Lists:

The list is Dynamic in size, meaning they can grow or shrink as needed. You can add or remove elements without having to specify the size beforehand. contain elements of different data types within the same list.

Python lists are

* Mutable,
* can hold duplicate values,
* Ordered and supports indexing.

|  |  |  |
| --- | --- | --- |
|  |  | List |
|  |  |  |
| Creation | Simple Method | abc = ["apple", "banana", "cherry", "orange"] |
|
| by USING Constructor | abc = list(("apple", "banana", "cherry", "orange")) |
|
|  |  |  |
| Access / Slicing Items | | print(abc[1])  ans= [apple] |
|
|
| print(abc[-1])  ans= [mango] |
|
|
| print(abc[2:5]) ans=[cherry, orange, kiwi] |
|
|
| print(abc[:4]) ans=[apple, banana, cherry, orange] |
|
|
| print(abc[2:]) [cherry, orange, kiwi, melon, mango] |
|
|
| print(abc[-4:-1]) [orange, kiwi, melon] |
|
|
|  |  |  |
| Loop Through | | #To print all value:- |
| for x in abc:  print(x) |
|
|
| apple banana cherry orange kiwi melon mango |
|
|
|
|
|
|
|
|  |  |  |
| Length | | len(abc) |
|
|  |  |  |
| Check if Item Exists | | if "apple" in abc:  print("Yes, apple is in the fruits list") |
|
|
|  |  |  |
| Changing the item | | abc[1] = "blackcurrant" print(abc) |
|
|
|  |  |  |
| Adding an Item | To add an item at the Last index | abc.append("orange") |
|
|  |  |
| To add an item at the specified index | abc.insert(1, "orange") |
|
|  |  |  |
| Remove | To remove the specified item: | abc.remove("banana") |
|
| abc.pop("banana") |
|
| To removes the Last item: | abc.pop() |
| To removes by Index | del abc[0] |
|  |  |  |
| Clear | To empty the list: | abc.clear() |
| Delete | To delete the Data Completely | del abc |
|  |  |  |
| Copy | Method 01 | mylist = abc.**copy()** print(mylist) |
|
|
| ["apple", "banana", "cherry", "orange"] |
|
| Method 02 | mylist = abc.list() print(mylist) |
|
|
| ["apple", "banana", "cherry", "orange"] |
|
|  |  |  |
| Join |  | List Example list1 = ["a", "b" , "c"] list2 = [1, 2, 3] |
|
|
|
| Method 01 by + operator | list3 = list1 + list2 print(list3) |
|
|
| Method 02 by using append | for x in list2:  list1.**append**(x)  print(list1) |
|
|
|
|
| Method 03 by using extend | list1.extend(list2) print(list1) |
|
|
|  |  |  |
| Number of elements in dataset | | list1.count(2) |
|
| 1 |
|
| Index of the first element with the specified value | | index() |
|
| Reverses the order | | reverse() |
|
| Sorts the list | | sort() |
|
| Sorts the list | | sorted() |
|

List comprehension is a **concise way to create lists in Python**. It allows you to generate lists based on existing iterables like **lists, strings, or other iterable objects**. The syntax for list comprehension is compact and readable, making it a preferred choice over traditional loops in many cases.

**[exp for x in iterable if condition ]**

**[x\*\*2 for x in range(10) if x % 2==0 ]**

**[exp for x in iterable if cond1 if cond2 ... ]**

**[x for x in range(10) if num % 2 == 0 if num > 5 ]**

**[exp1 if cond1 else exp2 if cond2 else exp3 for x in iterable]**

**[x\*\*2 if x<5 else x//2 if x>5 else x for x in iterable]**