In Python, lists are one of the most versatile and commonly used data structures, especially in **data science**. They allow you to store, manipulate, and analyze collections of data efficiently.

1. Basics of Lists

A list in Python is an **ordered, mutable collection** that can store elements of different data types.

Creating Lists

```
empty_list = []

numbers = [1, 2, 3, 4, 5]
strings = ["apple", "banana", "cherry"]
mixed = [1, "hello", 3.14, True]

nested_list = [[1, 2, 3], [4, 5, 6]]
```

Accessing Elements

```
# Indexing (0-based)
print(numbers[0])
print(numbers[-1])

# Slicing
print(numbers[1:4])
print(numbers[:3])
print(numbers[::2])
```

Modifying Lists

```
numbers[0] = 10
print(numbers)

numbers.append(6)
numbers.insert(2, 99)
print(numbers)

numbers.remove(3)
popped_value = numbers.pop()
```

2. Common List Operations

Length of a List

```
print(len(numbers))
```

Looping Through a List

```
# Using for loop
for num in numbers:
    print(num)
```

3. Useful List Methods

```
numbers.sort()
numbers.sort(reverse=True)

numbers.reverse()
```

```
count_twos = numbers.count(2)

new_list = numbers.copy()
```

4. Converting Lists to Other Data Structures

```
# List to tuple
tuple_numbers = tuple(numbers)

# List to set (removes duplicates)
set_numbers = set(numbers)
```

Python List Operations

Operation	Example	Result
Creating a list	lst = [1,2,3]	[1,2,3]
Indexing	lst[0]	1
Slicing	lst[1:3]	[2,3]
Append	lst.append(4)	[1,2,3,4]
Insert	lst.insert(1,99)	[1,99,2,3]
Remove	lst.remove(2)	[1,99,3]
Pop	lst.pop()	3
Sort	lst.sort()	[1,2,3]
Reverse	lst.reverse()	[3,2,1]