Day-2-Python Basics-2

## **Python Lists: A Comprehensive Guide**

In Python, a **list** is a fundamental data structure that allows you to store a collection of items in a particular order. Think of it like a dynamic array where you can keep various types of data—numbers, strings, even other lists—all within a single variable.

### **Key Characteristics of Lists**

* **Ordered:** The items in a list maintain a specific order, and this order will not change unless explicitly modified. This means you can access items by their position (index).
* **Mutable:** This is a crucial characteristic. "Mutable" means that you can **change the content of a list after it's been created** without changing the list's identity (its memory location). You can add, remove, or modify elements.
* **Allows Duplicates:** Lists can contain multiple instances of the same item.
* **Heterogeneous:** A single list can hold items of different data types (e.g., integers, strings, floats).

### **Creating a List**

You create a list by enclosing a comma-separated sequence of items within square brackets [].

Python

fruits = ['apple', 'banana', 'grapes']

print(fruits)

# Output: ['apple', 'banana', 'grapes']

### **Modifying List Elements**

Since lists are mutable, you can easily change their contents.

* **Changing an Item at a Specific Index:** You can replace an item at a given position by assigning a new value to that index.
* Python

fruits = ['apple', 'banana', 'grapes']

fruits[1] = 'mango' # Replaces 'banana' with 'mango'

print(fruits)

# Output: ['apple', 'mango', 'grapes']

* **Adding an Item to the End (append()):** The append() method adds a new item to the very end of the list.
* Python

fruits = ['apple', 'mango', 'grapes']

fruits.append('guava')

print(fruits)

# Output: ['apple', 'mango', 'grapes', 'guava']

* **Inserting an Item at a Specific Index (insert()):** The insert() method allows you to add an item at a specified index. Existing items from that index onwards are shifted to the right.

fruits = ['apple', 'mango', 'grapes', 'guava']

fruits.insert(1, 'pineapple') # Inserts 'pineapple' at index 1

print(fruits)

# Output: ['apple', 'pineapple', 'mango', 'grapes', 'guava']

### **Removing List Elements**

Python provides several ways to remove items from a list:

* **Removing an Item by Value (remove()):** The remove() method deletes the first occurrence of a specified value from the list.

fruits = ['apple', 'pineapple', 'mango', 'grapes', 'guava']

fruits.remove('guava')

print(fruits)

# Output: ['apple', 'pineapple', 'mango', 'grapes']

* **Removing an Item by Index (pop()):** The pop() method removes and returns the item at a specified index. If no index is provided, it removes and returns the last item.
* Python

fruits = ['apple', 'pineapple', 'mango', 'grapes']

popped\_fruit = fruits.pop(1) # Removes item at index 1 ('pineapple')

print(fruits)

# Output: ['apple', 'mango', 'grapes']

print(f"Removed fruit: {popped\_fruit}")

# Output: Removed fruit: pineapple

### **Sorting Lists**

You can sort lists in various ways using the sort() method, which modifies the list in-place.

* **Ascending Order (Default):** By default, sort() arranges elements in ascending order.

list1 = [1, 2, 5, 4, 6]

list1.sort() # Equivalent to list1.sort(key=None, reverse=False)

print(list1)

# Output: [1, 2, 4, 5, 6]

* **Descending Order:** To sort in descending order, set the reverse parameter to True.

list1 = [1, 2, 5, 4, 6]

list1.sort(reverse=True)

print(list1)

# Output: [6, 5, 4, 2, 1]

* **Custom Sorting with key:** The key parameter allows you to specify a function that will be called on each list item before comparisons are made for sorting. This is useful for custom sorting criteria.
* Python

words = ['banana', 'apple', 'kiwi']

words.sort(key=len) # Sorts based on the length of each string

print(words)

# Output: ['kiwi', 'apple', 'banana']

Lists are incredibly versatile and are one of the most commonly used data structures in Python programming. Understanding their mutability and the various methods for modification and sorting is key to effective Python development.