- **Listenin oluşturulması ve özellikleri:**

d=[] veya
d=list() d isimli boş liste oluşturur.

d=[10,12,6] Üç tane öğesi olan ilk değerleri 10,12 ve 6 olan liste oluşturur.

d=[10,"abc",20.2,10] liste öğeleri farklı veri tipinde olabilir.

d=[10,12,[23,54],28] listenin öğeleri de liste olabilir.

- **Listeye erişim ve indisler**

d=[56, 98, 23, 86]
a=d[1]
d listesinin 2. öğesi olan 98 değeri a değişkenine aktarılır.
- indisler 0 (sıfır) dan başlar.
- - (eksi) işaretli indis kullanılabilir. Son öğeden başlanarak sayılır.

a=d[-2]
d listesinin sondan 2. öğesi olan 23 değeri a değişkenine aktarılır.
- Olmayan bir indis değeri belirtildiğinde "IndexError" hatası oluşur.

a=d[10]
IndexError: list index out of range

Liste içinde liste olduğu durumda içteki liste için indis belirtilebilir:

```
d=[10,12,[23,54],28]
a=d[2]
```

d listesinin 2 numaralı indisine karşılık gelen değer [23,54] bir listedir.

```
a=d[2][1]
```

d listesinin 2 numaralı indisine karşılık gelen listenin 1 numaralı indisinde yer olan 54 değerine erişilir.

Örnek:

```
m=[[0,1],[1,0]]
```

m listesi matrise karşılık:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

gibi düşünülebilir.

- **Metin tipinde olan değişkenler liste ailesindendir.**

  s="Dolar yine yükseldi"

  a=s[0] a değişkeni "D" değerini alır.

  a=s[-1] a değişkeni "i" değerini alır.

- **Range() fonksiyonunun verdiği indisler liste gibi kullanılabilir.**

  a=range(5) a değişkeni "range" tipindedir (Class).

  b=a[1] b değişkeni 1 tamsayı değerini alır.

- **len() fonksiyonu ile listenin uzunluğu öğrenilebilir.**

```
d=["y","d","i"]
u=len(d)
```

u değişkeni 3 değerini alır.

- **Listenin ekrana yazılması**

```
f=[10,"g",20,95]
print(f)
```

ekranda:

[10, 'g', 20, 95]

görünür.

Veya döngü ile yazılabilir:

```
f=[10,"g",20,95]
for i in range(len(f)):
    print(f[i])
```

Veya:

```
f=[10,"g",20,95]
for i in f:
    print(i)
```

Ekranda:

10
g
20
95

Görünür.

- **Liste çoğaltılabilir:**

```
f=[1,3,5]
f=2*f
print(f)
```

```
[1, 3, 5, 1, 3, 5]
```

Veya listeyi çoğaltmadan:

```
f=[1,3,5]
print(2*f)
print(f)
```

```
[1, 3, 5, 1, 3, 5]
[1, 3, 5]
```

- **For döngüsünün indisi döngü içinde yapılan değişiklikten etkilenmez.**

```
f=[1,3,5]
for a in f:
    print(a)
    a+=4
    print(a)
print(f)
```

```
1
5
3
7
5
9
[1, 3, 5]
```

- **Liste indisi aralık gösterecek biçimde yazılabilir.**

liste[başlangıç:bitiş:adım]

```
r=[3,5,7,9,11,13,15]
s=r[1:5:2]
print(s)
```

```
[5, 9]
```

```
r=[3,5,7,9,11,13,15]
s=r[1::2]
print(s)
```

```
[5, 9, 13]
```

```
r=[3,5,7,9,11,13,15]
s=r[::2]
print(s)
```

[3, 7, 11, 15]

```
r=[3,5,7,9,11,13,15]
s=r[1:4]
print(s)
```

[5, 7, 9]

```
r=[3,5,7,9,11,13,15]
s=r[4:]
print(s)
```

[11, 13, 15]

- **Liste öğelerinin değerleri değişebilir.**

```
r=[3,5,7,9]
r[1]=11
print(r)
```

```
[3, 11, 7, 9]
```

```
r=[3,5,7,9]
r[-1]=11
print(r)
```

```
[3, 5, 7, 11]
```

```
r=[3,5,7,9]
r[1:3]=11,12
print(r)
```

```
[3, 11, 12, 9]
```

- **Listeler birleştirilebilir.**

```
r=[3,5,7,9]
r=r+[11,13]
print(r)
```

```
[3, 5, 7, 9, 11, 13]
```

```
r=[3,5,7,9]
r=r+[11]
print(r)
```

```
[3, 5, 7, 9, 11]
```

```
r=[3,5,7,9]
r=r+11
print(r)
```

```
TypeError: can only concatenate list
```

Liste metotları

- liste.**append(x)** : x değerini listenin sonuna ekler.

Örnek:

```
a=[4,6,8]
x=10
a.append(x)
print(a)
# Eşdeğerleri
x=12
a[len(a):] = [x]
print(a)
a=a+[14]
print(a)
```

```
[4, 6, 8, 10]
[4, 6, 8, 10, 12]
[4, 6, 8, 10, 12, 14]
```

- liste.**extend(yeni liste veya gezilebilir (iterable) yapı)**:

  Listenin sonuna bir başka liste veya gezilebilir bir yapı ekler.

  Örnek:

```
a=[6,8,10]
a.extend([1,3])
print(a)
#Eşdeğerleri
a[len(a):]=[5,7]
print(a)
a=a+[9,11]
print(a)
```

```
[6, 8, 10, 1, 3]
[6, 8, 10, 1, 3, 5, 7]
[6, 8, 10, 1, 3, 5, 7, 9, 11]
```

- **liste.insert(i,x)** : x değeri listenin i indisinin öncesine eklenir.

Örnek:

```
a=[6,8,10]
a.insert(1,7)
print(a)
a.insert(0,5)
print(a)
a.insert(len(a),11)
print(a)
```

```
[6, 7, 8, 10]
[5, 6, 7, 8, 10]
[5, 6, 7, 8, 10, 11]
```

- **liste.remove(x)** : x değerine eşit olan ilk öğeyi listeden çıkarır. x değerine eşit öğe yoksa ValueError ortaya çıkar.

Örnek:

```
a=[6,8,10,8]
a.remove(8)
print(a)
```

```
[6, 10, 8]
```

```
a=[6,8,10,8]
a.remove(9)
print(a)
```

```
ValueError: list.remove(x): x not in list
```

- **liste.pop([i]):** i indisi ile gösterilen öğeyi çıkarır ve öğenin değerini verir. Eğer indis belirtilmezse listenin son öğesini çıkarır (Notasyonda görünen köşeli parantezler i indisini yazmanın zorunlu olmadığını gösterir).

  örnek:

```
a=[6,9,10,8]
b=a.pop(2)
print(a)
print(b)
b=a.pop()
print(a)
print(b)
```

```
[6, 9, 8]
10
[6, 9]
8
```

- **liste.clear()** : Listenin tüm öğelerini (değerlerini) siler. Boş liste kalır.

Örnek:

```
a=[6,9,10,8]
a.clear()
print(a)
#eşdeğeri
a=[6,9,10,8]
del a[:]
print(a)
```

```
[]
[]
```

- **del deyimi**: Listenin tamamını veya bir kısmını bellekten silmek için kullanılır.

Örnek:

```
a=[6,9,10,8]
del a[1]
print(a)
a=[6,9,10,8]
del a[1:3]
print(a)
```

```
[6, 10, 8]
[6, 8]
```

```
a=[6,9,10,8]
del a
```

a listesinin tamamını bellekten siler. print(a) yazılırsa NameError hatası ortaya çıkar.

- **liste.index(x[, başlangıç[, bitiş]])** : Liste içinde x değerinin bulunduğu ilk indisi verir. Başlangıç ve bitiş indisleri zorunlu olmadığı için köşeli parantez içinde yazılmıştır. x değeri liste içinde yok ise ValueError ortaya çıkar. Başlangıç ve bitiş indisleri verildiğinde x değerinin indisi istenen parça içinde değil, tüm liste içindeki indistir.

Örnek:

```
a=[6,9,10,8,8]
i=a.index(9)
print(i)
i=a.index(8)
print(i)
i=a.index(10,1,3)
print(i)
i=a.index(20)
```

```
1
3
2
ValueError
```

- **liste.count(x)** : Liste içinde x değerinin kaç kez tekrarlandığını verir.

Örnek:

```
a=[6,9,10,8,8]
k=a.count(8)
print(k)
k=a.count(20)
print(k)
```

```
2
0
```

- **liste.sort(key=None, reverse=False)** : Listeyi kendi üzerinde sıralar. key ve reverse parametreleri belirtilmediğinde varsayılan değerleri, None ve False olur. Azalan sıralama için reverse=True olmalıdır. Key parametresi sıralama ölçütü oluşturmak amacıyla kullanılan fonksiyon adıdır. Örneğin metin öğeleri uzunluklarına göre sıralamak amacıyla kullanılabilir.

  Örnek:

```
a=[6,9,10,8,8]
a.sort()
print(a)
```

```
[6, 8, 8, 9, 10]
```

```
a=[6,9,10,8,8]
a.sort(reverse=True)
print(a)
```

```
[10, 9, 8, 8, 6]
```

```
a=[6,9,10,8,8,"12"]
a.sort(key=int)
print(a)
```

```
[6, 8, 8, 9, 10, '12']
```

```
a=[6,9,10,8,8,"12"]
a.sort()
```

```
TypeError: '<' not supported between
instances of 'str' and 'int'
```

- **sorted() fonksiyonu** : Listeyi veya gezilebilir (iterable) bir yapının sıralanmış durumunu verir. Benzer olarak, key ve reverse parametreleri vardır.

Örnek:

```
a=[6,9,10,8,8]
b=sorted(a)
print(b)
```

```
[6, 8, 8, 9, 10]
```

```
a=[6,9,10,8,8]
b=sorted(a,reverse=True)
print(b)
```

```
[10, 9, 8, 8, 6]
```

- **liste.reverse()** : Liste öğelerini kendi üzerinde tersine çevirir.

Örnek:

```
a=[6,9,10,8,7]
a.reverse()
print(a)
```

```
[7, 8, 10, 9, 6]
```

```
a=[6,9,[10,8],7]
a.reverse()
print(a)
```

```
[7, [10, 8], 9, 6]
```

```
a=[6,9,[10,8],7]
a[2].reverse()
print(a)
```

```
[6, 9, [8, 10], 7]
```

- **liste.copy()** : Listenin kopyasını verir. Farklı adreslerde iki tane liste oluşur.

  Örnek:

```
a=[6,9,10,8,7]
b=a.copy()
print(a)
print(b)
```

```
[6, 9, 10, 8, 7]
[6, 9, 10, 8, 7]
```

- **Eşdeğeri**:

```
a=[6,9,10,8,7]
b=a[:]
print(a)
print(id(a))
print(b)
print(id(b))
```

```
[6, 9, 10, 8, 7]
2518355459968
[6, 9, 10, 8, 7]
2518328957248
```

- **Benzer olarak atama ile liste kopyalama**: Aynı bellek adresini paylaşırlar. Bir tanesinde yapılan değişiklik diğerini de etkiler.

```
a=[6,9,10,8,7]
b=a
print(a)
print(id(a))
print(b)
print(id(b))
b[1]=3
print(a)
print(b)
print(id(a))
print(id(b))
```

```
[6, 9, 10, 8, 7]
2518320103360
[6, 9, 10, 8, 7]
2518320103360
[6, 3, 10, 8, 7]
[6, 3, 10, 8, 7]
2518320103360
2518320103360
```

- **Listede kullanılabilen fonksiyonlar**
- **min() fonksiyonu**: Liste veya tekrarlayabilir (iterable) yapılar içinde en küçük değeri bulur.

Örnek:

```
a=[6,9,10,8,7]
mn=min(a)
print(mn)
mn=min(a[2:4])
print(mn)
```

```
6
8
```

- **max() fonksiyonu**: Liste veya tekrarlayabilir (iterable) yapılar içinde en büyük değeri bulur.

Örnek:

```
a=[6,9,10,8,7]
mx=max(a)
print(mx)
mx=max(a[3:4])
print(mx)
```

```
10
8
```

- **sum(tekrarlanabilir yapı, start=0) fonksiyonu** : Liste veya tekrarlanabilir (iterable) yapılar içinde bulunan değerlerin toplamını bulur. Değerler sayısal olmalıdır. Start parametresi 3.8 versiyonunda eklenmiştir.

  Örnek:

```
a=[6,9,10,8,7]
toplam=sum(a)
print(toplam)
toplam=sum(a,start=10)
print(toplam)
```

```
40
50
```