Listeler

Şimdi de yeni bir veritipimiz olan **listeleri** öğrenmeye çalışalım. Listeler yapıları gereği stringlere oldukça benzerler ve kullanıldıkları yerler de çok yararlı olan bir veritipidir. Tıpkı stringler gibi ,indekslenirler,parçalanırlar ve üzerinde değişik işlemler yapabildiğimiz metodlar bulunur. Ancak listelerin stringlerden önemli farkları da bulunmaktadır. Stringler konusundan bildiğimiz kadarıyla stringler değiştirilemez bir veri tipidir. Ancak, listelerimiz değiştirilebilir bir veritipidir.

Bir listede her veritipinden elemanı saklayabiliriz. Bu anlamda sıralı bir diziye benzer. Peki bu konuda ne öğreneceğiz?

```
1.Liste oluşturma
2.Indeksleme ve Parçalama
3.Temel Liste Metodları ve İşlemleri
4.İç içe Listeler
```

Liste Oluşturma

Listeler bir *[]* köşeli parantez içine veriler veya değerler konarak oluşturulabilir.

In [1]:
```
# liste değişkeni. Değişik veri tiplerinden değerleri saklayabiliyoruz.
liste =  [3,4,5,6,"Elma",3.14,5.324]
liste
```

Out[1]:
```
[3, 4, 5, 6, 'Elma', 3.14, 5.324]
```

In [2]:
```
liste2 = [3,4,5,6,7,8,9]
liste2
```

Out[2]:
```
[3, 4, 5, 6, 7, 8, 9]
```

In [3]:
```
# Boş liste
bos_liste = []
bos_liste
```

Out[3]:
```
[]
```

In [4]:
```
# Boş liste . list() fonksiyonuyla da oluşturulabilir.
bos_liste = list()
bos_liste
```

Out[4]:
```
[]
```

In [5]:
```
# len fonksiyonu listeler üzerinde de kullanılabilir.
liste3 = [3,4,5,6,6,7,8,9,0,0,0]
len(liste3)
```

Out[5]:
```
11
```

Bir string *list()* fonksiyonu yardımıyla listeye dönüştürülebilir.

In [6]:
```
s =  "Merhaba"
lst =  list(s)
lst
```

Out[6]:
```
['M', 'e', 'r', 'h', 'a', 'b', 'a']
```

Listeleri Indeksleme ve Parçalama

Listeleri indeksleme ve parçalama stringlerle birebir olarak aynıdır.

In [7]:
```
liste = [3,4,5,6,7,8,9,10]
liste
```

Out[7]:
```
[3, 4, 5, 6, 7, 8, 9, 10]
```

In [8]:
```
# 0. eleman
liste[0]
```

Out[8]:
```
3
```

In [9]:
```
# 4. eleman
liste[4]
```

Out[9]:
```
7
```

In [10]:
```
# Sonuncu Eleman
liste[len(liste)-1]
```

Out[10]:
```
10
```

In [11]:
```
# Sonuncu Eleman
liste[-1]
```

Out[11]:
```
10
```

In [12]:
```
# İlk Eleman
liste[-len(liste)]
```

Out[12]:
```
3
```

In [13]:
```
# Baştan 4. indekse kadar (dahil değil)
liste[:4]
```

Out[13]:
```
[3, 4, 5, 6]
```

In [14]:
```
# 1.indeksten 5.indekse kadar
liste[1:5]
```

Out[14]:
```
[4, 5, 6, 7]
```

In [15]:
```
liste[5:]
```

Out[15]:
```
[8, 9, 10]
```

In [16]:
```
liste[::2]
```

Out[16]:
```
[3, 5, 7, 9]
```

In [17]:
```
liste[::-1]
```

Out[17]:
```
[10, 9, 8, 7, 6, 5, 4, 3]
```

Temel Liste Metodları ve İşlemleri

Bu kısımda da listelerde yapabileceğimiz temel işlemleri ve bazı temel metodları öğreneceğiz. Listelerin daha bir çok metodunu kursun ileriki kısımlarında görüyor olacağız.

Bir listeyi birbirine toplama işlemini stringlerdeki gibi yapabiliriz.

In [18]:
```
liste1 =  [1,2,3,4,5]
liste2 =  [6,7,8,9,10]
liste1 + liste2
```

Out[18]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Bir listeye bir tane eleman eklemek içinse aşağıdaki işlemi uygulayabiliriz.

In [19]:
```
liste = [1,2,3,4]
liste =  liste + ["Murat"]
```

In [20]:
```
liste
```

Out[20]:
```
[1, 2, 3, 4, 'Murat']
```

In [21]:
```
# Listeler direk olarak değiştirilebilirler.
liste[0] = 10
```

In [22]:
```
liste
```

Out[22]:
```
[10, 2, 3, 4, 'Murat']
```

In [23]:
```
# Şöyle bir kullanım da mümkündür.
liste[:2] = [40,50]
```

In [24]:
```
liste
```

Out[24]:
```
[40, 50, 3, 4, 'Murat']
```

Bir listeyi bir tamsayıyla da çarpabiliriz.

In [25]:
```
liste = [1,2,3,4,5]
liste * 3
```

Out[25]:
```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

In [26]:
```
liste # Ama gördüğümüz gibi listemiz değişmiyor.
```

Out[26]:
```
[1, 2, 3, 4, 5]
```

In [27]:
```
liste = liste * 3
```

In [28]:
```
liste # Eşitleme yaptığımız için liste değişti.
```

Out[28]:
```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

Şimdi de isterseniz temel bazı liste metodlarına bakalım. Listeler, diğer programlama dillerindeki **arraylere** göre oldukça esnektir. Belli bir boyutları yoktur ve ekleme, çıkarma yapmak oldukça kolaydır.

append metodu

append metodu, verdiğimiz değeri listeye eklememizi sağlar.

In [29]:
```
liste = [3,4,5,6]
liste.append(7)
liste
```

Out[29]:
```
[3, 4, 5, 6, 7]
```

In [30]:
```
liste.append("Murat")
```

In [31]:
```
liste
```

Out[31]:
```
[3, 4, 5, 6, 7, 'Murat']
```

pop metodu

Bu metod, içine değer vermezsek listenin son indeksindeki elemanı, değer verirsek verdiğimiz değere karşılık gelen indeksteki elemanı listeden atar ve attığı elemanı ekrana basar.

In [32]:
```
liste = [1,2,3,4,5]
liste.pop()
```

Out[32]:
```
5
```

In [33]:
```
liste
```

Out[33]:
```
[1, 2, 3, 4]
```

In [34]:
```
eleman = liste.pop(2)
```

In [35]:
```
eleman
```

Out[35]:
```
3
```

In [36]:
```
liste
```

Out[36]:
```
[1, 2, 4]
```

In [37]:
```
liste.append("Murat")
liste
```

Out[37]:
```
[1, 2, 4, 'Murat']
```

In [38]:
```
liste.pop()
```

Out[38]:
```
'Murat'
```

```
In [39]: liste

Out[39]: [1, 2, 4]
```

Aslında zamanı gelmişken söylemekte fayda var. Liste elemanlarına ulaşırken eğer olmayan bir indeksi verirsek *Python* bizlere hata verecektir.

```
In [40]: liste = [12,54,67,67]

In [41]: liste[50]

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-41-621e5ad1f358> in <module>()
----> 1 liste[50]

IndexError: list index out of range
```

sort metodu

sort metodu listenin elemanlarını sıralamamızı sağlar. Hemen kullanımına bakalım.

```
In [ ]: liste = [34,1,56,334,23,2,3,19]
        liste.sort() # Küçükten büyüğe sıralar.
        liste
```

```
In [ ]: liste.sort(reverse = True) # Büyükten küçüğe sıralar.
        liste
```

```
In [ ]: liste = ["Elma","Armut","Muz","Kiraz"]
        liste.sort() # Alfabetic olarak küçükten büyüğe
        liste
```

```
In [ ]: liste = ["Elma","Armut","Muz","Kiraz"]
        liste.sort(reverse = True) # Alfabetic olarak büyükten küçüğe
        liste
```

İç içe Listeler

Bir listenin içinde başka bir liste bulundurmak mümkündür. Bunlara Pythonda içiçe listeler denmektedir. Bu tip bir özellik, Pythonda ağaç yapılarında veya matris yapılarında oldukça kullanışlı olmaktadır.

```
In [ ]: # 3 Tane liste oluşturalım.

        liste1 = [1,2,3]
        liste2 = [4,5,6]
        liste3 = [7,8,9]

        yeniliste = [liste1,liste2,liste3]
        yeniliste
```

Şimdi, 2. listenin ilk elemanına nasıl ulaşacağımıza bakalım.

In [ ]:
```
# 1.elemanın 0.elemanı
yeniliste[1][0]
```

Pythonda listeler şimdilik bu kadar! Listelerin geri kalan özelliklerini ileride göreceğiz.

In [ ]: