3. List, Tuple ve Dictionary veri tipi ve Metotları

Liste veri tipi

- Liste [] (köşeli parantez) ile tanımlanır ve diğer veri tiplerini barındırabilir.
- Python'da yaygın şekilde kullanılan ve işlemleri kolaylaştıran bir yapıdır.
- Tıpkı Stringler gibi parçalama, indeksleme gibi değişik işlemler yapabilir.
- Stringlerde değişme yapmak olanaksızken, listelerde değişiklik yapabilmek mümkündür.

```
liste = ["Çay", 35, "Game of Thrones"]
liste
```

```
['Çay', 35, 'Game of Thrones']
```

```
type(liste)
```

```
list
```

```
liste2 = [1, 2, 3, "Tuba Ünsal", 9, "Üç kalp"]
liste2
```

```
[1, 2, 3, 'Tuba Ünsal', 9, 'Üç kalp']
```

```
len(liste2)
```

```
6
```

```
liste = [2,4,6,8,10,12,14,16,18,20]
liste
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

```
liste
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

```
liste[2]
```

```
6
```

```
liste[9]
```

```
20
```

```
liste[-1]
```

```
20
```

```
len(liste)
```

```
10
```

Listenin son elemanını görmek amacıyla aşağıdaki işlem yapılabilir.

```
liste[len(liste)-1]
```

```
20
```

```
liste
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

```
liste[4:]
```

```
[10, 12, 14, 16, 18, 20]
```

```
liste[:3]
```

```
[2, 4, 6]
```

```
liste [0:7:2]
```

```
[2, 6, 10, 14]
```

```
liste[::2]
```

```
[2, 6, 10, 14, 18]
```

```
liste[::-1]
```

```
[20, 18, 16, 14, 12, 10, 8, 6, 4, 2]
```

Temel Liste İşlemleri

Listeler, birbileriyle toplanabilir ve bir başka liste değişkeni olarak yazılabilir.

```
liste1 = [1,2,3,4,5]
liste2 = [6,7,8,9,10]

liste3 = liste1 + liste2
liste3
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
l = liste1 * 3
l
```

```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

```
liste1
```

```
[1, 2, 3, 4, 5]
```

```
liste1 = liste1 * 3
liste1
```

```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

```
liste2
```

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

```
liste2[2] = 44
liste2
```

```
[6, 7, 44, 9, 10]
```

```
liste2[:2]
```

```
[6, 7]
```

```
liste2[:2] = [60,70]
liste2
```

```
[60, 70, 44, 9, 10]
```

List veri tipi Metotları

.append() metodu

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

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

```
len(liste)
```

```
6
```

```
liste.append("python")
liste
```

```
[1, 2, 3, 4, 5, 6, 'python']
```

```
len(liste)
```

```
7
```

```
liste
```

```
[1, 2, 3, 4, 5, 6, 'python']
```

```
liste.append(200)
liste
```

```
[1, 2, 3, 4, 5, 6, 'python', 200]
```

.pop()

pop() listeden eleman silmek için kullanılır. İçine hiçbirşey yazılmazsa son elemanı siler ve sildiği elemanı gösterir. Haricinde atılmak istenen elemanın indeks numarası verilirse, bu eleman listeden atılır.

```
liste
```

```
[1, 2, 3, 4, 5, 6, 'python', 200]
```

```
liste.pop() # listenin son elemanı atıldı
liste
```

```
[1, 2, 3, 4, 5, 6, 'python']
```

```
liste
```

```
[1, 2, 3, 4, 5, 6, 'python']
```

```
liste.pop()
```

```
'python'
```

```
liste
```

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

```
liste.pop(0) # atılmak istenen elemanın index değeri parametre olarak verildi
```

```
1
```

```
liste
```

```
[2, 3, 4, 5, 6]
```

.sort () metodu

```
liste = [36, 16, 80, 34, 32, 100, 800, 24]
```

```
liste.sort() # küçükten büyüğe sıralar
liste
```

```
[16, 24, 32, 34, 36, 80, 100, 800]
```

Görüldüğü gibi sort metodu, büyükten küçüğe doğru listeyi sıralar.

```
liste.sort(reverse = True) # büyükten küçüğe sıralar
liste
```

```
[800, 100, 80, 36, 34, 32, 24, 16]
```

reverse = true komutu, listenin büyükten küçüğe yazılmasını sağlar.

Liste elemanları string ise sort metodu, stringlerin alfabetik sırayla yazılmasını sağlar.

```
liste = ["Python", "Java", "C++", "Visual Basic"]

liste.sort()
liste
```

```
['C++', 'Java', 'Python', 'Visual Basic']
```

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

```
['Visual Basic', 'Python', 'Java', 'C++']
```

.insert()

```
a = [1,2,3,4,5]

a.insert(2,"Ulubatlı Sooness")
a
```

```
[1, 2, 'Ulubatlı Sooness', 3, 4, 5]
```

```
len(a)
```

```
6
```

```
a
```

```
[1, 2, 'Ulubatlı Sooness', 3, 4, 5]
```

.remove()

```
a.remove("Ulubatlı Sooness")
a
```

```
[1, 2, 3, 4, 5]
```

del list[index no]

```
del a[1]
a
```

```
[1, 3, 4, 5]
```

```
del a # listeyi olduğu gibi siler
a
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[50], line 2
      1 del a # listeyi olduğu gibi siler
----> 2 a

NameError: name 'a' is not defined
```

Liste kopyalama

List bir class ve bellekte referans tip olarak ele alınır dolayısıyla bir listeyi başka bir listeye atamak istediğimizde liste elamanları kopyalanmaz bunun yerine listenin bellekteki adres bilgisi kopyalanır.

```
a = ["elma","muz"]
b = ["kayısı","karpuz"]
a = b
```

```
a # b listesi, a^' nın adresine atanmıştır
```

```
['kayısı', 'karpuz']
```

```
b
```

```
['kayısı', 'karpuz']
```

```
a[0] = "armut"
```

```
a
```

```
['armut', 'karpuz']
```

```
b # a daki değişim b yi etkilemiştir.
```

```
['armut', 'karpuz']
```

```
a = ["elma","muz"]
b = ["kayısı","karpuz"]
```

```
a
```

```
['elma', 'muz']
```

```
b
```

```
['kayısı', 'karpuz']
```

```
a = b.copy()
a
```

```
['kayısı', 'karpuz']
```

```
b
```

```
['kayısı', 'karpuz']
```

```
a[0] = "armut"
a
```

```
['armut', 'karpuz']
```

```
b # b, değişimden etkilenmedi !
```

```
['kayısı', 'karpuz']
```

count()

```
numbers = [1, 10, 5, 16, 10, 9, 10]
letters = ['a', 'g', 's', 'b', 'y', 'a', 's']
```

```
numbers.count(10)  # 2
```

```
3
```

```
letters.count('a')
```

```
2
```

```
numbers
```

```
[1, 10, 5, 16, 10, 9, 10]
```

```
numbers.clear() # numbers (sayılar) listesini temizler, boş liste yapar

numbers

[]
```

İç içe listeler

```
liste = [[1,2], [3,4], [100,200]]
liste

[[1, 2], [3, 4], [100, 200]]

liste[2]

[100, 200]

liste[2][1]

200
```

Tuple (Demet) veri tipi

Demetler, listelere oldukça benzerdir.tek ve en önemli farkı elemanlarının değiştirilemez olmasıdır.

```
Demet = (1,2,3,4,5,6)
Demet

(1, 2, 3, 4, 5, 6)

type(Demet)

tuple

Demet[0]

1

Demet[7]

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[76], line 1
----> 1 Demet[7]
```

```
IndexError: tuple index out of range
```

```
Demet
```

```
(1, 2, 3, 4, 5, 6)
```

```
Demet[-1]
```

```
6
```

```
Demet[-1] = 8
Demet
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[79], line 1
----> 1 Demet[-1] = 8
      2 Demet

TypeError: 'tuple' object does not support item assignment
```

```
Demet[:4]
```

```
(1, 2, 3, 4)
```

```
Demet[::-1]
```

```
(6, 5, 4, 3, 2, 1)
```

Demetlerin metotları

.count()

count() fonksiyonu parantez içerisine yazılan değerin demetin içinde kaç defa geçtiğini sayar.

```
demet = (1,1,1,2,2,2,2,3,3,3,3,"Ajda")
demet
```

```
(1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 'Ajda')
```

```
demet.count(1)
```

```
3
```

```
demet.index("Ajda")
```

```
11
```

```
demet.count(18)
```

```
0
```

.index()

index() fonksiyonu, demet yapısı içerisinde, fonksiyonun parantezi içerisine yazılan değerin indeksini söyler.

```
demet2 = ("python", 2, 3, "logout", "GS", 100, "python")
demet2
```

```
('python', 2, 3, 'logout', 'GS', 100, 'python')
```

```
demet2.count("python")
```

```
2
```

```
demet2.index("python")
```

```
0
```

Dictionary dtype (Sözlük veri tipi)

- Dictionary (Sözlük) veri tipi, "anahtar" - "değer" ilişkisine sahiptir.
- Süslü parantez {} ile tanımlanır,
- anahtar ve değer, ":" iki nokta üst üste işareti ile ayrılır.

```
sözlük1 = {"bir":1, "iki":2, "üç":3, "dört":4}
sözlük1
```

```
{'bir': 1, 'iki': 2, 'üç': 3, 'dört': 4}
```

```
type(sözlük1)
```

```
dict
```

```
sözlük1["bir"]
```

```
1
```

```
sözlük1["dört"]
```

```
4
```

```
sözlük1["beş"] = 5
sözlük1
```

```
{'bir': 1, 'iki': 2, 'üç': 3, 'dört': 4, 'beş': 5}
```

```
A = {"bir":[1,2,3,4], "iki":[[1,2],[3,4],[8,9]]}
A
```

```
{'bir': [1, 2, 3, 4], 'iki': [[1, 2], [3, 4], [8, 9]]}
```

```
A["iki"]
```

```
[[1, 2], [3, 4], [8, 9]]
```

```
A["iki"][2][1]
```

```
9
```

```
sözlük1
```

```
{'bir': 1, 'iki': 2, 'üç': 3, 'dört': 4, 'beş': 5}
```

```
sözlük1["beş"] += 1
sözlük1
```

```
{'bir': 1, 'iki': 2, 'üç': 3, 'dört': 4, 'beş': 6}
```

İç içe sözlükler

```
B = {"sayılar":{"bir":1, "iki":2,"üç":3},"meyveler":{"elma":"yaz",
"portakal":"kış"}}
B
```

```
{'sayılar': {'bir': 1, 'iki': 2, 'üç': 3},
 'meyveler': {'elma': 'yaz', 'portakal': 'kış'}}
```

```
B["sayılar"]
```

```
{'bir': 1, 'iki': 2, 'üç': 3}
```

```
B["sayılar"]["iki"]
```

```
2
```

```
B["meyveler"]["portakal"]
```

```
'kış'
```

Sözlük metotları

.keys()

```
sözlük1

{'bir': 1, 'iki': 2, 'üç': 3, 'dört': 4, 'beş': 6}

sözlük1.keys()

dict_keys(['bir', 'iki', 'üç', 'dört', 'beş'])
```

.values()

```
sözlük1.values()

dict_values([1, 2, 3, 4, 6])

sözlük1.items()

dict_items([('bir', 1), ('iki', 2), ('üç', 3), ('dört', 4), ('beş',
6)])

for x,y in sözlük1.items():
        print(x,y)

bir 1
iki 2
üç 3
dört 4
beş 6

for x in B:
    print(x)

sayılar
meyveler

sözlük1

{'bir': 1, 'iki': 2, 'üç': 3, 'dört': 4, 'beş': 6}

for x,y in B.items():
        print(x,y)

sayılar {'bir': 1, 'iki': 2, 'üç': 3}
meyveler {'elma': 'yaz', 'portakal': 'kış'}
```

```
import pandas as pd

df = pd.DataFrame()
df["x"] = sözlük1.keys()
df["y"] = sözlük1.values()
df
```

```
      x  y
0   bir  1
1   iki  2
2    üç  3
3  dört  4
4   beş  6
```