PYTHON

- KOLEKSİYONLAR -

Liste

- list()
- Aynı ya da farklı türde çok sayıda elemanı bir arada tutan
- Eleman ekleme, silme, güncelleme işlemlerine imkan veren koleksiyonlardır.

Liste Tanımlama

- Liste tanımlanırken elemanlar köşeli parantez içinde ve aralarına virgül konarak belirtilir.
- Farklı liste tanımlama biçimleri:

```
liste=list()                    #boş liste tanımlama
liste=[]                        #boş liste tanımlamanın farklı yolu
liste=([1,2,3])                 #içinde 1,2,3 sayıları olan liste tanımlama
liste=[1,2,3]                   #üstteki tanımlamanın farklı yolu
liste=["python","java","csharp"] #elemanları string olan liste
liste=[1,"iki",3.5]             #farklı türde elemanlar barındıran liste
```

Liste Tanımlama

- Listedeki her bir eleman bilgisayarın belleğinde ayrı bölgelerde sıralı olarak yer alır.
- Liste değişkeni elemanların bulunduğu bölgenin başlangıç adresine işaret eder.
- liste = [10, 20, 30]

	Başlangıç bellek adresi	Sıra numarası (indis-index)	Eleman değeri
liste	140736706802528	0	10
		1	20
		2	30

Liste Elemanlarına Erişim

- Listeye eklenen her bir elemana otomatik olarak sıfırdan başlayan bir sıra numarası verilir.
- Liste elemanlarına erişirken elemanların sıra numarası (indis) değerleri kullanılır.

print(liste[0]) => 10
print(liste[2]) => 30

	Sıra numarası (indis-index)	Eleman değeri
liste	0	10
	1	20
	2	30

Liste Eleman Sayısı

- Listenin eleman sayısını bulmak için len() komutu kullanılır.
- len – length (uzunluk)

```
liste = [10, 20, 30]
elemansayisi=len(liste)
print("Listenin eleman sayısı:", elemansayisi)
```

Listenin eleman sayısı:3

Liste Eleman Sayısı

- Listeye eklenen elemanların indis değerleri sıfırdan başladığı için son eklenen elemanın indis değeri, eleman sayısının 1 eksiğidir.
- Yani 3 elemanlı bir dizinin son elemanının indisi 2'dir.
- Bu durumda son elemana şu şekilde de erişmek mümkündür:

son=liste[len(liste)-1] => 30

3-1=2
liste[2]

	İndis	Eleman değeri
liste	0	10
	1	20
	2	30

Listeyi Ekrana Yazma

- Her bir elemanı bağımsız olarak yazmak için döngülerden yararlanılır:

```
liste = [10, 20, 30]
for x in liste:
    print(x)
```

```
10
20
30
```

- x değişkeni listedeki her bir elemanı sıra ile alıp getirir.
- x , döngüye ait bir değişken olup liste o anki liste elemanının bir kopyasını tutar.

Listeyi Ekrana Yazma

```
liste = [10, 20, 30]
for x in liste:
    x*=2
print(liste)
```

10, 20, 30

- Döngü değişkeni liste elemanlarının kopyasını taşıdığı için x değişkeni üzerinde yapılan değişiklikler orijinal liste elemanlarını etkilemez.

Listeye Eleman İndisi ile Erişim

```
liste = [10, 20, 30]
for i in range(0,len(liste)):
    liste[i]*=2
print(liste)
```

20, 40, 60

- liste[i] üzerinde yapılacak değişiklik direkt orijinal eleman değerini değiştirir.

Liste Parçalama

- Liste içerisinde belirli bir aralıkta yer alan elemanları almak için parçalama işlemi uygulanır.
- Liste parçalama işlemi genel anlamda şu şekilde yapılır:

liste[başlangıç:bitiş:artış]

- Liste içerisinde belirli bir başlangıç indisinden itibaren, bitiş indisine kadar, belirlenen artış miktarı ile elemanlar alınır.
- Başlangıç indisi aralığa dahil, bitiş indisi hariçtir.

Liste Parçalama

liste = [10, 20, 30, 40, 50, 60]
liste[1:5:2] #1, 3. elemanı alır. 5. eleman aralık dışında

- Listenin 1 numaralı elemanından itibaren (20), 5 numaralı elemanına (60) kadar 2'şer atlayarak elemanlar alınmaktadır.
- Bu parçalama sonucu elde edilen liste:

[20,40]

Liste Parçalama

- Liste parçalama işlemi range() fonksiyonunun kullanımına benzer. Başlangıç indisi belirtilmezse 0 kabul edilir. Bitiş indisi belirtilmezse listenin sonuna kadar gider. Artış miktarı verilmezse 1 olarak kabul edilir:

  liste[2:5] #2. elemandan 5. elemana kadar (5 hariç) 1'er atlayarak
  liste[2:] #2. elemandan liste sonuna kadar 1'er atlayarak
  liste[:5] #listenin ilk elemanından (0. indis), 5. elemana kadar (5 hariç)

Liste Elemanlarını Değiştirme

- Değiştirilecek indisteki elemana = ile yeni değer atanır.

```
liste = [10, 20, 30, 40, 50, 60]
liste[2] = 25
print(liste)
```

```
[10, 20, 25, 40, 50, 60]
```

Liste Elemanlarını Değiştirme

- Belirli bir aralık seçilerek toplu değer ataması da yapılabilir.

```
liste = [10, 20, 30, 40, 50, 60]
liste[2:5] = [25, 26, 27]
print(liste)
```

```
[10, 20, 25, 26, 27, 60]
```

Liste Birleştirme

- Birden çok listenin elemanları + operatörü ile yan yana getirilip tek listeye dönüştürülür.

```
liste1 = [10, 20, 30]
liste2 = [40, 50, 60]
liste3 = liste1+liste2
print(liste3)
```

```
[10, 20, 30, 40, 50, 60]
```

Listeye Eleman Ekleme

- Bir listenin sonuna yeni eleman eklemek için append() komutu kullanılır:

```
liste = [10, 20, 30]
liste.append(40)
print(liste)
```

```
[10, 20, 30, 40]
```

Listeye Eleman Ekleme

- Bir listede belirtilen indise yeni eleman eklemek için insert() komutu kullanılır:

```
liste = [10, 20, 30]
liste.insert(2, 25)        #2. indise 25 değerini yerleştir.
print(liste)
```

```
[10, 20, 25, 30]
```

Listeden Eleman Silme

- Listeden eleman silmek için remove() komutu kullanılır.

```
liste = [10, 20, 30]
liste.remove(20)          #Listeden değeri 20 olan elemanı siler.
print(liste)
```

[10, 30]

Listeden Eleman Silme

- Listenin belirli bir elemanı del komutu ile de şu şekilde silinebilir:

```
liste = [10, 20, 30]
del liste[2]        #listenin 2 indisli elemanını (30) sil.
print(liste)
```

```
[10, 20]
```

Listeden Eleman Silme

- Listede belirli bir aralıkta eleman şu şekilde silinebilir:

```
liste = [10, 20, 30, 40, 50, 60]
del liste[2:5]      #listenin 2 indisli elemandan 5 indisli elemana kadar sil.
print(liste)
```

```
[10, 20, 60]
```

Listeyi Silme

- del komutu bir listeyi bellekten tümüyle silmek için de kullanılır:

```
liste = [10, 20, 30]
del liste   #listeyi bellekten tümüyle sil
print(liste)
```

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

Listede Elemanın Varlığını Bulma

- Aitlik operatörleri in ve not in ile herhangi bir elemanın listedeki varlık ya da yokluk durumu boolean olarak alınabilir:

```
liste = [10, 20, 30]
m1 = 20 in liste          #listede 20 değerine sahip eleman var mı?
m2 = 50 not in liste      #listede 50 değerine sahip eleman yok mu?
print(m1)
print(m2)
```

```
True      #listede 20 değerine sahip eleman var
True      #listede 50 değerine sahip eleman yok
```

Listede Elemanın İndisini Bulma

- Liste içerisinde herhangi bir elemanın hangi indiste yer aldığı index() komutu ile elde edilir.

```
liste = [10, 20, 30]
indis = liste.index(20)  #20 değerine sahip eleman hangi indiste
print(indis)
```

```
1                #listede 20 değerine sahip eleman 1. indiste yer alıyor
```

Listede Elemanın İndisini Bulma

- index() komutu ile belirli bir indisten başlayarak da arama yapılabilir.

```
liste = [10, 20, 30, 20, 40]
indis = liste.index(20, 2)        #2. indisten itibaren 20 nerede?
print(indis)
```

```
3     #2. indis ve sonrasında 20 değeri 3. indiste yer alıyor
```

Listede Elemanın İndisini Bulma

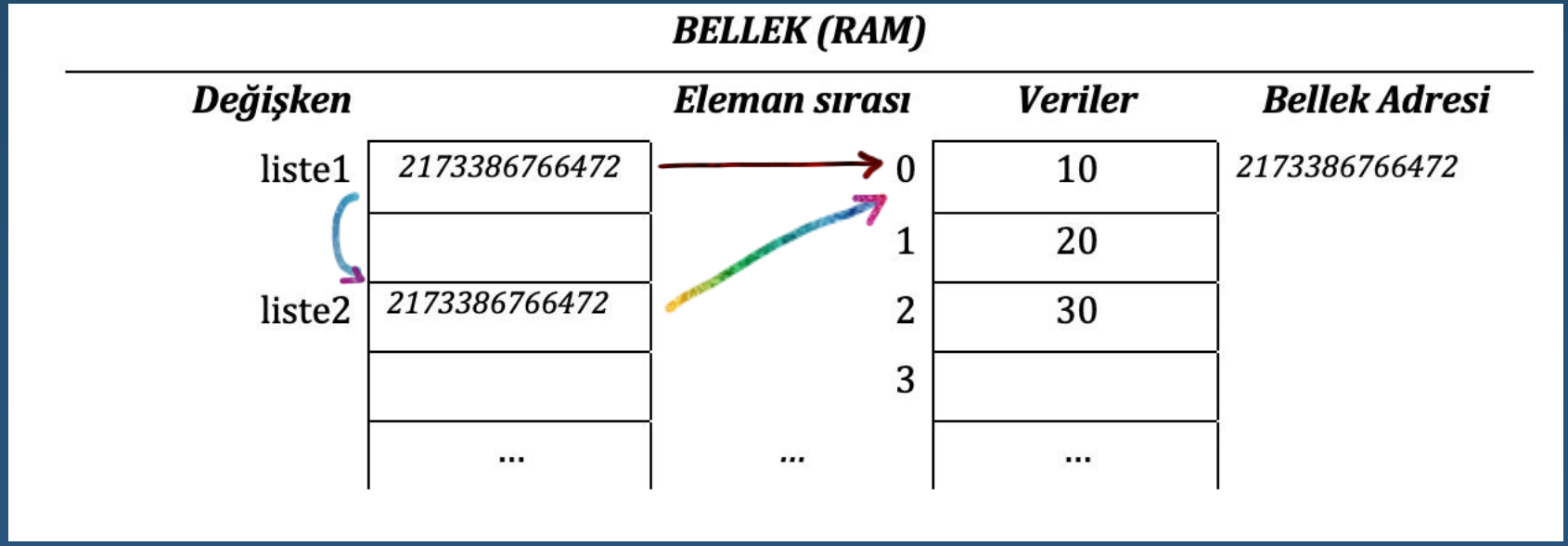
- index() aranan elemanı bulamazsa hata üretir.

```
liste = [10, 20, 30]
indis = liste.index(50)  #listede 50 nerede?
print(indis)
```

```
ValueError: 50 is not in list     #listede 50 yok
```

Liste Kopyalama

liste1 = [10,20,30]
liste2 = liste1



Liste Kopyalama

- liste1'in liste2'ye atanması demek, liste1 içerisinde yer alan bellek adresinin liste2'ye atanması demektir.
- liste1 ve liste2'nin aynı bellek adresini gösterdiğini id() operatörü ile görebiliriz.

```
print(id(liste1))
print(id(liste2))
```

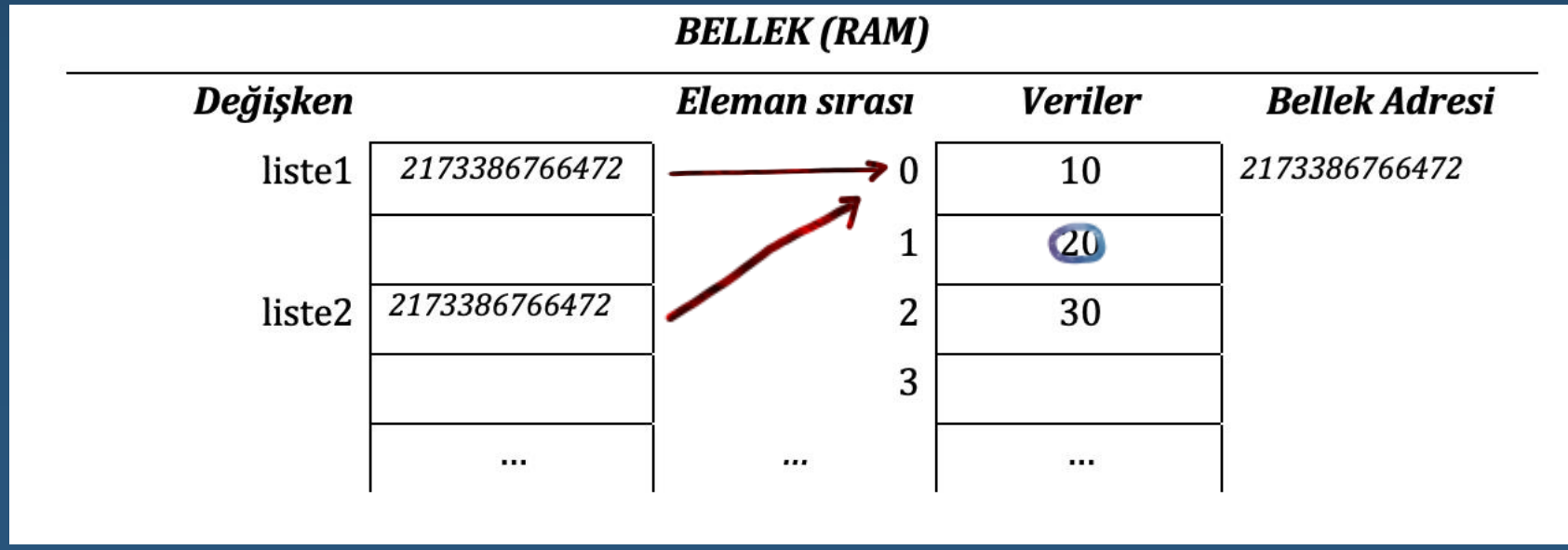
```
2173386766472
2173386766472
```

Liste Kopyalama

- Birbirine kopyalanan iki liste bellekte aynı yere işaret ettiğinden dolayı listelerden biri üzerinde yapılan değişiklik otomatik olarak diğerini de etkiler.

```
liste1 = [10,20,30]
liste2 = liste1
liste2[1]=15
print(liste1[1])
```

15



Liste Kopyalama

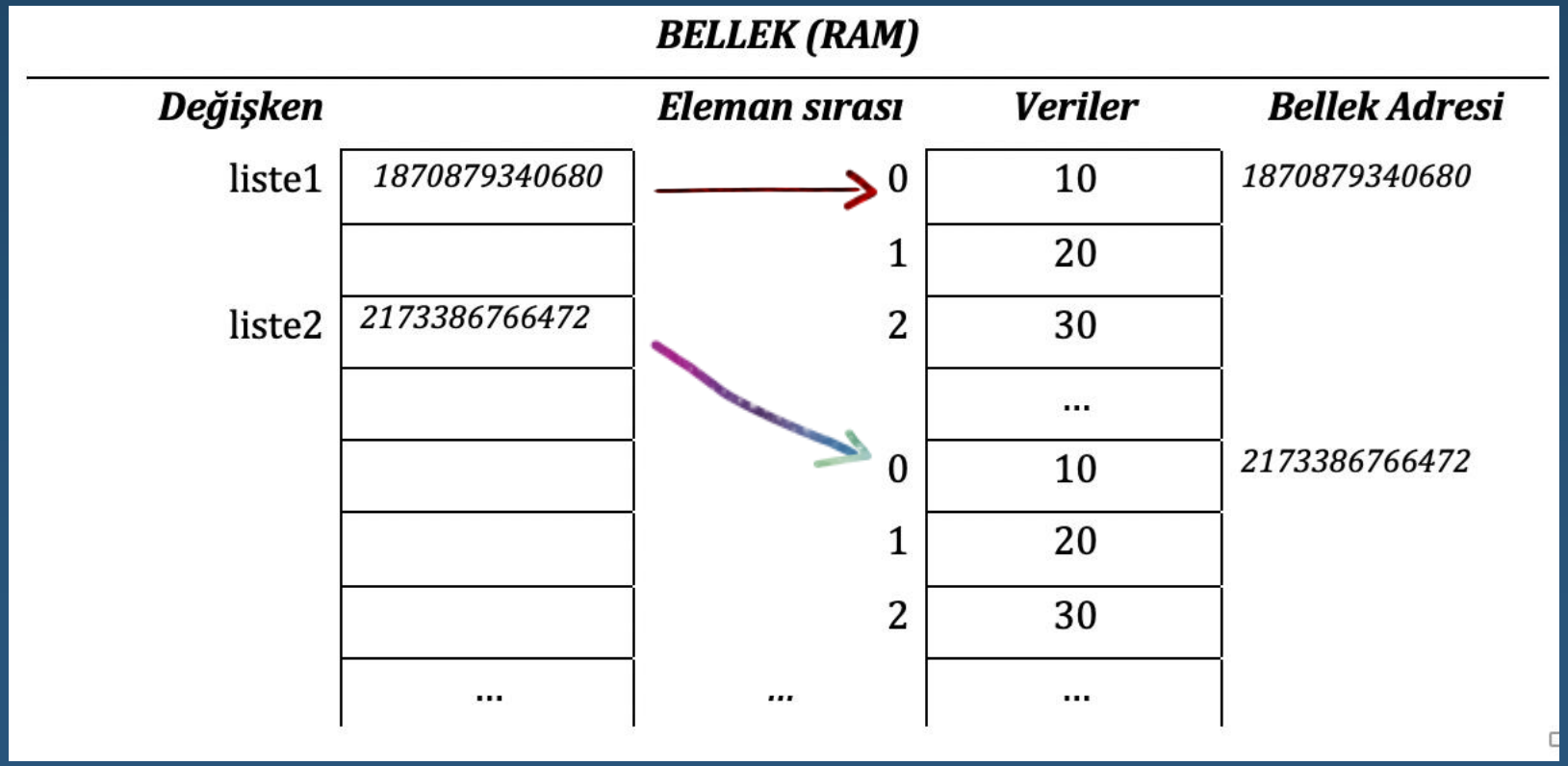
- Görüldüğü üzere bir listeyi = operatörü ile başka bir listeye atamak elemanlarının bir kopyasını başka bir listeye atamak anlamına gelmez.
- Bir listedeki elemanların her birinin birer kopyasını oluşturularak başka bir listeye atamak için elemanların tek tek kopyalanması gerekir.
- Bu işlem şu şekilde yapılabilir:

```
liste1 = [10,20,30]
liste2 = liste1[:]
```

Liste Kopyalama

- Bu durumda liste1'in her bir elemanı tek tek liste2 içine kopyalanır.

liste1 = [10,20,30]
liste2 = liste1[:]



Liste Kopyalama

- Bu durumda bir liste üzerinde yapılan değişiklik diğerini etkilemez.
- Bir listenin tümünü değil yalnız belli bir kısmını da kopyalamak mümkündür:

```
liste1 = [10,20,30,40,50,60]
liste2 = liste1[2:5]
print(liste2)
```

```
[30, 40, 50]
```

Listenin En Büyük ve En Küçük Elemanı

- Bir listenin en büyük ve en küçük elemanlarını bulmak için max() ve min() komutları kullanılır.

```
liste=[30,50,20,40,10]
a=max(liste)
b=min(liste)
print("Listenin en büyük elemanı:",a,", en küçük elemanı:",b)
```

Listenin en büyük elemanı:50, en küçük elemanı:10

Listenin En Büyük ve En Küçük Elemanı

- Karakter dizilerinde alfabetik olarak en büyük ya da en küçük elemanı verir.

```
kelime=["zürafa","aslan","fil"]
a=max(kelime)
b=min(kelime)
print("Listenin en büyük elemanı:",a,", en küçük elemanı:",b)
```

Listenin en büyük elemanı:zürafa, en küçük elemanı:aslan

Liste Elemanlarının Toplamını Bulma

- Sayısal listelerde elemanların toplamı sum() komutu ile bulunur.

```
liste=[10,20,30]
toplam=sum(liste)
print(toplam)
```

```
60
```

Liste Üreteçleri

- Belirli bir düzende otomatik listeler oluşturmak için liste üreteçleri kullanılır.
- Üreteçlerin kullanımında for döngüsüne aşina olmanız gerekir.

```
liste=[a for a in range(5)]
print(liste)
```

```
[0,1,2,3,4]
```

Liste Üreteçleri

```
liste=[a+1 for a in range(5)]
print(liste)
```

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

```
liste=[a for a in range(11) if a%2==0]
print(liste)
```

```
[0,2,4,6,8,10]
```

Demet

- Demetler (tuples) listelere benzerdir.
- Ancak demetler oluşturulduktan sonra üzerinde ekleme, silme, güncelleme işlemleri yapılamaz.
- Demetler sadece okunabilir listeler olarak düşünülebilir.

Demet

- Bir demet aşağıdaki yollardan biri ile tanımlanabilir:

```
demet = tuple()      #boş demet tanımlaması
demet = ()           #boş demet tanımlamanın farklı yolu
demet = tuple([1,2,3]) #içerisinde 1,2,3 değerleri olan demet tanımlama
demet = (1,2,3) #üstteki tanımlamanın farklı yolu
demet = ("python","java","assembly") #string demeti
demet = (1,2,"kiraz", 3.22) #farklı türde elemanlardan oluşan demet
```

Demet Elemanlarına Erişim

- Listelerde olduğu gibi demet içerisindeki her bir elemanın 0'dan başlayan bir indis numarası vardır.
- Demet içindeki bir elemana erişirken [] içerisinde indis numarası yazılır.

```
demet=(10,20,30)
a=demet[1]
print(a)
```

```
20
```

Demet Eleman Sayısını Bulma

- Demetin eleman sayısı len() komutu ile aşağıdaki gibi bulunur:

```
demet=(10,20,30)
a=len(demet)
print("Demetin eleman sayısı:",a)
```

Demetin eleman sayısı:3

Demeti Ekrana Yazdırma

- Demet aşağıdaki şekilde tümüyle ekrana yazılır:

```
demet=(10,20,30)
print(demet)
```

```
(10,20,30)
```

Demeti Ekrana Yazdırma

- Demetin her bir elemanı döngü yardımıyla da listelenebilir:

```
demet=(10,20,30)
for i in demet:
    print(i)
```

```
10
20
30
```

```
demet=(10,20,30)
for i in range(len(demet)):
    print(demet[i])
```

```
10
20
30
```

Demet Parçalama

- Demetin içerisinden belirli bir aralıktaki elemanları almak için, listelerde olduğu gibi şu yöntem kullanılır:

  demet[başlangıç:bitiş:artış]

- Başlangıç aralığa dahil, bitiş hariçtir.
- Başlangıç değeri verilmezse 0, bitiş değeri verilmezse demetin sonu, artış miktarı verilmezse 1 varsayılan değerdir.

Demet Parçalama

- Demetin içerisinden belirli bir aralıktaki elemanları almak için, listelerde olduğu gibi şu yöntem kullanılır:

```
demet=(10,20,30,40,50)
a=demet[1:4]
print(a)
```

```
(20,30,40)
```

Demette Eleman Varlığını Bulma

- Demet içerisinde herhangi bir elamanın var olup olmadığını bulmak için in ve not in operatörleri kullanılır:

```
demet=(10,20,30,40,50)
print(30 in demet)          # demet içinde 30 değeri var mı?
print(70 not in demet)      # demet içinde 70 değeri yok mu?
```

```
True
True
```

Demet İçindeki Elemanın İndisini Bulma

- Demet içerisinde yer alan bir elemanın indisi index komutu ile bulunur.
- Eleman birden çok kez varsa ilk indis değerini verir.
- Eleman yoksa hata üretir.

```
demet=(10,20,30,40,50)
indis=demet.index(30)          # demet içinde 30 değeri nerede?
print(indis)
```

```
2
```

Demet İçindeki Eleman Sayısını Bulma

- Bir elemanın demet içinde kaç kez yer aldığı count() komutu ile bulunur:

```
demet=(10,20,30,20,20)
adet=demet.count(20)        # demet içinde 20 kaç kez var?
print(adet)
```

```
3
```

Demetin En Büyük ve En Küçük Elemanı

- Demetin en büyük elemanı max() komutu ile, en küçük elemanı min() komutu ile bulunur.

```
demet=(10,20,30,40,50)
a=max(demet)
b=min(demet)
print("Demetin en büyük elemanı:",a,", en küçük elemanı:",b)
```

Demetin en büyük elemanı:50, en küçük elemanı:10

Demet Elemanlarının Toplamı

- Demet elemanlarının toplamını bulmak için sum() komutu kullanılır:

```
demet=(10,20,30)
a=sum(demet)
print("Demet elemanlarının toplamı:",a)
```

Demetin elemanlarının toplamı:60

Sözlük

- Liste ya da demet gibi birden çok elemanı barındıran koleksiyonlarda her bir elemanın eşsiz bir indis numarası vardır.
- Bu eşsiz indis numaraları elemanlara erişimimizi sağlayan birer anahtardır.

liste=[10,20,30]
liste[1] *-> 20*

Sözlük

- Koleksiyon içerisinde aynı değere sahip birden çok eleman olsa bile her birinin indisi farklıdır.

İndis Numarası	0	1	2	3	…
Eleman Değeri	10	20	10	40	…

- Listede iki tane 10 değeri olsa da ilkinin indisi 0, diğerininki 2’dir.

Sözlük

- Koleksiyon tanımlarken indis numaraları yerine kendi belirlediğimiz anahtarları kullanmak istersek sözlük tanımlamalarını kullanırız.
- Sözlükler elemanları bizim belirlediğimiz anahtar ve değer (key-value) ikilileri şeklinde saklarlar.
- Bu şekilde herhangi bir elemana erişirken kendi belirlediğimiz anahtarı kullanırız.

Sözlük

- Örneğin İngilizce-Türkçe kelimelerden oluşan sözlük programı yapmak istersek kelimenin Türkçesi değer, buna erişirken kullandığımız İngilizcesi ise anahtar olur.
- Öğrenci bilgilerini saklayan bir programda öğrenci numarası anahtar, adı değer olabilir.

Sözlük

- Sözlükler Python dilindeki anahtar-değer(key-value) ikilileri saklayan koleksiyonlardır.
- Sözlükler üzerinde ekleme, silme, güncelleme gibi düzenleme işlemleri yapılabilir.

Sözlük Tanımlama

- Bir sözlüğün genel tanımlama biçimi şu şekildedir:

  sozluk = {anahtar1:değer1, anahtar2:değer2, ….. }

- Değerleri sonradan belirlenmek üzere boş bir sözlük şöyle tanımlanabilir:

  sozluk={}

Sözlük Tanımlama

- İngilizce-Türkçe kelimeleri tutacak bir sözlük şöyle tanımlanabilir:

sozluk = {"book":"kitap","apple":"elma","pen":"kalem"}

Anahtar	book	apple	pen
Değer	kitap	elma	kalem

Sözlük Elemanlarına Erişim

- Sözlükteki herhangi bir değere erişmek için ilgili anahtar kullanılır.
  sozluk = {"book":"kitap","apple":"elma","pen":"kalem"}

Anahtar	book	apple	pen
Değer	kitap	elma	kalem

kelime =sozluk["apple"] ➡ *elma*

Sözlük Elemanlarına Erişim

- Sözlükte olmayan bir anahtar kullanılırsa hata alınır.
  sozluk = {"book":"kitap","apple":"elma","pen":"kalem"}

Anahtar	book	apple	pen
Değer	kitap	elma	kalem

kelime =sozluk["computer"]

KeyError: 'computer'

Sözlüğe Eleman Ekleme

- Var olan sözlüğe eleman eklemek için yeni anahtar tanımlanıp değer atanır.

```
sozluk = {"book":"kitap","apple":"elma","pen":"kalem"}
sozluk["computer"] = "bilgisayar"
print(sozluk)
```

```
{'book': 'kitap', 'pen': 'kalem', 'apple': 'elma', 'computer': 'bilgisayar'}
```

Sözlük Elemanını Değiştirme

- Mevcut sözlük elemanını değiştirmek için ilgili anahtara yeni değer atanır.

```
sozluk = {"book":"kitap","apple":"elma","pen":"kalem"}
sozluk["pen"] = "dolma kalem"
print(sozluk)
```

{'book': 'kitap', 'pen': ‘dolma kalem', 'apple': 'elma', 'computer': 'bilgisayar'}

Sözlükten Eleman Silme

- Sözlükten eleman silmek için del komutu kullanılır.

```
sozluk = {"book":"kitap","apple":"elma","pen":"kalem"}
del sozluk["pen"]
print(sozluk)
```

```
{'book': 'kitap', 'apple': 'elma', 'computer': 'bilgisayar'}
```

Sözlüğü Bellekten Silme

- Sözlüğü bellekten silmek için del komutu kullanılır.

```
sozluk = {"kitap":50,"kalem":10}
del sozluk
print(sozluk)
```

NameError: name ‘sozluk' is not defined

Sözlükten Eleman Silme

- Sözlükteki tüm elemanları silmek için clear() komutu kullanılır.

```
sozluk = {"book":"kitap","apple":"elma","pen":"kalem"}
sozluk.clear()
print(sozluk)
```

```
{ }
```

Sözlük Elemanlarını Listeleme

- Sözlükteki elemanlar for döngüsü yardımıyla şu şekilde listelenir.

```
sozluk = {"book":"kitap","apple":"elma","pen":"kalem"}
for k in sozluk:
    print("İngilizcesi:",k," Türkçesi:",sozluk[k])
```

```
İngilizcesi: book Türkçesi: kitap
İngilizcesi: apple Türkçesi: elma
İngilizcesi: pen Türkçesi: kalem
```

Sözlük Elemanlarını Listeleme

- Sözlükteki anahtar ve değerleri aynı anda listelemek için item() komutu kullanılır.

```
sozluk = {"book":"kitap","apple":"elma","pen":"kalem"}
print(sozluk.items())
```

```
dict_items([('book', 'kitap'), ('apple', 'elma'), ('pen', 'kalem')])
```

Sözlük Elemanlarını Listeleme

- Sözlükteki anahtarları almak için keys() komutu kullanılır.

```
sozluk = {"book":"kitap","apple":"elma","pen":"kalem"}
for k in sozluk.keys():
    print(k)
```

```
book
apple
pen
```

Sözlük Elemanlarını Listeleme

- Sözlükteki değerleri almak için values() komutu kullanılır.

```
sozluk = {"book":"kitap","apple":"elma","pen":"kalem"}
for v in sozluk.values():
    print(v)
```

```
kitap
elma
kalem
```

Sözlükteki Eleman Sayısı

- Sözlükteki her anahtar değer çifti tek bir elemandır. Eleman sayısını bulmak için len() komutu kullanılır.

```
sozluk = {"book":"kitap","apple":"elma","pen":"kalem"}
print(len(sozluk))
```

```
3
```

Sözlükte Anahtar Varlığını Kontrol Etme

- Herhangi bir anahtarın sözlükte var olup olmadığını kontrol etmek için in ve not in komutları kullanılır.

  sozluk = {"book":"kitap","apple":"elma","pen":"kalem"}

  print("apple" in sozluk) => apple sözlükte var mı? - True

  print("orange" not in sozluk) => orange sözlükte yok mu? - True

Sözlüklerin Eşitliğini Kontrol Etme

- Sözlüklerin eşit olması anahtar-değer çiftlerinin eşit olması demektir.
- İki sözlüğün eşitliği == ile, farklılığı != ile kontrol edilir.

sozluk1 = {"kitap":50,"kalem":10}
sozluk2 = {"kalem":10,"kitap":50}

print(sozluk1==sozluk2) => True

- Sözlük elemanların yerlerinin değişik olması eşitliği engellemez. Elemanların eşit olması yeterlidir.

Sözlük Güncelleme

- Bir sözlüğü daha sonra oluşturulan başka bir sözlüğün değerlerine göre güncelleme işlemi şöyle yapılır.

```
sozluk1 = {"kitap":50,"kalem":10}
sozluk2 = {"kitap":45,"kalem":10,"silgi":8}
sozluk1.update(sozluk2)
```

- sozluk1 içindeki değerler sozluk2'deki değerler ile güncellenir. kitap 45, kalem 10 değerine güncellenip, silgi anahtarı ve değeri 8 eklenir.

Sözlük Kopyalama

- Sözlükler de listeler gibi birer referans tipidir. İki sözlük = ile birbirine atanırsa bellek adresleri eşitlenir ve aynı bölgeye işaret ederler.

  sozluk = {"kitap":50,"kalem":10}

  sozluk2=sozluk

  Bu durumda sözlüklerden biri üzerinde yapılan değişiklik diğerini etkiler.

  sozluk2["kitap"]=45

  print(sozluk["kitap"])

  45

Sözlük Kopyalama

- Sözlüğün içeriğini kopyalamak için copy() komutu kullanılır.

```
sozluk = {"kitap":50,"kalem":10}
sozluk2=sozluk.copy()
```

Bu durumda sözlüklerin değerleri kopyalanır ve aynı bellek bölgesini göstermez.

```
sozluk2["kitap"]=45
print(sozluk["kitap"])
```

```
50
```