For Döngüleri

Bu konuda Pythondaki **for** döngülerinin yapısını ve for döngülerinin kullanım alanlarını öğreneceğiz. Ancak ondan önce , Pythondaki in operatörünü öğrenmeye çalışalım.

in Operatörü

Pythondaki *in* operatörü , bir elemanın başka bir listede,demette veya stringte (karakter dizileri) bulunup bulunmadığını kontrol eder. Kullanımı şu şekildedir;

```
In [3]: "a" in "merhaba"
Out[3]: True

In [4]: "mer" in "merhaba"
Out[4]: True

In [5]: "t" in "merhaba"
Out[5]: False

In [6]: 4 in [1,2,3,4]
Out[6]: True

In [7]: 10 in [1,2,3,4]
Out[7]: False

In [8]: 4 in (1,2,3)
Out[8]: False
```

for Döngüsü

for Döngüsü , listelerin ,demetlerin, stringlerin ve hatta sözlüklerin üzerinde dolaşmamızı sağlayan bir döngü türüdür. Yapısı şu şekildedir.

```
for eleman in veri_yapısı(liste,demet vs):
    Yapılacak İşlemler
```

Bu yapı bize şunu söyler;

```
eleman değişkeni her döngünün başında listenin,demetin vs. her bir elemanına eşit
olacak ve her döngüde
bu elemanla işlem yapılacak.
```

for döngüsünü daha iyi anlamak için örneklerimize bakalım.

Listeler Üzerinde Gezinmek

In [1]:
```
liste = [1,2,3,4,5,6,7]

for eleman in liste:
    print("Eleman",eleman)
```

```
Eleman 1
Eleman 2
Eleman 3
Eleman 4
Eleman 5
Eleman 6
Eleman 7
```

In [2]:
```
# Liste elemanlarını toplama
liste = [1,2,3,4,5,6,7]
toplam = 0
for eleman in liste:
    toplam += eleman
print("Toplam",toplam)
```

```
Toplam 28
```

In [3]:
```
# Çift elemanları bastırma
liste = [1,2,3,4,5,6,7,8,9]

for eleman in liste:
    if eleman % 2 == 0:
        print(eleman)
```

```
2
4
6
8
```

Karakter Dizileri Üzerinde Gezinmek (Stringler)

In [4]:
```
s =  "Python"
for karakter in s:
    print(karakter)
```

```
P
y
t
h
o
n
```

In [5]:
```
# Her bir karakterleri 3 ile çarpma
s = "Python"

for karakter in s:
    print(karakter * 3)
```

```
PPP
yyy
ttt
hhh
ooo
nnn
```

Demetler üzerinde gezinmek (Demetler)

In [6]:
```
# Listelerle aynı mantık
demet =  (1,2,3,4,5,6,7)

for eleman in demet:
    print(eleman)
```

```
1
2
3
4
5
6
7
```

Demetlerin üzerinde for döngüsü uygularken aslında çok pratik bir yöntem bulunuyor. Aşağıdaki örneğe bakalım.

In [9]:
```
# Listelerin için 2 boyutlu demetler

liste = [(1,2),(3,4),(5,6),(7,8)]

for eleman in liste:
    print(eleman) # Herbir elemanın  demet olduğu görebiliyoruz.
```

```
(1, 2)
(3, 4)
(5, 6)
(7, 8)
```

In [11]:
```
# Demet içindeki herbir elemanı almak için pratik yöntem
liste = [(1,2),(3,4),(5,6),(7,8)]

for (i,j) in liste:
    print(i,j)
```

```
1 2
3 4
5 6
7 8
```

In [12]:
```
# Demet içindeki elemanları çarpalım.
liste = [(1,2,3),(4,5,6),(7,8,9),(10,11,12)]
for (i,j,k) in liste:
    print(i * j * k)
```

```
6
120
504
1320
```

Sözlükler üzerinde gezinmek (Dictionary)

Hatırlarsanız, sözlükler konusunda 3 adet metod görmüştük. (*keys(),values(),*items()*). İsterseniz bunları burada hatırlayalım.

In [13]:
```
sözlük = {"bir":1,"iki":2,"üç":3,"dört":4}

sözlük.keys()
```

Out[13]:
```
dict_keys(['bir', 'iki', 'üç', 'dört'])
```

In [14]:
```
sözlük.values()
```

Out[14]:
```
dict_values([1, 2, 3, 4])
```

In [15]:
```
sözlük.items()
```

Out[15]:
```
dict_items([('bir', 1), ('iki', 2), ('üç', 3), ('dört', 4)])
```

Python sonuçları *dict_keys,dict_values,dict_items* olarak vermesine rağmen , bunlar üzerinde liste veya demet üzerinde gezinir gibi *for* döngüsüyle gezinebiliriz.

In [17]:
```
# Metodları kullanmadan sözlük üzerinde gezinmek - Sadece anahtarları alabiliyoruz.
sözlük = {"bir":1,"iki":2,"üç":3,"dört":4}

for eleman in sözlük:
    print(eleman)
```

```
bir
iki
üç
dört
```

In [19]:
```
# keys() - Aynı şey
sözlük = {"bir":1,"iki":2,"üç":3,"dört":4}

for eleman in sözlük.keys():
    print(eleman)
```

```
bir
iki
üç
dört
```

In [22]:
```
# values()
sözlük = {"bir":1,"iki":2,"üç":3,"dört":4}

for eleman in sözlük.values():
    print(eleman)
```

```
1
2
3
4
```

In [25]:
```
# items()
sözlük = {"bir":1,"iki":2,"üç":3,"dört":4}

for (i,j) in sözlük.items():
    print("Anahtar:",i,"Değer:",j)
```

```
Anahtar: bir Değer: 1
Anahtar: iki Değer: 2
Anahtar: üç Değer: 3
Anahtar: dört Değer: 4
```

İşte *for* döngüsü bu kadar! Bir sonraki konuda *while* döngülerini anlamaya çalışacağız.

In [ ]:
```

```