**LAB 4.2:**  **Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques**

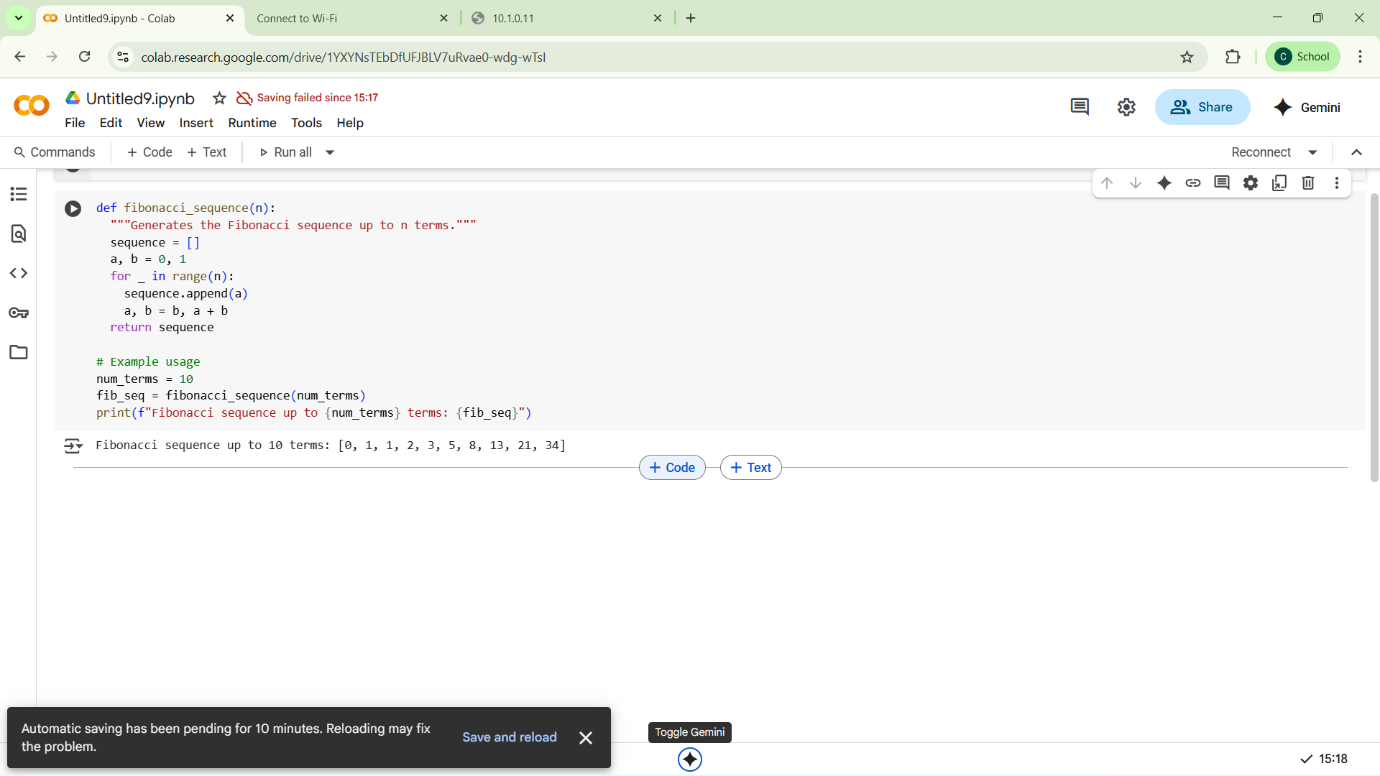
# Assignment 1

**Name :CHADA SATHWIKA**

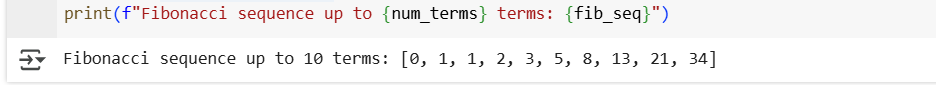
**HTNO:2403a51334**

**BTNO:24BTCAICSB14**

**Prompt:** Write a Python function to generate the Fibonacci sequence up to n terms

**CODE:** ****

Output:

****

**Code explanation:**

1. Function Definition

def fibonacci\_sequence(n):

* Defines a function named fibonacci\_sequence.
* Takes n as input → number of terms you want in the Fibonacci sequence.

2. Sequence Initialization

sequence = []

a, b = 0, 1

* sequence = [] → creates an empty list to store Fibonacci numbers.
* a = 0, b = 1 → first two Fibonacci numbers.

So at the start:

a = 0

b = 1

3. Loop for Generating Terms

for \_ in range(n):

sequence.append(a)

a, b = b, a + b

* for \_ in range(n): → runs the loop n times. (\_ is used since the loop variable isn’t needed.)
* sequence.append(a) → adds the current number a to the list.
* a, b = b, a + b → updates values simultaneously:
  + a becomes the old b
  + b becomes the sum of old a + b

This is the Fibonacci rule:

F(n) = F(n-1) + F(n-2)

4. Returning the Sequence

return sequence

* After generating all terms, the list is returned.

5. Example Usage

num\_terms = 10

fib\_seq = fibonacci\_sequence(num\_terms)

print(f"Fibonacci sequence up to {num\_terms} terms: {fib\_seq}")

* Sets num\_terms = 10.
* Calls the function → fib\_seq = fibonacci\_sequence(10)
* Prints the result.

🔢 Step-by-Step Output (for 10 terms)

* Start: a=0, b=1
* Iterations:
  + 1st → add 0, update → a=1, b=1
  + 2nd → add 1, update → a=1, b=2
  + 3rd → add 1, update → a=2, b=3
  + 4th → add 2, update → a=3, b=5
  + 5th → add 3, update → a=5, b=8
  + 6th → add 5, update → a=8, b=13
  + 7th → add 8, update → a=13, b=21
  + 8th → add 13, update → a=21, b=34
  + 9th → add 21, update → a=34, b=55
  + 10th → add 34, update → a=55, b=89

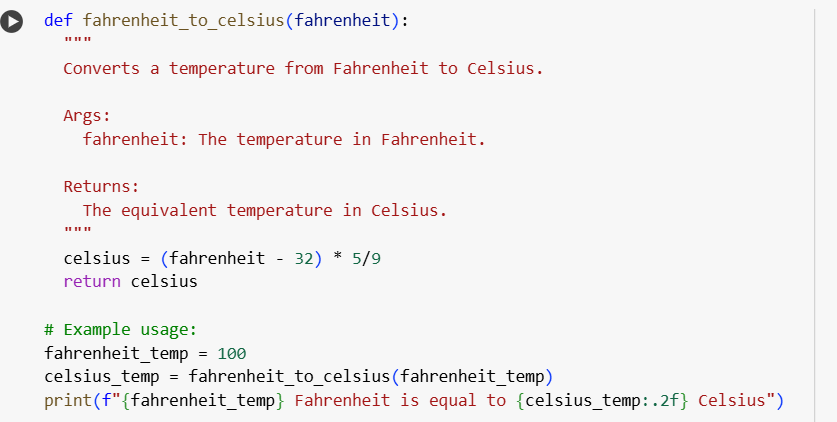
Final list:

[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

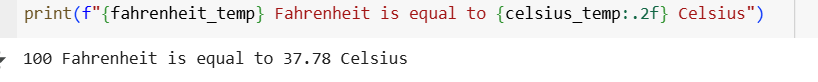
**TASK2:**

**PROMPT:** Write a Python function that takes a temperature in Fahrenheit as input and returns the equivalent Celsius value.

Input: 100  
Output: 37.78

**CODE:** ****

**Output**

****

**Code explanation:**

1. Function Definition
2. def fahrenheit\_to\_celsius(fahrenheit):
   * Here, a function named fahrenheit\_to\_celsius is created.
   * It takes one input parameter: fahrenheit (the temperature in Fahrenheit).
3. Docstring (Explanation inside quotes)
4. """
5. Converts a temperature from Fahrenheit to Celsius.
6. Args:
7. fahrenheit: The temperature in Fahrenheit.
8. Returns:
9. The equivalent temperature in Celsius.
10. """
    * This is just documentation for the function.
    * It explains what the function does, what input it expects, and what it returns.
    * It doesn’t affect the program’s execution.
11. Conversion Formula
12. celsius = (fahrenheit - 32) \* 5/9
    * This line converts Fahrenheit to Celsius.
    * Formula:

Celsius=(Fahrenheit−32)×59\text{Celsius} = (\text{Fahrenheit} - 32) \times \frac{5}{9}Celsius=(Fahrenheit−32)×95​

* + Example: If fahrenheit = 100 →

(100−32)×59=68×0.555...=37.78(100 - 32) \times \frac{5}{9} = 68 \times 0.555... = 37.78(100−32)×95​=68×0.555...=37.78

1. Return Statement
2. return celsius
   * The function returns the converted Celsius value.
   * It does not print inside the function — it just gives the value back to the caller.
3. Example Usage
4. fahrenheit\_temp = 100
5. celsius\_temp = fahrenheit\_to\_celsius(fahrenheit\_temp)
   * fahrenheit\_temp is assigned the value 100.
   * Then, the function is called: fahrenheit\_to\_celsius(100) → returns 37.78.
   * This result is stored in celsius\_temp
6. Printing Output
7. print(f"{fahrenheit\_temp} Fahrenheit is equal to {celsius\_temp:.2f} Celsius")
   * This uses f-string formatting.
   * {celsius\_temp:.2f} → formats the Celsius value to 2 decimal places.
   * Output:
   * 100 Fahrenheit is equal to 37.78 Celsius

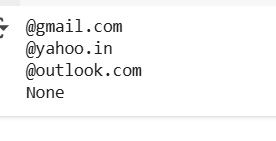
**TASK3:**

**Prompt:** Write a Python function that extracts and returns only the domain name Input: "john.doe@gmail.com" Output: "@gmail.com" Input: "alice123@yahoo.in" Output: "@yahoo.in" Input: "student@outlook.com" Output: "@outlook.com"

**Code:**

****

**Output:**

****

1. Code explanation: Function Definition
2. def extract\_domain(email):
   * A function named extract\_domain is created.
   * It takes one argument: email (a string, expected to be an email address).
3. Docstring (Explanation inside quotes)
4. """
5. Extracts the domain name from an email address.
6. ...
7. """
   * This is documentation, not code.
   * It explains what the function does, the input, and what it returns.
8. Checking for "@" in the email
9. if "@" in email:
   * Every valid email should contain "@".
   * This condition checks if the string has "@".
   * If yes → extract the domain.
   * If no → return None.
10. Splitting and Extracting the Domain
11. return "@" + email.split("@")[-1]
    * email.split("@") → splits the string into parts using "@".
      + Example: "john.doe@gmail.com".split("@") → ["john.doe", "gmail.com"].
    * [-1] → takes the last part (i.e., "gmail.com").
    * "@" + ... → adds back the @ symbol in front.
      + Result: "@gmail.com".
12. If Email is Invalid
13. else:
14. return None
    * If there’s no "@" in the string (like "invalid-email"), the function returns None.
    * This way, the function handles invalid inputs gracefully.
15. Example Usage and Output
16. print(extract\_domain("john.doe@gmail.com")) # @gmail.com
17. print(extract\_domain("alice123@yahoo.in")) # @yahoo.in
18. print(extract\_domain("student@outlook.com")) # @outlook.com
19. print(extract\_domain("invalid-email")) # None

✅ Output will be:

@gmail.com

@yahoo.in

@outlook.com

None

**TASK4:**

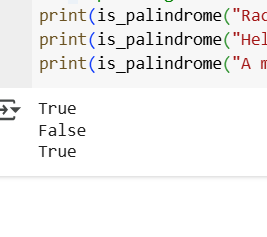
**Zero shot prompting:**

**Prompt:** write a python program that checks whether a string Is palindrome or not

**Code:**

****

**Output:**

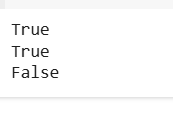
****

**Few shot**

**Prompt**: write a python program that checks whether a given string is palindrome or not input madam output true input:racecar output:true input :hello output false

****

**Output:**

****

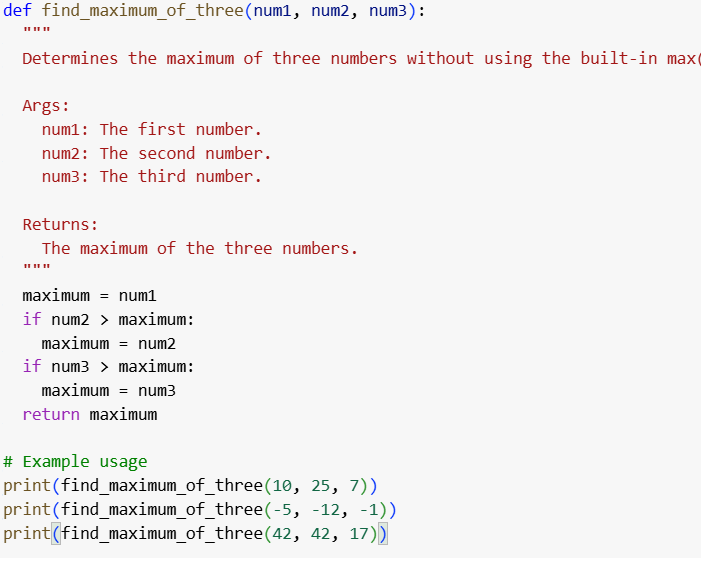
**Difference between zero and few shot**

| Aspect | Zero-shot Prompt | Few-shot Prompt |
| --- | --- | --- |
| Prompt Text | *"Write a Python function that checks if a word or sentence is a palindrome, ignoring punctuation and case."* | *"Write a Python function that checks if a word or sentence is a palindrome, ignoring punctuation and case.*  Examples: Input: 'Racecar' → Output: True Input: 'Madam, I’m Adam' → Output: True Input: 'Hello' → Output: False" |
| Clarity | May be slightly ambiguous since no examples are given. The model has to infer how to handle case and punctuation. | Very clear because the examples show exactly how case and punctuation should be treated. |
| Likelihood of Correct Output | Medium – the model might only check strict equality (e.g., "Racecar" vs "racecar") and ignore punctuation handling. | High – because the examples demonstrate lowercase conversion and punctuation removal. |
| Flexibility | More general, but risk of incorrect interpretation. | More specific, guides the model to follow a particular behavior. |
| Use Case | Useful when the task is simple and well-defined. | Useful when the task is complex or could be misinterpreted (like ignoring punctuation and case). |

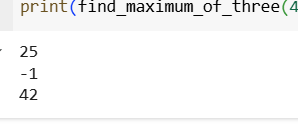
**Task 5:**

**Prompt:** Write a Python function that determines the maximum of three numbers without using the built-in max() function. Here are some examples: Input: (10, 25, 7) Output: 25 Input: (-5, -12, -1) Output: -1 Input: (42, 42, 17) Output: 42 Now, write the function that works correctly for any three numbers.

**Code:**

****

**Output:**

****

**Code explanation**

1. Function Definition
2. def find\_maximum\_of\_three(num1, num2, num3):
   * A function named find\_maximum\_of\_three is defined.
   * It takes three numbers as input: num1, num2, num3.
3. Docstring
4. """
5. Determines the maximum of three numbers without using the built-in max() function.
6. ...
7. """
   * This is documentation describing the purpose of the function.
   * It tells us the arguments and the return value.
8. Initial Maximum Assumption
9. maximum = num1
   * At first, assume the maximum value is num1.
   * Later, compare it with the other numbers to check if a bigger one exists.
10. Compare with num2
11. if num2 > maximum:
12. maximum = num2
    * If num2 is greater than the current maximum (num1 initially),  
      update maximum to num2.
13. Compare with num3
14. if num3 > maximum:
15. maximum = num3
    * Then check if num3 is greater than the current maximum.
    * If yes, update maximum to num3.
16. Return the Result
17. return maximum
    * After comparing all three, the largest value is stored in maximum.
    * The function returns this value.
18. Example Usage
19. print(find\_maximum\_of\_three(10, 25, 7)) # 25
20. print(find\_maximum\_of\_three(-5, -12, -1)) # -1
21. print(find\_maximum\_of\_three(42, 42, 17)) # 42

✅ Output:

25

-1

42