As Per CBSE Syllabus

Computer Science

Class XI

List in Python

Introduction to Lists

In Python, a **List** is a data structure that stores an ordered collection of items. Each item in a list is called an element, and lists are versatile and commonly used in programming.

Python provides built-in support for lists, making it easy to work with data in a structured manner. Lists are defined by enclosing elements in square brackets and separating them with commas.

List Indexing

Backward Indexing:	-8	-7	-6	-5	-4	-3	-2	-1
	10	**20**	**30**	**50**	**12**	**82**	**36**	**20**
Forward Indexing:	0	1	2	3	4	5	6	7

List Declaration

How to Declare a List:

In Python,. you can declare a list by enclosing a sequence of elements in square brackets.

For example:

```
my_list = [1, 2, 3, 4, 5]                              #creates a list named my_list containing five elements

my_list1 = [“kvs”,  2,  “cs”,  41,  50, "python”]    #creates a list named my_list1 containing six elements

my_list2 =[ ]                                          #creates an empty list named my_list2
```

List Declaration and Indexing

Accessing Elements by Index: - Lists uses both forward indexing and backward indexing, meaning in case of forward indexing, the first element is at index 0, the second at index 1, and so on. You can access elements of the list by specifying their index

```
my_list[0]=[1, 2, 3,”kvs”, 5, ”cs”]

my_list[0]   # would return the first element of the list -  1

my_list[1]   # would return the second element of the list  - 2

my_list[3]   # would return the fourth element of the list  - “kvs”

my_list[-1]   # would return the last element of the list (backword index)  - “cs”
```

List Operations:

List Concatenation:

List concatenation is the process of combining two or more lists to create a new list. You can use the `+` operator to concatenate lists,

```
list1=[1,3,5]
list2=[8,0]
list3=list1 + list2
print(list3).        #would print [1,3,5,8,0]
```

List Repetition:

List repetition involves repeating the elements of a list multiple times. You can use the `*` operator to repeat a list,

```
list1=[1,3,5]
list2=list1 *  3
print(list3).        #would print [1,3,5,1,3,5,1,3,5]
```

List Operations:

Membership Testing (in and not in):

You can check if an element is present in a list using the `in` and `not in` operators. For example:

```
list1=[1,3,5]
Print(5 in list1).       # would print True
Print(7 in list1).       # would print False
```

Slicing Lists:

Slicing allows you to extract a portion of a list by specifying a start and end index. For example:

```
list1=[1,3,5,6,9,0,12,3]
print(list1[1:5]).      # would print [3,5,6,9]
print(list[ :7])        #would print [1,3,5,6,9,0,12]
print(list[ 4: ])       #would print [9,0,12,3]
```

Traversing a List Using Loops

Using `for` loop to iterate through a list: You can use a `for` loop to traverse (iterate through) all elements in a list. This is a common way to process each item in a list one by one.

Example: Printing each element of a list:

```
list1=[1,3,5]

for item in list1:

        print(item)

# This code will print each element of `my_list` on a separate line.
1
3
5
```

Built-in List Functions

Sl No	Function	Explanation
1	append()	adds an element to the end of a list.
2	extend()	appends the elements of another list to the end of the current list.
3	insert()	is used to insert an element at a specified position in the list.
4	count()	returns the number of occurrences of a specific element in the list.
5	index()	finds the index of the first occurrence of a specified element in the list.
6	remove()	removes the first occurrence of a specified element from the list.
7	pop()	removes and returns an element by its index.
8	clear()	Removes all elements of the list
9	sorted()	returns a new list with elements sorted in ascending order, leaving the original list unchanged.
10	reverse()	reverses the order of elements in the list in-place.
11	sort()	is used to sort a list in ascending order (in-place).
12	min()	finds the minimum value in a list.
13	max()	finds the maximum value in a list.
14	sum()	calculates the sum of all elements in a list.

Nested Lists

Definition and Usage of Nested Lists: - Nested lists are lists that contain other lists as their elements. They are used to represent multi-dimensional data or hierarchical structures.

Example of a Nested List: - An example of a nested list could be a list of lists, such as

```
list1 = [[1, 2, 3],  [4, 5, 6], 10, 20]

print(list1[0])        #would print the first element of the list – [1,2,3]

print(list2[1])        #would print the second element of the list – [4,5,6]

print(list1[1][0])     #would print 4
```

Practice Programs Example :

1. **Removing Duplicates from a List:**

Write a program that takes a list as input and removes duplicate elements, returning a new list with unique values.

2. **List Comprehension for Squares:**

Create a program that generates a new list containing the squares of each element from an input list using list comprehension.

3. **List Concatenation and Sorting:**

Write a program that concatenates two lists, sorts the resulting list in descending order, and displays the sorted list.

4. **List Intersection:**

Implement a program that takes two lists as input and returns a new list containing elements that are common to both input lists (intersection).

5. **List Shuffling:**

Create a program that shuffles the elements of a list randomly. Ensure that no element appears in its original position after shuffling.

Practice Programs Example :

6. Palindrome Check:

Write a program that checks if a given list of words is a palindrome, i.e., it reads the same forwards and backward.

7. List Rotation:

Implement a program that rotates the elements of a list to the right by a specified number of positions. The rotated list should wrap around.

8. List Filtering: Write a program that filters a list of integers, keeping only the even or odd numbers based on user input.

9. Finding Common Elements in Multiple Lists: Create a program that finds and displays elements that appear in all of a given set of lists.

10. Merging Two Sorted Lists: Write a program that merges two sorted lists into a single sorted list efficiently, without using built-in sorting functions.