Q1. Discuss string and provide examples.

Ans . It follows the format string[start:end], where start is the index where slicing begins (inclusive) and end is the index where it ends (exclusive). For example, with the string my_string = "Hello, World!", if you use my_string[7:12], it will return "World". If you omit start, it starts from the beginning of the string.

Q2. Explain the key features of lists in python.

Ans. Lists are mutable and dynamic; list items can be added, removed or changed after the list is defined. Lists are ordered; newly added items will be placed at the end of the list. Lists use zero-based indexing; every list item has an associated index, and The first item's index is 0.

Q3. Describe how to access,modify and delete elements in a list with examples.

Ans. Python list elements are ordered by index, a number referring to their placement in the list. List indices start at 0 and increment by one. To access a list element by index, square bracket notation is used: list[index] .

Modify an Item Within a List in Python

1. Step 1: Create a List. To start, create a list in Python. For example: list_names = ["Jon", "Bill", "Maria", "Jenny", "Jack"] ...
2. Step 2: Modify an Item within the list. For example, to modify the third item (index of 2) in the list from “Maria” to “Mona“: list_names[2] = "Mona"

You can either use the del keyword or the pop() function to remove an element from a list in Python by its index

Q4. Compare and contrast tuples and lists with examples.

Ans. Use a list if you need a mutable collection of items where you may need to add, remove, or change elements. Lists are more flexible and have more built-in methods, making them ideal for dynamic collections. Use a tuple if you need an immutable collection where the elements won't change after creation.

Q5.Describe the key features of sets and provide examples of their uses.

Ans.A set is represented by a capital letter symbol and the number of elements in the finite set is represented as the cardinal number of a set in a curly bracket {…}. For example, set A is a collection of all the natural numbers, such as A = {1,2,3,4,5,6,7,8,…..∞}. Also, check sets here.

Q6.Discuss the use cases of tuples and sets in python programming.

Ans. Tuples are used to store multiple items in a single variable. Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Set, and Dictionary, all with different qualities and usage. A tuple is a collection which is ordered and unchangeable.

A common use of sets in Python is computing standard math operations such as union, intersection, difference, and symmetric difference. The image below shows a couple standard math operations on two sets A and B. The red part of each Venn diagram is the resulting set of a given set operation.

Q7. Describe how to add, modify, and delete items in a dictionary with examples.

Ans. How to Add an Item to a Dictionary in Python

1. Method 1: Using The Assignment Operator. ...
2. Method 2: Using update() ...
3. Method 3: Using dict() Constructor. ...
4. Method 4: Using __setitem__ ...
5. Method 5: Using The ** Operator. ...
6. Method 6: Checking If A Key Exists. ...
7. Method 7: Using A For Loop. ...
8. Method 8: Using zip.

How to modify a dictionary Python

- To update a dictionary in Python, you can use the update() method, like dict1. update({'b': 3, 'c': 4}) . This method allows you to add new items or change the value of existing items in a Python dictionary. In this example, we have a dictionary dict1 with keys 'a' and 'b'.

How to remove a key from a Python dictionary.

To remove a key from a dictionary, you can use the del statement. Here's an example: my_dict = {'a': 1, 'b': 2, 'c': 3} del my_dict['b'] print(my_dict) # Output: {'a': 1, 'c': 3} Alternatively, yo…

Q8.Discuss the importance of dictionary keys being immutable and provide examples.

Ans. In a dictionary, the key part must be immutable. This means that once a key is created, it cannot be changed. Immutability is essential for keys because it ensures that the dictionary can efficiently look up values based on their keys.If the key were a mutable object, its value could change, and thus its hash could also change. But since whoever changes the key object can't tell that it was being used as a dictionary key, it can't move the entry around in the dictionary.