

Assignment 4.3

Task 1: Zero-Shot Prompting – Leap Year Check

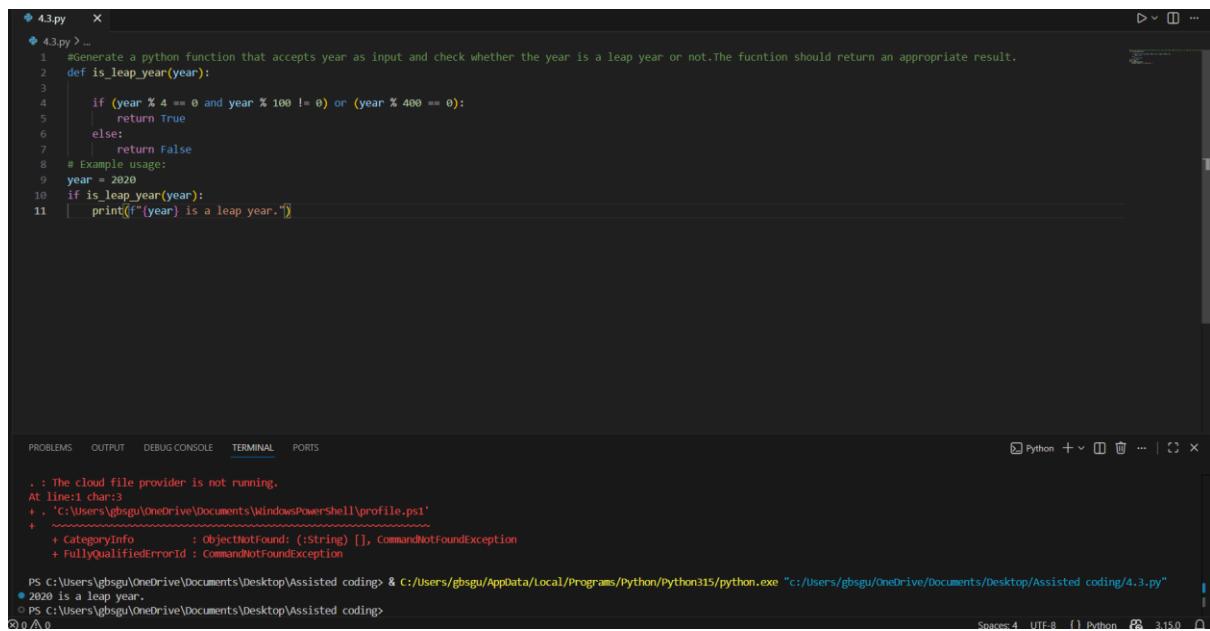
Scenario

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



```
#generate a python function that accepts year as input and check whether the year is a leap year or not.The fucntion should return an appropriate result.
def is_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False
# Example usage:
year = 2020
if is_leap_year(year):
    print(f"{year} is a leap year.")
```

The screenshot shows a code editor window with a Python file named "4.3.py". The code defines a function "is_leap_year" that checks if a given year is a leap year according to the rules: if the year is divisible by 4 but not by 100, or if it is divisible by 400. It includes a comment and an example usage with year 2020. Below the code editor is a terminal window showing the output of running the script, which prints "2020 is a leap year." The terminal also shows some PowerShell error messages about the cloud file provider not running.

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

The screenshot shows a code editor interface with a dark theme. At the top, there's a status bar with icons for file operations like 'New', 'Open', 'Save', etc. Below the status bar is a navigation bar with tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected), and 'PORTS'. The main area contains a Python script named '4.3(2).py'.

```
❶ 4.3(2).py > ...
1 #Generate a python function that converts centimeters to inches.
2 #Example : input 10cm -> output 3.94inches
3 def cm_to_inches(cm):
4     inches = cm / 2.54
5     return round(inches, 2)
6 # Example usage:
7 cm_value = 38
8 inches_value = cm_to_inches(cm_value)
9 print(f"{cm_value} cm is equal to {inches_value} inches.")
```

Below the script, the terminal pane displays the execution of the script and its output:

```
+ CategoryInfo          : ObjectNotFound: (:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFound
```

```
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/4.3(2).py"
❷ 10 cm is equal to 3.94 inches.
❸ 20 cm is equal to 7.87 inches.
❹ 30 cm is equal to 11.81 inches.
❺ PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/4.3(2).py"
❻ 10 cm is equal to 3.94 inches.
❼ 20 cm is equal to 7.87 inches.
❽ 30 cm is equal to 11.81 inches.
❾ PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding>
```

Task 3: Few-Shot Prompting – Name Formatting

Scenario

Few-shot prompting improves accuracy by providing multiple examples.

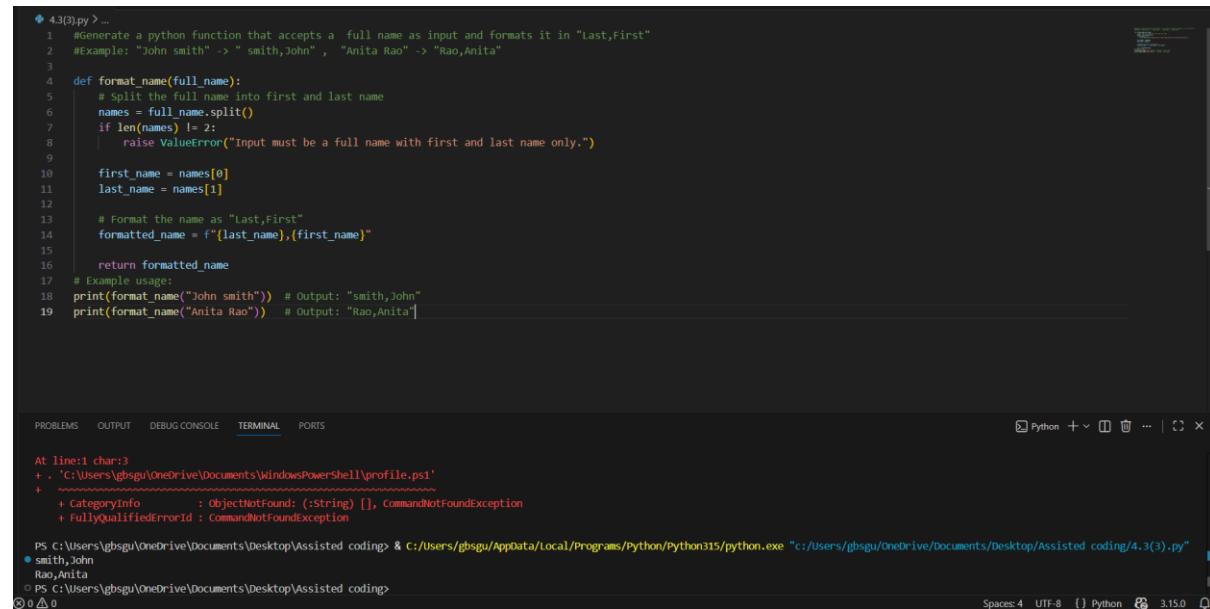
Task Description

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"



```
◆ 4.3(3).py > ...
1  #Generate a python function that accepts a full name as input and formats it in "Last,First"
2  #Example: "John smith" -> " smith,John" , "Anita Rao" -> "Rao,Anita"
3
4  def format_name(full_name):
5      # split the full name into first and last name
6      names = full_name.split()
7      if len(names) != 2:
8          raise ValueError("Input must be a full name with first and last name only.")
9
10     first_name = names[0]
11     last_name = names[1]
12
13     # Format the name as "Last,First"
14     formatted_name = f'{last_name},{first_name}'
15
16     return formatted_name
17
18 # Example usage:
19 print(format_name("John smith")) # Output: "smith,John"
20 print(format_name("Anita Rao")) # Output: "Rao,Anita"
```

The screenshot shows a code editor window with the Python file 4.3(3).py open. The code defines a function format_name that takes a full name as input, splits it into first and last names, and returns them in 'Last, First' format. It includes two print statements demonstrating the usage with 'John Smith' and 'Anita Rao'. Below the code editor, a terminal window shows the output of running the script: 'smith,John' and 'Rao,Anita'. The terminal also shows some PowerShell error messages related to command not found.

The screenshot shows a code editor window with a dark theme. At the top, there's a status bar with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, showing a command-line interface with the following output:

```
+ CategoryInfo          : ObjectNotFound: (:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/documents/Desktop/Assisted coding/4.3(3).py"
smith,John
Rao,Anita
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/documents/Desktop/Assisted coding/4.3(3).py"
Rushikesh,Eaga
Guptha,Bhargav
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding>
```

The code itself is a Python script named 4.3(3).py, which defines a function format_name that takes a full name as input and returns it in "Last,First" format. It includes error handling for cases where the input is not a full name.

```
4.3(3).py > ...
1 #Generate a python function that accepts a full name as input and formats it in "Last,First"
2 #example: "John smith" -> " smith,John" , "Anita Rao" -> "Rao,Anita"
3
4 def format_name(full_name):
5     # split the full name into first and last name
6     names = full_name.split()
7     if len(names) != 2:
8         raise ValueError("Input must be a full name with first and last name only.")
9
10    first_name = names[0]
11    last_name = names[1]
12
13    # Format the name as "Last,First"
14    formatted_name = f"({last_name}),({first_name})"
15
16    return formatted_name
17
18 # Example usage:
19 print(format_name("Eaga Rushikesh"))
20 print(format_name("Bhargav Guptaha"))
```

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

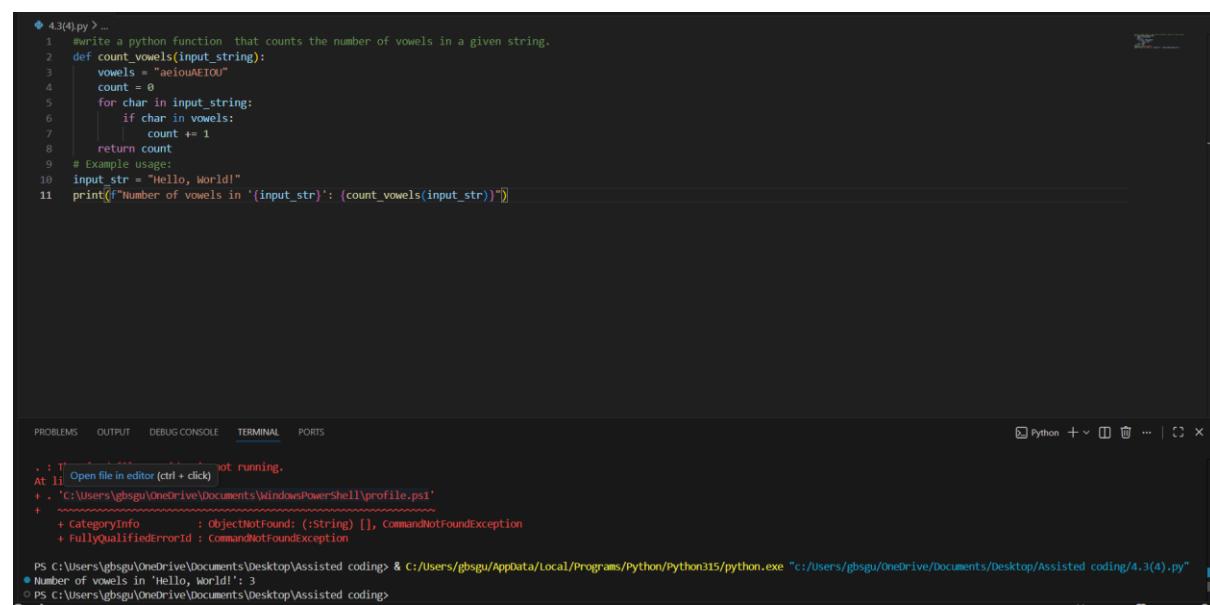
Scenario

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

Zero shot prompt:



```
4.3(4).py > ...
1 #write a python function that counts the number of vowels in a given string.
2 def count_vowels(input_string):
3     vowels = "aeiouAEIOU"
4     count = 0
5     for char in input_string:
6         if char in vowels:
7             count += 1
8     return count
9 # Example usage:
10 input_str = "Hello, World!"
11 print(f"Number of vowels in '{input_str}': {count_vowels(input_str)}")
```

The screenshot shows a terminal window with the following content:

```
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/4.3(4).py"
● Number of vowels in 'Hello, World!': 3
○ ps C:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding
```

Few shot prompt:

```
4.3(4).py > ...
1  #Write a python function that counts vowels in a given string.
2  #Example: input: "Hello world"
3  #           output: 3
4  def count_vowels(input_string):
5      vowels = "aeiouAEIOU"
6      count = 0
7      for char in input_string:
8          if char in vowels:
9              count += 1
10     return count
11 # Example usage:
12 input_string = "Hello world"
13 print(count_vowels(input_string)) # Output: 3
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

+ . 'c:\Users\gbsgu\OneDrive\Documents\WindowsPowerShell\profile.ps1'
+ ~~~~~CategoryInfo : ObjectNotFound: (:)[], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/4.3(4).py"
● Number of vowels in 'Hello, World!': 3
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/4.3(4).py"
● 3
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding>

Ln 13, Col 47 Spaces: 4 UTF-8 CRLF {} Python 3.15.0

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

```
4.3(5).py
1 #write a python function that reads a text file and counts the number of lines in it.
2 def count_lines_in_file(file_path):
3     try:
4         with open(file_path, 'r') as file:
5             lines = file.readlines()
6             return len(lines)
7     except FileNotFoundError:
8         print("The file was not found.")
9         return 0
10    except Exception as e:
11        print(f"An error occurred: {e}")
12    return 0
13 # Example usage
14 file_path = 'Sample.txt'
15 line_count = count_lines_in_file(file_path)
16 print(f"The number of lines in the file is: {line_count}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + v 🌐 ⚙️ ⚡ ⚓ ✎
```

```
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/4.3(5).py"
The number of lines in the file is: 7
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding>
```

The AI understands the requirement through few-shot prompting by observing example input and output. It opens the text file safely using with open() in read mode. The file is read line by line, avoiding loading the entire file into memory. Each line increments a counter to determine the total number of lines. The final count is returned as the output.