Python'da **kümeler** (sets), benzersiz ve sırasız elemanlardan oluşan veri tipleridir. Kümeler, matematiksel kümelere benzer şekilde çalışır ve elemanların tekrarlanmamasını sağlar. Python'da kümeler üzerinde matematiksel işlemler (kesişim, birleşim, fark gibi) yapılabilir.

Temel Özellikler:

1. **Benzersiz Elemanlar**: Kümelerde aynı eleman birden fazla kez bulunamaz.
2. **Sırasız Yapı**: Kümelerde elemanların belirli bir sırası yoktur; bu nedenle indeksleme veya dilimleme yapılmaz.
3. **Değiştirilebilir**: Kümelere eleman ekleyebilir veya çıkarabilirsin.
4. **Karışık Veri Tipleri**: Bir küme içinde farklı türde veri tipleri olabilir, ancak kümelerin elemanları **değiştirilemez (immutable)** olmalıdır. Yani bir kümenin elemanı `list` olamaz ama `tuple` olabilir.

Python'da Küme Tanımlama

Boş Küme

Boş bir küme oluşturmak için `set()` fonksiyonu kullanılır. `{}` işaretleri boş bir sözlük oluşturur, boş küme değil!

```
bos_kume = set()
```

Elemanlarla Birlikte Küme

Küme, `{}` süslü parantezler ile tanımlanabilir.

```
kume = {1, 2, 3, 4}
```

Temel Küme İşlemleri

- **Eleman Ekleme**: `add()` metodu kullanılarak kümeye eleman eklenir.

```
kume = {1, 2, 3}
kume.add(4)
print(kume)  # Çıktı: {1, 2, 3, 4}
```

- **Eleman Silme**: `remove()` metodu ile belirli bir eleman kümeden silinir. Eğer eleman yoksa hata verir. Alternatif olarak `discard()` metodu eleman yoksa hata vermez.

```
kume.remove(2)
kume.discard(5)  # Eleman yoksa hata vermez
```

- **Küme Uzunluğu**: `len()` fonksiyonu kümenin eleman sayısını döner.

```
print(len(kume))  # Kümedeki eleman sayısı
```

- **Eleman Arama**: `in` anahtar kelimesi ile bir elemanın kümede olup olmadığını kontrol edebilirsin.

```
print(3 in kume)  # True ya da False döner
```

Küme Operasyonları

Python'da kümeler üzerinde matematiksel işlemler yapmak oldukça kolaydır.

- **Birleşim**: İki kümenin birleşimi, her iki kümedeki tüm elemanları içerir.

```
kume1 = {1, 2, 3}
kume2 = {3, 4, 5}
print(kume1 | kume2)  # Çıktı: {1, 2, 3, 4, 5}
```

- **Kesişim**: İki kümenin kesişimi, her iki kümede bulunan ortak elemanları içerir.

```
print(kume1 & kume2)  # Çıktı: {3}
```

- **Fark**: Bir kümeden diğerini çıkarır, yalnızca birinci kümede bulunan elemanları döner.

```
print(kume1 - kume2)  # Çıktı: {1, 2}
```

- **Simetrik Fark**: İki kümenin simetrik farkı, her iki kümede olup da ortak olmayan elemanları içerir.

```
print(kume1 ^ kume2)  # Çıktı: {1, 2, 4, 5}
```

Örnek Kullanım

Kümeler özellikle tekrar eden verilerden kurtulmak veya iki veri kümesi arasındaki benzersiz elemanları bulmak için kullanışlıdır.

```
liste = [1, 2, 2, 3, 4, 4, 5]
benzersiz = set(liste)  # Tekrar eden elemanlar silinir
print(benzersiz)  # Çıktı: {1, 2, 3, 4, 5}
```

Dondurulmuş Kümeler (frozenset)

Eğer bir kümenin elemanlarının değiştirilmesini istemiyorsan, `frozenset` kullanarak değiştirilemez (immutable) bir küme oluşturabilirsin.

```
donmus_kume = frozenset([1, 2, 3])
# donmus_kume.add(4)  # Hata verir, çünkü eleman eklenemez
```

Kümelerin Avantajları:

- Benzersiz elemanlardan oluştuğu için tekrarlayan verileri yönetmede etkilidir.

- Matematiksel kümeler gibi birleşim, kesişim ve fark işlemleri kolaylıkla yapılabilir.
- Kümeler oldukça hızlıdır çünkü arka planda hash yapısı kullanır. Bu da büyük veri kümelerinde eleman aramayı ve işlemleri hızlandırır.

Python'daki kümeler, özellikle büyük veri setlerinde işlem yaparken ve benzersiz veri kümeleri oluşturmak istediğinde oldukça kullanışlıdır.