

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: M.Tech. and MCA		Assignment Type: Lab	
Course Coordinator Name		Venkataramana Veeramsetty	
Course Code		Course Title	AI Assisted Problem Solving Using Python
Year/Sem	I/I	Regulation	R24
Date and Day of Assignment	Week3 - Monday	Time(s)	
Duration	2 Hours	Applicable to Batches	M.Tech. and MCA
AssignmentNumber: 4.3(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>Lab Objectives:</p> <ul style="list-style-type: none"> • To explore and apply different levels of prompt examples in AI-assisted code generation. • To understand how zero-shot, one-shot, and few-shot prompting affect AI output quality. • To evaluate the impact of context richness and example quantity on AI performance. • To build awareness of prompt strategy effectiveness for different problem types. <p>Lab Outcomes (LOs):</p> <p>After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> • 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 	Week3 - Monday

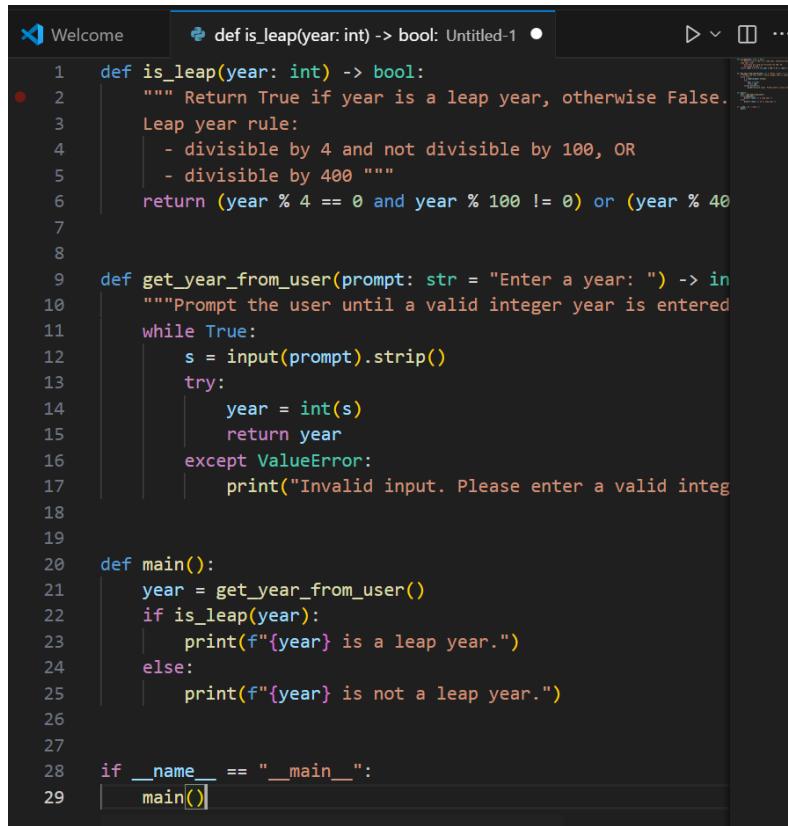
	<p>responses.</p> <ul style="list-style-type: none"> • Compare AI outputs across the three prompting strategies. <p>Task Description#1</p> <ul style="list-style-type: none"> • Zero-shot: Prompt AI to write a function that checks whether a given year is a leap year. <p>Expected Output#1</p> <ul style="list-style-type: none"> • AI-generated function with no examples provided <p>Task Description#2</p> <ul style="list-style-type: none"> • One-shot: Give one input-output example to guide AI in writing a function that converts centimeters to inches. <p>Expected Output#2</p> <ul style="list-style-type: none"> • Function with correct conversion logic <p>Task Description#3</p> <ul style="list-style-type: none"> • Few-shot: Provide 2–3 examples to generate a function that formats full names as “Last, First”. <p>Expected Output#3</p> <ul style="list-style-type: none"> • Well-structured function respecting the examples <p>Task Description#4</p> <ul style="list-style-type: none"> • Compare zero-shot and few-shot prompts for writing a function that counts the number of vowels in a string. <p>Expected Output#4</p> <ul style="list-style-type: none"> • Functional output and comparative reflection <p>Task Description#5</p> <ul style="list-style-type: none"> • Use few-shot prompting to generate a function that reads a .txt file and returns the number of lines. <p>Expected Output#5</p> <ul style="list-style-type: none"> • Working file-processing function with AI-guided logic <p>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</p> <p>Evaluation Criteria:</p>	
--	--	--

	Criteria	Max Marks	
	Zero Shot (Task #1)	2.5	
	One Shot (Task#2)	2.5	
	Few Shot (Task#3 & Task #5)	2.5	
	Comparison (Task#4)	2.5	
	Total	10 Marks	

Task Description#1

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

PROMPT: Write a program that takes a year from the user and prints whether it is a leap year or not.



```

Welcome      def is_leap(year: int) -> bool: Untitled-1 ●
1  def is_leap(year: int) -> bool:
2      """ Return True if year is a leap year, otherwise False.
3      Leap year rule:
4          - divisible by 4 and not divisible by 100, OR
5          - divisible by 400 """
6      return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)
7
8
9  def get_year_from_user(prompt: str = "Enter a year: ") -> int:
10     """Prompt the user until a valid integer year is entered
11     while True:
12         s = input(prompt).strip()
13         try:
14             year = int(s)
15             return year
16         except ValueError:
17             print("Invalid input. Please enter a valid integer")
18
19
20 def main():
21     year = get_year_from_user()
22     if is_leap(year):
23         print(f"{year} is a leap year.")
24     else:
25         print(f"{year} is not a leap year.")
26
27
28 if __name__ == "__main__":
29     main()

```

Expected Output#1

- AI-generated function with no examples provided

Practical output:

```

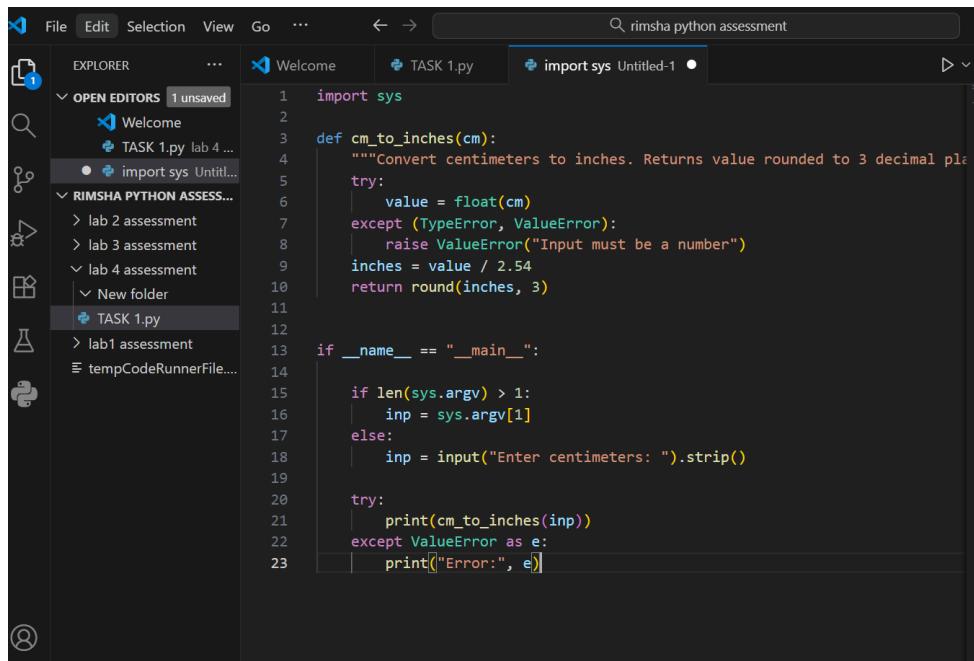
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> python -u "c:
● \Users\rimsha\OneDrive\Desktop\rimsha python assessment\tempCodeRunnerFile.
python"
Enter a year: 2012
2012 is a leap year.
❖ PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment>

```

Task Description#2

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

PROMPT: Write a function that converts a value in centimeters to inches.



```

File Edit Selection View Go ...
EXPLORER ... Welcome TASK 1.py import sys Untitled-1
OPEN EDITORS 1 unsaved
RIMSHA PYTHON ASSESS...
lab 2 assessment
lab 3 assessment
lab 4 assessment
New folder
TASK 1.py
lab1 assessment
tempCodeRunnerFile...
import sys
def cm_to_inches(cm):
    """Convert centimeters to inches. Returns value rounded to 3 decimal places."""
    try:
        value = float(cm)
    except (TypeError, ValueError):
        raise ValueError("Input must be a number")
    inches = value / 2.54
    return round(inches, 3)

if __name__ == "__main__":
    if len(sys.argv) > 1:
        inp = sys.argv[1]
    else:
        inp = input("Enter centimeters: ").strip()

    try:
        print(cm_to_inches(inp))
    except ValueError as e:
        print("Error:", e)

```

Expected Output#2

- Function with correct conversion logic

Practical output:

```
> TERMINAL
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> python -u "c:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\tempCodeRunnerFile.python"
Enter centimeters: 6
2.362
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment>
```

Task Description#3

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

PROMPT: Write a function that asks for a full name and prints it as ‘Last, First’.

```
Welcome TASK 1.py TASK 2.py TASK 3.py CHAT
lab 4 assessment > TASK 3.py > get_formatted_name
1 def get_formatted_name(prompt="Enter full name: "):
2     """
3         Prompts the user for a full name and returns it formatted as "Last, First".
4         If only one name is entered, it is returned unchanged.
5     """
6     full = input(prompt).strip()
7     if not full:
8         return ""
9     parts = full.split()
10    if len(parts) == 1:
11        return parts[0]
12    last = parts[-1]
13    first = " ".join(parts[:-1])
14    return f"[last], {first}"
15
16 if __name__ == "__main__":
17     result = get_formatted_name()
18     if result:
19         print(result)
```

Expected Output#3

- Well-structured function respecting the examples

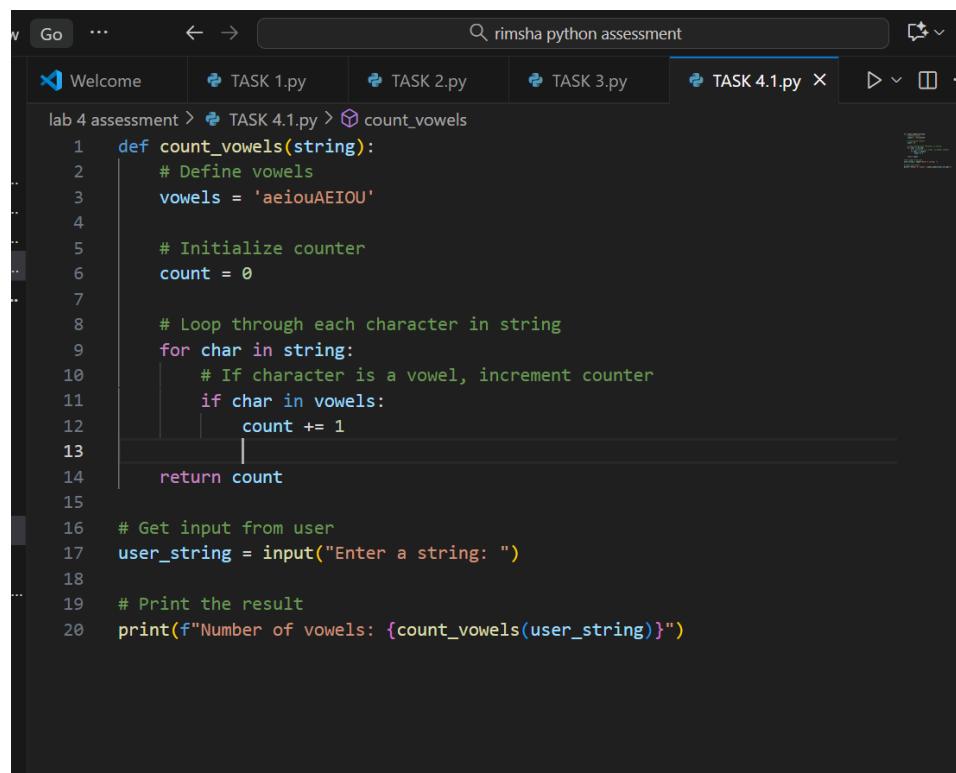
Practical output:

```
● PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> python -u "c:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\tempCodeRunnerFile.python"
Enter full name: RAHIMATHUNNISA RIMSHA
RIMSHA, RAHIMATHUNNISA
❖ PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment>
```

Task Description#4

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

ZERO-SHOT PROMPT: Write a function that takes a string as input and returns the number of vowels in it.



```
lab 4 assessment > TASK 4.1.py > count_vowels
1 def count_vowels(string):
2     # Define vowels
3     vowels = 'aeiouAEIOU'
4
5     # Initialize counter
6     count = 0
7
8     # Loop through each character in string
9     for char in string:
10         # If character is a vowel, increment counter
11         if char in vowels:
12             count += 1
13
14     return count
15
16 # Get input from user
17 user_string = input("Enter a string: ")
18
19 # Print the result
20 print(f"Number of vowels: {count_vowels(user_string)}")
```

FEW-SHOT PROMPT: Write a user input function that reads a string and prints how many vowels it has.

The screenshot shows a code editor interface with a dark theme. The title bar says "rimsha python assessment". Below the title bar, there are tabs for "TASK 1.py", "TASK 2.py", "TASK 3.py", "TASK 4.1.py", and "TASK 4.2.py". The "TASK 4.2.py" tab is active. The code editor displays the following Python script:

```
def count_vowels(text: str) -> int:
    """
    Return the number of vowels (a, e, i, o, u) in the given string.
    Counts both uppercase and lowercase vowels.
    """
    if text is None:
        return 0
    vowels = set("aeiouAEIOU")
    return sum(1 for ch in text if ch in vowels)

# Example usage / quick tests
if __name__ == "__main__":
    samples = [
        "Hello, World!",
        "AEIOU aeiou",
        "",
        "Python Programming",
        None
    ]
    for s in samples:
        print(repr(s), "->", count_vowels(s))
```

Difference Between Zero-Shot and Few-Shot Prompts

- Zero-Shot Prompt:

The model is given only the instruction (no examples).

Example: Write a function to count vowels.

- Few-Shot Prompt:

The model is given one or more examples along with the instruction.

Example: Showing sample input-output before asking to write the function.

Expected Output#4

- Functional output and comparative reflection

Practical output:

ZERO-SHOT:

```
● PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> python -u "c:\Users\ri  
rive\Desktop\rimsha python assessment\tempCodeRunnerFile.python"  
Enter a string: HLO WORLD  
Number of vowels: 2  
● PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> python -u "c:\Users\ri  
rive\Desktop\rimsha python assessment\tempCodeRunnerFile.python"  
Enter a string: HI AM RIMSHA  
Number of vowels: 4  
❖ PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> █
```

FEW-SHOT:

A screenshot of a terminal window in Visual Studio Code. The terminal tab is selected at the top. The output shows a Python script running, with several test cases and their results. The results are color-coded: green for strings like 'Hello, World!', red for 'AEIOU aeiou', blue for '' (empty string), and purple for 'Python Programming' and 'None'.

```
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> python -u "c:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\tempCodeRunnerFile.python"
'Hello, World!' -> 3
'AEIOU aeiou' -> 10
'' -> 0
'Python Programming' -> 4
None -> 0
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment>
```

Task Description#5

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

lab 4 assessment > TASK 5.py > ...

```

1  # ...existing code...
2  def _count_lines_in_file(path: str) -> int:
3      """Return the number of lines in the given file path."""
4      with open(path, "r", encoding="utf-8") as f:
5          return sum(1 for _ in f)
6
7  def count_lines_in_file(path: str) -> int:
8      """
9          Return the number of lines in the given .txt file path.
10         Appends '.txt' if the extension is omitted and raises exceptions on error.
11     """
12
13     if not path:
14         raise ValueError("path must be a non-empty string")
15     path = path.strip().strip('"')
16     if not path.lower().endswith(".txt"):
17         path += ".txt"
18     return _count_lines_in_file(path)
19
20  def count_lines_from_user_input(prompt: str = "Enter path to a .txt file: "):
21      """
22          Prompt the user for a .txt filename (or path), open it and return the number of lines.
23          The function will append '.txt' if the user omits the extension and will raise an exception if the file does not exist.
24      """
25
26      while True:
27          try:
28              user_input = input(prompt).strip().strip('"')
29          except KeyboardInterrupt, EOFError:
30              raise # let caller handle interruption
31
32  
```

lab 4 assessment > TASK 5.py > ...

```

19  def count_lines_from_user_input(prompt: str = "Enter path to a .txt file: "):
20
21      if not user_input:
22          print("No filename entered. Please try again.")
23          continue
24
25      if not user_input.lower().endswith(".txt"):
26          user_input += ".txt"
27
28      try:
29          return _count_lines_in_file(user_input)
30      except FileNotFoundError:
31          print(f"File not found: {user_input}")
32      except IsADirectoryError:
33          print(f"Path is a directory, not a file: {user_input}")
34      except Exception as e:
35          print(f"Error opening file: {e}")
36
37  if __name__ == "__main__":
38      try:
39          count = count_lines_from_user_input()
40          print(count)
41      except Exception:
42          print("Operation cancelled or failed.")
43
44  # ...existing code...
45
46  
```

Expected output:

```
● PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> python -u "c:\Users\rimsha\OneD  
rive\Desktop\rimsha python assessment\tempCodeRunnerFile.python"  
Enter path to a .txt file: "C:\Users\rimsha\OneDrive\Desktop\SAMPLE_txt.txt"  
7  
❖ PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> █
```