Lists in Python

The list is the most versatile datatype available in Python. We can create a list in Python by enclosing a comma-separated sequence of elements within square brackets ([]).

Eg- my_list = [1, 2, 3, 4, 5]

Accessing List Elements:

You can access individual elements in a list using their index. In Python, indexing starts from 0.

EG-first_element = my_list[0] # This will be 1

#List Manipluation Techniques

1. Appending Elements

We can add elements to the end of a list using the append() method.

Eg-my_list.append(6) # Adds 6 to the end of the list.

2. Inserting Elements

We can insert elements at a specific index using the insert() method.

Eg-my_list.insert(2, 7) # Inserts 7 at index 2, shifting other elements

3. Removing Elements

To remove elements from a list, you can use methods like remove(), pop(), or del.

Eg- my_list.remove(3) # Removes the first occurrence of 3

popped_element = my_list.pop(1) # Removes and returns the element at index 1

del my_list[0] # Deletes the element at index 0

4. Slicing Lists

We can extract a portion of a list using slicing. Slicing creates a new list containing the selected elements.

Eg- subset = my_list[1:4] # Gets elements from index 1 (inclusive) to 4 (exclusive)

5. List Concatenation

We can concatenate two or more lists using the + operator.

Eg- list1 = [1, 2, 3]

list2 = [4, 5, 6]

concatenated_list = list1 + list2 # Results in [1, 2, 3, 4, 5, 6]

6. List Length

We can find the length of a list using the len() function.

Eg-length = len(my_list) # Returns the number of elements in the list.

7. Sorting Lists

We can sort a list using the sort() method.

Eg- my_list.sort() # Sorts the list in ascending order

#To sort in descending order- my_list.sort(reverse=True) # Sorts the list in descending order.

8. List Comprehensions

List comprehensions provide a concise way to create lists based on existing lists. They are a powerful tool for manipulation.

Eg- squared_values = [x ** 2 for x in my_list] # Creates a new list with squared values.

9. List Iteration

We can iterate over the elements of a list using a for loop. This allows you to perform operations on each element.

```
Eg-my_list = [1, 2, 3, 4, 5]
  for item in my_list:
     print(item)
```

10. List Membership

We can check if an element is present in a list using the in keyword.

```
Eg- if 3 in my_list:
      print("3 is in the list")
```

11. Counting Occurrences

To count the number of times an element appears in a list, we can use the count() method.

Eg- count_of_3 = my_list.count(3) # Returns 1 (3 appears once)

12. Finding Index

To find the index of the first occurrence of an element, we can use the index() method.

Eg- my_list.reverse() # Reverses the list in-place

13. List Reversal

We can reverse the order of elements in a list using the reverse() method.

Eg- my_list.reverse() # Reverses the list in-place

14. List Copy

Creating a copy of a list can be done using slicing or the copy() method to avoid modifying the original list.

Eg- # Using slicing

    copied_list = my_list[:] # Creates a new list with the same elements

 # Using copy() method

    copied_list = my_list.copy() # Creates a new list with the same elements

15. List Clearing

We can remove all elements from a list using the clear() method.

Eg- my_list.clear() # Removes all elements from the list, making it empty

16. List Filtering

We can create a new list by filtering elements from an existing list based on a condition using list comprehensions.

```
Eg- even_numbers = [x for x in my_list if x % 2 == 0]  # Creates a list of
even     numbers
```

17. List Concatenation with Extend

The extend() method allows us to add elements from another iterable (e.g., another list) to an existing list.

```
Eg- list1 = [1, 2, 3]
    list2 = [4, 5, 6]
    list1.extend(list2)  # Extends list1 with elements from list2
```

18. List Min and Max

We can find the minimum and maximum values in a list using the min() and max() functions.

```
Eg-  minimum_value = min(my_list)
     maximum_value = max(my_list)
```