

AI Assisted Coding

Assignment – 4.3

Name :Y. Pranith Sai

Roll No : 2303A54072

Batch : 48

Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques

Task 1: Zero-Shot Prompting – Leap Year Check

Scenario:

Zero-shot prompting involves giving instructions without providing examples.

Prompt used : #Write a Python function that takes a year as input and checks whether it is a leap year.

#The function should return an appropriate message indicating whether the year is a leap year or not.

#Do not include any input-output examples.

- **Generated code :**

```
def is_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False

year = int(input("Enter year: "))
print(is_leap_year(year))
```

- **Sample Input :**

Enter year:2024

- **Sample Output :**

True

- **Short Explanation of Logic :**

the program checks the leap year conditions using logical operators .a year is a leap year if the it is divisible by 400 or divisible by 4 but not divisible by 100 . the logic is implemented directly without providing any prior .

```

1 #Write a python function that takes a year as input and checks whether it is a leap year.
2 #The function should return an appropriate message indicating whether the year is a leap year.
3 #Do not include any input-output examples.
4
5 def is_leap_year(year):
6     if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
7         return True
8     else:
9         return False
10
11 year = int(input("Enter year: "))
12 print(is_leap_year(year))
13
14

```

The function has been added to the file. It correctly implements the leap year logic: a year is a leap year if it's divisible by 4, but not by 100 unless also divisible by 400. The code has no syntax errors.

Read task1.py
Edited task1.py +0
Ran pylanceSyntaxErrors - pylance mcp server (MCP Server)

Write a Python function that takes a year as input and checks whether it is a leap year.
The function should return an appropriate message indicating whether the year is a leap year or not.
Do not provide any examples.

Click to Edit

Read task1.py
Edited task1.py +5
Ran pylanceSyntaxErrors - pylance mcp server (MCP Server)

The function has been added to the file. It checks if the given year is a leap year and returns a message stating whether it is or not. The code has no syntax errors.

Grok Code Fast 1 + 1x

task1.py
Describe what to build next
Agent Auto

Ln 14, Col 1 Spaces: 4 UTF-8 CRLF Python 3.14.2

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

❖ Scenario : One-shot prompting guides AI using a single example.

- **Prompt used :**
- # Write a Python function to convert centimeters to inches
- # Use the formula inches = centimeters / 2.54
- # Example:
- # Input: 10 cm
- # Output: 3.94 inches
- **Generated code :**

```

def cm_to_inches(centimeters):
    """
    Convert centimeters to inches.

    Args:
        centimeters: A number representing length in centimeters

    Returns:
        A float representing the length in inches
    """
    inches = centimeters / 2.54
    return round(inches, 2)

# Example usage
if __name__ == "__main__":
    cm_value = 10
    result = cm_to_inches(cm_value)
    print(f"Input: {cm_value} cm")
    print(f"Output: {result} inches")

```

- **Sample Input :**
 - Enter value in cm: 25.4
 - **Sample Output :**

Short Explanation of Logic :

The function takes centimeters as input and converts it into inches using the standard formula Cm %2.54 one example in the prompt helps the ai to generate the logic accurately in one short

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows a project structure under "AI CODING" containing files like assignment 4.3.py, task1.py, task2.py, etc.
- Editor:** The main editor area displays the content of task2.py. The code defines a function `cm_to_inches` that converts centimeters to inches using the formula $\text{inches} = \text{centimeters} / 2.54$. It includes docstrings and example usage.
- Terminal:** The terminal window shows command-line interactions for running task1.py and task2.py, and for testing the `cm_to_inches` function with input 10 cm resulting in 3.94 inches.
- Bottom Status Bar:** Shows file path (master), line count (Ln 20), character count (Col 38), spaces count (Spaces: 4), encoding (UTF-8), language (Python), and line height (3142).

Task3: Few-shot prompting-Name Formatting

Scenario:

Few-shot prompting improves accuracy by providing multiple examples.



- **Prompt used :**
- # Write a Python function to format names as "Last, First"
- # Examples:
- # "John Smith" -> "Smith, John"
- # "Anita Rao" -> "Rao, Anita"

- **Generated code :**

```
"""  
  
def format_name(full_name):  
    """  
        Format a full name as "Last, First"  
  
    Args:  
        full_name (str): A full name in "First Last" format  
  
    Returns:  
        str: The name formatted as "Last, First"  
    """  
  
    parts = full_name.split()  
    if len(parts) >= 2:  
        first_name = parts[0]  
        last_name = parts[-1]  
        return f"{last_name}, {first_name}"  
    return full_name  
  
  
# Test examples  
print(format_name("John Smith"))      # Smith, John  
print(format_name("Anita Rao"))       # Rao, Anita
```

- **Sample Input :**

Enter full name: John Smith

- **Sample Output : Smith, John**

Short Explanation of Logic : The function splits the full name into first and last names and rearranges them in the required format it providing multiple examples improves few short

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists several files: task1.py, task2.py, task3.py, assignment 4.3.py, task1.py, task2.py, task3.py, assignment1st.py, assignment3.1.py, task2.py, task2.py, task3.py, task4py, and task5py. The task3.py file is open in the editor, displaying the following Python code:

```

File Edit Selection View Go Run Terminal Help ← → Q: AI CODING
EXPLORER ... task1.py task2.py task3.py
assignment 4.3.py > task3.py > ...
1 def format_name(full_name):
2     """
3         Format a full name as "Last, First"
4
5     Args:
6         full_name (str): A full name in "First Last" format
7
8     Returns:
9         str: The name formatted as "Last, First"
10    """
11    parts = full_name.split()
12    if len(parts) >= 2:
13        first_name = parts[0]
14        last_name = parts[-1]
15        return f'{last_name}, {first_name}'
16    return full_name
17
18
19 # Test examples
20 print(format_name("John Smith")) # Smith, John
21 print(format_name("Anita Rao")) # Rao, Anita

```

The bottom right corner of the code editor shows the status bar with "In 21, Col 51 Spaces:4 UTF-8 {} Python 3.14.2". Below the code editor is the Terminal panel, which shows the command line output:

```

PROBLEMS OUTPUT TERMINAL PORTS
DEBUG CONSOLE Filter (e.g. text, exclude, \escape)
TERMINAL
ers\baidu\OneDrive\Desktop\AI CODING\assignment 4.3.py\task2.py"
Input: 10 cm
Output: 3.94 inches
PS C:\Users\baidu\OneDrive\Desktop\AI CODING> & C:/Users/baidu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/baidu/OneDrive/Desktop/Assignment 4.3.py/task3.py"
Smith, John
Rao, Anita
PS C:\Users\baidu\OneDrive\Desktop\AI CODING>

```

The status bar at the bottom of the terminal panel also shows "In 21, Col 51 Spaces:4 UTF-8 {} Python 3.14.2".

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

- **Scenario:**
Different prompt strategies may produce different code quality.
- **Prompt 1: Zero-Shot Prompting**

Write a Python function that counts the number of vowels in a given string.

The function should return the total count.

Do not provide any examples.

Generated code:

```
def count_vowels(string):
    vowels = "aeiouAEIOU"
    count = 0
    for char in string:
        if char in vowels:
```

- **Sample Input :**

Enter string: Hello World

- **Sample Output :**

Number of vowels: 3

- **Prompt 2: Few-Shot Prompting**

- Write a Python function to count vowels in a string.
- Examples:
- Input: "hello" → Output: 2
- Input: "AI Assisted Coding" → Output: 7

❖ **Generated code:**

```
❖ ef count_vowels(string):  
❖     """Count the number of vowels in a string."""  
❖     vowels = "aeiouAEIOU"  
❖     return sum(1 for char in string if char in vowels)  
❖  
❖ # Test cases  
❖ print(count_vowels("hello")) # Output: 2  
❖ print(count_vowels("AI Assisted Coding")) # Output: 7
```

- **Sample Input :**
Enter string: Hello World
- **Sample Output :3**

Explanation (Few-Shot);

The function uses a predefined vowel set and python's sum with the generator expression to count vowels efficiently the logic is compact and easier to understand due to example .

```
File Edit Selection View Go Run Terminal Help <- > AI CODING
```

```
EXPLORER ... task1.py task2.py task3.py task401.py x
```

```
assignment 4.3.py
```

```
task1.py task2.py task3.py task401.py
```

```
assignment1st assignment3.1.py
```

```
task2.py task3.py task4.py task5.py
```

```
task401.py
```

```
assignment 4.3.py
```

```
task1.py task2.py task3.py task401.py
```

```
def count_vowels(string):  
    vowels = "aeiouAEIOU"  
    count = 0  
    for char in string:  
        if char in vowels:  
            count += 1  
    return count
```

```
PROBLEMS OUTPUT TERMINAL PORTS
```

```
TERMINAL
```

```
DING> & C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODING/assignment 4.3.py/task3.py"  
Smith, John  
● Ran, Anita  
PS C:/Users/bindu/OneDrive/Desktop/AI CODING/& C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODING/assignment 4.3.py/task401.py"  
PS C:/Users/bindu/OneDrive/Desktop/AI CODING>
```

```
> OUTLINE  
> TIMELINE  
> PROJECTS
```

```
master* 0 11 0 0 > Indexing completed.
```

```
Ln 7, Col 17 Spaces:4 UTF-8 Python 3.14.2
```

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows files in the "AI CODING" folder, including assignment 4.3.py, task1.py, task2.py, task3.py, task42.py, task401.py, assignment1st.py, assignment3.1.py, task2.py, task3.py, task4.py, and task5.py.
- Editor:** The current file is task42.py, which contains a Python function to count vowels in a string.
- Terminal:** The terminal window shows the output of running task401.py and task42.py. It includes a prompt for "Rao, Anita" and the command "Open file in editor (ctrl + click)".
- Bottom Status Bar:** Shows indexing completed, master branch, 0 errors, 0 warnings, 0 info, 0 tips, and Python 3.14.2.

Comparison Table

Criteria	Zero-Shot	Few-Shot
Accuracy	Correct	Correct
Readability	Medium	High
Logical Clarity	Simple	Very clear
Code Length	Longer	Shorter

Task 5: Few-Shot Prompting – File Handling (Line Count)

Scenario : You are building a text file analyzer that counts the number of lines in a file. Clear examples help ensure correct file handling.

Prompt used :

```
# Write a Python function to read a text file
# The function should take the file name as input
# Open the file in read mode# Count the total number of lines in the file
#Return the line count as an integer
# Example:
# If "file.txt" contains 3 lines, the output should be 3
```

Generated code :

```
def count_lines(filename):
    """
    Read a text file and count the total number of lines.

    Args:
        filename (str): The name of the file to read

    Returns:
        int: The total number of lines in the file
    """
    try:
        with open(filename, 'r') as file:
            line_count = sum(1 for line in file)
        return line_count
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
        return 0

# Example usage
if __name__ == "__main__":
    result = count_lines("file.txt")
    print(f"Total lines: {result}")
```

- **Sample Input :**
- `print(count_lines("file.txt"))`
- **Sample Output :**

3

Short Explanation of Logic :

The program reads a text file in read mode and counts the number of lines present in it. A counter variable is used to keep track of the number of lines. Each time a line is read from the file, the counter increases by one. After reading the entire file, the function returns the total number of lines. This method ensures accurate line counting and is easy to understand.

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files task1.py, task2.py, task3.py, task401.py, task42.py, and task5.py.
- Code Editor:** Displays a Python script named assignment 4.3.py. The code defines a function count_lines that reads a file and counts its lines. It includes error handling for file not found.
- Run and Debug View:** Shows a "Run and Debug" button highlighted. A tooltip says: "To customize Run and Debug create a launch.json file. Debug using a terminal command or in an interactive chat." Below it, there's a "Show automatic Python configurations" button.
- Terminal:** Shows the output of running the script: "PS C:\Users\bindu\Desktop\AI CODING> & C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/Desktop/AI CODING/assignment 4.3.py/task5.py" Error: File 'file.txt' not found. Total lines: 0".
- Breakpoints:** Shows a list of breakpoints: Raised Except..., Uncaught Except..., and User Uncaught... None are currently active.

```
assignment 4.3.py > task5.py > ...
1 def count_lines(filename):
2     """
3         Read a text file and count the total number of lines.
4     """
5     Args:
6         filename (str): The name of the file to read
7
8     Returns:
9         int: The total number of lines in the file
10    """
11    try:
12        with open(filename, 'r') as file:
13            line_count = sum(1 for line in file)
14        return line_count
15    except FileNotFoundError:
16        print(f"Error: File '{filename}' not found.")
17    return 0
18
19
20 # Example usage
21 if __name__ == "__main__":
22     result = count_lines("file.txt")
23     print(f"Total lines: {result}")
```


