Python'da List Comprehension ve İlgili Konular

1. List Comprehension Nedir?

List Comprehension, Python'da bir listeyi daha kısa ve okunaklı bir şekilde oluşturmak için kullanılan bir yöntemdir. Döngüleri ve koşullu ifadeleri tek bir satırda kullanarak liste oluşturmayı sağlar.

1.1. Temel Sözdizimi

```
[expression for item in iterable if condition]
```

- **expression**: Her öğe için uygulanacak işlem.
- **item**: Döngüdeki mevcut öğe.
- **iterable**: Üzerinde döngü çalıştırılacak veri yapısı.
- **condition** (isteğe bağlı): Koşul sağlanıyorsa öğe listeye eklenir.

1.2. Örnekler

Basit Liste Oluşturma

```
numbers = [x for x in range(5)]
print(numbers)  # [0, 1, 2, 3, 4]
```

Koşullu Liste

```
even_numbers = [x for x in range(10) if x % 2 == 0]
print(even_numbers)  # [0, 2, 4, 6, 8]
```

İç İçe Döngüler

```
matrix = [(x, y) for x in range(3) for y in range(2)]
print(matrix)  # [(0, 0), (0, 1), (1, 0), (1, 1), (2, 0), (2, 1)]
```

2. Set Comprehension

Set Comprehension, küme (set) oluşturmanın kısa yoludur. List Comprehension ile benzerdir ama sonuç küme olarak döner.

```
unique_numbers = {x for x in range(10) if x % 2 == 0}
print(unique_numbers)  # {0, 2, 4, 6, 8}
```

3. Tuple Comprehension (Generator Expression)

Tuple comprehension, list comprehension gibi çalışır ama sonuç bir **generator object** olur.

```
generator = (x for x in range(5))
```

```
print(tuple(generator))  # (0, 1, 2, 3, 4)
```

4. Dictionary Comprehension

Dict Comprehension, anahtar-değer çiftlerinden oluşan bir sözlük oluşturmak için kullanılır.

```
squares = {x: x**2 for x in range(5)}
print(squares)  # {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

5. map() Fonksiyonu

`map()` fonksiyonu, bir fonksiyonu listedeki her elemana uygulayıp yeni bir liste oluşturur.

```
def square(n):
    return n * n

numbers = [1, 2, 3, 4]
squared_numbers = list(map(square, numbers))
print(squared_numbers)  # [1, 4, 9, 16]
```

Lambda ile Kullanımı:

```
squared_numbers = list(map(lambda x: x**2, numbers))
print(squared_numbers)  # [1, 4, 9, 16]
```

6. filter() Fonksiyonu

`filter()`, belirli bir koşulu sağlayan elemanları seçer.

```
numbers = [1, 2, 3, 4, 5, 6]
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print(even_numbers)  # [2, 4, 6]
```

7. List Comprehension vs map() & filter()

Yöntem	Kullanım Amacı	Hız
List Comprehension	Daha okunaklı ve esnek	Hızlı
`map()`	Bir fonksiyonu listeye uygulamak	Daha az okunaklı
`filter()`	Belirli koşullara göre eleme	Daha az esnek

Bu döküman Python'daki List Comprehension ve ilgili konular hakkında detaylı bilgi vermektedir. Kullanım örnekleriyle birlikte konuları pekiştirebilirsiniz. 🚀