

LAB ASSIGNMENT

NAME: M.TANVI

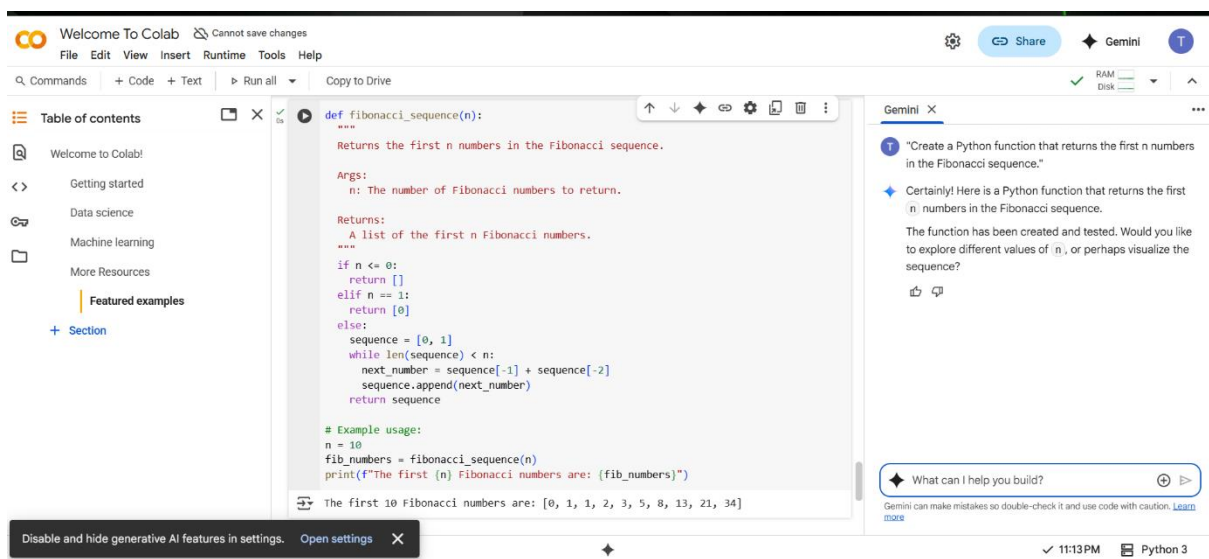
ROLL NO:2403A510A4

BRANCH:CSE AIML

TASK 1

PROMPT

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



The screenshot displays a Google Colab environment. On the left, a 'Table of contents' sidebar lists 'Welcome to Colab!', 'Getting started', 'Data science', 'Machine learning', and 'More Resources'. The main code editor contains a Python function `def fibonacci_sequence(n):` that returns the first `n` numbers of the Fibonacci sequence. The function includes docstrings for its purpose, arguments, and return values. It uses a `while` loop to generate the sequence. Below the function, an example usage is shown: `n = 10`, `fib_numbers = fibonacci_sequence(n)`, and `print(f"The first (n) Fibonacci numbers are: {fib_numbers}")`. The output of the code is displayed at the bottom: 'The first 10 Fibonacci numbers are: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]'. On the right, a Gemini AI chat interface is open, showing a prompt: 'Create a Python function that returns the first n numbers in the Fibonacci sequence.' and a response: 'Certainly! Here is a Python function that returns the first n numbers in the Fibonacci sequence. The function has been created and tested. Would you like to explore different values of n, or perhaps visualize the sequence?'. A status bar at the bottom indicates '11:13 PM' and 'Python 3'.

```
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 a Google Colab notebook interface. The top screenshot shows a code cell with a function definition for converting Fahrenheit to Celsius, an example usage, and the output: "77 degrees Fahrenheit is equal to 25.0 degrees Celsius." The bottom screenshot shows the same notebook with additional examples added to the code cell, including one-shot examples for 200 and 100 degrees Fahrenheit, and their corresponding Celsius outputs. The Gemini chat interface on the right provides context and suggestions for using the function.

Top Screenshot:

```
[2] def fahrenheit_to_celsius(fahrenheit):  
    """Converts Fahrenheit to Celsius."""  
    celsius = (fahrenheit - 32) * 5/9  
    return celsius  
  
# Example usage:  
temp_f = 77  
temp_c = fahrenheit_to_celsius(temp_f)  
print(f"{temp_f} degrees Fahrenheit is equal to {temp_c} degrees Celsius.")
```

Output: 77 degrees Fahrenheit is equal to 25.0 degrees Celsius.

Bottom Screenshot:

```
[5] def fahrenheit_to_celsius(fahrenheit):  
    """Converts Fahrenheit to Celsius."""  
    celsius = (fahrenheit - 32) * 5/9  
    return celsius  
  
# Example usage:  
temp_f = 77  
temp_c = fahrenheit_to_celsius(temp_f)  
print(f"{temp_f} degrees Fahrenheit is equal to {temp_c} degrees Celsius.")  
  
# One-shot example:  
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.")  
  
# Another one-shot example:  
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.")
```

Output: 77 degrees Fahrenheit is equal to 25.0 degrees Celsius.
200 degrees Fahrenheit is equal to 93.33 degrees Celsius.
100 degrees Fahrenheit is equal to 37.78 degrees Celsius.

Gemini Chat:

Using only the above example, create a function that converts a temperature value from Fahrenheit to Celsius.

Okay, I can create a function based on the examples provided in the notebook that converts Fahrenheit to Celsius.

Great! The function has been created and tested with example.

What would you like to do next? I could:

- Explain how the function works.
- Show you how to use the function with a different temperature.
- Introduce you to other features of Colab, like working with data or machine learning.

What can I help you build?

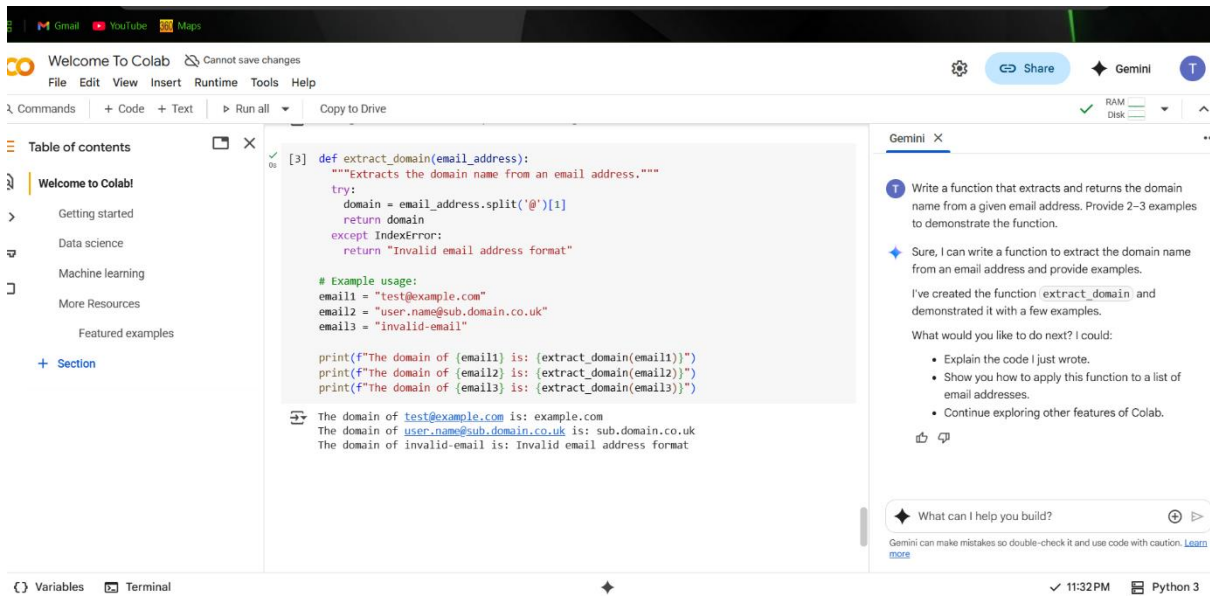
Gemini can make mistakes so double-check it and use code with caution.

TASK 3

PROMPT

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

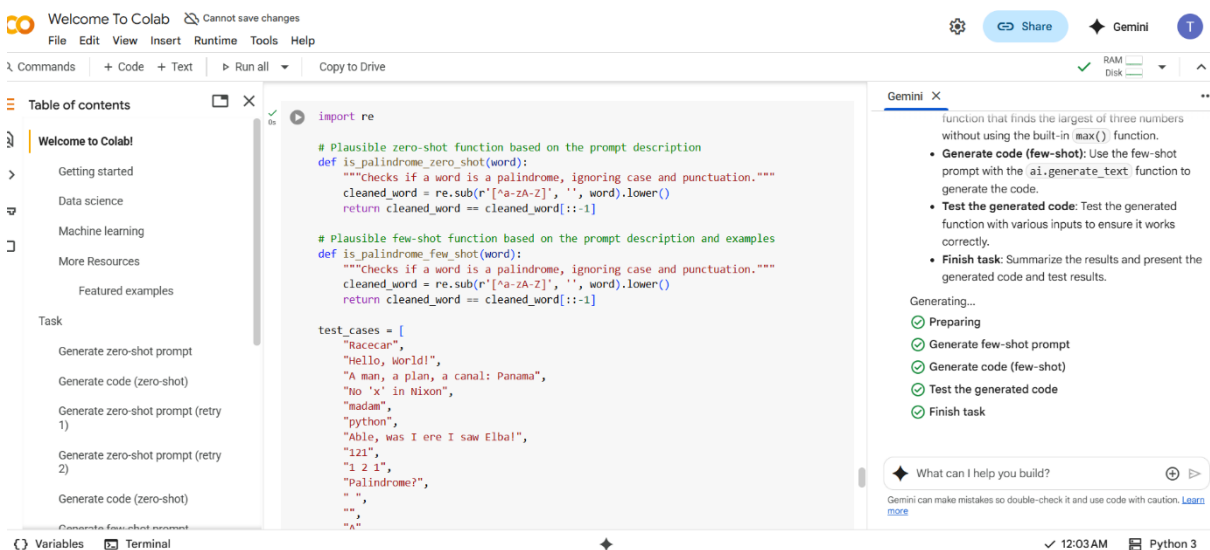
Provide **2–3 examples** to demonstrate the function.



TASK 4

PROMPT

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



Welcome To Colab Cannot save changes

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)

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](#)

```

def is_palindrome_zero_shot(test_case):
    return test_case == test_case[::-1]

def is_palindrome_few_shot(test_case):
    return test_case == test_case[::-1]

print("Comparing Zero-shot and Few-shot Palindrome Checkers:")
print("-" * 60)

for test_case in test_cases:
    zero_shot_output = is_palindrome_zero_shot(test_case)
    few_shot_output = is_palindrome_few_shot(test_case)

    print(f"Input: '{test_case}'")
    print(f"Zero-Shot Output: {zero_shot_output}")
    print(f"Few-Shot Output: {few_shot_output}")
    print("-" * 20)

Input: 'Racecar'
Zero-Shot Output: True
Few-Shot Output: True
-----
Input: 'Hello, World!'
Zero-Shot Output: False
Few-Shot Output: False
-----
Input: 'A man, a plan, a canal: Panama'
Zero-Shot Output: True
Few-Shot Output: True
-----

```

Welcome To Colab Cannot save changes

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)

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](#)

```

def is_palindrome_zero_shot(test_case):
    return test_case == test_case[::-1]

def is_palindrome_few_shot(test_case):
    return test_case == test_case[::-1]

print("Comparing Zero-shot and Few-shot Palindrome Checkers:")
print("-" * 60)

for test_case in test_cases:
    zero_shot_output = is_palindrome_zero_shot(test_case)
    few_shot_output = is_palindrome_few_shot(test_case)

    print(f"Input: '{test_case}'")
    print(f"Zero-Shot Output: {zero_shot_output}")
    print(f"Few-Shot Output: {few_shot_output}")
    print("-" * 20)

Input: 'Racecar'
Zero-Shot Output: True
Few-Shot Output: True
-----
Input: 'Hello, World!'
Zero-Shot Output: False
Few-Shot Output: False
-----
Input: 'A man, a plan, a canal: Panama'
Zero-Shot Output: True
Few-Shot Output: True
-----
Input: 'No 'x' in Nixon'
Zero-Shot Output: True
Few-Shot Output: True
-----
Input: 'madam'
Zero-Shot Output: True
Few-Shot Output: True
-----
Input: 'python'
Zero-Shot Output: False
Few-Shot Output: False
-----
Input: 'Able, was I ere I saw Elba!'
Zero-Shot Output: True
Few-Shot Output: True
-----
Input: '121'
Zero-Shot Output: True
Few-Shot Output: True
-----
Input: '1 2 1'
Zero-Shot Output: True

```

Welcome To Colab Cannot save changes

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)

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](#)

```

def is_palindrome_zero_shot(test_case):
    return test_case == test_case[::-1]

def is_palindrome_few_shot(test_case):
    return test_case == test_case[::-1]

print("Comparing Zero-shot and Few-shot Palindrome Checkers:")
print("-" * 60)

for test_case in test_cases:
    zero_shot_output = is_palindrome_zero_shot(test_case)
    few_shot_output = is_palindrome_few_shot(test_case)

    print(f"Input: '{test_case}'")
    print(f"Zero-Shot Output: {zero_shot_output}")
    print(f"Few-Shot Output: {few_shot_output}")
    print("-" * 20)

Input: 'Racecar'
Zero-Shot Output: True
Few-Shot Output: True
-----
Input: 'Hello, World!'
Zero-Shot Output: False
Few-Shot Output: False
-----
Input: 'A man, a plan, a canal: Panama'
Zero-Shot Output: True
Few-Shot Output: True
-----
Input: 'No 'x' in Nixon'
Zero-Shot Output: True
Few-Shot Output: True
-----
Input: 'madam'
Zero-Shot Output: True
Few-Shot Output: True
-----
Input: 'python'
Zero-Shot Output: False
Few-Shot Output: False
-----
Input: 'Able, was I ere I saw Elba!'
Zero-Shot Output: True
Few-Shot Output: True
-----
Input: '121'
Zero-Shot Output: True
Few-Shot Output: True
-----
Input: '1 2 1'
Zero-Shot Output: True

```

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 'Table of contents' sidebar with links like 'Welcome to Colab', 'Getting started', 'Data science', 'Machine learning', 'More Resources', and 'Task'. The main area is split into two panes. The left pane contains a code editor with the following text:

```
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)
```

Below the code, it says 'Write a Python function that takes three numbers as input and returns the largest one w'. Then 'Here are a few examples:' followed by three examples:

```
Input: 1, 5, 2
Output: 5

Input: 10, 3, 7
Output: 10

Input: 4, 9, 9
Output: 9
```

The right pane shows a Gemini chat window with the following text:

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

At the bottom of the chat window, there is a text input field with the placeholder 'What can I help you build?' and a 'Send' button.

The bottom status bar of the Colab interface shows 'Variables', 'Terminal', '11:53 PM', and 'Python 3'.