ASSIGNMENT-4.3

TASK-1

CODE:

A group of colorful objects

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

AI-generated content may be incorrect.

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

EXPLANATION:

Here's a step-by-step explanation:

1. **Markdown Cell (Cell ID: bea066c3):** This cell introduces the concept of using zero-shot prompting for the task of checking whether a year is a leap year. It sets the context for the code that follows.
2. **Code Cell (Cell ID: 4463fd63):** This is the main cell where the zero-shot prompting is performed.
   * zero\_shot\_prompt\_leap\_year = """...""": This multiline string variable contains the prompt that will be sent to the Gemini model. It includes:
     + An instruction: "Write a Python function called is\_leap\_year that takes a year (integer) as input and returns True if it is a leap year, and False otherwise." This clearly tells the model what function to create and what its purpose is.
     + Since this is a zero-shot prompt, there are *no* examples provided. The model must rely on its internal knowledge and training data to understand the concept of a leap year and generate the correct Python logic for the function.
   * response\_zero\_shot\_leap\_year = model.generate\_content(zero\_shot\_prompt\_leap\_year): This line calls the generate\_content method of the model object (which should be an initialized Gemini model). It sends the zero\_shot\_prompt\_leap\_year string to the model. The model processes this prompt, using only the instruction to generate a Python function. The model's response is stored in the response\_zero\_shot\_leap\_year variable.
   * print("Zero-shot response for leap year check:"): This line prints a descriptive label before displaying the model's output.
   * print(response\_zero\_shot\_leap\_year.text): This line retrieves the generated text from the model's response (response\_zero\_shot\_leap\_year.text) and prints it to the console. This is where you will see the Python function that the Gemini model generated based solely on the zero-shot prompt.
3. **Markdown Cell (Cell ID: c9734398):** This cell provides a detailed explanation of the code in the previous cell. It breaks down the components of the prompt and explains how the generate\_content method is used in the zero-shot context. It also mentions what to expect in the output.

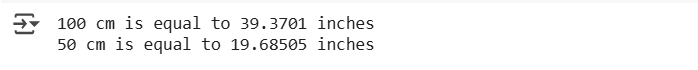
TASK-2

CODE:

A white background with red and green text

AI-generated content may be incorrect.

OUTPUT:



EXPLANATION;

Here's a step-by-step explanation:

1. **Markdown Cell (Cell ID: bf0a103d):** This cell introduces the concept of using one-shot prompting for the task of converting centimeters to inches. It sets the context for the code that follows.
2. **Code Cell (Cell ID: 59e93f87):** This is the main cell where the one-shot prompting is performed.
   * one\_shot\_prompt\_cm\_to\_inches = """...""": This multiline string variable contains the prompt that will be sent to the Gemini model. It includes:
     + An instruction: "Write a Python function called cm\_to\_inches that takes a measurement in centimeters (number) as input and returns the equivalent measurement in inches." This clearly tells the model what function to create and what its purpose is.
     + Example:: This keyword signifies the beginning of the one-shot example.
     + One example: Input: 10 and Output: 3.93701. This single example provides the model with a concrete instance of the desired input (a number representing centimeters) and the expected output (the corresponding value in inches). Even with just one example, the model can learn the input/output format and infer the underlying relationship or conversion.
   * response\_one\_shot\_cm\_to\_inches = model.generate\_content(one\_shot\_prompt\_cm\_to\_inches): This line calls the generate\_content method of the model object (which should be an initialized Gemini model). It sends the one\_shot\_prompt\_cm\_to\_inches string to the model. The model processes this prompt, using the instruction and the single example to generate a Python function. The model's response is stored in the response\_one\_shot\_cm\_to\_inches variable.
   * print("One-shot response for cm to inches conversion:"): This line prints a descriptive label before displaying the model's output.
   * print(response\_one\_shot\_cm\_to\_inches.text): This line retrieves the generated text from the model's response (response\_one\_shot\_cm\_to\_inches.text) and prints it to the console. This is where you will see the Python function that the Gemini model generated based on the one-shot prompt.
3. **Markdown Cell (Cell ID: 1fe38b26):** This cell provides a detailed explanation of the code in the previous cell. It breaks down the components of the prompt and explains how the generate\_content method is used. It also mentions what to expect in the output.

TASK-3

CODE:

A screenshot of a computer code

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

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

AI-generated content may be incorrect.

EXPLANATION:

Certainly! I'll provide a step-by-step explanation of the code I generated for formatting names using few-shot prompting.

Here's a breakdown of the cells and their purpose:

1. **Markdown Cell (Cell ID: 7b79521d):** This cell serves as an introduction to the task. It explains that we will be using few-shot prompting to generate a Python function for formatting full names as "Last, First" and that we will provide examples to guide the model.
2. **Code Cell (Cell ID: a80ca26a):** This is the core cell where the few-shot prompting happens.
   * few\_shot\_prompt\_name\_format = """...""": This multiline string variable holds the prompt that will be sent to the Gemini model. It contains:
     + An instruction: "Write a Python function called format\_name that takes a full name (string) as input and returns the name formatted as 'Last, First'." This tells the model what kind of function to create and what it should do.
     + Examples:: This keyword signals the start of the few-shot examples.
     + Three examples: Each example shows an Input: (a full name) and the desired Output: (the name formatted as "Last, First"). These examples are crucial for few-shot prompting as they demonstrate the expected behavior and output format to the model.
   * response\_few\_shot\_name\_format = model.generate\_content(few\_shot\_prompt\_name\_format): This line calls the generate\_content method of the model object (which should be an initialized Gemini model). It passes the few\_shot\_prompt\_name\_format string as input to the model. The model processes the prompt and generates a response based on the instructions and examples. The response is stored in the response\_few\_shot\_name\_format variable.
   * print("Few-shot response for formatting names:"): This line prints a descriptive label before the model's response.
   * print(response\_few\_shot\_name\_format.text): This line accesses the text content of the model's response (response\_few\_shot\_name\_format.text) and prints it. This is where you will see the Python function generated by the Gemini model based on the few-shot prompt.
3. **Markdown Cell (Cell ID: edcf32c3):** This cell provides an explanation of the previous code cell, detailing what each part of the code does and how it contributes to the few-shot prompting process. It reiterates the purpose of the prompt, the examples, and the call to the model.

TASK-4

PROMPT:

A close up of text

AI-generated content may be incorrect.

CODE:

A screenshot of a computer

AI-generated content may be incorrect.

EXPLANATION:

Absolutely! Let me explain the code I generated.

The notebook demonstrates the difference between zero-shot and few-shot prompting using the Gemini API to generate a Python function that counts vowels in a string.

1. **Set up the API key:** This cell imports the necessary libraries and retrieves your Google API key from Colab secrets to configure the Gemini API.
2. **Initialize the Generative Model:** This cell initializes the GenerativeModel with the specified model ('gemini-1.5-flash-latest') which will be used to generate the code.
3. **Zero-shot prompting:**
   * This markdown cell introduces the concept of zero-shot prompting.
   * The following code cell defines a zero\_shot\_prompt which is a simple instruction for the model to write the count\_vowels function without any examples.
   * It then calls the model.generate\_content() method with this prompt and prints the model's response.
4. **Few-shot prompting:**
   * This markdown cell introduces the concept of few-shot prompting.
   * The following code cell defines a few\_shot\_prompt which includes the same instruction as the zero-shot prompt but also provides three examples of input strings and their expected vowel counts.
   * It then calls the model.generate\_content() method with this prompt and prints the model's response.

By comparing the output of the two generate\_content() calls, you can see how providing examples in the few-shot prompt influences the generated function compared to the zero-shot approach. The few-shot response is likely to be more accurate or robust because the examples guide the model's understanding of the desired behavior.

TASK-5

CODE:

A white rectangular object with black text

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

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

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

AI-generated content may be incorrect.

A close-up of a text

AI-generated content may be incorrect.