Koleksiyonlar

Koleksiyon veri tipleri birçok değişkeni bir arada tutmayı sağlar. 4 çeşit koleksiyon veri tipi vardır.

- List:
- Tuples
- Set
- Dictionary

List

- Sıralanabilir ve değiştirilebilir koleksiyon veri tipidir. Liste içerisinde aynı elemandan birden fazla bulundurabiliriz. Bu özellik Duplicate olarak adlandırılır.
- Diğer dillerden farklı olarak Python'da, aynı listede farklı veri tipleri bulunabilir.
- Liste yapmak için köşeli parantez yani "[sayi1, sayi2, ...]" kullanırız.
- Listelerin içindeki verilere, tıpkı string indexlemede yaptığımız gibi listede de indexleme yaparak [0] ulaşabiliriz.

```
list = ["Başar", "Yağız", "Talha", "Efe", "Ömer"]
print(list)
```

```
['Başar', 'Yağız', 'Talha', 'Efe', 'Ömer']
```

Listede Indexleme Nasıl Yapılır ?

```
list_eestec = ["Kayra", "Başar", "Yağız", "Talha", "Efe", "Ömer", "1", "2", "3"]
print(list_eestec[0])

print(list_eestec[1])
```

```
Kayra
Başar
```

Listenin son index'i -1'den başlar ve listenin ilk elemanına doğru azalarak devam eder

```
print(list_eestec[-1]) # -1 index'i listenin son elemanını temsil eder.

print(list_eestec[-3])
```

```
3
1
```

Düzenlemek için çift tıklayın (veya enter tuşuna basın)

```
list_eestec[1:4]
```

```
['Başar', 'Yağız', 'Talha']
```

Listeler değiştirilebilir olduğundan dolayı içindeki verileri rahatlıkla değiştirebiliriz. Bu, listelerin mutable olmasından dolayıdır.

```
list_eestec[0] = "Ahmet"
list_eestec
```

```
['Ahmet', 'Başar', 'Yağız', 'Talha', 'Efe', 'Ömer', 1, 2, 3]
```

Listelerde birden çok elemanı da değiştirebiliriz.

```
list_eestec[1:4] = "EESTEC", "LC", "Istanbul"
list_eestec
```

```
['Ahmet', 'EESTEC', 'LC', 'Istanbul', 'Efe', 'Ömer', 1, 2, 3]
```

Len() komutu listede kaç eleman olduğunu gösterir.

```
len(list_eestec)
```

```
9
```

Listenin sonuna eleman eklemek için .append() komutunu kullanırız.

```
list_eestec
```

```
['Kayra', 'Başar', 'Yağız', 'Talha', 'Efe', 'Ömer', 1, 2, 3]
```

```
list_eestec.append(1773)
```

```
list_eestec
```

```
['Kayra', 'Başar', 'Yağız', 'Talha', 'Efe', 'Ömer', 1, 2, 3, 1773]
```

Listenin sonuna birden çok eleman eklemek için .extend() komutunu kullanırız

```
list_eestec
```

```
['Kayra', 'Başar', 'Yağız', 'Talha', 'Efe', 'Ömer', 1, 2, 3, 1773]
```

```
list_eestec.extend([20, 30, 40])
```

```
list_eestec
```

```
['Kayra', 'Başar', 'Yağız', 'Talha', 'Efe', 'Ömer', 1, 2, 3, 1773, 20, 30, 40]
```

Belirli bir indexe eleman eklemek için .insert() komutunu kullanırız.

```
list_eestec
```

```
['Kayra', 'Başar', 'Yağız', 'Talha', 'Efe', 'Ömer', 1, 2, 3, 1773, 20, 30, 40]
```

```
list_eestec.insert(0, 19)
```

```
list_eestec
```

```
[19, 'Kayra', 'Başar', 'Yağız', 'Talha', 'Efe', 'Ömer', 1, 2, 3]
```

Listeden belirli bir elemanı silmek için .remove() komutunu kullanırız. Eğer silmek istediğimiz eleman listede yoksa error verir.

```
list_eestec
```

```
[19, 'Kayra', 'Başar', 'Yağız', 'Talha', 'Efe', 'Ömer', 1, 2, 3]
```

```
list_eestec.remove(19)
```

```
list_eestec
```

```
['Kayra', 'Başar', 'Yağız', 'Talha', 'Efe', 'Ömer', 1, 2, 3]
```

Listenin belirli indexindeki elemanı silmek için .pop() komutunu kullanırız.

```
list_eestec.pop(3)
```

```
list_eestec
```

```
['Kayra', 'Başar', 'Yağız', 'Efe', 'Ömer', 1, 2, 3]
```

Listedeki bir elemanın kaç kez tekrar ettiğini bulmak için .count() komutunu kullanırız.

```
list_eestec.count(3)
```

```
0
```

```
list_eestec.count("Kayra")
```

```
1
```

Çünkü 3 sayısı listemizde string olarak yer almakta.

Listeyi terse çevirmek için .reverse() komutunu kullanırız.

```
list_eestec
```

```
['Kayra', 'Başar', 'Yağız', 'Efe', 'Ömer', 1, 2, 3]
```

```
list_eestec.reverse()
```

```
list_eestec
```

```
['3', '2', '1', 'Ömer', 'Efe', 'Talha', 'Yağız', 'Başar', 'Kayra']
```

Listedeki elemanları küçükten büyüğe veya alfabetik olarak sıralamak için sorted() ve .sort() komutlarını kullanırız.

Bu iki komut arasındaki fark şudur:

- **sorted() komutu listeyi güncellemez.**
- **.sort() komutu listeyi günceller.**

```
list_eestec
```

```
['3', '2', '1', 'Ömer', 'Efe', 'Talha', 'Yağız', 'Başar', 'Kayra']
```

```
sorted(list_eestec)
```

```
['1', '2', '3', 'Başar', 'Efe', 'Kayra', 'Talha', 'Yağız', 'Ömer']
```

```
list_eestec
```

```
['3', '2', '1', 'Ömer', 'Efe', 'Talha', 'Yağız', 'Başar', 'Kayra']
```

```
list_eestec.sort()
```

```
list_eestec
```

```
['1', '2', '3', 'Başar', 'Efe', 'Kayra', 'Talha', 'Yağız', 'Ömer']
```

Tuple

- Tuple birden çok farklı veri tipini bir arada tutmamızı sağlar
- Tuple'lar listelerin aksine değiştirelemezler yani sabittirler. Buna **immutable** olma özelliği diyoruz.
- Değişmesini istemediğimiz değerler için Tuple kullanmak faydalıdır.
- İçerisinde farklı veri yapıları hatta tuple bile içerebilirler.

```
itu = (1773, 2022)
```

```
itu[0]
```

```
1773
```

```
itu[0] = 12
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-51-9cf1dd9f61b2> in <module>()
----> 1 itu[0] = 12

TypeError: 'tuple' object does not support item assignment

SEARCH STACK OVERFLOW
```

Tuple'lar değiştirilemez yani sabit değerler olduğu için error verdi

```
eestec = (1, 5, 7, "Ahmet", "Kayra", ("Hello", "World!"))
```

```
eestec
```

```
(1, 5, 7, 'Ahmet', 'Kayra', ('Hello', 'World!'))
```

Elemanları değiştirmek istiyorsam:

- Elemanları tuple aracılığıyla rahatlıkla değiştirebilirim.

```
a = 5
```

```
b = 7
```

```
(a, b) = (b, a)
```

```
a
```

```
7
```

```
b
```

```
5
```

Parantez koymasak bile Python programlama dili otomatik olarak tuple olarak algılıyor.

```
c = 1, 3, 5, 7
```

```
c
```

```
(1, 3, 5, 7)
```

```
type(c)
```

```
tuple
```

Bu yüzden değerleri değiştirmek için de parantez kullanmamıza gerek kalmaz

```
q = 3
w = 5
```

```
q, w = w, q
```

```
q
```

```
5
```

```
w
```

```
3
```

Dictionary

- Dictionary hayatımızı birçok alanda kolaylaştırmaktadır. Örneğin;
- Bir araba galerisinde her arabaya ait farklı fiyatlandırmalar var.

- Biz bu fiyatlandırmaları her arabaya farklı değerleri atamak için yeni bir liste oluşturmamız gerekir.
- Hem araba modelleri hem de fiyatlar için 2 farklı liste olacaktır.
- Bu işlemi gerçekleştirmenin kısa yolu dictionary kullanmaktır.

Gelin, size örneklerle açıklayalım.

```
fiyatlar = {"Audi": 10, "Mercedes": 20, "Volkswagen": 30}
```

Audi'nin fiyatını öğrenmek için:

```
fiyatlar["Audi"]
```

```
10
```

Bu işlemi daha da spesifik bir şekilde yazabiliriz.

```
arabalar = {"Audi": {"fiyat":10, "yıl": 2002}, "Mercedes": {"fiyat":20, "yıl": 2008},
```

```
arabalar["Audi"]["fiyat"]
```

```
10
```

```
arabalar["Audi"]["yıl"]
```

```
2002
```

Dictionary key:value sistemi ile çalıştığı için index sorgusu yapamayız. Örneğin;

```
arabalar[0]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-75-dd7b097758cc> in <module>()
----> 1 arabalar[0]

KeyError: 0
```

SEARCH STACK OVERFLOW

Arabalar dictionary'sinde "0" adlı bir key olmadığı için error verdi.

Arabanın fiyatını değiştirmek istersek:

```
arabalar["Audi"]["fiyat"] = arabalar["Audi"]["fiyat"] + 5
```

```
arabalar["Audi"]["fiyat"]
```

```
15
```

```
arabalar
```

```
{'Audi': {'fiyat': 15, 'yıl': 2002},
 'Mercedes': {'fiyat': 20, 'yıl': 2008},
 'Volkswagen': {'fiyat': 30, 'yıl': 2010}}
```

len() komutu bize dictionary'deki key sayısını verir.

```
len(arabalar)
```

```
3
```

Bir dictionary'e eleman eklemek çok basittir.

```
fiyatlar = {"Audi": 10, "Mercedes": 20, "Volkswagen": 30}
```

```
fiyatlar["TOGG"] = 50
```

```
fiyatlar
```

```
{'Audi': 10, 'Mercedes': 20, 'TOGG': 50, 'Volkswagen': 30}
```

Bir dictionary'den eleman silmek için del komutu kullanılmalıdır.

```
del fiyatlar["TOGG"]
```

```
fiyatlar
```

```
{'Audi': 10, 'Mercedes': 20, 'Volkswagen': 30}
```

Bir dictionary'de sadece immutable yapıdaki veriler key olabilir. Eğer mutable bir değer verirsek error verir.

```
d = {[1, 2]: 3}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-89-bf7f06622b3f> in <module>()
----> 1 d = {[1, 2]: 3}

TypeError: unhashable type: 'list'
```

SEARCH STACK OVERFLOW

Boş bir dictionary yaratarak içine veriler ekleyebiliriz.

```
d1 = {}
```

```
d1["Başar"] = "Çelebi"
```

```
d1
```

```
{'Başar': 'Çelebi'}
```

Bir elemanın key değerleri içerisinde olup olmadığını sorgulamak için in komutu kullanırız.

```
"Başar" in d1
```

```
True
```

```
"Volvo" in fiyatlar
```

```
False
```