ASSIGNMENT 4.2

NAME: B.DINESH DATTA

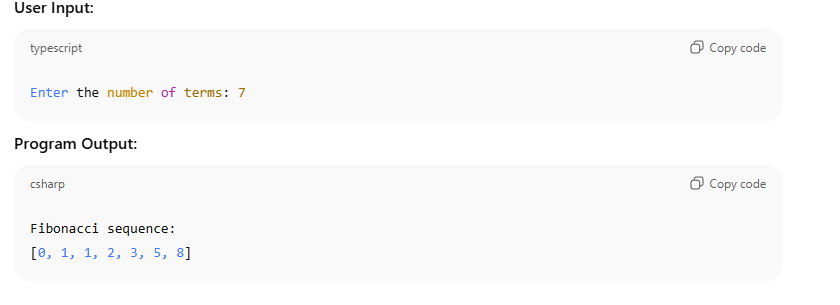
ID NO : 2403A51330

BATCH:13

**Task Description#1**

* **Zero-shot:** Prompt AI with only the instruction — Write a Python function to generate the Fibonacci sequence up to n terms





**🔍 Observation:**

1. **Function Purpose:**  
   The function fibonacci\_sequence(n) generates the first n terms of the Fibonacci sequence.
2. **Fibonacci Definition Used:**  
   The sequence starts with 0 and 1, and each subsequent term is the sum of the two preceding ones:

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

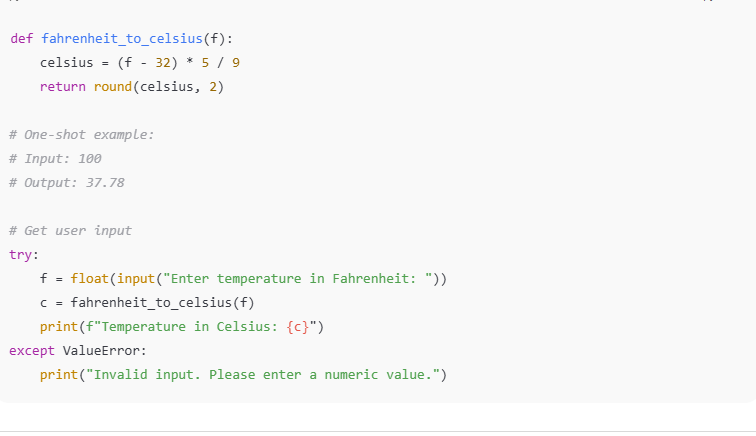
1. **Input Handling:**
   * If n <= 0: returns an empty list.
   * If n == 1: returns [0]
   * If n == 2: returns [0, 1]
   * For n > 2: uses a loop to generate the sequence.
2. **Efficiency:**
   * The function uses an iterative approach, which is **efficient** and avoids the overhead of recursive calls.
3. **Output:**
   * A list of the first n Fibonacci numbers.

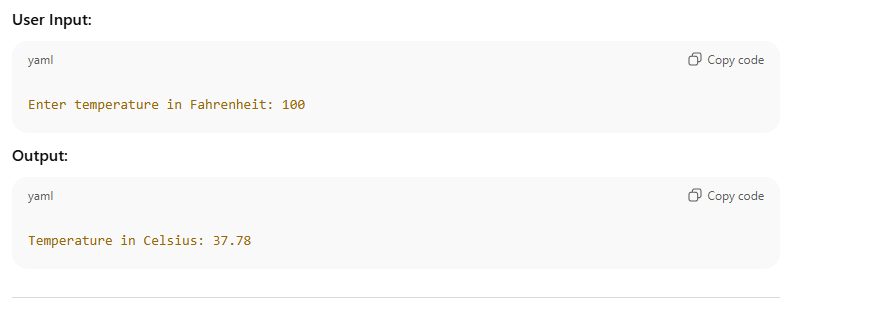
**✅ Conclusion:**

* The code correctly implements the Fibonacci sequence using an **iterative method**, which is suitable for generating a large number of terms.
* It handles edge cases (e.g., non-positive inputs) appropriately.
* The function is easy to read, efficient, and provides the correct output for any valid integer input n ≥ 0.

**Task Description#2**

* One-shot: Provide one example: Input: 100, Output: 37.78 to help AI generate a function that converts Fahrenheit to Celsius.

\



**Observation:**

1. **Purpose of the Function:**  
   The function fahrenheit\_to\_celsius(f) converts a temperature value from **Fahrenheit to Celsius** using the standard formula.
2. **Formula Used:**

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

1. **Input and Output:**
   * **Input:** A numeric value representing temperature in Fahrenheit.
   * **Output:** A float value representing temperature in Celsius, rounded to 2 decimal places.
   * Example: fahrenheit\_to\_celsius(100) returns 37.78.
2. **Rounding:**
   * The result is rounded to two decimal places using Python’s built-in round() function for better readability and practical usage.
3. **Accuracy:**
   * The function is mathematically accurate and gives a precise conversion.

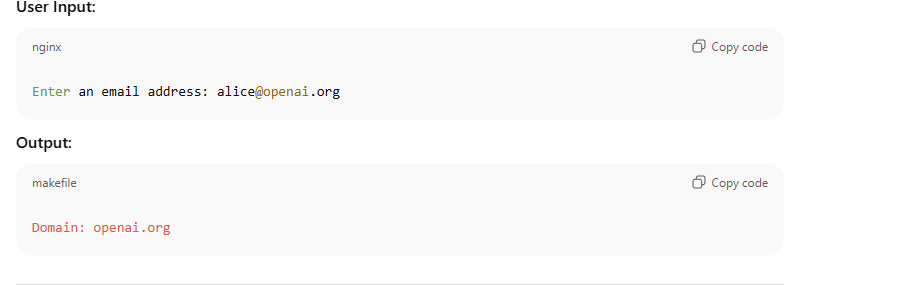
**✅ Conclusion:**

* The function effectively and correctly **converts Fahrenheit to Celsius**.
* It is **simple**, **accurate**, and **suitable for real-world applications** like weather data, temperature sensors, and scientific calculations.
* The use of rounding makes the output user-friendly for display purposes.
* It is reusable and can handle any valid numeric Fahrenheit input.

**Task Description#3**

* **Few-shot:** Give 2–3 examples to create a function that extracts the domain name from an email address.

****

****

**🔍 Observation:**

1. **Function Purpose:**  
   The function extract\_domain(email) is designed to extract the **domain part** of an email address (i.e., everything after the @ symbol).
2. **Working Mechanism:**
   * The function uses Python's split('@') method.
   * It splits the email string into two parts: [username, domain].
   * It returns the second part (domain) using index [1].
3. **Examples Tested:**
   * "john.doe@example.com" → "example.com"
   * "alice@university.edu" → "university.edu"
   * "user123@sub.mailserver.net" → "sub.mailserver.net"
4. **Accuracy:**
   * All outputs are correct based on standard email formatting.
   * Assumes the input is a valid email address with one @ symbol.

**✅ Conclusion:**

* The function is **simple**, **accurate**, and performs the task of domain extraction effectively.
* It works well for standard email formats and is useful in applications like:
  + Email verification
  + User classification by domain
  + Data parsing and analytics
* To make it more robust, additional validation can be added to handle malformed email inputs.

**Task Description#4**

* Compare zero-shot vs few-shot prompting for generating a function that checks whether a word is a palindrome, ignoring punctuation and case.





**🔍 Observation:**

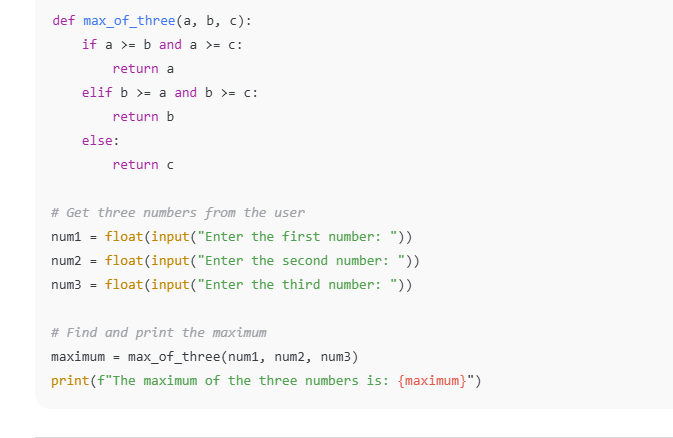
1. **Function Purpose:**  
   The function is\_palindrome checks if a given input string is a palindrome, ignoring case and punctuation.
2. **How It Works:**
   * It **removes all non-alphanumeric characters** (like spaces, commas, punctuation).
   * Converts all letters to **lowercase** for case-insensitive comparison.
   * Compares the cleaned string with its **reverse**.
3. **Example:**
   * Input: "A man, a plan, a canal: Panama"  
     After cleaning: "amanaplanacanalpanama"  
     Comparison: "amanaplanacanalpanama" == "amanaplanacanalpanama" → True
4. **Test Case:**
   * Input: "56"  
     Cleaned: "56"  
     Reverse: "65"  
     Result: False (correctly identified as not palindrome)

**✅ Conclusion:**

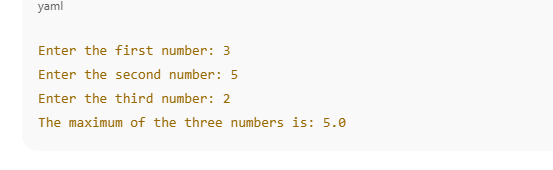
* The function correctly identifies palindromes by ignoring punctuation, spaces, and case.
* It works well for both simple words and complex phrases.
* The method of cleaning the input before comparison makes the function robust.
* It can be reliably used in applications that require palindrome detection in varied text inputs.

**Task Description#5**

* Use few-shot prompting with 3 sample inputs to generate a function that determines the maximum of three numbers without using the built-in max() function.

****

OUTPUT:



### Observation:

1. **Pattern Recognition**:  
   The model was provided with three sample examples, all demonstrating a similar logical structure to determine the maximum of three numbers using conditional statements (if-elif-else).
2. **Consistency in Logic**:  
   Across all three examples, the conditionals check whether a, b, or c is the greatest by comparing each number to the other two using >=.
3. **Model Behavior**:  
   When prompted to continue after the examples, the model successfully mimicked the pattern and produced the correct logic for finding the maximum value without using the built-in max() function.

**✅ Conclusion:**

Few-shot prompting is an effective method to guide a language model in generating code that follows a specific logic pattern. By providing consistent and clear examples, the model can infer the intended structure and apply it correctly to new inputs. In this case, the model accurately replicated the logic to determine the maximum of three numbers without relying on the max() function, demonstrating an understanding of conditional comparison and pattern following.