pythonda liste, demet, sözlük ve küme yapılarını açıkla kullanım örnekleri ve tüm metotlarını bütün olarak anlatımı

Python'da **liste**, demet (tuple), sözlük (dict) ve küme (set) hepsi koleksiyon (collection) veri tipleridir; her biri elemanları farklı kurallarla saklar ve farklı metotlara sahiptir.[1]

Aşağıda her yapı için:

- Tanım
- Oluşturma ve temel kullanım
- En sık kullanılan metotlar (kısa açıklama + örnek)

verilmiştir. Tam "tüm metotlar" listesi için `dir(tip_adi)` kullanmak en pratik ve güncel yoldur.[2]

Liste (list)

Listeler sıralı, değiştirilebilir (mutable) ve tekrar eden eleman tutabilen koleksiyonlardır.[1]

Oluşturma ve kullanım

```
liste = [1, 2, 3, "python"]
bos_liste = []
karisik = [1, 3.14, True, ["a", "b"]]

print(liste[^1_0])      # 1
print(liste[-1])     # "python"
liste[^1_1] = 99        # eleman değiştirme
print(liste)         # [1, 99, 3, "python"]
```

Dilimleme (slicing):

```
liste = [10, 20, 30, 40, 50]
print(liste[1:4])    # [20, 30, 40]
print(liste[:3])     # [10, 20, 30]
print(liste[::2])    # [10, 30, 50]
```

Önemli liste metotları

Tam liste için: `dir(list)` veya `dir(liste)` kullanılır.[2]

En sık kullanılanlar:

- `append(x)` : Sona eleman ekler.

```
liste = [1, 2, 3]
liste.append(4)
print(liste)   # [1, 2, 3, 4]
```

- `insert(i, x)` : Belirtilen indekse eleman ekler.

```
liste = [1, 3, 4]
liste.insert(1, 2)
print(liste)   # [1, 2, 3, 4]
```

- `extend(iterable)` : Başka bir liste/iterable'ın tüm elemanlarını ekler.

```
liste = [1, 2, 3]
liste.extend([4, 5])
print(liste)   # [1, 2, 3, 4, 5]
```

- `remove(x)` : Değerin ilk geçtiği elemanı siler (yoksa hata).

```
liste = [1, 2, 2, 3]
liste.remove(2)
print(liste)   # [1, 2, 3]
```

- `pop(i=-1)` : Verilen indeksteki (varsayılan son) elemanı listeden çıkarır ve döndürür.

```
liste = [10, 20, 30]
son = liste.pop()
print(son, liste)    # 30 [10, 20]
```

- `clear()` : Tüm elemanları siler.[3]

```
liste = [1, 2, 3]
liste.clear()
print(liste)         # []
```

- `index(x[, start[, end]])` : İlk geçtiği konumun indeksini döndürür.[3]

```
liste = ["a", "b", "c", "b"]
print(liste.index("b"))    # 1
```

- `count(x)` : Elemanın kaç kez geçtiğini verir.[3]

```
liste = [1, 2, 2, 2, 3]
print(liste.count(2))      # 3
```

- `sort(key=None, reverse=False)` : Listeyi yerinde sıralar.

```
liste = [3, 1, 4, 2]
liste.sort()
print(liste)               # [1, 2, 3, 4]
liste.sort(reverse=True)
print(liste)               # [4, 3, 2, 1]
```

- `reverse()` : Listeyi ters çevirir (sıralama yapmadan).[3]

```
liste = [1, 2, 3]
liste.reverse()
print(liste)               # [3, 2, 1]
```

- `copy()` : Yüzeysel (shallow) kopya döndürür.[3]

```
liste = [1, 2, 3]
kopya = liste.copy()
```

Demet (tuple)

Demetler sıralı ama **değiştirilemeyen** (immutable) koleksiyonlardır. Genellikle sabit veri grupları için kullanılır.[4]

Oluşturma ve kullanım

```
demet = (1, 2, 3)
tek_elemanli = (1,)      # virgül çok önemli
karisik = (1, "a", True)

print(demet[^1_0])          # 1
print(demet[-1])         # 3
```

Pek çok yerde liste gibi indeksleme ve dilimleme yapabilirsiniz.[5]

Önemli demet metotları

Demetlerin metot sayısı azdır; tipik olarak sadece:

- `count(x)` : Elemanın sayısı.
- `index(x[, start[, end]])` : İlk indeks.[5]

```
demet = (1, 2, 2, 3)
print(demet.count(2))     # 2
```

```
print(demet.index(3))     # 3
```

Diğer işlemler fonksiyonlar/operatörlerle yapılır:

- Uzunluk: `len(demet)` [5]
- Birleştirme: `yeni = demet1 + demet2` [6]
- Tekrarlama: `demet * 3`
- Üyelik: `x in demet`

```
a = (1, 2)
b = (3, 4)
c = a + b
print(c)          # (1, 2, 3, 4)
print(2 in c)     # True
```

Sözlük (dict)

Sözlükler anahtar–değer (key–value) çiftleri tutan, sırasal gösterimi olan ama indeks yerine anahtarlarla erişilen, değiştirilebilir yapılardır. [7]

Oluşturma ve temel kullanım

```
sozluk = {
    "ad": "Ali",
    "yas": 25,
    "dil": "Python"
}

print(sozluk["ad"])       # Ali
sozluk["yas"] = 26        # değer değiştirme
sozluk["sehir"] = "İzmir" # yeni anahtar ekleme
```

Anahtara güvenli erişim için `get` kullanılır:

```
print(sozluk.get("meslek"))        # None
print(sozluk.get("meslek", "Yok")) # "Yok"
```

Sözlük üzerinde dolaşma:

```
for k in sozluk:             # anahtarlar
    print(k)

for k, v in sozluk.items():  # anahtar, değer
    print(k, v)
```

Önemli sözlük metotları

Tam liste için `dir(dict)` veya `dir(sozluk)` kullanılabilir.[8]

En sık kullanılanlar:

- `keys()` : Tüm anahtarlar görünümü.[8]

```
print(list(sozluk.keys()))
```

- `values()` : Tüm değerler görünümü.[8]

```
print(list(sozluk.values()))
```

- `items()` : (anahtar, değer) çiftleri.[8]

```
for k, v in sozluk.items():
    print(k, v)
```

- `get(key[, default])` : Güvenli anahtar erişimi (yoksa hata vermez).[8]
- `pop(key[, default])` : Anahtarı siler ve değerini döndürür.[7] [8]

```
yas = sozluk.pop("yas")
print(yas)         # 26
print(sozluk)
```

- `popitem()` : Son eklenen çifti siler ve döndürür.[8]
- `update([other])` : Başka sözlük/diziyle günceller (varsa üzerine yazar).[1]

```
sozluk.update({"yas": 30, "ulke": "TR"})
```

- `clear()` : Tüm öğeleri siler.[8]
- `copy()` : Yüzeysel kopya.[8]
- `fromkeys(iterable, value=None)` : Verilen anahtarlardan yeni sözlük üretir.[8]

```
anahtarlar = ["ad", "yas"]
yeni = dict.fromkeys(anahtarlar, 0)
print(yeni)   # {'ad': 0, 'yas': 0}
```

Küme (set)

Kümeler sırasız, benzersiz elemanlardan oluşan koleksiyonlardır; matematikteki küme yapısına benzer ve değiştirilebilirdir.[9] [10]

Oluşturma ve temel kullanım

```
kume = {1, 2, 3, 3, 2}
print(kume)          # {1, 2, 3}  (tekrar yok)
bos_kume = set()     # {} sözlük anlamına gelir

print(2 in kume)     # True
```

Bir iterable'dan küme:

```
kume = set([1, 2, 2, 3])
print(kume)          # {1, 2, 3}
```

Küme işlemleri (operatörler)

- Birleşim: `A | B` veya `A.union(B)` [9]
- Kesişim: `A & B` veya `A.intersection(B)` [9]
- Fark: `A - B` veya `A.difference(B)` [9]
- Simetrik fark: `A ^ B` veya `A.symmetric_difference(B)` [9]

```
A = {1, 2, 3}
B = {3, 4, 5}

print(A | B)  # {1, 2, 3, 4, 5}
print(A & B)  # {3}
print(A - B)  # {1, 2}
print(A ^ B)  # {1, 2, 4, 5}
```

Önemli küme metotları

Tam liste için `dir(set)` kullanılır. [11] [9]

En sık kullanılanlar:

- `add(x)` : Eleman ekler. [11]

```
kume = {"elma", "muz"}
kume.add("kiraz")
print(kume)
```

- `update(iterable)` : Birden fazla eleman ekler (liste, başka set vb.). [9]

```
kume.update([1, 2, 3])
```

- `remove(x)` : Elemanı siler, yoksa hata. [9]
- `discard(x)` : Elemanı varsa siler, yoksa hiçbir şey yapmaz. [9]

- `pop()` : Rastgele bir elemanı silip döndürür (kümeler sırasız).[9]
- `clear()` : Tüm elemanları siler.[9]
- `union(*others)` : Birleşim kümesi döndürür.[9]
- `intersection(*others)` / `intersection_update(*others)` : Kesişim alır; `intersection_update` mevcut kümeyi günceller.[9]
- `difference(*others)` / `difference_update(*others)` : Fark alır.[9]
- `symmetric_difference(other)` / `symmetric_difference_update(other)` : Simetrik fark.[9]
- Alt/üst küme kontrolleri:
  - `issubset(other)` : Alt küme mi?[9]
  - `issuperset(other)` : Üst küme mi?[9]
  - `isdisjoint(other)` : Ortak eleman yok mu?[9]

```
A = {1, 2}
B = {1, 2, 3}
print(A.issubset(B))   # True
print(B.issuperset(A)) # True
```

Bu özet, pratikte en çok kullanılan metotları kapsıyor; istersen her tip için ayrı ayrı "detaylı pratik soru–çözüm" şeklinde de ilerlenebilir.

⁂

her biri için 3 er farklı alıştırmalar ve cevapları

Aşağıda her veri tipi için 3'er **farklı** alıştırma ve ayrıntılı cevapları var (toplam 12 alıştırma). Sorular temel–orta seviye karışık tutuldu.[21]

Liste için 3 alıştırma

1) Çift sayıları filtrele

Bir listede sadece çift sayıları tutan yeni bir liste oluştur.

```
sayilar = [3, 4, 7, 10, 12, 15, 20]

# Çözüm 1: döngüyle
ciftler = []
for s in sayilar:
    if s % 2 == 0:
        ciftler.append(s)

print(ciftler)   # [4, 10, 12, 20]

# Çözüm 2: list comprehension
```

```
ciftler2 = [s for s in sayilar if s % 2 == 0]
print(ciftler2)  # [4, 10, 12, 20]
```

2) Listedeki tekrar edenleri kaldır

Verilen bir listeden tekrar eden elemanları silip, sadece benzersiz elemanlardan oluşan bir liste üret.

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

benzersiz = []
for eleman in liste:
    if eleman not in benzersiz:
        benzersiz.append(eleman)

print(benzersiz)    # [1, 2, 3, 4, 5]
```

3) İç içe listeden düz liste (flatten)

İç içe bir listeyi tek seviyeli listeye çevir.

```
icerik = [1, [2, 3], [4, [5, 6]], 7]

def flatten(l):
    sonuc = []
    for eleman in l:
        if isinstance(eleman, list):
            sonuc.extend(flatten(eleman))
        else:
            sonuc.append(eleman)
    return sonuc

düz = flatten(icerik)
print(düz)   # [1, 2, 3, 4, 5, 6, 7]
```

Demet (tuple) için 3 alıştırma

1) Demet elemanlarını değiş-tokuş et

İki demetin içeriğini birbirleriyle değiştir.

```
t1 = (11, 22)
t2 = (99, 88)

t1, t2 = t2, t1
```

```
print(t1)   # (99, 88)
print(t2)   # (11, 22)
```

2) Belirli elemanları yeni demete kopyala

`tuple1 = (11, 22, 33, 44, 55, 66)` içinden `44` ve `55` değerlerinden oluşan yeni bir demet oluştur.

```
tuple1 = (11, 22, 33, 44, 55, 66)

tuple2 = tuple1[3:5]   # 3. ve 4. indeks

print(tuple2)          # (44, 55)
```

3) Listeyi demete çevir ve uzunluk bul

Bir listeyi demete çevir ve oluşan demetin uzunluğunu ekrana yazdır.

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

my_tuple = tuple(my_list)
uzunluk = len(my_tuple)

print(my_tuple)   # (10, 20, 30, 40)
print(uzunluk)    # 4
```

Sözlük (dict) için 3 alıştırma

1) İki sözlüğü birleştir

İki sözlüğü `update` kullanarak birleştir.

```
d1 = {"ad": "Ali", "yas": 25}
d2 = {"sehir": "İzmir", "dil": "Python"}

d1.update(d2)

print(d1)
# {'ad': 'Ali', 'yas': 25, 'sehir': 'İzmir', 'dil': 'Python'}
```

2) Sözlükteki sayı değerleri topla

Değerleri sayı olan bir sözlükte tüm değerlerin toplamını bul.

```
notlar = {
    "Ali": 70,
    "Ayşe": 85,
    "Mehmet": 90
}

toplam = sum(notlar.values())
print(toplam)        # 245
```

3) Değerleri liste olan sözlük oluştur

İki listeyi anahtar–değer eşlemesi yapan sözlüğe dönüştür.

```
anahtarlar = ["ad", "yas", "sehir"]
degerler  = ["Ali", 25, "Ankara"]

sonuc = {}
for i in range(len(anahtarlar)):
    sonuc[anahtarlar[i]] = degerler[i]

print(sonuc)
# {'ad': 'Ali', 'yas': 25, 'sehir': 'Ankara'}
```

Küme (set) için 3 alıştırma

1) İki listeden ortak elemanları bul

İki listede geçen ortak elemanları küme kullanarak bul.

```
l1 = [1, 2, 3, 4, 5]
l2 = [4, 5, 6, 7]

k1 = set(l1)
k2 = set(l2)

ortak = k1 & k2   # intersection

print(ortak)      # {4, 5}
```

2) Kümeden birden çok elemanı sil

Bir kümeden belirli elemanları (varsa) güvenli şekilde sil.

```
kume = {1, 2, 3, 4, 5}
sil = [2, 5, 10]   # 10 yok, hata almadan geç

for x in sil:
    kume.discard(x)

print(kume)       # {1, 3, 4}
```

3) Cümledeki farklı kelime sayısı

Bir cümlede kaç farklı kelime geçtiğini bul.

```
cumle = "python öğrenmek güzeldir python programlama güzeldir"

kelimeler = cumle.split()
kume = set(kelimeler)
farkli_sayi = len(kume)

print(kume)          # {'python', 'öğrenmek', 'güzeldir', 'programlama'}
print(farkli_sayi)   # 4
```

İstersen bir sonraki adımda bu alıştırmaların zorluğunu artırıp “sadece list comprehension ile çöz”, “sadece dict metotları kullan” gibi kısıtlar da ekleyebiliriz.

⁂

her biri için metotların kullanımını gösteren 10 farklı kısa günlük hayattan örnekler

Her bir veri tipi için 10’ar tane, günlük hayattan, tek metot (veya çok az ek işlem) içeren **kısa** örnekler var. Toplam 40 küçük senaryo hazırladım.[45]

Liste: 10 günlük hayat örneği

1. **Alışveriş listesine ürün ekleme –** `append`

```
alisveris = ["ekmek", "sut"]
alisveris.append("yumurta")
print(alisveris)  # ['ekmek', 'sut', 'yumurta']
```

2. **Listeye aynı anda birkaç ürün ekleme –** `extend`

```
alisveris = ["ekmek"]
alisveris.extend(["peynir", "zeytin"])
print(alisveris)  # ['ekmek', 'peynir', 'zeytin']
```

3. **Araya ürün sıkıştırma –** `insert`

```
alisveris = ["ekmek", "yumurta"]
alisveris.insert(1, "sut")
print(alisveris)  # ['ekmek', 'sut', 'yumurta']
```

4. **Bitmiş ürünü listeden çıkarma –** `remove`

```
alisveris = ["ekmek", "sut", "yumurta"]
alisveris.remove("sut")
print(alisveris)  # ['ekmek', 'yumurta']
```

5. **Son eklenen görevi silme –** `pop`

```
gorevler = ["mail kontrol", "toplantı", "rapor yaz"]
son = gorevler.pop()
print(son)       # 'rapor yaz'
print(gorevler)  # ['mail kontrol', 'toplantı']
``]
[web:59]

6. **En sevilen filmi kaç kere izlemişsin – `count`**

```python
filmler = ["Matrix", "Inception", "Matrix", "Matrix"]
print(filmler.count("Matrix"))  # 3
```

[web:59]

7. **Bir kitabın sırasını bulma –** `index`

```
kitaplar = ["Simyacı", "Kürk Mantolu", "Tutunamayanlar"]
print(kitaplar.index("Kürk Mantolu"))  # 1
```

[web:13]

8. **Doğum yıllarını sıralama –** `sort`

```
dogum_yillari =
dogum_yillari.sort()
print(dogum_yillari)  #
```

[web:59]

9. **Son aramaları tersine çevirme –** `reverse`
```

```
aramalar = ["anne", "banka", "kargo"]
aramalar.reverse()
print(aramalar)  # ['kargo', 'banka', 'anne']
```

[web:13]

10. **Orijinal listeyi bozmadan kopya ile oynama –** `copy`**,** `clear`

```
favoriler = ["kahve", "çay", "limonata"]
kopya = favoriler.copy()
kopya.remove("çay")
favoriler.clear()
print(kopya)     # ['kahve', 'limonata']
print(favoriler) # []
```

[web:13][web:59]

Demet: 10 günlük hayat örneği

Demetlerde veri sabit kalsın, çoğu örnek **kullanma/erişim** üzerine.[web:18]

1. **Koordinat saklama (x, y)**

```
konum = (40.65, 29.27)   # Yalova koordinatı gibi düşün
enlem, boylam = konum
print(enlem, boylam)
```

[web:18]

2. **Gün, ay, yıl bilgisini paketleme**

```
dogum_tarihi = (4, "Ocak", 2000)
gun, ay, yil = dogum_tarihi
print(ay)  # 'Ocak'
```

[web:18]

3. **Ürün bilgisi: (ad, fiyat, stok)**

```
urun = ("Kulaklık", 350.0, 12)
ad, fiyat, stok = urun
print(f"{ad} - {fiyat} TL")
```

[web:18]

4. **Demette bir değerin kaç kez geçtiğini bul –** `count`

```
notlar = (5, 4, 5, 3, 5)
print(notlar.count(5))  # 3
```

[web:18]

5. **Belirli elemanın indeksini bulma –** `index`

```
gunler = ("Pzt", "Sal", "Çar", "Per", "Cum")
print(gunler.index("Çar"))  # 2
```

[web:18]

6. **Sabit menü (değişmesini istemiyorsun)**

```
menu = ("Çorba", "Ana Yemek", "Tatlı")
for item in menu:
    print(item)
```

[web:18]

7. **Listeyi demete kilitleme (değişmesin) –** `tuple`

```
sehirler_list = ["İstanbul", "Ankara", "İzmir"]
sehirler = tuple(sehirler_list)
# sehirler = "Bursa"  # Hata, değiştirilemez
```

[web:18]

8. **Kullanıcı adı, rol ikililerini tutma (tuple listesi)**

```
kullanicilar = [
    ("ali", "admin"),
    ("ayse", "editor"),
    ("mehmet", "user")
]

for isim, rol in kullanicilar:
    print(isim, "=>", rol)
```

[web:55]

9. **Demet birleştirme**

```
aile1 = ("anne", "baba")
aile2 = ("çocuk1", "çocuk2")
aile = aile1 + aile2
print(aile)  # ('anne', 'baba', 'çocuk1', 'çocuk2')
```

[web:18]

10. **Demeti tekrar etme (örneğin haftalık plan şablonu)**

```
sablon = ("ders", "spor")
haftalik = sablon * 3
```

```
print(haftalik)  # ('ders', 'spor', 'ders', 'spor', 'ders', 'spor')
```

[web:18]

Sözlük: 10 günlük hayat örneği

1. **Kişi bilgisi saklama ve güvenli erişim –** `get`

```
kisi = {"ad": "Ali", "yas": 25}
print(kisi.get("sehir", "bilgi yok"))  # 'bilgi yok'
```

[web:67]

2. **Sözlüğe yeni bilgi ekleme / güncelleme – köşeli parantez**

```
kisi["sehir"] = "Yalova"
kisi["yas"] = 26
print(kisi)
```

[web:67]

3. **Sadece anahtarları listeleme –** `keys`

```
urun = {"ad": "Telefon", "fiyat": 15000, "renk": "siyah"}
print(list(urun.keys()))  # ['ad', 'fiyat', 'renk']
```

[web:67][web:64]

4. **Sadece değerleri alma –** `values`

```
print(list(urun.values()))  # ['Telefon', 15000, 'siyah']
```

[web:67]

5. **Anahtar–değer çiftleriyle dolaşma –** `items`

```
for k, v in urun.items():
    print(k, "=>", v)
```

[web:67]

6. **İki sözlükten profil oluşturma –** `update`

```
profil = {"ad": "Ali"}
ek = {"soyad": "Yılmaz", "sehir": "İzmir"}
profil.update(ek)
print(profil)
```

[web:67][web:64]

7. **Silinen görevi geri almak için –** `pop`

```
gorevler = {"g1": "mail", "g2": "toplantı", "g3": "fatura"}
silinen = gorevler.pop("g2")
print(silinen)   # 'toplantı'
print(gorevler)
```

[web:67]

8. **Son eklenen kaydı kaldırmak –** `popitem`

```
kayitlar = {"ilk": 1, "ikinci": 2}
son = kayitlar.popitem()
print(son)       # ('ikinci', 2)
print(kayitlar)  # {'ilk': 1}
```

[web:61][web:67]

9. **Varsa getir, yoksa ekle –** `setdefault`

```
ayarlar = {"tema": "koyu"}
deger = ayarlar.setdefault("dil", "tr")
print(deger)     # 'tr'
print(ayarlar)   # {'tema': 'koyu', 'dil': 'tr'}
```

[web:61]

10. **Aynı varsayılan değere sahip çoklu anahtar –** `fromkeys`

```
alanlar = ["ad", "soyad", "yas"]
varsayilan = dict.fromkeys(alanlar, "")
print(varsayilan)  # {'ad': '', 'soyad': '', 'yas': ''}
```

[web:61][web:73]

Küme: 10 günlük hayat örneği

1. **Arkadaş listelerinin kesişimi –** `intersection` **/** `&`

```
senin = {"Ali", "Ayşe", "Mehmet"}
arkadasin = {"Ayşe", "Zeynep"}
ortak = senin & arkadasin
print(ortak)  # {'Ayşe'}
```

[web:65][web:71]

2. **Bir kişinin arkadaşını ekleme –** `add`

```
arkadaslar = {"Ali", "Ayşe"}
arkadaslar.add("Mehmet")
print(arkadaslar)
```

[web:62][web:68]

3. **Artık görüşmediğin kişiyi çıkarma –** `discard`

```
arkadaslar.discard("Ali")    # Yoksa da hata vermez
print(arkadaslar)
```

[web:62]

4. **Takip ettiğin ama seni takip etmeyen hesaplar –** `difference` **/** `-`

```
sen = {"a", "b", "c", "d"}
seni_takip_edenler = {"b", "d"}
tek_yonlu = sen - seni_takip_edenler
print(tek_yonlu)   # {'a', 'c'}
```

[web:71][web:68]

5. **İki sosyal çevrenin tamamen farklı olup olmadığını kontrol –** `isdisjoint`

```
is_arkadaslari = {"Ahmet", "Fatma"}
mahalle_arkadaslari = {"Mehmet", "Zeynep"}
print(is_arkadaslari.isdisjoint(mahalle_arkadaslari))  # True
```

[web:71][web:68]

6. **Tüm etkinlik katılımcılarını birleştirme –** `union` **/** `|`

```
konferans = {"Ali", "Ayşe"}
workshop = {"Mehmet", "Ali"}
toplam = konferans | workshop
print(toplam)   # {'Ali', 'Ayşe', 'Mehmet'}
```

[web:65][web:68]

7. **Alt/üst küme ilişkisi –** `issubset`**,** `issuperset`

```
okunan = {"Python 101", "Veri Bilimi"}
hedef = {"Python 101", "Veri Bilimi", "Algoritmalar"}

print(okunan.issubset(hedef))    # True
print(hedef.issuperset(okunan))  # True
```

[web:71][web:68]

8. **Tekrarlı öğeleri otomatik temizleme (email listesi)**

```
emailler = ["a@mail.com", "b@mail.com", "a@mail.com"]
benzersiz = set(emailler)
print(benzersiz)  # {'a@mail.com', 'b@mail.com'}
```

[web:65][web:71]

9. **Kümeyi tamamen temizleme –** `clear`

```
etiketler = {"spor", "yemek", "gezi"}
etiketler.clear()
print(etiketler)  # set()
```

[web:62][web:68]

10. **Bir elemanı rastgele çekip çıkarma –** `pop`

```
hediyeler = {"kalem", "defter", "ajanda"}
verilen = hediyeler.pop()
print(verilen)
print(hediyeler)
```

[web:62][web:68]

İstersen bir sonraki adımda her yapı için “sadece bu metotları kullanarak mini proje” (örneğin basit yapılacaklar listesi, sözlükle mini rehber, kümeyle arkadaş önerisi sistemi) de yazabiliriz.

⁂

1. https://docs.python.org/tr/3/tutorial/datastructures.html
2. https://python.yazbel.com/listelerin_ve_demetlerin_metotlari.html
3. https://www.mobilhanem.com/python-liste-ve-listenin-metotlari/
4. https://realpython.com/python-tuple/
5. https://www.programiz.com/python-programming/tuple
6. https://python.sitesi.web.tr/python-tuple.html
7. https://www.sadikturan.com/python-objeleri-ve-veri-yapilari/python-dictionary/1379
8. https://www.cagrigungor.com/python-sozluk-dictionary-olusturma-ve-metotlar/
9. https://www.snakify.org/tr/lessons/sets/
10. https://www.mobilhanem.com/python-set-kume-ve-fronzsetkisitlanmis-kume/
11. https://python.sitesi.web.tr/python-set.html
12. https://docs.python.org/tr/3.10/tutorial/datastructures.html
13. https://caglarbostanci.com.tr/python-liste-kullanimi-ve-fonksiyonlari-python-list-methods/
14. https://tr.linkedin.com/pulse/pythonda-listeler-liste-metotları-ve-tuplelar-mehmet-özkaya
15. https://www.geeksforgeeks.org/python/python-tuples/
16. https://www.geeksforgeeks.org/python/list-methods-python/

17. https://python.sitesi.web.tr/python-dict.html
18. https://www.youtube.com/watch?v=gEfvbNNyp0g
19. https://www.w3schools.com/python/python_lists_methods.asp
20. https://www.youtube.com/watch?v=YErwf3J-lZ4
21. https://pynative.com/python-exercises-with-solutions/
22. https://www.w3resource.com/python-exercises/tuple/
23. https://pynative.com/python-dictionary-exercise-with-solutions/
24. https://www.tutorjoes.in/python_programming_tutorial/list_comprehensions_exercises_in_python
25. https://www.w3schools.com/python/python_tuples_exercises.asp
26. https://pythonlobby.com/dictionary-exercises-in-python-with-solution/
27. https://www.tutorjoes.in/python_programming_tutorial/list_programs_in_python
28. https://www.tutorjoes.in/python_programming_tutorial/tuple_comprehensions_exercises_in_python
29. https://www.w3resource.com/python-exercises/dictionary/
30. https://www.geeksforgeeks.org/python/python-tuple-exercise/
31. https://www.w3schools.com/python/python_dictionaries_exercises.asp
32. https://pynative.com/python-list-exercise-with-solutions/
33. https://www.reddit.com/r/learnpython/comments/147pr2e/can_someone_provide_me_with_the_list_of_python/
34. https://www.programiz.com/python-programming/tuple
35. https://python.sitesi.web.tr/python-dict.html
36. https://www.snakify.org/tr/lessons/sets/
37. https://python.sitesi.web.tr/python-set.html
38. https://www.w3resource.com/python-exercises/list/
39. https://www.geeksforgeeks.org/python/python-list-exercise/
40. https://www.w3schools.com/python/python_lists_exercises.asp
41. https://pythonlobby.com/list-exercises-in-python-with-solution/
42. https://pynative.com/python-tuple-exercise-with-solutions/
43. https://pynative.com/python-set-exercise-with-solutions/
44. https://www.w3resource.com/python-exercises/list-advanced/index.php
45. https://docs.python.org/3/tutorial/datastructures.html
46. https://sparkbyexamples.com/python/python-dictionary-methods-examples/
47. https://realpython.com/python-sets/
48. https://cs.du.edu/~intropython/intro-to-programming/list_methods.html
49. https://www.programiz.com/python-programming/dictionary
50. https://www.w3schools.com/python/python_ref_set.asp
51. https://www.youtube.com/watch?v=u0yr9B3nH8c
52. https://www.programiz.com/python-programming/set
53. https://www.pythonsnacks.com/p/python-list-methods-functions-example
54. https://www.freecodecamp.org/news/create-a-dictionary-in-python-python-dict-methods/

55. https://www.codeguage.com/courses/python/sets-set-methods
56. https://www.geeksforgeeks.org/python/list-methods-python/
57. https://www.scaler.com/topics/list-methods-in-python/
58. https://www.w3schools.com/python/python_lists_methods.asp
59. https://www.w3schools.com/python/python_ref_list.asp
60. https://www.programiz.com/python-programming/list
61. https://developers.google.com/edu/python/lists
62. https://dev.to/usooldatascience/a-quick-guide-to-python-dictionary-methods-with-examples-2gfb
63. https://www.geeksforgeeks.org/python/python-set-methods/
64. https://www.datacamp.com/tutorial/python-list-function