Output: 3.94 inches

### Expected Output

- Python function with correct conversion logic
- Accurate calculation
- Sample test cases and outputs



```

1602 asi-4.3.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Start coding or generate with AI.
+ Gemini
def cm_to_inches(cm: float) -> float:
    """
    Converts a value from centimeters to inches.

    Args:
        cm: The value in centimeters.

    Returns:
        The converted value in inches.
    """
    # 1 inch = 2.54 centimeters
    return cm / 2.54

# Sample test cases
print("10 cm = " + cm_to_inches(10).__str__())
print("25.4 cm = " + cm_to_inches(25.4).__str__())
print("1 cm = " + cm_to_inches(1).__str__())
print("0 cm = " + cm_to_inches(0).__str__())

```

... 10 cm = 3.94 inches  
25.4 cm = 10.00 inches  
1 cm = 0.39 inches  
0 cm = 0.00 inches

+ Code + Text

Generate a Python function that converts a value from centimeters to inches using the correct mathematical formula. Example: Input: 10 cm → Output: 3.94 inches. The function should return the converted value and also include a few sample test cases with their outputs.

Accept and run Accept Cancel

What can I help you build?

Gemini 2.5 Flash

## CODE EXPLANATION:

- Formula used:  
**1 inch = 2.54 cm**
- One-shot prompting uses **one example** (10 cm → 3.94 inches)

---

### Task 3: Few-Shot Prompting – Name Formatting

#### Scenario

Few-shot prompting improves accuracy by providing multiple examples.

#### Task Description

Use few-shot prompting with 2–3 examples to generate a Python function that:

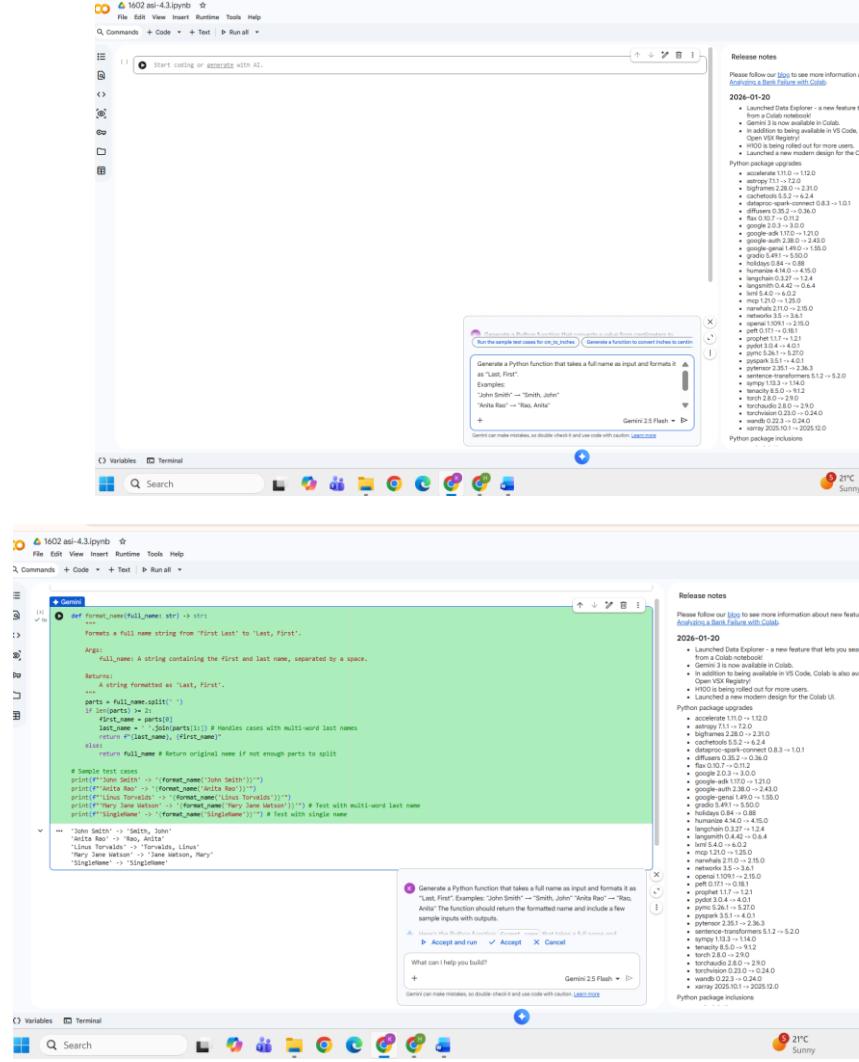
- Accepts a full name as input
- Formats it as “Last, First”

#### Example formats:

- “John Smith” → “Smith, John”
- “Anita Rao” → “Rao, Anita”

#### Expected Output

- Well-structured Python function
- Output strictly following example patterns
- Correct handling of names
- Sample inputs and outputs



## CODE EXPLANATION:

- The name is split into first and last parts
  - Output follows the pattern:  
**Last, First**
  - Few-shot prompting provides  
**multiple examples**

## Task 4: Comparative Analysis – Zero-Shot vs Few-Shot

### Scenario

Different prompt strategies may produce different code quality.

### Task Description

- Use zero-shot prompting to generate a function that counts vowels in a string
- Use few-shot prompting for the same problem
- Compare both outputs based on:
  - Accuracy
  - Readability
  - Logical clarity

### Expected Output

- Two vowel-counting functions
- Comparison table or short reflection paragraph
- Conclusion on prompt effectiveness

The screenshot shows the Gemini AI interface with two separate windows side-by-side. Both windows have a header bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help' and 'Commands', 'Code', 'Text', 'Run all' buttons.

**Top Window (Zero-Shot Prompt):**

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

    Args:
        input_string: The string to analyze.

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

**Bottom Window (Few-Shot Prompt):**

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

    Args:
        input_string: The string to analyze.

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

**Release Notes:**

2026-01-20

- Launched Data Explorer - a new feature that lets you search for data in Gemini. It's now available in Colab.
- Gemini 3 is now available in VS Code. Colab is also available.
- Open VSX Registry for more extensions.
- Python package design for the Colab UI.

Python package upgrades

- accelerate 0.12.0 → 1.12.0
- astropy 3.11 → 3.20.0
- bcolz 0.20.0 → 2.30.0
- cachetools 5.5.2 → 6.2.4
- dill 0.3.1 → 0.3.2
- diffutils 0.30 → 0.36.0
- fastparquet 0.1.0 → 0.1.0
- google 2.0.3 → 3.0.0
- google-auth 2.18.0 → 2.43.0
- google-auth-oauthlib 0.10.0
- gradeit 1.4.1 → 1.5.0.0
- h5py 5.1.0 → 5.2.0
- humanize 4.1.0 → 4.1.0
- jupyterlab 1.0.0 → 1.0.0
- langchain 0.4.2 → 0.4.4
- m1t 1.1.0 → 1.1.0
- msgpack 1.0.6 → 1.0.6
- networkx 3.5 → 3.6.0
- numpy 1.25.0 → 1.26.0
- petl 0.97.1 → 0.98.1
- pyparallel 0.1.1
- pydantic 2.1.0 → 2.1.0
- pygments 3.1.1 → 3.1.2
- pygments-pprint 2.3.1
- sentence-transformers 3.12 → 5.2.0
- smisc 0.1.0 → 0.1.0
- torchtext 0.1.0 → 0.1.2
- torchaudio 0.14.0 → 0.14.0
- wands 0.2.1 → 0.3.0.0
- xarray 2026.01.12.0

Python package inclusions

2026-01-20

- accelerate
- astropy 7.1
- bcolz
- cachetools
- dataprof-n
- diffutils
- fastparquet
- google 2.0
- google-auth
- google-auth-oauthlib
- gradeit
- humanize
- jupyterlab
- langchain
- lmfit 5.4.0
- m1t
- msgpack
- networkx
- numpy
- petl 0.171
- pygments
- pygments-pprint
- pyparallel
- sentence-transformers
- smisc
- torch 2.0
- torchtext 0.1.0
- wands 0.2
- xarray 2026.01.12.0

# CODE EXPLANATION:

## Task 5: Few-Shot Prompting – File Handling

### Scenario

File processing requires clear logical understanding.

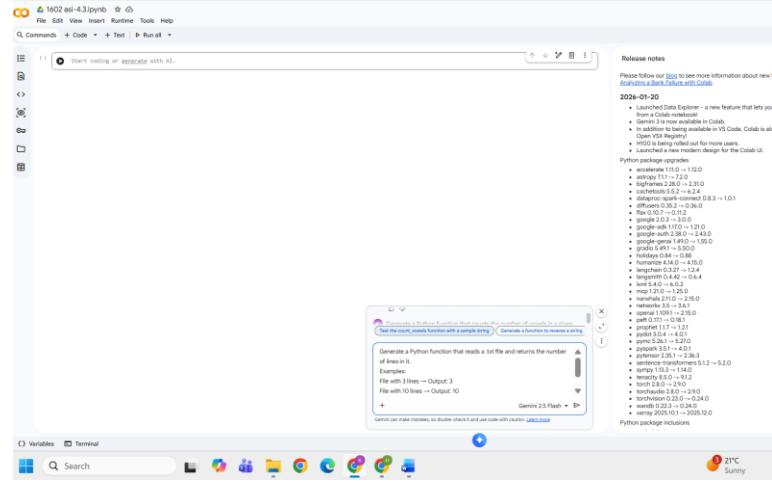
### Task Description

Use few-shot prompting to generate a Python function that:

- Reads a .txt file
- Counts the number of lines in the file
- Returns the line count

### Expected Output

- Working Python file-processing function
- Correct line count
- Sample .txt input and output
- AI-assisted logic explanation



The screenshot shows a Gemini 2.5 interface. On the left is a code editor with the following Python code:

```

1602 asl-4.3.ipynb  ☆
File Edit View Insert Runtime Tools Help
Commands + Code - Test | Run all ▾

def count_lines_in_file(file_path):
    """
    Returns:
        ... The number of lines in the file, or -1 if the file is not found.

    try:
        with open(file_path, 'r') as f:
            return sum(1 for _ in f)
    except FileNotFoundError:
        return -1

    sample_file_path = "sample_file.txt"
    with open(sample_file_path, "w") as f:
        f.write("Line 1\nLine 2\nLine 3\n")

    lines_count = count_lines_in_file(sample_file_path)
    print(f"The file '{sample_file_path}' has {lines_count} lines.")

    non_existent_file_path = "non_existent.txt"
    lines_count_2 = count_lines_in_file(non_existent_file_path)
    print(f"Line count result for missing file: {lines_count_2}")

    ...
    File 'sample_file.txt' has 3 lines.
    Line count result for missing file: -1

```

In the center, a modal window titled "Generate a Python function that reads a .txt file and returns the number of lines in it. Examples: File with 3 lines → Output: 3 File with 10 lines → Output: 10" is displayed. It includes a "Accept and run" button.

On the right, there is a sidebar titled "Release note" with the date "2026-01-20" and a list of updates, and a "Python package" sidebar listing various libraries like accelerate, astropy, numpy, etc.

## CODE EXPLANATION:

- Opens the file in read mode
- Counts lines using `sum(1 for _ in f)`
- Returns `-1` if file does not exist
- Few-shot prompting uses **multiple example outputs**

**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.**