Python'da **liste (list)**, sıralı ve **değiştirilebilir (mutable)** bir veri yapısıdır. Listeler, farklı veri tiplerini içerebilen bir dizi öğeyi bir arada tutmak için kullanılır. Listeler, **[ ]** köşeli parantezler içinde tanımlanır ve öğeler virgülle ayrılır. Python'da en çok kullanılan veri tiplerinden biridir çünkü esneklik sağlar ve çok yönlüdür.

Liste'nin Temel Özellikleri:

1. **Sıralı (Ordered)**: Listeler, elemanların eklenme sırasını hatırlar. İlk öğe `0` indisinde, ikinci öğe `1` indisinde vb. bulunur.
2. **Değiştirilebilir (Mutable)**: Listeler oluşturulduktan sonra elemanlar eklenebilir, değiştirilebilir veya silinebilir.
3. **Heterojen (Farklı Türlerde Öğeler İçerebilir)**: Bir liste, farklı veri tiplerinden öğeler içerebilir (integer, string, float, başka bir liste vb.).
4. **Tekrar Eden Değerler**: Listelerde aynı öğe birden fazla kez bulunabilir.

Liste Oluşturma

Listeler, **[]** köşeli parantezlerle tanımlanır. Liste içinde farklı türde veriler olabilir.

Örnekler:

```
# Boş liste
liste1 = []

# Birden fazla öğeli liste
liste2 = [1, 2, 3, 4]

# Farklı veri tiplerini içeren liste
liste3 = [1, "Python", 3.14, True]

# Liste içinde liste
liste4 = [1, [2, 3], 4]
```

Liste Elemanlarına Erişim

Listelerde, elemanlara indeks ile erişilir. İndeksleme 0'dan başlar ve negatif indeksleme ile sondan da elemanlara erişilebilir.

Örnekler:

```
# İndeksleme
liste = [10, 20, 30, 40, 50]

# İlk elemana erişim
print(liste[0])  # 10

# Son elemana erişim
print(liste[-1])  # 50

# Üçüncü elemana erişim
print(liste[2])  # 30
```

Liste Elemanlarını Güncelleme

Listeler **değiştirilebilir** olduklarından, liste içindeki öğeleri değiştirmek mümkündür.

Örnek:

```
liste = [10, 20, 30]

# İlk elemanı güncelle
liste[0] = 100
print(liste)  # [100, 20, 30]
```

Listeye Eleman Ekleme

Listeye eleman eklemek için çeşitli yöntemler kullanılabilir:

- `append()`: Listenin sonuna eleman ekler.
- `insert()`: Belirtilen indekse eleman ekler.
- `extend()`: Bir listeyi başka bir listeye ekler.

Örnekler:

```
liste = [1, 2, 3]

# Sonuna eleman ekle
liste.append(4)
print(liste)  # [1, 2, 3, 4]

# Belirli bir indekse eleman ekle
liste.insert(1, 10)
print(liste)  # [1, 10, 2, 3, 4]

# Bir listeyi diğerine ekle
liste2 = [5, 6]
liste.extend(liste2)
print(liste)  # [1, 10, 2, 3, 4, 5, 6]
```

Listeden Eleman Silme

Bir listeden eleman silmek için çeşitli yöntemler vardır:

- `remove()`: Belirtilen değere sahip ilk elemanı siler.
- `pop()`: Belirtilen indeksteki elemanı siler ve o elemanı döner. İndeks verilmezse son elemanı siler.
- `del`: Belirtilen indeksteki elemanı siler veya tüm listeyi yok eder.
- `clear()`: Listenin tüm elemanlarını siler.

Örnekler:

```
liste = [10, 20, 30, 40, 50]
```

```
# Değer bazlı eleman silme
liste.remove(30)
print(liste)  # [10, 20, 40, 50]

# İndeks bazlı eleman silme
liste.pop(1)
print(liste)  # [10, 40, 50]

# Son elemanı sil
liste.pop()
print(liste)  # [10, 40]

# Tüm elemanları sil
liste.clear()
print(liste)  # []
```

Liste İçinde Dilimleme (Slicing)

Listelerde dilimleme, bir liste içindeki belli bir aralıktaki öğelere erişmek için kullanılır.

Örnekler:

```
liste = [10, 20, 30, 40, 50]

# İlk 3 elemanı al
print(liste[:3])  # [10, 20, 30]

# 2. indeksten itibaren elemanları al
print(liste[2:])  # [30, 40, 50]

# 1. ve 3. indeksler arasındaki elemanları al
print(liste[1:4])  # [20, 30, 40]
```

Liste Metodları

Python listeleri, listelerle çalışmayı kolaylaştıran birçok yerleşik metoda sahiptir:

1. `append(x)` : Listeye eleman ekler.

```
liste.append(5)
```

2. `insert(i, x)` : Belirtilen indekse eleman ekler.

```
liste.insert(1, 10)
```

3. `remove(x)` : Belirtilen değeri listeden siler (ilk bulduğu değeri).

```
liste.remove(10)
```

4. `pop(i)` : Belirtilen indeksteki elemanı siler ve döner.

```
liste.pop(2)
```

5. `clear()` : Listedeki tüm elemanları siler.

```
liste.clear()
```

6. `index(x)` : Belirtilen değerin ilk geçtiği indeksi döner.

```
print(liste.index(20))
```

7. `count(x)` : Belirtilen değerin listede kaç kez geçtiğini döner.

```
print(liste.count(20))
```

8. `sort()` : Listeyi küçükten büyüğe sıralar.

```
liste.sort()
```

9. `reverse()` : Listeyi tersine çevirir.

```
liste.reverse()
```

10. `extend(iterable)`: Başka bir iterable'ı listeye ekler.

```
liste.extend([6, 7, 8])
```

Listeler Arasında Dönüşüm

Listeleri diğer veri türlerine (tuple, string) dönüştürmek mümkündür.

Örnekler:

```
# Listeyi string'e dönüştürme
liste = ['Python', 'programlama']
string = ' '.join(liste)
print(string)  # "Python programlama"

# Listeyi tuple'a dönüştürme
liste = [1, 2, 3]
tuple_yapisi = tuple(liste)
print(tuple_yapisi)  # (1, 2, 3)
```

Listelerin Avantajları:

1. **Esneklik**: Farklı veri tiplerini bir arada tutabilir ve bu veriler üzerinde çeşitli işlemler yapabilirsiniz.
2. **Değiştirilebilirlik**: Elemanlar üzerinde değişiklik yapabilir, eleman ekleyip çıkarabilirsiniz.
3. **Zengin Fonksiyonellik**: Çok sayıda yerleşik fonksiyon ve metod ile listeler üzerinde kompleks işlemler gerçekleştirilebilir.

Özet:

- Python listeleri, sıralı ve değiştirilebilir veri tipleridir.
- Farklı veri tiplerini içerebilirler ve elemanlara indeksleme ile erişilebilir.
- Birçok işlevsel yerleşik metodla güçlü veri manipülasyon imkânı sağlarlar.
- Liste üzerinde ekleme, silme, değiştirme gibi işlemler kolayca yapılabilir.