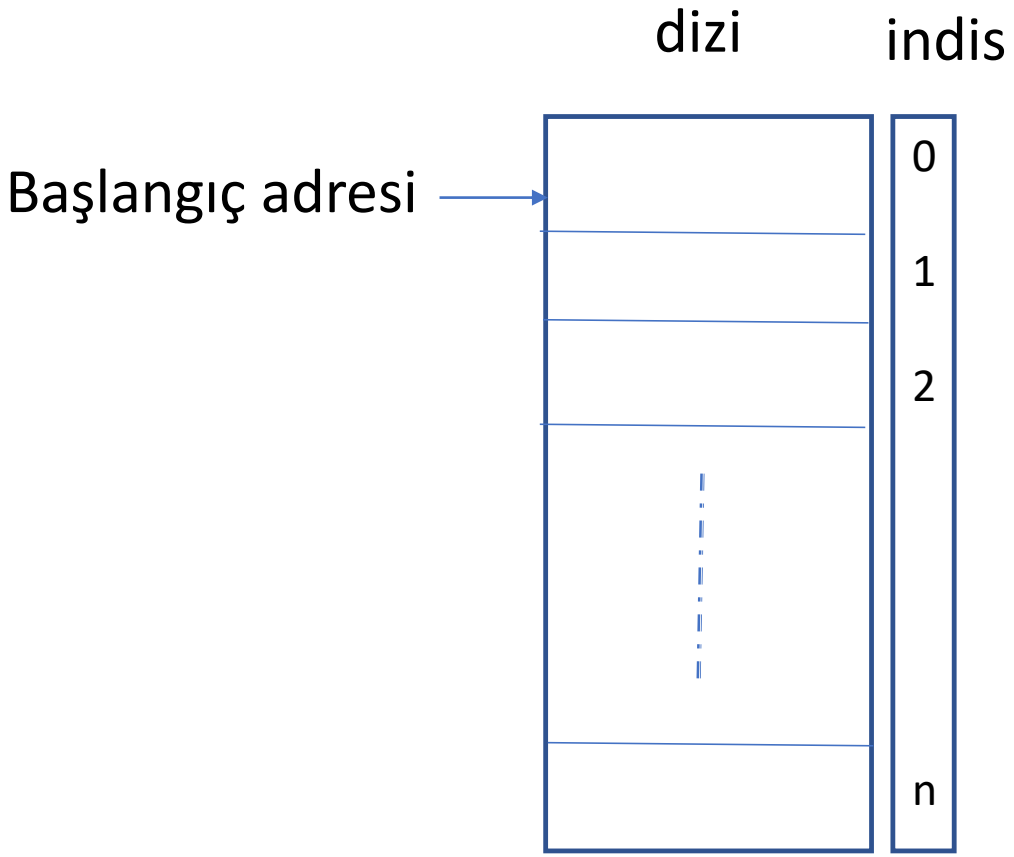
Hacettepe Üniversitesi İstatistik Bölümü İST281 Bilgisayar Programlama Dersi

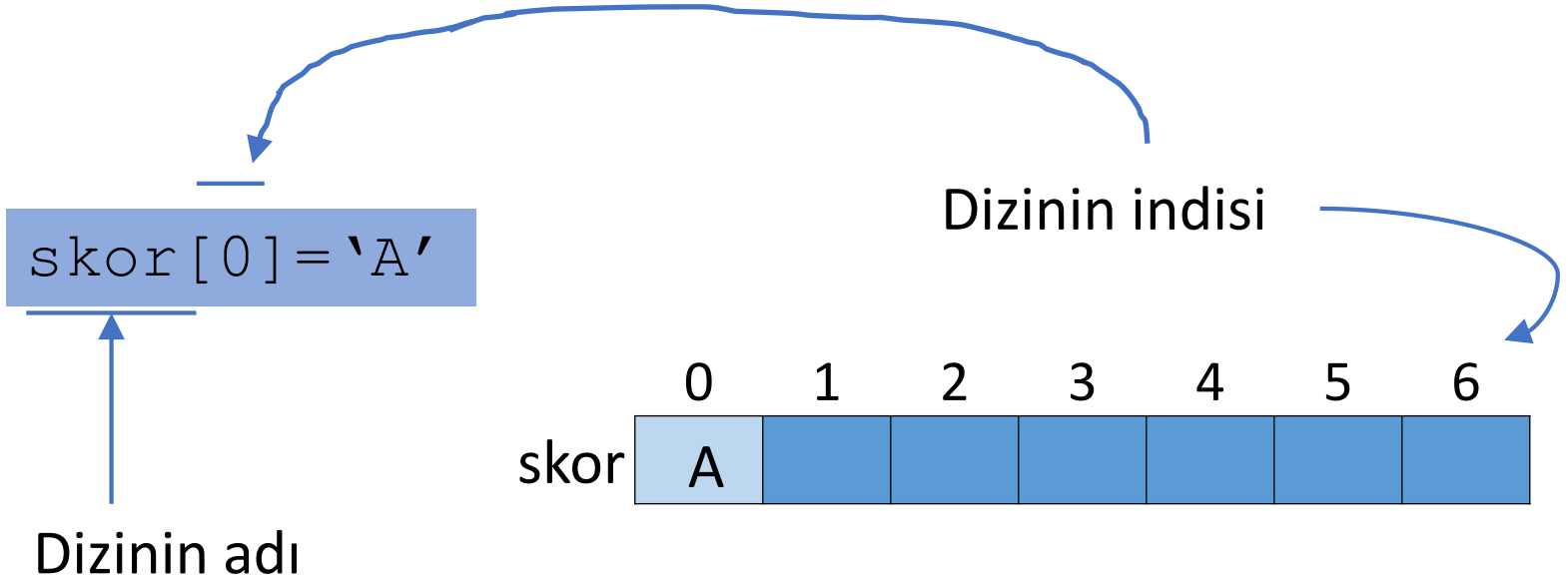
List Veri Yapısı

Python'da alışılmış dizi (array) veri yapısı yerine liste (linked list, list) veri yapısı kullanılmaktadır. Her ne kadar standart kütüphane içinde "array" modülü bulunsa da alışılmış dizi kullanımı için uygun değildir.

Alışılmış dizi ve matris işlemleri için çoğunlukla NumPy kütüphanesi tercih edilmektedir.

Dizi veri yapısında bir bellek adresinden başlayıp, önceden belirlenmiş sayıda bellek alanına tek bir ad verilir. Dizinin satır ve sütun sayısı önceden belirlenmelidir ve tüm öğeler aynı veri tipinde olmalıdır. Dizi ardışık (sequential) olarak indekslenir. İndeksler aracılığı ile erişim sağlanır.




skor[0]='A'
Dizinin indisi
0 1 2 3 4 5 6
skor
A
Dizinin adı
Bellek bölgesi

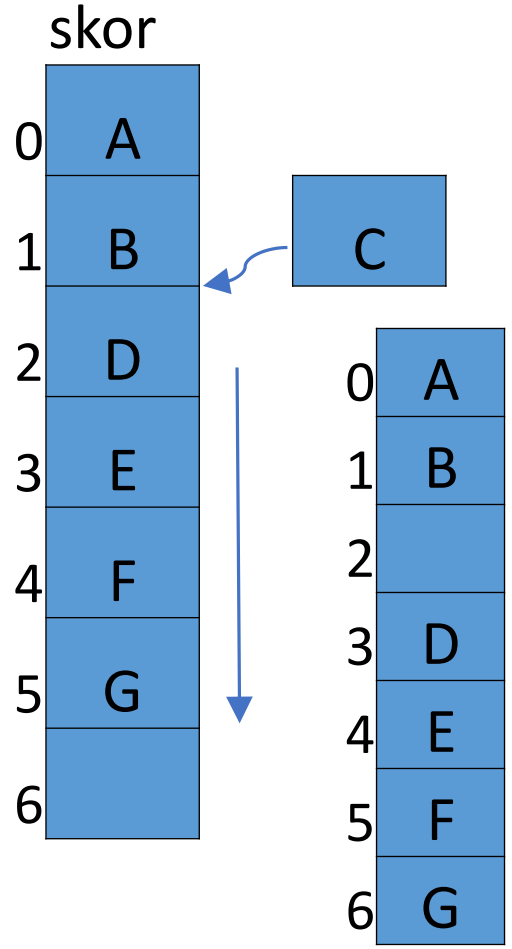
C'de dizi

Liste yapısı

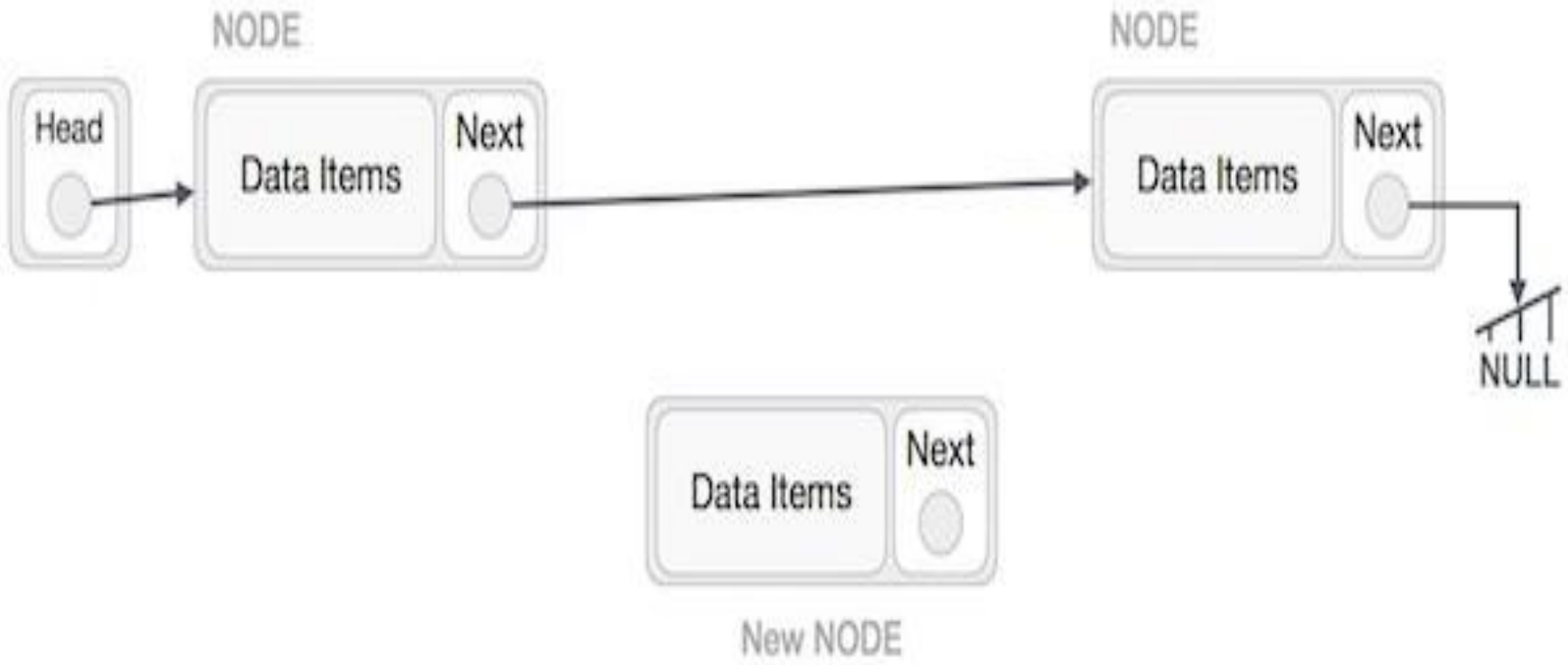
Basit liste yapısı birbiri ile bağlantılı bellek alanlarından oluşur. Önceden kaç tane öğeden oluşacağı belirlenmez. Her öğe farklı veri tipine sahip olabilir. Ayrıca her öğe bir başka liste olabilir.

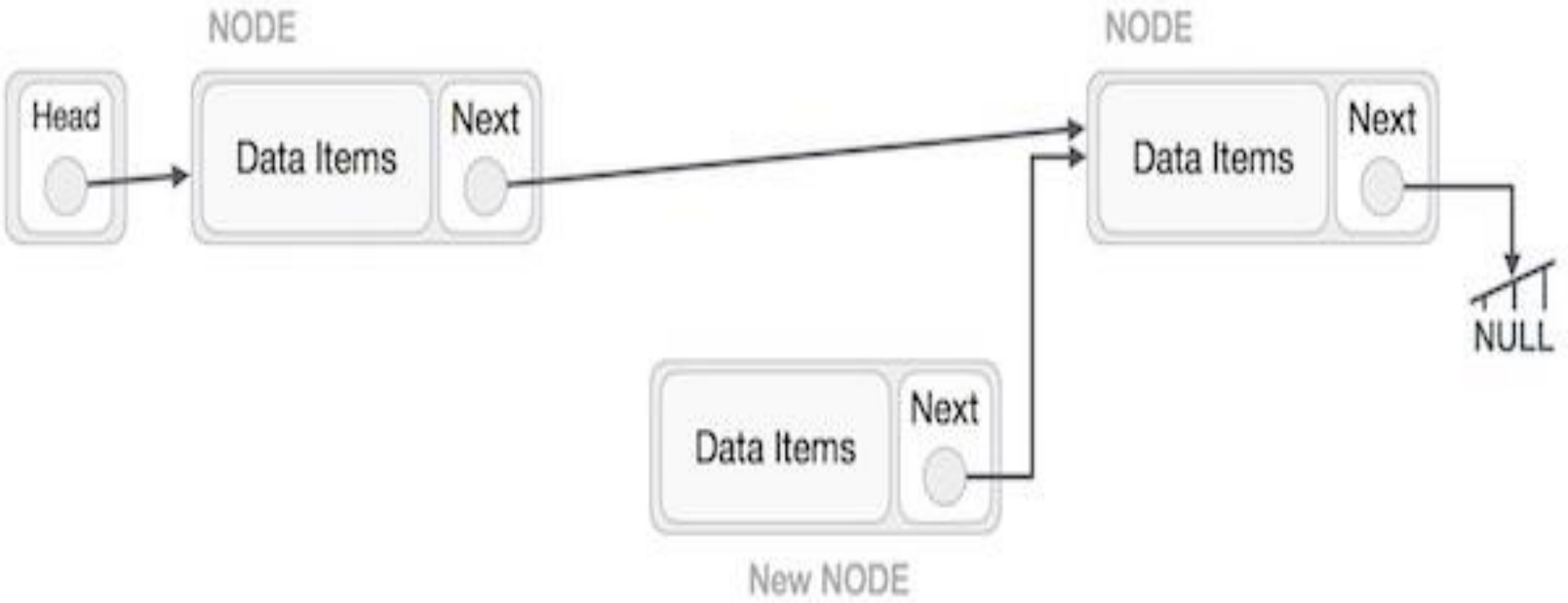
Operasyonlar

Liste yapısında bazı operasyonlar için çok az işlem gerekir. Örneğin araya yeni bir öğe ekleme operasyonu, basit liste veri yapısında 2 tane işlemle yapılabilir. Dizi veri yapısında araya yeni bir öğe ekleme operasyonu değişen sayıda işlem gerektirir. En iyi durum sona öğe eklemektir. Hiçbir öğeyi taşımak gerekmez. En kötü durum ise başa öğe eklemektir. Kendisinden sonra gelen tüm öğeleri bir sonraya taşımak gerekir.

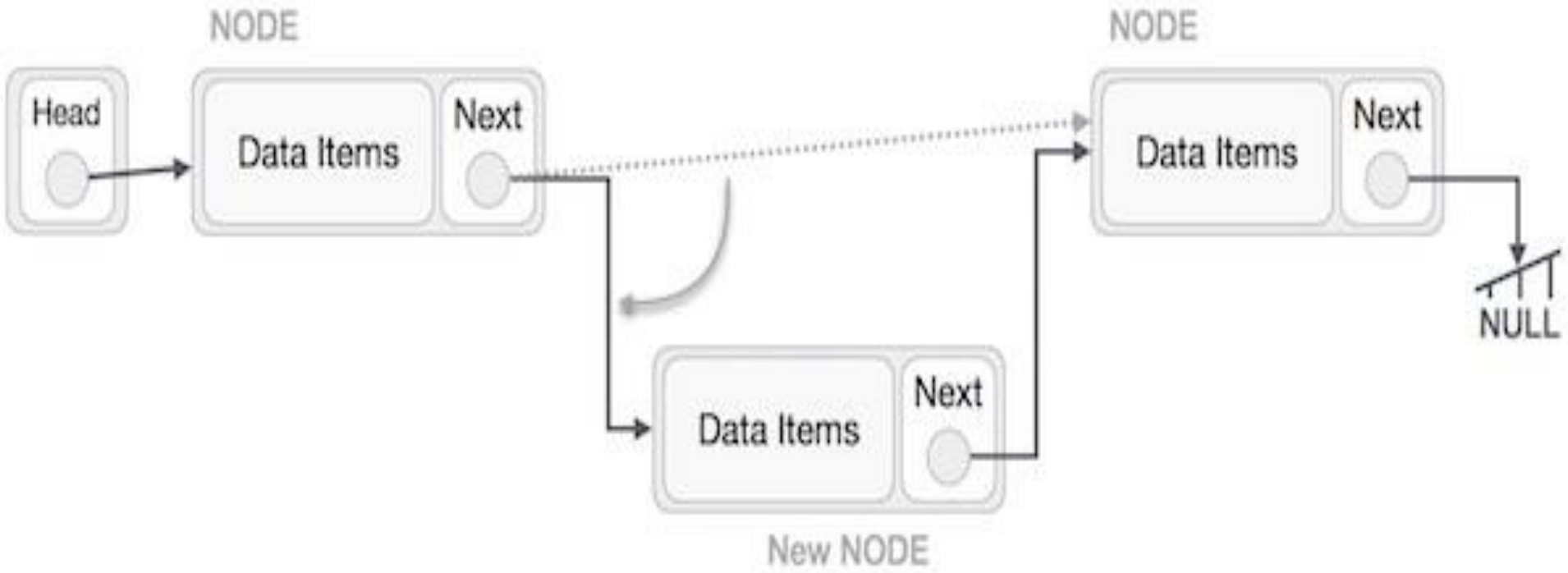


Liste veri yapısında araya düğüm ekleme (insert) işlemi:

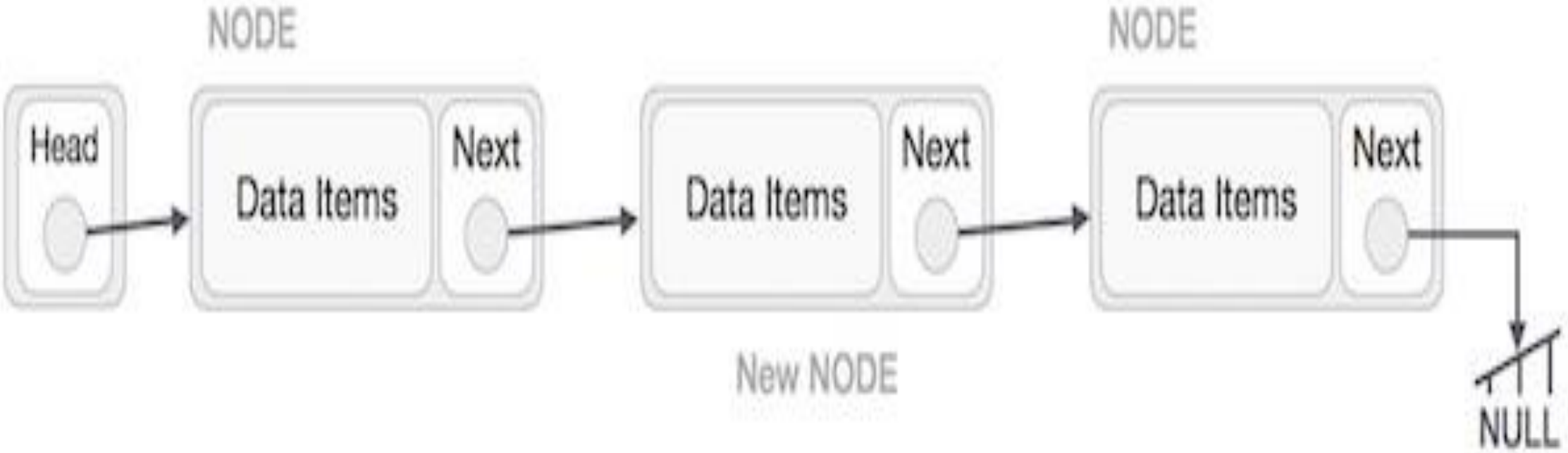




Yeni düğümün bağlacı ayarlanır.



Ek yerinin bağlacı koparılır. Yeni düğümü gösterecek biçimde ayarlanır.


NODE
NODE
Head
Data Items
Next
Data Items
Next
Data Items
Next
New NODE
NULL

- **Listenin oluşturulması ve özellikleri:**

d=[] veya
d=list() d isimli boş liste oluşturur.

d=[10,12,6] Üç tane öğesi olan ilk değerleri 10,12 ve 6 olan liste oluşturur.

d=[10,"abc",20.2,10] liste öğeleri farklı veri tipinde olabilir.

d=[10,12,[23,54],28] listenin öğeleri de liste olabilir.

- **Listeye erişim ve indisler**

d=[56, 98, 23, 86]

a=d[1]

d listesinin 2. öğesi olan 98 değeri a değişkenine aktarılır.

- indisler 0 (sıfır) dan başlar.
- - (eksi) işaretli indis kullanılabilir. Son öğeden başlanarak sayılır.

a=d[-2]

d listesinin sondan 2. öğesi olan 23 değeri a değişkenine aktarılır.

- Olmayan bir indis değeri belirtildiğinde "IndexError" hatası oluşur.

a=d[10]

IndexError: list index out of range

Liste içinde liste olduğu durumda içteki liste için indis belirtilebilir:

```
d=[10,12,[23,54],28]
a=d[2]
```

d listesinin 2 numaralı indisine karşılık gelen değer [23,54] bir listedir.

```
a=d[2][1]
```

d listesinin 2 numaralı indisine karşılık gelen listenin 1 numaralı indisinde yer olan 54 değerine erişilir.

Örnek:

m=[[0,1],[1,0]]

m listesi matrise karşılık:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

gibi düşünülebilir.

- **Metin tipinde olan değişkenler liste ailesindendir.**

  s="Dolar yine yükseldi"

  a=s[0] a değişkeni "D" değerini alır.

  a=s[-1] a değişkeni "i" değerini alır.

- **Range() fonksiyonunun verdiği indisler liste gibi kullanılabilir.**

  a=range(5) a değişkeni "range" tipindedir (Class).

  b=a[1] b değişkeni 1 tamsayı değerini alır.

- **len() fonksiyonu ile listenin uzunluğu öğrenilebilir.**

  d=["y","d","i"]

  u=len(d)

  u değişkeni 3 değerini alır.

- **Listenin ekrana yazılması**

  f=[10,"g",20,95]

  print(f)

  ekranda:

  [10, 'g', 20, 95]

  görünür.

Veya döngü ile yazılabilir:

```
f=[10,"g",20,95]
for i in range(len(f)):
    print(f[i])
```

Veya:

```
f=[10,"g",20,95]
for i in f:
    print(i)
```

Ekranda:

10
g
20
95

Görünür.

- **Liste çoğaltılabilir:**

```
f=[1,3,5]
f=2*f
print(f)
```

```
[1, 3, 5, 1, 3, 5]
```

Veya listeyi çoğaltmadan:

```
f=[1,3,5]
print(2*f)
print(f)
```

```
[1, 3, 5, 1, 3, 5]
[1, 3, 5]
```

- **For döngüsünün indisi döngü içinde yapılan değişiklikten etkilenmez.**

```
f=[1,3,5]
for a in f:
    print(a)
    a+=4
    print(a)
print(f)
```

```
1
5
3
7
5
9
[1, 3, 5]
```

- **Liste indisi aralık gösterecek biçimde yazılabilir.**

liste[başlangıç:bitiş:adım]

```
r=[3,5,7,9,11,13,15]
s=r[1:5:2]
print(s)
```

```
[5, 9]
```

```
r=[3,5,7,9,11,13,15]
s=r[1::2]
print(s)
```

```
[5, 9, 13]
```

```
r=[3,5,7,9,11,13,15]
s=r[::2]
print(s)
```

[3, 7, 11, 15]

```
r=[3,5,7,9,11,13,15]
s=r[1:4]
print(s)
```

[5, 7, 9]

```
r=[3,5,7,9,11,13,15]
s=r[4:]
print(s)
```

[11, 13, 15]

```
r=[3,5,7,9,11,13,15]
s=r[::-1]
print(s)
```

```
[15, 13, 11, 9, 7, 5, 3]
```

```
r=[3,5,7,9,11,13,15]
s=r[::-2]
print(s)
```

```
[15, 11, 7, 3]
```

- **Liste öğelerinin değerleri değişebilir.**

```
r=[3,5,7,9]
r[1]=11
print(r)
```

```
[3, 11, 7, 9]
```

```
r=[3,5,7,9]
r[-1]=11
print(r)
```

```
[3, 5, 7, 11]
```

```
r=[3,5,7,9]
r[1:3]=11,12
print(r)
```

```
[3, 11, 12, 9]
```

- **Listeler birleştirilebilir.**

```
r=[3,5,7,9]
r=r+[11,13]
print(r)
```

```
[3, 5, 7, 9, 11, 13]
```

```
r=[3,5,7,9]
r=r+[11]
print(r)
```

```
[3, 5, 7, 9, 11]
```

```
r=[3,5,7,9]
r=r+11
print(r)
```

```
TypeError: can only concatenate list
```

Sorular

1) Verilen bir sayısal listenin öğelerinin toplamını ve çarpımını bulan bir program yazınız.
2) Verilen bir sayısal listenin en küçük öğesini bulan bir program yazınız.
3) Öğeleri metin tipinde olan bir listenin, uzunluğu 2'den büyük öğelerinin ilk ve son karakteri aynı olanların sayısını bulunuz. Örneğin, liste=["123","121","ahmet","xy","xyx"] ise sonuç 2 olur.
4) Verilen iki tane listenin ortak olan en az 1 tane öğesi varsa "Ortak değer var.", yoksa "Ortak değer yok." mesajı veren bir program yazınız.