|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | | | |
| **Program Name:** B. Tech | | | | **Assignment Type: Lab** | | | **Academic Year:**2025-2026 | | |
| **Course Coordinator Name** | | | | Venkataramana Veeramsetty | | | | | |
| **Instructor(s) Name** | | | | |  | | --- | | Dr. V. Venkataramana (Co-Ordinator) | | Dr. T. Sampath Kumar | | Dr. Pramoda Patro | | Dr. Brij Kishor Tiwari | | Dr.J.Ravichander | | Dr. Mohammand Ali Shaik | | Dr. Anirodh Kumar | | Mr. S.Naresh Kumar | | Dr. RAJESH VELPULA | | Mr. Kundhan Kumar | | Ms. Ch.Rajitha | | Mr. M Prakash | | Mr. B.Raju | | Intern 1 (Dharma teja) | | Intern 2 (Sai Prasad) | | Intern 3 (Sowmya) | | NS\_2 ( Mounika) | | | | | | |
| **Course Code** | | | 24CS002PC215 | **Course Title** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **Date and Day**  **of Assignment** | | | Week2 - Monday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | |  | | | |
| **Assignment Number:4.1**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***Expected Time***  ***to complete*** |  |
|  | 1 | Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques  **Lab Objectives:**   * 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.   **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 the three prompting strategies.   **Task #1 – Zero-Shot Prompting with Conditional Validation**  Objective  Use zero-shot prompting to instruct an AI tool to generate a function that validates an Indian mobile number.  Requirements   * The function must ensure the mobile number:   + Starts with 6, 7, 8, or 9   + Contains exactly 10 digits   Expected Output   * A valid Python function that performs all required validations without using any input-output examples in the prompt.   **Task #2 – One-Shot Prompting with Edge Case Handling**  Objective  Use one-shot prompting to generate a Python function that calculates the factorial of a number.  Requirements   * Provide one sample input-output pair in the prompt to guide the AI. * The function should handle:   + 0! correctly   + Negative input by returning an appropriate message   Expected Output   * A Python function with correct factorial logic and edge case handling, generated from a single example.   **Task #3 – Few-Shot Prompting for Nested Dictionary Extraction**  Objective  Use few-shot prompting (2–3 examples) to instruct the AI to create a function that parses a nested dictionary representing student information.  Requirements   * The function should extract and return:   + Full Name   + Branch   + SGPA   Expected Output   * A reusable Python function that correctly navigates and extracts values from nested dictionaries based on the provided examples.   **Task #4 – Comparing Prompting Styles for File Analysis**  Objective  Experiment with zero-shot, one-shot, and few-shot prompting to generate functions for CSV file analysis.  Requirements   * Each generated function should:   + Read a .csv file   + Return the total number of rows   + Count the number of empty rows   + Count the number of words across the file   Expected Output   * Working Python functions for each prompting style, with a brief reflection comparing their accuracy, clarity, and efficiency.   **Task #5 – Few-Shot Prompting for Text Processing and Word** **Frequency**  Objective  Use few-shot prompting (with at least 3 examples) to generate a Python function that processes text and analyzes word frequency.  Requirements  The function must:   * Accept a paragraph as input * Convert all text to lowercase * Remove punctuation * Return the most frequently used word   Expected Output   * A functional Python script that performs text cleaning, tokenization, and returns the most common word using only the examples provided in the prompt   **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  **Evaluation Criteria:**   | **Criteria** | **Max Marks** | | --- | --- | | Zero Shot (Task #1) | 0.5 | | One Shot (Task#2) | 0.5 | | Few Shot (Task#3, Task#4 & Task #5) | 1.5 | | **Total** | **2.5 Marks** | | | | | | | Week2 - Monday |  |

**Assignment-4.1**

**NAME:** J.SAI GANESH **DATE**:13-08-2025

**ROLL\_NO**:2403A51301

**#TASK1(ASKED IN CHATGPT5)**

**PROMPT**: Create a Python function named validate-mobile that:

* Takes a single parameter (the mobile number as a string)
* if it starts with 6, 7, 8, or 9 and has exactly 10 digits

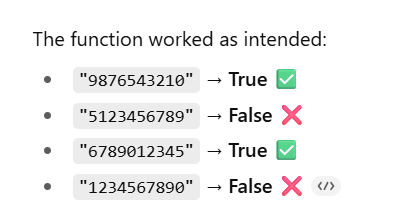
**CODE**:  
 def validate-mobile(mobile-number: str) -> bool:

if len(mobile-number) == 10 and mobile-number.isdigit() and mobile-number[0] in {'6', '7', '8', '9'}:

return True

return False

**OUT PUT:**



**OBSERVATION:** “When tested, it worked as expected — returning True for valid numbers and False for invalid ones. “ ,there fore it takes it’s own input while we running it show us out put.

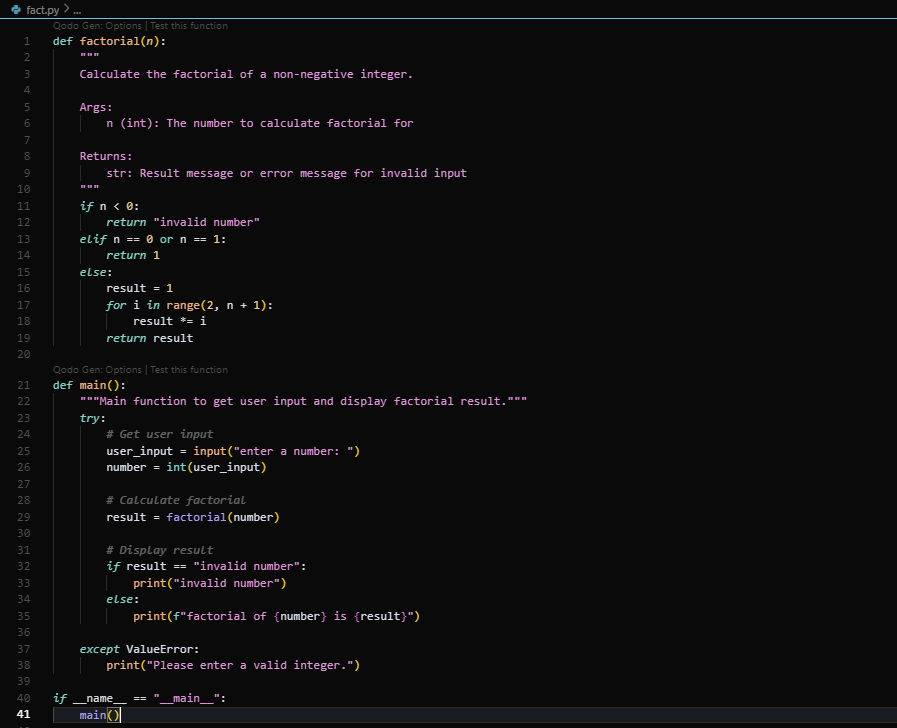
**#TASK2(cursor)  
PROMPT:**

generate a python function that calculates factorial of a number,if user provides negativenumber it should return invalid number.

Sample example:

input: enter a number:3

output: factiorial of 3 is 6  
  
**CODE:**



**Output:**



**OBSERVATION:** Providing a single input-output example helped guide the function to handle both normal and edge cases. The one-shot prompt improved clarity compared to zero-shot.

**#TASK3**

**Prompt:** I want a Python function that extracts student details (Full Name, Branch, SGPA) from nested dictionaries.

Here are examples:

Ex 1:Input: { "student": { "personal": {"first\_name": "SAIGANESH", "last\_name": "JOGIPARTHY"},

"academic": {"branch": "CSE", "sgpa": 8.2}}}

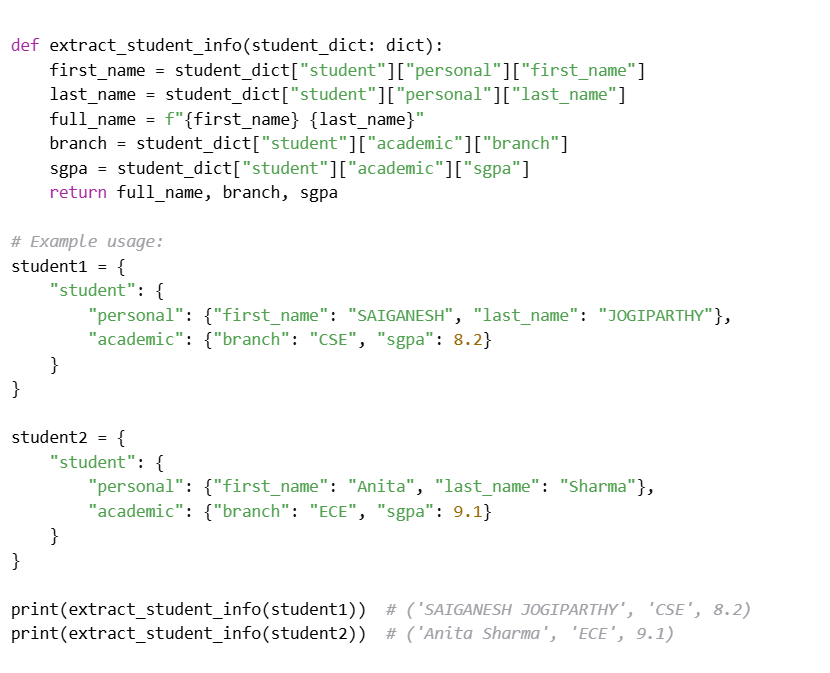
Output: ("Ravi Kumar", "CSE", 8.2)

Ex 2:Input: {"student": {"personal": {"first\_name": "Anita", "last\_name": "Sharma"},

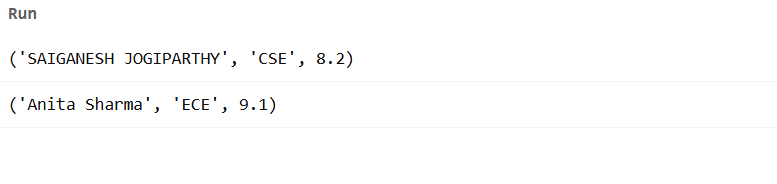
"academic": {"branch": "ECE", "sgpa": 9.1} }}

Output: ("Anita Sharma", "ECE", 9.1)

Now write a reusable Python function that works for such nested dictionaries.

**CODE:**

OUT PUT:



**OBSERVATION:** Using multiple examples ensured the function extracted values from different nested structures accurately. Few-shot prompting is effective for structured data parsing**.**

**#TASK4:  
PROMPT:  
ZERO-SHOT PROMPT**: Write a Python function that reads a .csv file and returns:

• Total number of rows

• Number of empty rows

• Total number of words across the file

Do not use examples. Just provide the function.  
**ONE-SHOT PROMPT**: Write a Python function that reads a .csv file and returns:

• Total number of rows

• Number of empty rows

• Total number of words across the file

Ex:Suppose sample.csv has the following:

Name, Age

Ravi, 21

Anita, 22(empty row)

The output should be:

Total rows: 4

Empty rows: 1

Total words: 5

Now write the function.  
**FEW-SHOT PROMPT**: I want a Python function that analyzes a .csv file.

Here are examples:

Ex1:Input File:

A,B

1,2

3,4(empty row)

Output: Total rows = 3, Empty rows = 1, Total words = 4

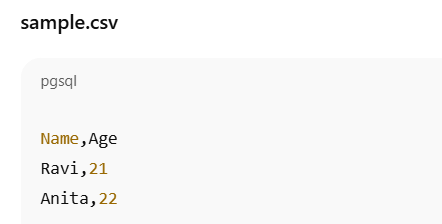
Ex2:Input File:

Name, Age

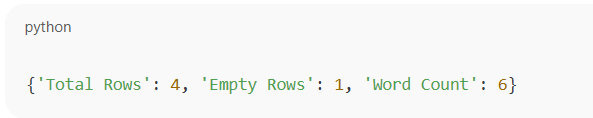
Ravi, 21

Anita, 22(empty row)

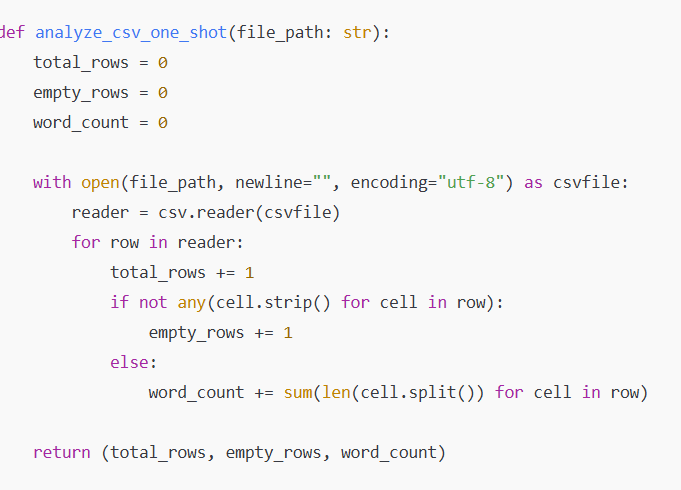
Output: Total rows = 4, Empty rows = 1, Total words = 5

Now generate a reusable function that works for any .csv file and gives these three counts.  
**Zero-Shot CODE:**  


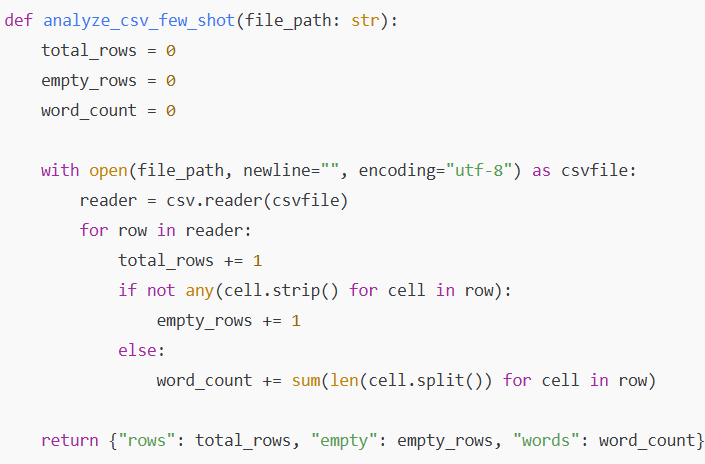
OUT PUT:



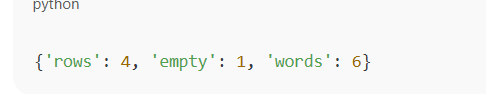
**ONE-SHOT CODE:**



  
**FEW-SHOT CODE:**



**OUTPUT:**



**OBSERVATION**: All three prompting styles produced the same correct result. Zero-shot worked, but one-shot and few-shot made requirements clearer, reducing ambiguity in expected output format.

**#TASK5:  
PROMPT:** Write a Python function that processes text and returns the most frequently used word.

The function must:

• Accept a paragraph as input

• Convert all text to lowercase

• Remove punctuation

• Return the most common word

Here are examples:

Ex1:Input: "Hello world! Hello AI."

Output: "hello"

Ex2:Input: "Python is great, and python is powerful."

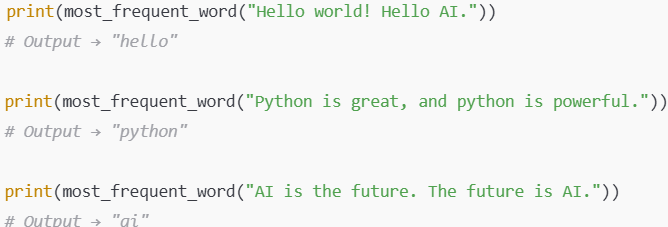
Output: "python"

Ex3:Input: "AI is the future. The future is AI."

Output: "is"

Now write the function.  
**CODE:**

  
**OUT PUT:**

****

**OBSERVATION:** Multiple examples guided the function to correctly clean, tokenize, and count words. Few-shot prompting is most useful in text-processing tasks where multiple rules must be combined.