**Q1. Question 1: Given the sets of events, sports\_winners = set(["Jake", "John",Eric","Geetha","Femi","Kala"]) quiz\_winners = set(["John", "Jill","Eric","Geetha","Femi"])**

1. print the winners who got sports prize and quiz prize

The code snippet uses the & operator between the sports\_winners set and the quiz\_winners set. This operator is an intersection operation, which returns a new set containing elements that are common to both sets. In this case, it identifies the winners who received both sports and quiz prizes. When you print the result of this operation, it displays the names of individuals who excelled in both sports and quiz competitions.

1. print the winners who got sports prize and not quiz prize

The code snippet uses the - operator between the sports\_winners set and the quiz\_winners set. This operator performs a set difference operation, which returns a new set containing elements that are in the first set (sports\_winners) but not in the second set (quiz\_winners). In this context, it identifies the winners who received sports prizes but did not win any quiz prizes.

1. print the winners who either sports prize or quiz prize

The code snippet utilizes the union method to combine the sports\_winners set and the quiz\_winners set. The union operation creates a new set containing all unique elements from both sets. In this context, it provides a comprehensive list of individuals who won either a sports prize or a quiz prize or both.

1. print the winners who got quize prize and not sports prize

The code snippet employs the - operator to perform a set difference operation. It takes the quiz\_winners set and subtracts the elements that are also present in the sports\_winners set. In this specific context, the operation identifies and isolates the winners who exclusively achieved success in the quiz competition but did not receive any sports prizes

**Q2. Question 2: Consider the set with some integers given below myset = {12,34,56,3,45,67,89,1,6}**

1. Create set by filtering even numbers and
2. Create set by filtering odd numbers

The given code snippet works with two sets, myset1 and myset2, which are initially empty. It iterates through the elements in the myset set, which contains a collection of integers.

For each element val in myset, it checks if val is even (divisible by 2). If val is even, it adds it to the myset1 set using the add method.

If val is not even (i.e., it's odd), it adds it to the myset2 set using the add method

This code snippet serves to categorize the elements from the myset set into two different sets, myset1 and myset2, based on whether they are even or odd.

myset1 will contain all the even numbers from myset.

myset2 will contain all the odd numbers from myset

1. Create set by filtering numbers>20

A new set called myset3 is introduced, initially empty. The code iterates through the elements in the myset set, which contains a collection of integers.

For each element val in myset, it checks if val is greater than 20.

If val is indeed greater than 20, it adds it to the myset3 set using the add method.

Finally, the code prints the contents of the myset3 set.

The resulting myset3 set will contain all the integers from myset that are greater than 20.

1. Remove the item 56 and create a new set

This part of the code has two distinct operations:

The myset.remove(56) operation removes the element 56 from the myset set if it is present. If 56 was originally in the set, it is now removed, altering the contents of myset.

The assignment myset4 = myset creates a new variable myset4 that references the same set in memory as myset. This means that any changes made to myset will also be reflected in myset4, as they both point to the same set object.

**Question 3: Date and Time**

1. Print the Date after 1 week and 6 days from the current date.

It obtains the current date using the datetime.date.today() method.

It calculates a new date, new\_date, by adding 13 days to the current date using the timedelta class.

The result is then printed to the console, displaying the date that is 13 days ahead of the current date.

1. Given the birthday as date(2000,10,2), find the age

It calculates the age of a person with a birthdate of October 2, 2000, based on the current date.

It computes the age in years with fractional years, taking into account leap years.

The calculated age is then displayed as a floating-point number, indicating the number of years elapsed since the provided birthdate.

1. Subtract five days from the current date and display it.

It calculates a date that is 5 days before the current date (current\_date).

The calculated date is stored in the variable Day5minus.

The code then prints the Day5minus date, which represents a date that is 5 days in the past relative to the current date.

1. Display the dates of yesterday, today and tomorrow

It calculates and displays the date that is one day before the current date, referred to as "yesterday."

It calculates and displays the date that is one day after the current date, referred to as "tomorrow."

It also displays the current system date, which is stored in the variable current\_date.

**Question 4: Consider the folowing strings, string1="Great", string2="Learning" and string3="Python". Perform the following operations on the given strings.**

* 1. Concatenate them to create a single string called "single\_string"

This Python code snippet combines three individual strings, "Great," "Learning," and "Python," into a single string named single\_string. The concatenation is achieved using the + operator, and the resulting string contains the text "GreatLearningPython."

* 1. Access and print the first and last characters of "single\_string"

This Python code snippet demonstrates the use of string indexing to access specific characters within the single\_string variable, which is the concatenated string "GreatLearningPython."

single\_string[0] retrieves the first character, which is "G."

single\_string[-1] retrieves the last character, which is "n."

* 1. Reverse the "single\_string" and save it in "rev\_string"

This Python code snippet demonstrates the reversal of a string's characters. It takes the original single\_string variable, which contains the concatenated string "GreatLearningPython," and creates a new string rev\_string with the characters reversed.

For example, if single\_string was "GreatLearningPython," then rev\_string will be "nohtyPgninraeLtaerG," which is the original string reversed.

* 1. Extract the word "Learning" from "single\_string"

This Python code snippet successfully extracts the word 'Learning' from the single\_string variable, which is the result of concatenating 'Great', 'Learning', and 'Python'. The slice [5:13] precisely captures the desired substring, starting at index 5 and ending at index 12 (the character at index 13 is not included).

**Question 5.**

1. print(sentense.index('Learning')): Prints the index of the first occurrence of "Learning" in the sentence.

2. print(sentense.index('e')): Prints the index of the first occurrence of the letter "e" in the sentence.

3. print(sentense.index('e', 5, 10)): Attempts to find "e" between positions 5 and 10 in the sentence, but it may raise an error if not found.

4. print(sentense.split()): Splits the sentence into words using spaces as separators and prints the list of words.

**Question 6.**

1. It defines a function Analyzetext that takes a text string (txt) as input.

2. Inside the function, it initializes an empty dictionary called dict\_text\_document.

3. It then splits the input text into words and iterates through them.

4. For each word, it updates its count in the dictionary using setdefault.

5. After counting all words, it sorts the dictionary in reverse order by count.

6. It returns the sorted dictionary.

7. The program defines a sample text document and calls the Analyzetext function with it.

8. It initializes a counter variable (count) to 0.

9. It iterates through the sorted dictionary items and prints each word and its count.

10. It breaks the loop after printing the top 5 most common words.

**Question 7.**

1. It defines a function find\_best\_employer that takes a list of tuples (employee\_lst) as input.

2. Inside the function, it sorts the list of tuples in reverse order based on the total bonus points using a lambda function.

3. It checks if there are at least 3 employees in the list.

4. If there are 3 or more employees, it prints the names of the top three employees using string formatting.

5. The program defines a list of employee tuples and calls the find\_best\_employer function with it