Demetler (Tuplelar)

Demetler veya İngilizce ismiyle tuplelar listelere oldukça benzer ancak farkları demetlerin değiştirilemez oluşudur. Bu yüzden programlarımızda değiştirilmesini istemediğimiz değerleri bir demet içinde depolayabiliriz. İsterseniz konumuza demetlerin oluşturulmasıyla başlayalım.

Demet Oluşturma

In [1]:
```
# Demet elemanları parantez içine alınarak demet oluşturulabilir.
demet = (1,2,3,4,5,6,7,8,9)
```

In [2]:
```
demet
```

Out[2]:
```
(1, 2, 3, 4, 5, 6, 7, 8, 9)
```

In [3]:
```
# type() fonksiyonu yardımıyla türünü öğrenelim.
type(demet)
```

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

Peki tek elemanlı bir demet nasıl tanımlanır ?

In [4]:
```
# Tek elemanlı demet bu şekilde tanımlanabilir.
demet = (1,)
```

In [5]:
```
demet
```

Out[5]:
```
(1,)
```

In [6]:
```
type(demet)
```

Out[6]:
```
tuple
```

In [7]:
```
demet = (1,2,3,4,5,6,7)
# 0. indekse  ulaşma
demet[0]
```

Out[7]:
```
1
```

In [8]:
```
# 4. indekse ulaşma
demet[4]
```

Out[8]:
```
5
```

In [9]:
```
demet[-1]
```

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

In [10]:
```
demet[2:]
```

Out[10]:
```
(3, 4, 5, 6, 7)
```

Demetlerin Temel Metodları

index metoduyla içine verdiğimiz elemanın hangi indekste olduğunu bulabiliriz.

In [11]:
```
# Demeti oluşturalım.
demet = (1,2,3,"Mustafa","Murat","Coşkun")
# "Mustafa" elemanının indeksini buluyoruz.
demet.index("Mustafa")
```

Out[11]: 3

In [12]:
```
demet.index(1)
```

Out[12]: 0

count metoduyla içine verdiğimiz değerin demette kaç defa geçtiğini bulabiliriz.

In [17]:
```
demet = (1,23,34,34,2,1,4,5,1,1,34)
demet.count(1)
```

Out[17]: 4

In [18]:
```
demet.count(34)
```

Out[18]: 3

Değiştirilmeme Özelliği

Demetlerin değiştirilemez olduğunu artık biliyoruz. İsterseniz bir deneme yapalım.

In [19]:
```
# Demet oluşturalım.

demet = ("Elma","Armut","Muz")
demet[0] = "Kiraz"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-19-b5e8086e73d6> in <module>()
      2
      3 demet = ("Elma","Armut","Muz")
----> 4 demet[0] = "Kiraz"

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

In [20]:
```
demet.remove("Elma")
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-20-d49dcfce475c> in <module>()
----> 1 demet.remove("Elma")

AttributeError: 'tuple' object has no attribute 'remove'
```

Demetleri Ne Zaman Kullanalım ?

Aslında Python programcıları demetlerden ziyade listeleri daha çok kullanır. Ancak eğer programınızda değiştirilmesini istemediğiniz bilgiler varsa (Android uygulama sabitleri gibi) bunları demet içinde depolayabilirsiniz. Aynı zamanda, Read Only(Sadece Okuma) bir veritipi olduğu için listelere göre biraz daha hızlı çalışırlar.