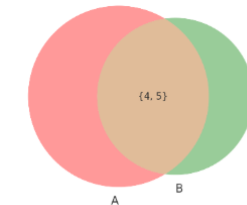
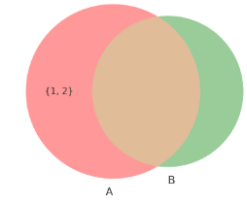


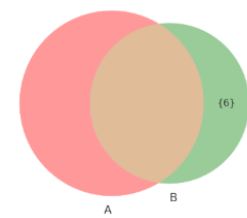
	LIST(LİSTE)	TUPLE (DEMET)	DICTIONARY (SÖZLÜK)	SET (KÜME)
ÖZELLİKLER	Değiştirilebilir Sıralı Kapsayıcı list	DEĞİŞTİRİLEMEZ Sıralı Kapsayıcı tuple	Değiştirilebilir SIRASIZ Kapsayıcı dict	Değiştirilebilir SIRASIZ-EŞSİZ Kapsayıcı set
TYPE OLUŞTURMA	<pre>N=[90,"d",8.7] M=[1,4,N] M #[1, 4, [90, 'd', 8.7]]</pre>	<pre>T= ("ali",4,7.2,[4,5,6]) ('ali',4,7.2,[4,5,6]) T= ("ali",) #tek elemanlıda yanına virgül konur</pre>	<pre>S={15:"Burdur",13:"Bitlis",34:"İstambul"} #Ç {15: 'Burdur', 13: 'Bitlis', 34: 'İstanbul'} S1={15:{100:"Burdur",200:"Bucak",},13:{130:"Bitlis",230:"Tatvan"},34:{400:"İstanbul",440:"Esenyurt"}} len(S) # Çıktı: 3</pre>	<pre>L=[1,"a",123] S=set(L) S #{1, 123, 'a'} T= ("a","ali") S=set(T) S #{'a', 'ali'} ad="naaciye" S=set(ad) S #{'a', 'c', 'e', 'i', 'n', 'y'} #her eleman tekilleştiriliyor</pre>
ELEMAN SAYISI	<pre>len(N) N.count(90) #kaç 90 var sayar</pre>	<pre>len(T) # Çıktı: 4</pre>		<pre>ad="naaciye" S=set(ad) len(S) #6</pre>
ELEMANA ERİŞME	<pre>N[0] #90 N[0:2] #[90, 'd'] N[:2] #[90, 'd'] N[2:] #[8.7] M[2][1] # 'd'</pre>	<pre>T= ("ali",4,7.2,[4,5,6]) T[0] # "ali" T[0:2]# ("ali", 4) T[:2] # ("ali", 4) T[2:]#(7.2,[4,5, 6]) T[3][1] #5</pre>	<pre>S[15] #'Burdur' S1[15][100] #'Burdur'</pre>	<pre>#SIRASIZ olduğu için doğrudan indeksle erişim yapılamaz.# Ancak elemanlar döngü ile işlenebilir. for eleman in S: print(eleman, end=" ") # Set'i listeye çevirerek indeksle erişim S_liste = list(S) print(S_liste) # Çıktı: ['n', 'a', 'c', 'i', 'y', 'e'] (sıra değişebilir) print(S_liste[0]) # Örnek: 'n' (sıra değişebilir)</pre>
SİLME	<pre>del N[0] # 0. indeks silinir. del N # Tüm liste sil N.remove(90) #Ç: ['d', 8.7] N.pop(0)# 0. İndeksi siler</pre>	<pre>del T[0] # Hata verir, çünkü tuple elemanları silinemez. del T # Tuple'ın kendisi silinebilir, bu tüm nesneyi bellekten kaldırır. T[3].remove(4) # Mümkün, çünkü T[3] bir liste. Bu işlem liste içindeki '4' elemanını kaldırır. T[3].pop(0) # Mümkün, çünkü T[3] bir liste. Bu işlem listenin ilk elemanını (şu anki haliyle) kaldırır.</pre>	<pre># 1. Belirli bir anahtarı silme del S[13] # Ç: {15: "Burdur", 34: "İstanbul"} bitlis = S.pop(13) # 13 anahtarı ve değeri silinir, değeri döner son silinen = S.popitem() # Sözlükte kalan son çift silinir print(S) # Ç:{15: 'Burdur', 13: 'Bitlis'} S.clear() # Tüm elemanları temizler del S # Sözlük tamamen silinir # print(S) # Hata verir: NameError: name 'S' is not defined</pre>	<pre>S.discard('a') S.remove('a') # 'a' elemanını siler, eleman yoksa hata vermez # Ç: {'n', 'c', 'i', 'y', 'e'} #Set'in tamamını temizleme S.clear() print(S) # Çıktı: set()</pre>
EKRAN YAZDIRMA	<pre>print(N) #[90, 'd', 8.7]</pre>	<pre>print(T) ('ali', 4, 7.2, [4, 5, 6])</pre>	<pre>print(S) #{15: 'Burdur', 13: 'Bitlis', 34: 'İstanbul'}</pre>	<pre>print(S) #{'a', 'ali'}</pre>
ELEMANLARI DEĞİŞTİRME - YENİ ELEMAN EKLE	<pre>N[0:2]=1,1 #[1, 1, 8.7] N=N+["k"] N.append(6) #[90, 'd', 8.7, 'k'] N.insert(0,6)# 0. İndekse 6 elemanını ekledi [6,90,'d',8.7] N.insert(len(N),"k") #son indekse eleman ekledi N.reverse() #ters sıralar M=N.copy() #kopyalar N.index(90) #90 bilgisi hangi indekse onu döner N.sort() #aynı tür verileri sıralar. N.clear() #listeyi temizler N.extend([1,2,3]) #yeni liste ile birleştirir.</pre>	<pre>T[3][1] = 10 # Mümkün # Çıktı: ('ali', 4, 7.2, [4, 10, 6])t tuple içindeki liste değiştirilebilir T[0] = "ahmet" # Hata verir: TypeError: 'tuple' object does not support item assignment</pre>	<pre>S[6]="Ankara" #{15: 'Burdur', 13: 'Bitlis', 34: 'İstanbul', 6: 'Ankara'} #key değerın aynısı yoksa ekleme-varsa veriye erişme #sözel değerlerde key olabilir.</pre>	<pre>#Eleman değiştirme (önce eski elemanı kaldır, sonra yenisini ekle) S=set(T) S.add("b") S #{'a', 'ali', 'b'} S.update([8,"l"]) #birden fazla eleman ekler S #{8, 'a', 'ali', 'l'}</pre>
			<pre>l=[1] t=(1,) s={1,2,3} s[1]=4 #bunu kabul etmez. liste değiştirilebilir. key sabit veri olmalı s[t]=8 #bunu kabul eder. tuple değiştirilemez. key sabit olur</pre>	<pre>#A - B: #A.difference(B): A'nın, B'de bulunmayan elemanlarını döndürür. (B'de olmayan elemanlarını) döndürür. #A ^ B: A ve B'nin her ikisinde de bulunmayan (sadece birinde olan) elemanları döndürür. #A & B: #A.intersection(B): A ile B'nin kesişimi (her ikisinde de bulunan elemanlar) döner. #A B: #A.union(B): A ile B'nin birleşimi (her iki setteki tüm elemanlar) döner.</pre>

```
# Örnek setler
A = {1, 2, 4, 5}
B = {4, 5, 6}
# (A - B): A'nın B'de olmayan elemanları
fark1 = A - B
fark3 = A.difference(B)
# Çıktı: {1, 2}
# Kesişim (A ∩ B): Hem A'da hem B'de olan elemanlar
kesisim = A & B
kesisim_metod = A.intersection(B) # Çıktı: {4, 5}
```

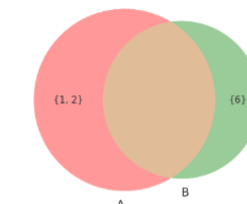
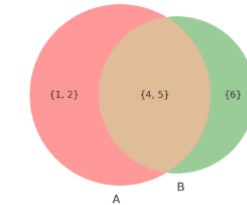


```
# Symmetric Difference (A ^ B): Her iki sette de bulunmayan elemanlar
fark5 = A ^ B # Çıktı: {1, 2, 6}
```

```
# B - A): B'nin A'da olmayan elemanları
fark2 = B - A
fark4 = B.difference(A) # Çıktı: {6}
```



```
# Birleşim (A ∪ B): A ve B'deki tüm elemanlar
birlesim = A | B
birlesim_metod = A.union(B) # Çıktı: {1, 2, 4, 5, 6}
```



HAZIRLARYAN ÖĞR. GÖR. NACİYE MACİT SEZİKLİ