#### **Scenario:** A program needs to find the second largest number in a given list of numbers. Write logic to find the second largest number in a given list.

**Ramishahope Artificial Intelligence Pvt Ltd**

**36, Old Anandas, SG Arcade, Marudhamalai Main Road, Vadavalli, Coimbatore -641041.**

**+91 6385383227 |** [**www.hopelearning.net**](http://www.hopelearning.net/) **|** [**mdaravind@hopelearning.net**](mailto:mdaravind@hopelearning.net) **| 33AAMCR3722R1ZU**

* **Def second largest(my\_list):**
* **Sorted\_list=sorted(my\_list) # To order the given list in accending order**
* **Second\_Largest=sorted\_list[-2]**
* **Print (Second\_Largest)**

#### **Scenario:** A function needs to convert an integer to its binary representation without using Python’s built-in bin() function. Write logic to convert a given integer to its binary representation.

* **Def find\_binary()**
* **Now create empty string to store temporary number**
* **While num>0:**
* **Reminder = num%2, binary=str(Reminder)+binary, num=num//2**
* **Repeat the above loop until num=0**
* **Then, reverse the binary (binary=str(Reminder)+binary no need to reverse)**
* **Return binary**
* **Num=int (input(“Enter the number”))**
* **Print(find\_binary(num)**

#### **Scenario:** A function needs to merge two sorted lists into a single sorted list efficiently. Write logic to merge two sorted lists into one sorted list.

* **def merge\_lists(a,b):**
* **c = sorted(a + b)**
* **return c**
* **a = [1, 3, 5,7]**
* **b = [2, 4, 6,8]**
* **print(merge\_lists(a,b))**

#### **Scenario:** A function needs to find the first non-repeating character in a string for text processing. Write logic to find the first non-repeating character in a given string.

* **def first\_non\_repeating\_char(s):**
* **char\_count = {}**
* **for char in s:**
* **char\_count[char] = char\_count.get(char, 0) + 1**
* **for char in s:**
* **if char\_count[char] == 1:**
* **return char**
* **return None**
* **s='satish'**
* **print (first\_non\_repeating\_char(s))**

#### **Scenario:** A program needs to identify common elements between two lists for data filtering. Write logic to find the common elements between two lists.

* **def find\_common\_elements(list1, list2):**
* **return list(set(list1) & set(list2))**
* **list1 = ['satish','hope','eingineer']**
* **list2 = ['hope','ai','engineer','satish'] #it works for numbers too**
* **common\_elements = find\_common\_elements(list1, list2)**
* **print("Common elements:", common\_elements)**

#### **Scenario:** A function is required to reverse a given number. Write logic to reverse a given number.

* **Def reverse(n):**
* **Reverse\_number=reverse(n)**
* **Print (Revese\_number)**
* **n=int(input(“enter the number”))**
* **reverse(n)**

#### **Scenario:** A program needs to count the number of words in a given sentence. Write logic to count the number of words in a given sentence.

* **def count\_words(sentence):**
* **words=sentence.split()**
* **return len(words)**
* **word= ('satish student of hope ai')**
* **print ("Number of words:",count\_words(word))**

#### **Scenario:** A function needs to compute the factorial of a number using iteration instead of recursion. Write logic to find the factorial of a given number using iteration.

* **def factorial\_iterative(n):**
* **if n < 0:**
* **return "Fallse" #Factorial not defined for negative numbers**
* **result = 1**
* **for i in range(2, n + 1):**
* **result = result\*i**
* **return result**
* **num = 5**
* **print("Factorial of", num , "is", factorial\_iterative(num))**

#### **Scenario:** A program is required to convert all strings in a list to uppercase. Write logic to convert all strings in a list to uppercase.

* **def convert\_to\_uppercase(string\_list):**
* **return [s.upper() for s in string\_list]**
* **words = ['hope', 'satish', 'python', 'ai']**
* **print("Uppercase list:", convert\_to\_uppercase(words))**

#### **Scenario:** A function is needed to compute the greatest common divisor (GCD) of two numbers using the Euclidean algorithm. Write logic to calculate the GCD of two numbers using the Euclidean algorithm.

* **import math**
* **def find\_GCD(a,b):**
* **return (math.gcd(a,b))**
* **print("GCD of given nuber is:", find\_GCD(48,18))**