Python'da **sözlük (dictionary)**, anahtar-değer (key-value) çiftleri şeklinde verileri saklayan bir veri tipidir. Sözlükler, her bir anahtarın (key) bir değerle (value) eşleştirildiği yapılardır. Sözlükler, **değiştirilebilir (mutable)** oldukları için yeni anahtar-değer çiftleri eklenebilir, mevcut çiftler silinebilir veya değiştirilebilir. Sözlükler **{}** süslü parantezler ile tanımlanır ve her anahtar-değer çifti iki nokta üst üste (`:`) ile ayrılır.

Sözlüklerin Temel Özellikleri:

1. **Anahtar-Değer İlişkisi (Key-Value Pairs)**: Her öğe, bir anahtar ve bu anahtara karşılık gelen bir değerden oluşur. Anahtar, sözlükte benzersiz olmalıdır ancak değerler aynı olabilir.
2. **Değiştirilebilir (Mutable)**: Sözlüklerdeki anahtar-değer çiftleri eklenebilir, güncellenebilir veya silinebilir.
3. **Sırasız (Unordered)**: Python 3.7'den önce sözlükler sırasızdı; Python 3.7 ve sonrasında ise eklenme sırasını hatırlamaya başladılar.
4. **Anahtarlar İmmutable Olmalı**: Anahtarlar sabit veri türlerinden olmalıdır (string, integer, tuple gibi). Listeler gibi değiştirilebilir veri türleri anahtar olarak kullanılamaz.
5. **Hızlı Erişim**: Sözlükler, veri yapılarına hızlı erişim sağlar çünkü veri saklama ve erişim hash tabanlıdır.

Sözlük Oluşturma

Sözlükler, **{ }** süslü parantezler ile tanımlanır ve her bir anahtar-değer çifti iki nokta üst üste (`:`) ile ayrılır. Anahtarlar benzersiz olmalıdır, aksi takdirde son eklenen anahtar-değer çifti geçerli olur.

Örnekler:

```
# Boş sözlük
sozluk = {}
```

```python
# Anahtar ve değerlerle sözlük oluşturma
sozluk = {"isim": "Ahmet", "yas": 25, "sehir": "İstanbul"}

# Farklı veri türlerinde anahtar ve değerler
sozluk = {1: "bir", 2: "iki", 3: "üç"}

# Tuple'ları anahtar olarak kullanma
sozluk = {(1, 2): "koordinatlar", (3, 4): "daha fazla koordinat"}
```

Sözlük Elemanlarına Erişim

Sözlükteki bir değere, ilgili anahtar ile erişilir. Listelerden farklı olarak, sözlüklerde indeksleme değil, anahtar kullanılır.

Örnekler:

```python
# Sözlük oluşturma
sozluk = {"isim": "Ayşe", "yas": 30, "sehir": "Ankara"}

# Anahtarla değerine erişim
print(sozluk["isim"])  # Ayşe
print(sozluk["yas"])   # 30

# Olmayan bir anahtara erişim hataya yol açar
# print(sozluk["meslek"])  # KeyError: 'meslek'
```

`.get()` Metodu ile Güvenli Erişim

Sözlükte anahtarın olup olmadığını bilmediğinizde, `.get()` metodunu kullanabilirsiniz. Eğer anahtar yoksa `None` döner veya isterseniz varsayılan bir değer belirleyebilirsiniz.

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

Sözlüğe Eleman Ekleme ve Güncelleme

Sözlüklere yeni anahtar-değer çiftleri ekleyebilir veya mevcut bir anahtarı güncelleyebilirsiniz.

Örnekler:

```
sozluk = {"isim": "Ayşe", "yas": 30}

# Yeni bir anahtar-değer ekleme
sozluk["meslek"] = "Mühendis"
print(sozluk)  # {"isim": "Ayşe", "yas": 30, "meslek": "Mühendis"}

# Mevcut bir anahtarı güncelleme
sozluk["yas"] = 31
print(sozluk)  # {"isim": "Ayşe", "yas": 31, "meslek": "Mühendis"}
```

Sözlükten Eleman Silme

Bir anahtar-değer çiftini sözlükten silmek için çeşitli yöntemler vardır:

- `del`: Belirtilen anahtarı ve ona bağlı değeri siler.
- `pop()`: Anahtar ve ona karşılık gelen değeri siler ve o değeri döner.
- `popitem()`: Son eklenen anahtar-değer çiftini siler ve döner.
- `clear()`: Tüm anahtar-değer çiftlerini siler.

Örnekler:

```
sozluk = {"isim": "Ayşe", "yas": 30, "meslek": "Mühendis"}

# 'yas' anahtarını sil
del sozluk["yas"]
print(sozluk)  # {"isim": "Ayşe", "meslek": "Mühendis"}

# 'meslek' anahtarını sil ve değerini döner
meslek = sozluk.pop("meslek")
```

```
print(meslek)  # "Mühendis"
print(sozluk)  # {"isim": "Ayşe"}

# Tüm elemanları sil
sozluk.clear()
print(sozluk)  # {}
```

Sözlük Metodları

Sözlüklerle çalışmayı kolaylaştıran birçok yerleşik metod vardır:

1. `keys()`: Sözlükteki tüm anahtarları döner.

```
sozluk = {"isim": "Ayşe", "yas": 30}
print(sozluk.keys())  # dict_keys(['isim', 'yas'])
```

2. `values()`: Sözlükteki tüm değerleri döner.

```
print(sozluk.values())  # dict_values(['Ayşe', 30])
```

3. `items()`: Sözlükteki anahtar-değer çiftlerini döner.

```
print(sozluk.items())  # dict_items([('isim', 'Ayşe'), ('yas',
30)])
```

4. `update()`: Başka bir sözlükle birleştirir veya günceller.

```
yeni_bilgiler = {"sehir": "Ankara", "yas": 31}
sozluk.update(yeni_bilgiler)
print(sozluk)  # {'isim': 'Ayşe', 'yas': 31, 'sehir': 'Ankara'}
```

5. `pop()`: Belirtilen anahtarı ve değerini siler ve döner.

```
yas = sozluk.pop("yas")
print(yas)  # 31
```

6. `popitem()` : Son eklenen anahtar-değer çiftini siler ve döner.

```
son_eleman = sozluk.popitem()
print(son_eleman)  # ('sehir', 'Ankara')
```

Sözlük ile Liste Arasında Dönüşüm

Sözlükler, listelere dönüştürülebilir ve tam tersi de yapılabilir.

Örnekler:

```
# Anahtarlar listesini elde etme
sozluk = {"isim": "Ayşe", "yas": 30}
anahtarlar = list(sozluk.keys())
print(anahtarlar)  # ['isim', 'yas']

# Değerler listesini elde etme
degerler = list(sozluk.values())
print(degerler)  # ['Ayşe', 30]

# Anahtar-değer çiftlerinden liste oluşturma
anahtar_deger = list(sozluk.items())
print(anahtar_deger)  # [('isim', 'Ayşe'), ('yas', 30)]
```

Sözlüklerin Kullanım Alanları

- **Veri Haritalama**: Bir anahtara karşılık gelen bir değeri saklamak için kullanılır (örneğin, bir kullanıcının bilgilerini saklamak).
- **Hızlı Arama**: Anahtarlara hızlı bir şekilde erişilebilir, bu da veri arama işlemlerini hızlandırır.
- **Etiketleme**: Verileri daha anlamlı ve organize bir şekilde etiketlemek için kullanılır

Python'da **sözlük (dictionary)**, anahtar-değer çiftleri şeklinde verileri saklayan, esnek ve güçlü bir veri yapısıdır. Veri organizasyonu ve hızlı arama gibi durumlar için

kullanışlıdır. Eğer başka veri tipleri veya konular hakkında daha fazla bilgiye ihtiyaç duyarsanız, sorularınızı memnuniyetle yanıtlarım!