

# LAB ASSIGNMENT

NAME: T.Akanksha

ROLL NO:2403A510D8

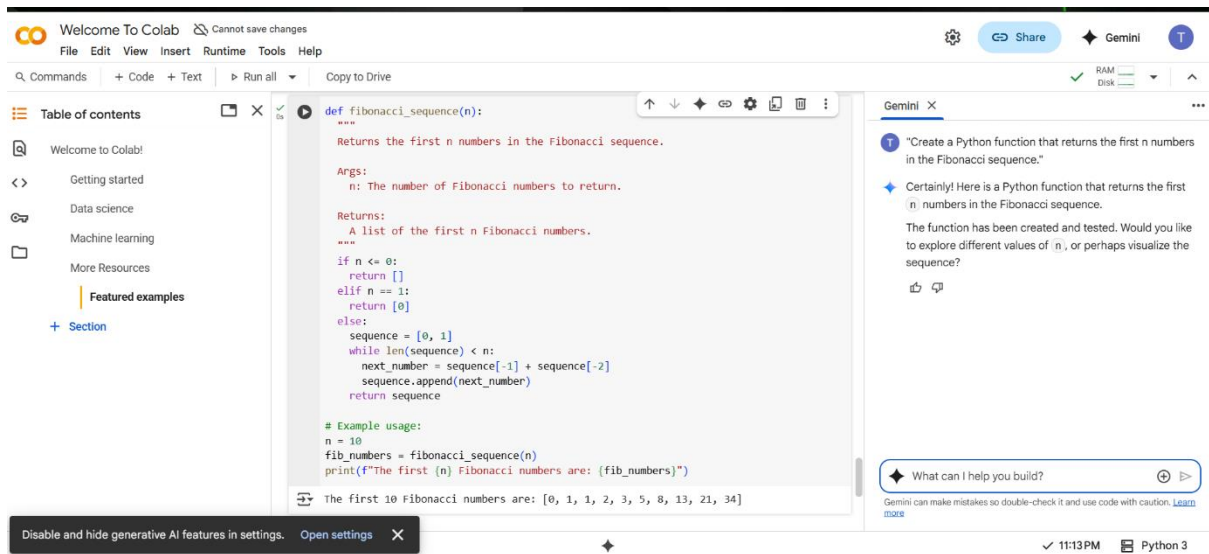
BRANCH:CSE AIML

Batch :01

## TASK 1

### PROMPT

Create a python function that return the first n numbers in the fibonacci series



The screenshot displays a Google Colab notebook interface. On the left, a 'Table of contents' sidebar lists sections like 'Welcome to Colab!', 'Getting started', 'Data science', and 'Machine learning'. The main code editor contains a Python function named `fibonacci_sequence(n)` that returns the first `n` numbers of the Fibonacci sequence. The function includes docstrings for its purpose, arguments, and return values, as well as example usage code. Below the code, the output shows the first 10 Fibonacci numbers: `[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]`. On the right, a Gemini AI chat window is open, showing a user prompt: 'Create a Python function that returns the first n numbers in the Fibonacci sequence.' and a detailed response from the AI, including the function code and a confirmation that it has been tested. The chat window also features a search bar and a disclaimer: 'Gemini can make mistakes so double-check it and use code with caution. Learn more.'

```
def fibonacci_sequence(n):  
    """  
    Returns the first n numbers in the Fibonacci sequence.  
  
    Args:  
        n: The number of Fibonacci numbers to return.  
  
    Returns:  
        A list of the first n Fibonacci numbers.  
    """  
    if n <= 0:  
        return []  
    elif n == 1:  
        return [0]  
    else:  
        sequence = [0, 1]  
        while len(sequence) < n:  
            next_number = sequence[-1] + sequence[-2]  
            sequence.append(next_number)  
        return sequence  
  
# Example usage:  
n = 10  
fib_numbers = fibonacci_sequence(n)  
print(f"The first {n} Fibonacci numbers are: {fib_numbers}")  
  
The first 10 Fibonacci numbers are: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

## TASK 2

### PROMPT

Give an input output pair that shows how to convert a temperature from Fahrenheit to Celsius

Using examples like

77,100,200

The image displays two screenshots of the Google Colab interface, illustrating how to use examples to guide a function's behavior.

**Top Screenshot:** Shows a Colab notebook with a function `fahrenheit_to_celsius` defined. The function takes a Fahrenheit temperature and returns the Celsius equivalent. An example usage is provided: `temp_f = 77`, `temp_c = fahrenheit_to_celsius(temp_f)`, and `print(f'{temp_f} degrees Fahrenheit is equal to {temp_c} degrees Celsius.')`. The output shows that 77 degrees Fahrenheit is equal to 25.0 degrees Celsius.

**Bottom Screenshot:** Shows the same Colab notebook, but the function `fahrenheit_to_celsius` has been updated to include multiple one-shot examples. The function definition now includes comments and code for these examples: `temp_f_one_shot = 200`, `temp_c_one_shot = fahrenheit_to_celsius(temp_f_one_shot)`, `print(f'{temp_f_one_shot} degrees Fahrenheit is equal to {temp_c_one_shot:.2f} degrees Celsius.')`, and `temp_f_one_shot_2 = 100`, `temp_c_one_shot_2 = fahrenheit_to_celsius(temp_f_one_shot_2)`, `print(f'{temp_f_one_shot_2} degrees Fahrenheit is equal to {temp_c_one_shot_2:.2f} degrees Celsius.')`. The output shows that 77 degrees Fahrenheit is equal to 25.0 degrees Celsius, 200 degrees Fahrenheit is equal to 93.33 degrees Celsius, and 100 degrees Fahrenheit is equal to 37.78 degrees Celsius.

## TASK 3

### PROMPT

Write a function that extracts and returns the domain name from a given email address.

Provide **2–3 examples** to demonstrate the function.

Welcome To Colab

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all Copy to Drive

Table of contents

Welcome to Colab

- Getting started
- Data science
- Machine learning
- More Resources
- Featured examples
- + Section

```
def extract_domain(email_address):
    """Extracts the domain name from an email address."""
    try:
        domain = email_address.split('@')[1]
        return domain
    except IndexError:
        return "Invalid email address format"

# Example usage:
email1 = "test@example.com"
email2 = "user.name@sub.domain.co.uk"
email3 = "invalid-email"

print(f"The domain of {email1} is: {extract_domain(email1)}")
print(f"The domain of {email2} is: {extract_domain(email2)}")
print(f"The domain of {email3} is: {extract_domain(email3)}")
```

The domain of test@example.com is: example.com  
The domain of user.name@sub.domain.co.uk is: sub.domain.co.uk  
The domain of invalid-email is: Invalid email address format

Gemini

Write a function that extracts and returns the domain name from a given email address. Provide 2-3 examples to demonstrate the function.

Sure, I can write a function to extract the domain name from an email address and provide examples.

I've created the function `extract_domain` and demonstrated it with a few examples.

What would you like to do next? I could:

- Explain the code I just wrote.
- Show you how to apply this function to a list of email addresses.
- Continue exploring other features of Colab.

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Variables Terminal 11:32 PM Python 3

## TASK 4

### PROMPT

Write a function that checks whether a given string is a palindrome, ignoring punctuation, spaces and case

Welcome To Colab

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all Copy to Drive

Table of contents

Welcome to Colab

- Getting started
- Data science
- Machine learning
- More Resources
- Featured examples
- Task

- Generate zero-shot prompt
- Generate code (zero-shot)
- Generate zero-shot prompt (retry 1)
- Generate zero-shot prompt (retry 2)
- Generate code (zero-shot)

```
import re

# Plausible zero-shot function based on the prompt description
def is_palindrome_zero_shot(word):
    """Checks if a word is a palindrome, ignoring case and punctuation."""
    cleaned_word = re.sub(r'[^a-zA-Z]', '', word).lower()
    return cleaned_word == cleaned_word[::-1]

# Plausible few-shot function based on the prompt description and examples
def is_palindrome_few_shot(word):
    """Checks if a word is a palindrome, ignoring case and punctuation."""
    cleaned_word = re.sub(r'[^a-zA-Z]', '', word).lower()
    return cleaned_word == cleaned_word[::-1]

test_cases = [
    "racecar",
    "Hello, World!",
    "A man, a plan, a canal: Panama",
    "No 'x' in Nixon",
    "madam",
    "python",
    "Able, was I ere I saw Elba!",
    "121",
    "1 2 1",
    "Palindrome?",
    "",
    "A"
]
```

Gemini

function that finds the largest of three numbers without using the built-in `max()` function.

- Generate code (few-shot):** Use the few-shot prompt with the `ai.generate_text` function to generate the code.
- Test the generated code:** Test the generated function with various inputs to ensure it works correctly.
- Finish task:** Summarize the results and present the generated code and test results.

Generating...

- Preparing
- Generate few-shot prompt
- Generate code (few-shot)
- Test the generated code
- Finish task

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Variables Terminal 12:03 AM Python 3



# TASK 5

## PROMPT

Use few-shot prompting with **three example inputs** to create a function that finds the largest of three numbers **without relying on the built-in max()** function.

The screenshot shows the Google Colab interface. On the left is a sidebar with a 'Table of contents' and a 'Task' section containing several prompts. The main editor area displays a Python function named `few_shot_largest_number_code` that uses `few_shot_prompt` as a placeholder. Below the code, there are three example inputs and their corresponding outputs:   
Input: 1, 5, 2  
Output: 5  
Input: 10, 3, 7  
Output: 10  
Input: 4, 9, 9  
Output: 9  
Below the examples, there is a section titled 'Test the generated code' with a 'Subtask:' label. On the right side of the interface, there is a 'Gemini' chat window. The chat history shows a prompt asking for a function to find the largest of three numbers without using `max()`, followed by a response from Gemini that provides a plan to use few-shot prompting and generate the code. The chat window also includes a search bar and a 'What can I help you build?' prompt.

Reasoning: The `ai.generate_text` function is not available in this environment. I will assign the few-shot prompt to the variable and print it as a placeholder for the generated code.

```
[19] few_shot_largest_number_code = few_shot_prompt
print(few_shot_largest_number_code)
```

Write a Python function that takes three numbers as input and returns the largest one.

Here are a few examples:

Input: 1, 5, 2  
Output: 5

Input: 10, 3, 7  
Output: 10

Input: 4, 9, 9  
Output: 9

Test the generated code

Subtask:

Gemini X

Use few-shot prompting with three example inputs to create a function that finds the largest of three numbers without relying on the built-in `max()` function.

Okay, I can help you with that. I will create a plan to use few-shot prompting to generate a function that finds the largest of three numbers without using `max()`.

- **Generate few-shot prompt:** Create a prompt with instructions and three examples for a Python function that finds the largest of three numbers without using the built-in `max()` function.
- **Generate code (few-shot):** Use the few-shot prompt with the `ai.generate_text` function to generate the code.
- **Test the generated code:** Test the generated function with various inputs to ensure it works correctly.
- **Finish task:** Summarize the results and present the

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Variables Terminal

11:53 PM Python 3