**AI Assisted coding**

**Assignment-4.1**

**Name: SRIMANI**

**HT NO.:2403A51275**

**Batch NO.:12**

**Task-1**

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

A screenshot of a computer program

AI-generated content may be incorrect.

Output:

A screenshot of a computer code

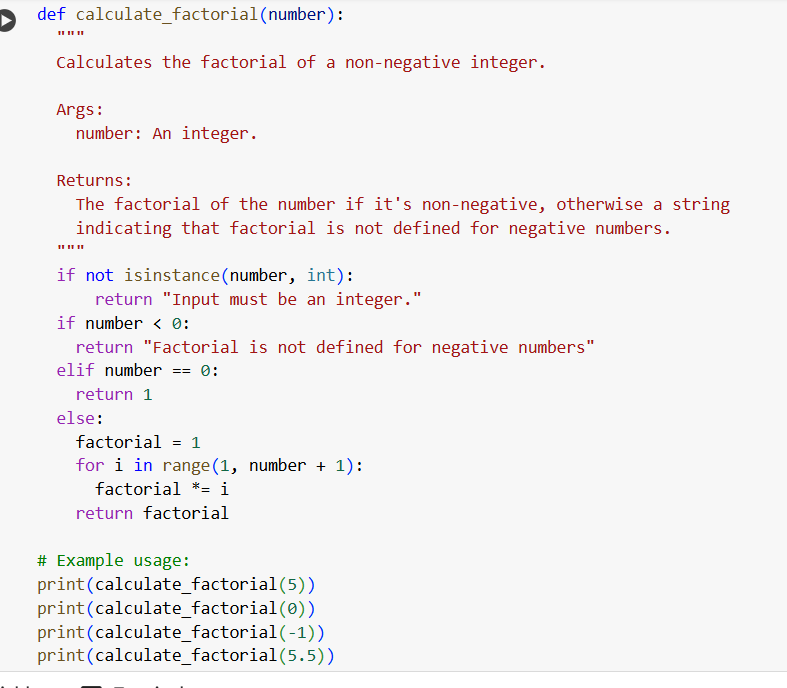
AI-generated content may be incorrect.

**Task-2:**

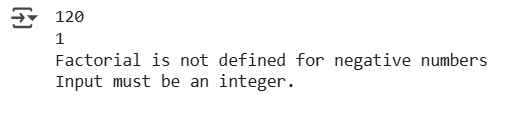
*Use one-shot prompting to generate a Python function that calculates the factorial of a number.*

*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*

**

**Output:**

****

**Task-3:**

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
* A screenshot of a computer program

  AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**Task-4:**

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
* A screenshot of a computer

  AI-generated content may be incorrect.
* A screenshot of a computer program

  AI-generated content may be incorrect.

A screenshot of a computer code

AI-generated content may be incorrect.

**Task-5:**

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



A screen shot of a computer code

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Explanation:**

Certainly. The code I provided demonstrates how to use few-shot prompting to generate a Python function for text processing and word frequency analysis.

Here's a breakdown of the code:

1. **Markdown Cells:** The markdown cells provide the context for the few-shot prompting. They introduce the task and present three examples (text\_data\_1, text\_data\_2, and text\_data\_3) with their expected outputs. These examples serve as the "shots" to guide the AI model in generating the desired function.
2. **Code Cell with Examples :** This cell defines the three example text snippets (text\_data\_1, text\_data\_2, text\_data\_3) that were mentioned in the markdown cell. These are the inputs used to illustrate the function's expected behavior.
3. **Code Cell with Function Definition :** This is the core of the solution. The analyze word frequency function is defined here.
   * It imports the string module for punctuation handling and Counter from the collections module for counting word frequencies.
   * It takes a paragraph string as input.
   * cleaned\_text = paragraph.lower(): Converts the input paragraph to lowercase to ensure that words like "This" and "this" are counted as the same word.
   * cleaned\_text = cleaned\_text.translate(str.maketrans('', '', string.punctuation)): Removes punctuation from the text. str.maketrans('', '', string.punctuation) creates a translation table that maps each punctuation character to None, effectively removing them when translate() is applied.
   * words = cleaned\_text.split(): Splits the cleaned text into a list of individual words based on whitespace.
   * word\_counts = Counter(words): Uses the Counter object to count the occurrences of each word in the words list.
   * if word\_counts: return word\_counts.most\_common(1)[0][0] else: return None: This part finds the most common word. word\_counts.most\_common(1) returns a list of the single most common word and its count (e.g., [('word', 3)]). [0] accesses the first element of this list (the tuple), and [0] again accesses the word itself from the tuple. The if word\_counts: check handles the case of an empty input paragraph.
4. **Testing the Function:** The print() statements at the end of the function definition cell call the analyze\_word\_frequency function with the example text data and print the returned most frequent word, demonstrating that the function works as expected based on the few-shot examples provided.

In essence, the few-shot prompting approach involved showing the AI what kind of input the function would receive and what the corresponding output should be for a few different cases. This guided the AI to generate the Python code that performs the necessary text cleaning, tokenization, and frequency analysis to achieve those results.