PROGRAMLAMA TEMELLERI DERS 3

> LISTELER - TEKRAR

- Virgülle ayrılmış öğelerin bir araya geldiği bir veri yapısıdır.
- Listeler, köşeli parantez içinde tanımlanır.

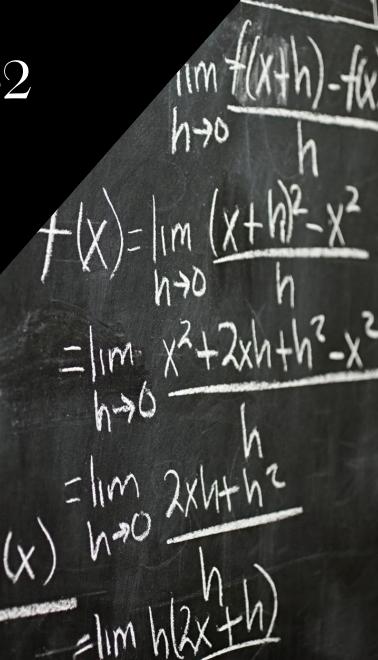
```
my_list = [1, 2, 3, 4, 5]
```

• Eleman Erişimi: Listenin her elemanına, sıfırdan başlayarak indeks kullanarak erişilebilir.

```
first_element = my_list[0] # 1. eleman
```

• Negatif İndeksleme: Listenin son elemanına -l indeksiyle erişilebilir. -2 indeksi, sondan bir önceki elemana denk gelir, ve böyle devam eder.

last_element = my_list[-1] # Son eleman

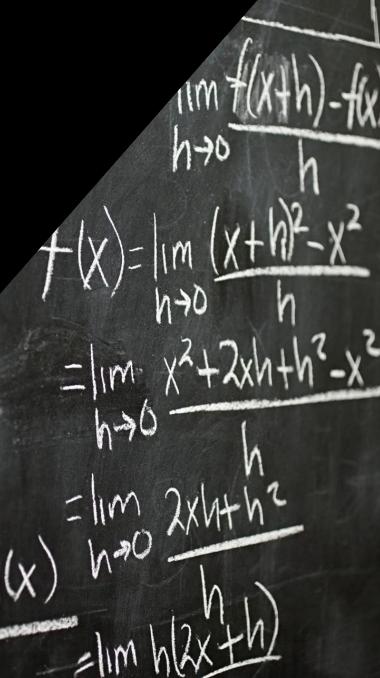


• Homojen ve Heterojen Elemanlar: Liste içinde aynı türden (homojen) veya farklı türden (heterojen) elemanlar bulunabilir.

```
list = [1, "iki", 3.0, True]
```

• Değiştirilebilir (Mutable) Yapı: Listeler, eleman ekleme, çıkarma, güncelleme gibi işlemlere açıktır ve değiştirilebilirler.

my_list[2] = 10 # 3. eleman1 10 ile değiştir



Slicing işlemi, bir liste, dize veya başka bir sıralı veri tipinin belirli bir aralığını seçmeye yarayan bir tekniktir. Atlama (step) değeri, slicing işleminde kaçar adım atılacağını belirler. Slicing start:stop:step şeklinde kullanılır.

Temel Kullanım -> Atlama değeri olmadan sadece start ve stop değerleri kullanarak bir aralık seçimi:

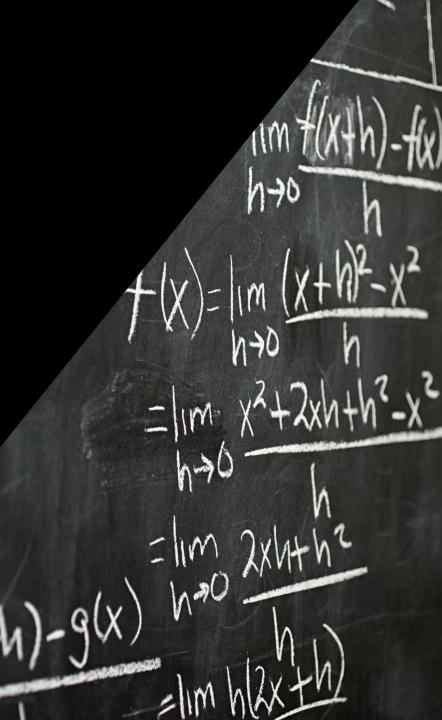
```
my_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
result = my_list[2:8] # 2. indeksten 8. indekse kadar (8 dahil değil)
print(result)
```

[2, 3, 4, 5, 6, 7]

Atlama İşlemi Eklemek -> step değeri ekleyerek belirli bir atlama ile eleman seçimi:

```
my_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
result = my_list[2:8:2] # 2. indeksten 8. indekse kadar, 2'şer adım
print(result)
```

[2, 4, 6]



Negatif Atlama Değeri -> Negatif bir step değeri kullanarak tersten seçim:

```
my_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
result = my_list[8:2:-1] # 8. indeksten 2. indekse kadar, tersten
print(result)
```

```
[8, 7, 6, 5, 4, 3]
```

```
asal_sayilar = [2, 3, 5, 7, 11, 13, 17, 19, 23]
print(asal_sayilar[0:6:2])
```

Verilen kodun çıktısı ne olur?

Verilen kod, asal_sayilar adlı bir liste oluşturur ve bu listenin 0 ile 5 indeksleri arasındaki elemanları, 2'şer adımlarla alarak bir dilimleme işlemi yapar.

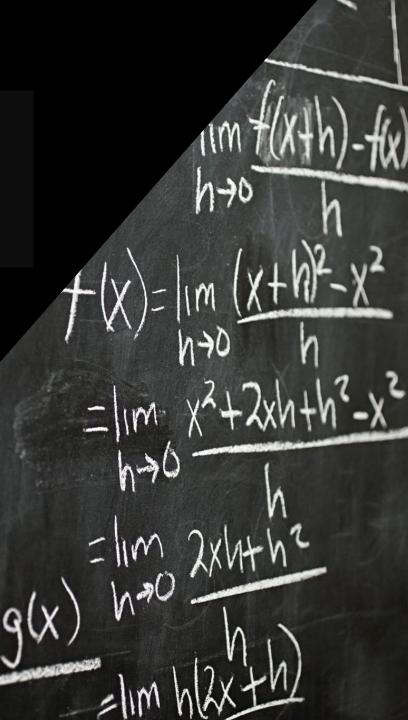
asal_sayilar[0:6:2] ifadesi, 0'dan başlayarak 6'ya kadar (6 dahil değil) olan indeks aralığını alır. 2'şer adımla alındığı için, 0, 2, 4 indekslerindeki elemanları seçer.

Sonuç olarak, çıktı [2, 5, 11] şeklinde olacaktır.

Bu çıktı, asal_sayilar listesinin 0, 2 ve 4. indekslerindeki elemanları içerir: 2, 5 ve 11. Bu elemanlar, dilimleme işlemi sırasında 2'şer adımla seçilmiştir.

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
result = numbers[1::2]
print(result)
```

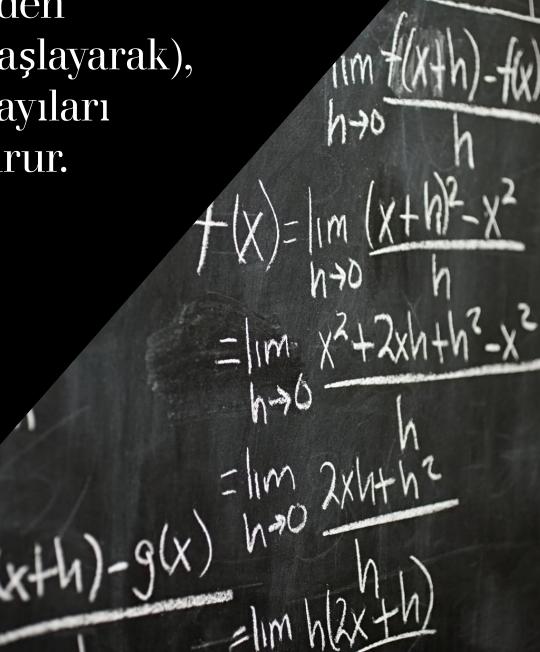
Verilen kodun çıktısı ne olur?



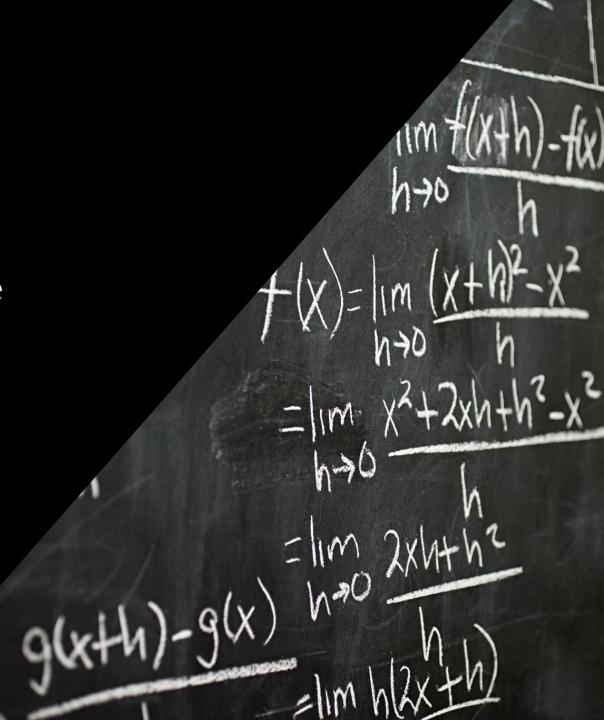
Bu kod, numbers listesinin 1. indeksinden başlayarak sona kadar (2. indeksten başlayarak), 2'şer adımla çift sayıları seçer ve bu sayıları içeren yeni bir liste olan resultı oluşturur.

Çıktı: [2, 4, 6, 8, 10]

Bu çıktı, numbers listesindeki çift sayıları 2'şer adımla seçtiğimizi gösterir.



"Listelerin fonksiyonları" ifadesi, genellikle bir programlama dilindeki listelerle çalışmak için kullanılabilen yerleşik fonksiyonları veya metodları ifade eder. Python'da liste veri tipi, listenin üzerinde çeşitli işlemler yapmak için kullanılabilecek bir dizi özel fonksiyon veya metoda sahiptir. İşte bazı yaygın list fonksiyonları:



Listelerin Fonksiyonları

append	extend	insert	remove	pop	clear
index	count	sort	reverse	copy	del