

## AI-ASSISTANT CODING – ASSIGNMENT 4.3

HALLTICKET:2303A54038

BATCH:48

### 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)

**Prompt : checking leap year or not**

```
def is_leap_year(year):  
    """  
    Check if a year is a leap year.  
  
    A year is a leap year if:  
    - It's divisible by 4 AND not divisible by 100, OR  
    - It's divisible by 400  
    """  
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):  
        return True  
    return False  
  
# Test the function  
year = int(input("Enter a year: "))  
if is_leap_year(year):  
    print(f"{year} is a leap year")  
else:  
    print(f"{year} is not a leap year")
```

Output:

```
Enter a year: 2026  
2026 is not a leap year
```

### 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

Prompt:input=10cm

output=3.94 inches

conversion from cm to inches using formula

```
# Convert centimeters to inches
# Formula: inches = centimeters / 2.54

cm = 10
inches = cm / 2.54

print(f"input={cm}cm")
print(f"output={inches:.2f} inches")
```

Output:

```
input=10cm
output=3.94 inches
```

Testcases:

```
# Test cases for cm_to_inches function
def test_cm_to_inches():
    # Test case 1: Standard conversion
    assert cm_to_inches(10) == 10 / 2.54
    assert round(cm_to_inches(10), 2) == 3.94

    # Test case 2: Zero centimeters
    assert cm_to_inches(0) == 0

    # Test case 3: Larger value
    assert round(cm_to_inches(100), 2) == 39.37

    # Test case 4: Decimal input
    assert round(cm_to_inches(2.54), 2) == 1.0

    # Test case 5: Small value
    assert round(cm_to_inches(1), 2) == 0.39

    print("All tests passed!")

if __name__ == "__main__":
    test_cm_to_inches()
```

```
input=10cm
output=3.94 inches
output=3.94 inches
All tests passed!•
```

### 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

Prompt:

Input="John Smith" → output = "Smith, John"

Input="Anita Rao" → output = "Rao, Anita"

Generate a code to print the above examples

usage of python function,handle names correctly

```
def format_name(full_name):
    """Convert 'First Last' to 'Last, First' format"""
    parts = full_name.split()
    if len(parts) == 2:
        first, last = parts
        return f"{last}, {first}"
    return full_name
print(format_name("John Smith"))
print(format_name("Anita Rao"))
```

Output:

```
PS C:\AI assistant> .\code.py
Smith, John
Rao, Anita
```

#### 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

Prompt:

**ZERO SHOT** : generate a code to count number of vowels

```
def count_vowels(text):
    vowels = "aeiouAEIOU"
    return sum(1 for char in text if char in vowels)

print(count_vowels("John Smith"))
print(count_vowels("Anita Rao"))
```

OUTPUT:

```
PS C:\AI assitant>
2
5
```

### ONE SHOT :

Input = "John Smith" → output = 2

Input = "Anita Rao" → output = 3

Generate a code that counts how many vowels in the above strings

```
def count_vowels(text):
    vowels = "aeiouAEIOU"
    return sum(1 for char in text if char in vowels)

print(count_vowels("John Smith"))
print(count_vowels("Anita Rao"))
```

Output:

```
PS C:\AI assitant>
2
5
```

### 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

#### Prompt:

##### Input:

File has 3 lines

##### Output:

Line count = 3

#### Example 2

##### Input:

File has 5 lines

##### Output:

Line count = 5

c:\Users\thimi\OneDrive\Documents\AI PROJECT PARENT(1).docx

- code to print the Count of the number of lines in the file

```
def count_lines(filename):  
    with open(filename, 'r') as file:  
        line_count = sum(1 for line in file)  
    return line_count  
  
filename = r"C:\AI assitant\lab 4.3.py"  
print(f"Line count = {count_lines(filename)}")
```

Output:

```
PS C:\AI assitant>  
Line count = 25
```

---