

# Python Programlama Dili (Devam)

## Liste kullanım uygulamaları

```
In [1]: # bir elemanın Listedeki sıra numarası  
# baştan itiraben sayılماaya başlandığında  
#      0  1  2  3  4  5  6  7  8  9  
liste1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
  
# sondan başa doğru sayılırsa  
#      -10 -9 -8 -7 -6 -5 -4 -3 -2  -1  
liste1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
In [2]: type(liste1)
```

```
Out[2]: list
```

```
In [3]: # boş liste de oluşturulabilir.  
bos_liste = []  
type(bos_liste)
```

```
Out[3]: list
```

```
In [4]: # aşağıdaki gibi index metodу kullanılarak bir Listeye  
# ait elamanın Listedeki sıra numarası elde edilebilir.  
liste1.index(10)
```

```
Out[4]: 9
```

```
In [5]: liste2 = [1, 2, 3, 4, 5, 1000, 6, 7, 8, 9, 1000]
```

```
In [6]: # aynı değere sahip iki eleman varsa ilk karşılaştığı  
# elamanın sıra numarasını verir.  
liste2.index(1000)
```

```
Out[6]: 5
```

```
In [7]: # Listeye ait olmayan bir eleman sorulursa  
# ValueError: (value) is not in list  
# hatalı verir. Anlamı bu değer listede yok!  
liste2.index(11)
```

```
-----  
ValueError                                                 Traceback (most recent call last)  
<ipython-input-7-311d26d08e22> in <module>  
      2 # ValueError: (value) is not in list  
      3 # hatalı verir. Anlamı bu değer listede yok!  
----> 4 liste2.index(11)
```

```
ValueError: 11 is not in list
```

```
In [8]: # count metodу ile bir Listedede ilgilendiğimiz bir elemandan  
# kaç tane var onu öğrenebiliriz.  
liste2.count(1000)
```

```
Out[8]: 2
```

```
In [9]: # clear metoduya bütün liste elemanları silinir ve  
# liste boş liste halini alır.  
liste2.clear()
```

```
In [10]: # list komutunu kullanarak aşağıdaki gibi  
# bir liste oluşturmak mümkündür.  
liste2 = list("Mesut Karakoç")  
print(liste2)  
['M', 'e', 's', 'u', 't', ' ', 'K', 'a', 'r', 'a', 'k', 'o', 'ç']
```

```
In [11]: # listenin sıralamasını reverse metoduya  
# ters çevirmek mümkündür.  
liste2.reverse()  
print(liste2)  
['ç', 'o', 'k', 'a', 'r', 'a', 'K', ' ', 't', 'u', 's', 'e', 'M']
```

```
In [12]: # copy metoduya bir listenin aynısından  
# yeni bir tane oluşturmak mümkündür.  
liste3 = liste2.copy()  
print(liste3)  
['ç', 'o', 'k', 'a', 'r', 'a', 'K', ' ', 't', 'u', 's', 'e', 'M']
```

```
In [13]: # aşağıdaki uygulamayla listenin yeni bir kopyası OLUŞMAZ!!!  
# Liste2 ve yeni değişken Liste4 aynı LİSTE nesnesine bakmaktadır.  
liste4 = liste2
```

```
In [14]: # Liste2'nin 0. elamanını değiştirmeden önce  
print(liste2[0])  
print(liste4[0])
```

ç  
ç

```
In [15]: # Liste2'nin 0. elamanını değiştirdik.  
liste2[0] = "A"  
print(liste2[0])
```

A

```
In [16]: # Liste4 değişkeni aynı liste nesnesine baktığı için  
# Liste4[0] yeni değeri göstermektedir.  
print(liste4[0])
```

A

```
In [17]: print(liste3[0])
```

ç

```
In [18]: # remove'la istenilen bir eleman silinebilir.  
liste2.remove('a')
```

```
In [19]: # listemizin son hali  
liste2
```

```
Out[19]: ['A', 'o', 'k', 'r', 'a', 'K', ' ', 't', 'u', 's', 'e', 'M']
```

```
In [20]: # Listedeki elamanları sıralar, kendine ait bir algoritmayla.  
liste2.sort()  
print(liste2)
```

```
[ ' ', 'A', 'K', 'M', 'a', 'e', 'k', 'o', 'r', 's', 't', 'u' ]
```

```
In [21]: # verdiğimiz indeks numarasındaki sıraya göre  
# verdiğimiz nesneyi yerleştirir.  
liste2.insert(1, "mesut")
```

```
In [22]: print(liste2)
```

```
[ ' ', 'mesut', 'A', 'K', 'M', 'a', 'e', 'k', 'o', 'r', 's', 't', 'u' ]
```

```
In [23]: # indeksini verdiğimiz elemani siler ve  
# sildiği elamanın kendisini gösterir.  
liste2.pop(1)
```

```
Out[23]: 'mesut'
```

```
In [24]: print(liste2)
```

```
[ ' ', 'A', 'K', 'M', 'a', 'e', 'k', 'o', 'r', 's', 't', 'u' ]
```

```
In [25]: # verilen herhangi bir Python nesnesini  
# listenin son elemani olarak ekler.  
liste2.append("bir başka şey")
```

```
In [26]: liste2
```

```
Out[26]: [ ' ', 'A', 'K', 'M', 'a', 'e', 'k', 'o', 'r', 's', 't', 'u', 'bir başka şey' ]
```

```
In [27]: # append metodunun eşdeğeri aşağıdaki gibidir  
liste2 = liste2 + ['yeni bir şey']
```

```
In [28]: print(liste2)
```

```
[ ' ', 'A', 'K', 'M', 'a', 'e', 'k', 'o', 'r', 's', 't', 'u', 'bir başka şey', 'yeni  
bir şey' ]
```

```
In [29]: # extend ile bir listenin sonuna bir başka  
# listenin elemanları ilave edilebilir.  
liste2.extend([1, 2, 3, 4, 5])
```

```
In [30]: print(liste2)
```

```
[ ' ', 'A', 'K', 'M', 'a', 'e', 'k', 'o', 'r', 's', 't', 'u', 'bir başka şey', 'yeni  
bir şey', 1, 2, 3, 4, 5]
```

## Listeyle başka bazı uygulamalar

```
In [31]: type(range(1, 12, 2))
```

```
Out[31]: range
```

```
In [32]: liste5 = list(range(1, 12, 2))
print(liste5)
```

```
[1, 3, 5, 7, 9, 11]
```

```
In [33]: # Listenin bir elamanına aşağıdaki
# gibi sıra numarası kullanılarak ulaşılabilir.
liste5[3]
```

```
Out[33]: 7
```

```
In [34]: # Listeler tersten de sıralıdır.
# Böylece listenin son elamanına
# şu şekilde ulaşılabilir.
liste5[-1]
```

```
Out[34]: 11
```

```
In [35]: # belirli bir sıra aralığındaklere ulaşım
liste5[2:5]
```

```
Out[35]: [5, 7, 9]
```

```
In [36]: # iki liste bir birine (+) işlemcisisiyle de eklenebilir.
liste6 = list(range(1,12,2))
liste7 = list(range(0,12,2))
```

```
In [37]: print(liste6)
print(liste7)
print(liste6 + liste7)
print(liste7 + liste6)
```

```
[1, 3, 5, 7, 9, 11]
[0, 2, 4, 6, 8, 10]
[1, 3, 5, 7, 9, 11, 0, 2, 4, 6, 8, 10]
[0, 2, 4, 6, 8, 10, 1, 3, 5, 7, 9, 11]
```

## Liste ve döngü içeren bir örnek

```
In [38]: #  $y = x^{**2}$ 'yi hesaplayan bir program

sayi_listesi = list(range(11))
print("listemiz = ", sayi_listesi)

for x in sayi_listesi:
    print(x, x**2)

listemiz =  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
0 0
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
10 100
```

```
In [39]: x = list(range(11))
x2 = []
for xi in x:
    x2 = x2 + [xi**2]

print(x)
print(x2)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
In [40]: x = list(range(11))
x2 = []
for xi in x:
    x2.append(xi**2)

print(x)
print(x2)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
In [41]: x = list(range(11))
x2 = []
for xi in x:
    print(x2)
    x2 = x2 + [xi**2]

print(x)
print(x2)
```

```
[]
[0]
[0, 1]
[0, 1, 4]
[0, 1, 4, 9]
[0, 1, 4, 9, 16]
[0, 1, 4, 9, 16, 25]
[0, 1, 4, 9, 16, 25, 36]
[0, 1, 4, 9, 16, 25, 36, 49]
[0, 1, 4, 9, 16, 25, 36, 49, 64]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
In [42]: x = list(range(11))
y = []
for xi in x:
    y = y + [xi**2]

print(" x  x^2")
for i, xi in enumerate(x):
    print("{:>2d} {:>4d}".format(xi, y[i]))
```

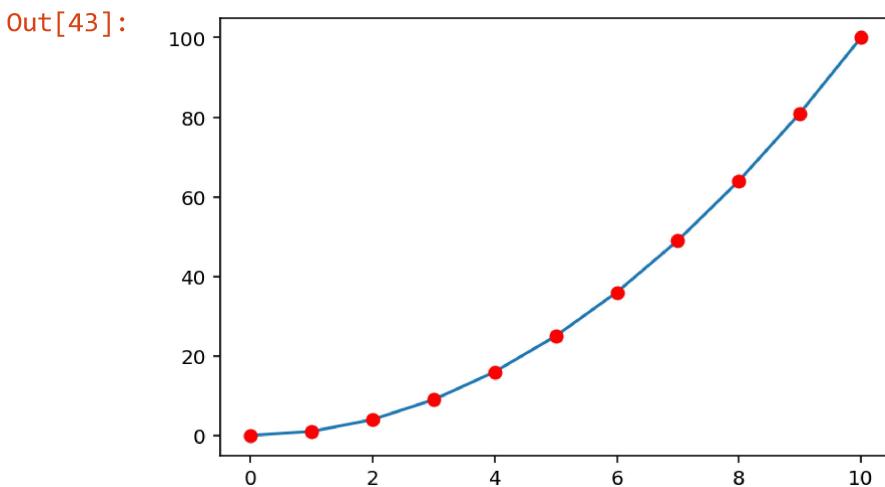
```
x  x^2
0   0
1   1
2   4
3   9
4   16
5   25
6   36
7   49
8   64
9   81
10  100
```

## Listeleri kullanarak çizim yapılabilir

Aşağıda  $y = x^2$  grafiği görülmektedir.

```
In [43]: import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(x, y)
plt.plot(x, y, "ro")
plt.show()
```



## Fibonacci sayılarından oluşan seri

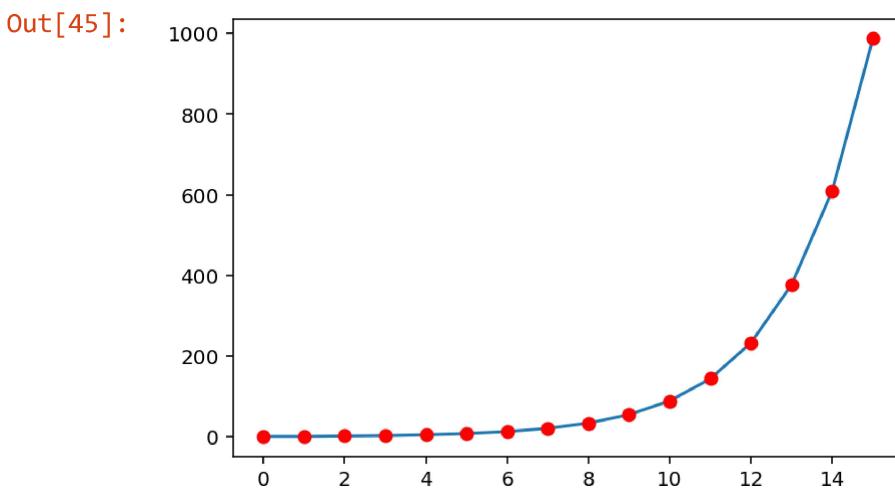
```
In [44]: n = list(range(16))
Fn = [1, 1]
for ni in n[2:]:
    Fn = Fn + [Fn[-2] + Fn[-1]]

print(" n  Fn")
for ni in n:
    print("{:>2d} {:>3d}".format(ni, Fn[ni]))
```

```
n  Fn
0   1
1   1
2   2
3   3
4   5
5   8
6  13
7  21
8  34
9  55
10 89
11 144
12 233
13 377
14 610
15 987
```

```
In [45]: import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(n, Fn)
plt.plot(n, Fn, "ro")
plt.show()
```



```
In [46]: print(n)
print(Fn)

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987]
```

## Liste kullanmadan Fibonacci sayıları

```
In [47]: F0, F1 = 1, 1
```

```
for i in range(14):
    F2 = F0 + F1
    F0 = F1
    F1 = F2
    print(F2)
```

```
2
3
5
8
13
21
34
55
89
144
233
377
610
987
```

## List comprehensions (Python'a özel bir liste oluşturma yöntemi)

```
In [48]: kare = []
```

```
for i in range(1, 11):
    kare.append(i**2)
#kare = kare + [i**2]

print(kare)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
In [49]: kare2 = [i**2 for i in range(1,11)]
print(kare2)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

## Python'da Fonksiyon

### Ürettigi veriyi ana programa aktarmayan fonksiyon örneği

```
In [50]: def topla():
```

```
    a = 5
    b = 10
    toplam = a + b
    print(a, b, toplam)
```

```
In [51]: # a ve b yerel (Local) değişkenlerdir.  
print(a, b)
```

```
NameError Traceback (most recent call last)  
<ipython-input-51-0751adecab6e> in <module>  
      1 # a ve b yerel (local) değişkenlerdir.  
----> 2 print(a, b)  
  
NameError: name 'a' is not defined
```

```
In [52]: topla()
```

```
5 10 15
```

```
In [53]: print(topla())
```

```
5 10 15  
None
```

## Yerel ve genel değişken örnekleri (Local and Global variables)

Python öncelikle varsa yerel değişkeni dikkate alır.

```
In [54]: a = 7 # genel değişken  
def topla2():  
    b = 10 # yerel değişken  
    toplam = a + b  
    print(a, b, toplam)
```

```
In [55]: topla2()
```

```
7 10 17
```

```
In [56]: a = 7 # genel değişken  
def topla3():  
    a = 5 # yerel değişken  
    b = 10  
    toplam = a + b  
    print(a, b, toplam)
```

```
In [57]: topla3()
```

```
5 10 15
```

## Fonksiyonlarda "return" ifadesinin görevi

Bir fonksiyon içinde üretilen veri return ifadesiyle ana programa aktarılabilir.

```
In [58]: def topla4():  
    a = 5  
    b = 10  
    toplam = a + b  
    return toplam
```

```
In [59]: topla4()
```

```
Out[59]: 15
```

```
In [60]: a_ve_b_toplami = topla4()
print(a_ve_b_toplami)
```

```
15
```

## Fonksiyonlarda parametre kullanımı

```
In [61]: def topla5(a, b):
    toplam = a + b
    return toplam
```

```
In [62]: topla5(12, 12)
```

```
Out[62]: 24
```

## Fibonacci sayılarını liste olarak üreten bir fonksiyon

```
In [63]: def fibonacci(nmax):
    n = list(range(nmax))
    Fn = [1, 1]
    for ni in n[2:]:
        Fn = Fn + [Fn[-2] + Fn[-1]]
    return Fn
```

```
In [64]: fibonacci(5)
```

```
Out[64]: [1, 1, 2, 3, 5]
```

```
In [65]: def fibonacci2(n0, nmax):
    n = list(range(nmax))
    Fn = [1, 1]
    for ni in n[2:]:
        Fn = Fn + [Fn[-2] + Fn[-1]]
    return Fn[n0:nmax]
```

```
In [66]: print(fibonacci2(0, 10))
print(fibonacci2(5, 10))
print(fibonacci2(9, 10))
```

```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

```
[8, 13, 21, 34, 55]
```

```
[55]
```

```
In [67]: def fibonacci22(n0, nmax):
    return fibonacci(nmax)[n0:nmax]
```

```
In [68]: print(fibonacci22(0, 10))
print(fibonacci22(5, 10))
print(fibonacci22(9, 10))
```

```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

```
[8, 13, 21, 34, 55]
```

```
[55]
```

## Bir fonksiyonun kullanımı hakkında bilgi verme

```
In [69]: def fibonacci3(n0, nmax):
    """
        fibonacci fonksiyonu fibonacci sayılarını hesaplar.
        kullanımı: fibonacci(n0, nmax)
    """
    n = list(range(nmax))
    Fn = [1, 1]
    for ni in n[2:]:
        Fn = Fn + [Fn[-2] + Fn[-1]]
    return Fn[n0:nmax]
```

```
In [70]: fibonacci3()
```

```
-----  
TypeError                                     Traceback (most recent call last)  
<ipython-input-70-20d3984f55b0> in <module>  
----> 1 fibonacci3()  
  
TypeError: fibonacci3() missing 2 required positional arguments: 'n0' and 'nmax'
```

```
In [71]: fibonacci3.__doc__
```

```
Out[71]: '\n    fibonacci fonksiyonu fibonacci sayılarını hesaplar.\n    kullanımı: fibonacci\n    (n0, nmax)\n    '
```

```
In [72]: print(fibonacci3.__doc__)
```

```
fibonacci fonksiyonu fibonacci sayılarını hesaplar.  
kullanımı: fibonacci(n0, nmax)
```

```
In [0]: fibonacci3?
```

## Fonksiyonlarda ön tanımlı parametreler

```
In [73]: def fibonacci4(n0=5, nmax=10):
    """
        fibonacci fonksiyonu fibonacci sayılarını hesaplar.
        kullanımı: fibonacci(n0, nmax)
    """
    n = list(range(nmax))
    Fn = [1, 1]
    for ni in n[2:]:
        Fn = Fn + [Fn[-2] + Fn[-1]]
    return Fn[n0:nmax]
```

```
In [74]: fibonacci4()
```

```
Out[74]: [8, 13, 21, 34, 55]
```

```
In [75]: fibonacci4(3,6)
```

```
Out[75]: [3, 5, 8]
```

```
In [76]: fibonacci4(3, nmax=6)
```

```
Out[76]: [3, 5, 8]
```

```
In [77]: fibonacci4(n0=3, nmax=6)
```

```
Out[77]: [3, 5, 8]
```

## Python'da Kütüphane Kullanımı

### numpy kütüphanesi

<http://www.numpy.org> (<http://www.numpy.org>)

nümerik hesaplamalar için gerekli araçları içerir.

### sympy kütüphanesi

<http://www.sympy.org> (<http://www.sympy.org>)

analitik hesaplamalar için gerekli araçları içerir.

### scipy kütüphanesi

<https://www.scipy.org/> (<https://www.scipy.org/>)

çeşitli bilimsel hesaplama araçlarını içerir.

### matplotlib kütüphanesi

<https://matplotlib.org/> (<https://matplotlib.org/>)

hesaplanan veya bir şekilde elde edilen verilerin çizimi için kullanılabilir.

### math ve cmath kütüphaneleri

math reel matematik araçlar içerir, cmath ise karmaşık matematik araçlar içerir.

## Kütüphaneleri çağrıma yöntemleri

### A) import (kütüphane adı)

bir kütüphanedeki bütün araçları çağrıır

örnek: sympy kütüphanesinden sinüs fonksyonunu kullanalım.

```
In [78]: import sympy
```

```
sympy.sin(3.14159)
```

```
Out[78]: 2.65358979335273 · 10-6
```

```
In [79]: sympy.sin(3.14159/2)
```

```
Out[79]: 0.9999999999912
```

In [80]: `import sympy`

```
print(sympy.sin(3.14159))
print(sympy.sin(sympy.pi))
print(sympy.pi)
print(sympy.N(sympy.pi, 100))
print(str(sympy.N(sympy.pi, 1000))[-1])
```

```
2.65358979335273e-6
0
pi
3.1415926535897932384626433832795028841971693993751058209749445923078164062862089986
28034825342117068
8
```

#### B) import (kütüphane adı) as (kısa/yeni ad)

In [81]: `import sympy as s`

```
print(s.sin(s.pi))
```

```
0
```

In [82]: `import numpy as np`

```
print(np.sin(np.pi))
```

```
1.2246467991473532e-16
```

In [83]: `np.pi`

Out[83]: 3.141592653589793

#### C) from (kütüphane adı) import \*

kütüphanedeki bütün araçları çağırır

In [84]: `from sympy import *`

```
sin(pi)
```

Out[84]: 0

In [85]: `from numpy import *`

```
sin(pi)
```

Out[85]: 1.2246467991473532e-16

#### D) from (kütüphane adı) import (araç adı)

In [86]: `from sympy import cos, pi`

In [87]: `cos(pi)`

Out[87]: -1

```
In [88]: sin(pi)
```

```
-----  
AttributeError Traceback (most recent call last)  
AttributeError: 'Pi' object has no attribute 'sin'
```

The above exception was the direct cause of the following exception:

```
TypeError Traceback (most recent call last)  
<ipython-input-88-69d9a90af4a5> in <module>  
----> 1 sin(pi)
```

```
TypeError: loop of ufunc does not support argument 0 of type Pi which has no callable  
e sin method
```

## E) from (kütüphane adı) import (araç adı) as (yeni/kısa ad)

```
In [89]: from sympy import cos as c  
from sympy import pi
```

```
In [90]: c(pi)
```

```
Out[90]: -1
```

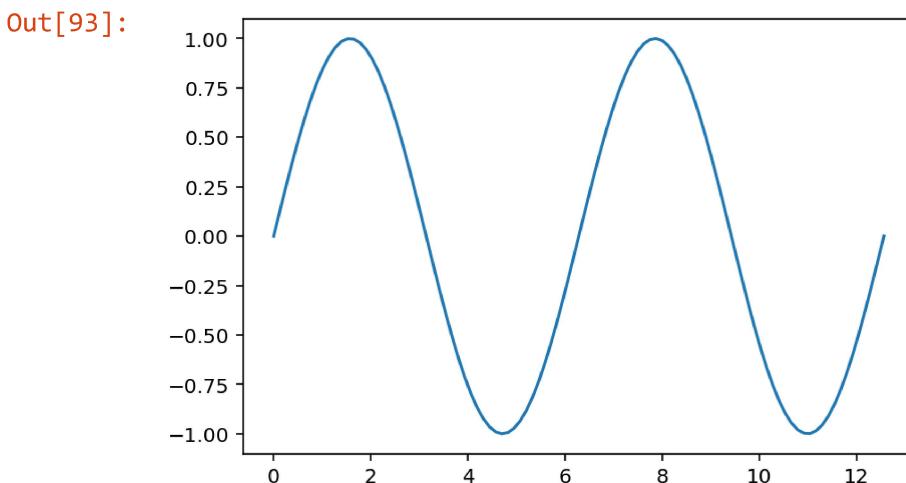
## Uygulama

$\sin(x)$  fonksiyonunu  $x \in [0, 4\pi]$  aralığında çizdirelim.

```
In [91]: pi = 3.14159  
  
xmin = 0  
xmax = 4*pi  
nx = 100  
dx = (xmax - xmin)/(nx-1)  
  
xlist = []  
for i in range(nx):  
    #xlist.append(xmin + i*dx)  
    xlist = xlist + [xmin + i*dx]
```

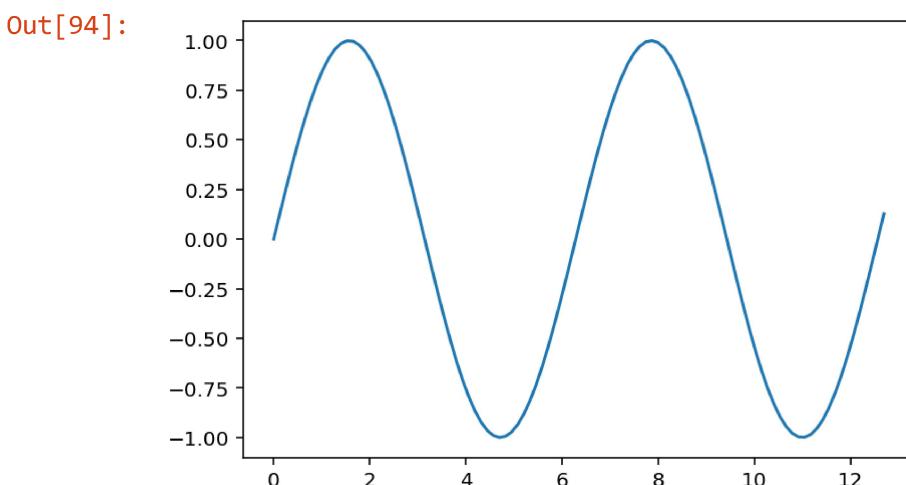
```
In [92]: from numpy import sin  
  
ylist = []  
for xi in xlist:  
    ylist = ylist + [sin(xi)]
```

```
In [93]: import matplotlib.pyplot as plt  
%matplotlib inline  
  
plt.plot(xlist, ylist)  
plt.show()
```



Yukarıdaki süreci tek bir hücrede de halledebiliriz.

```
In [94]: from numpy import sin  
import matplotlib.pyplot as plt  
%matplotlib inline  
  
pi = 3.14159  
  
xmin = 0  
xmax = 4 * pi  
nx = 100  
dx = (xmax - xmin)/(nx-1)  
  
xlist = []  
for i in range(nx+1):  
    #xlist.append(xmin + i*dx)  
    xlist = xlist + [xmin + i*dx]  
  
ylist = []  
for xi in xlist:  
    ylist = ylist + [sin(xi)]  
  
plt.plot(xlist, ylist)  
plt.show()
```



In [95]:

```
from numpy import sin
import matplotlib.pyplot as plt
%matplotlib inline

pi = 3.14159

xmin = 0
xmax = 4 * pi
nx = 100
dx = (xmax - xmin)/(nx-1)

xlist = []
for i in range(nx+1):
    #xlist.append(xmin + i*dx)
    xlist = xlist + [xmin + i*dx]

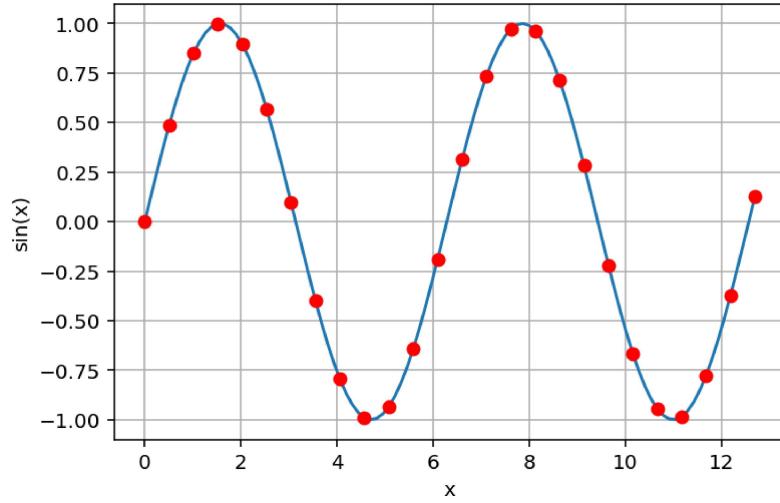
ylist = []
for xi in xlist:
    ylist = ylist + [sin(xi)]

plt.plot(xlist, ylist)

plt.plot(xlist[::4], ylist[::4], 'ro')

plt.xlabel('x')
plt.ylabel('sin(x)')
plt.grid()
plt.show()
```

Out[95]:



In [96]:

```
from numpy import exp
import matplotlib.pyplot as plt
%matplotlib inline

xmin = 0
xmax = 5
nx = 100
dx = (xmax - xmin)/(nx-1)

xlist = []
for i in range(nx+1):
    #xlist.append(xmin + i*dx)
    xlist = xlist + [xmin + i*dx]

ylist = []
for xi in xlist:
    ylist = ylist + [exp(xi)]

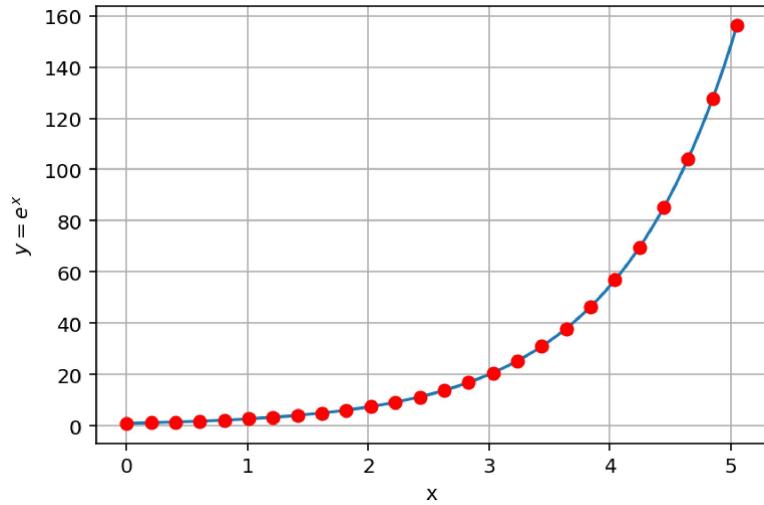
plt.plot(xlist, ylist)

plt.plot(xlist[::4], ylist[::4], 'ro')

plt.xlabel('x')
plt.ylabel('$y = e^x$')

plt.grid()
plt.show()
```

Out[96]:



In [97]:

```
from numpy import exp
import matplotlib.pyplot as plt
%matplotlib inline

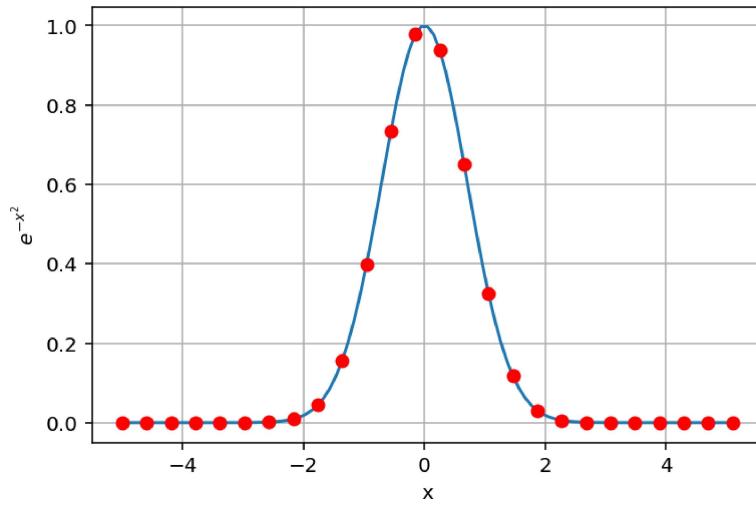
xmin = -5
xmax = 5
nx = 100
dx = (xmax - xmin)/(nx-1)

xlist = []
for i in range(nx+1):
    #xlist.append(xmin + i*dx)
    xlist = xlist + [xmin + i*dx]

ylist = []
for xi in xlist:
    ylist = ylist + [exp(-xi**2)]

plt.plot(xlist, ylist)
plt.plot(xlist[::4], ylist[::4], 'ro')
plt.xlabel('x')
plt.ylabel('$e^{-x^2}$')
plt.grid()
plt.show()
```

Out[97]:



In [98]:

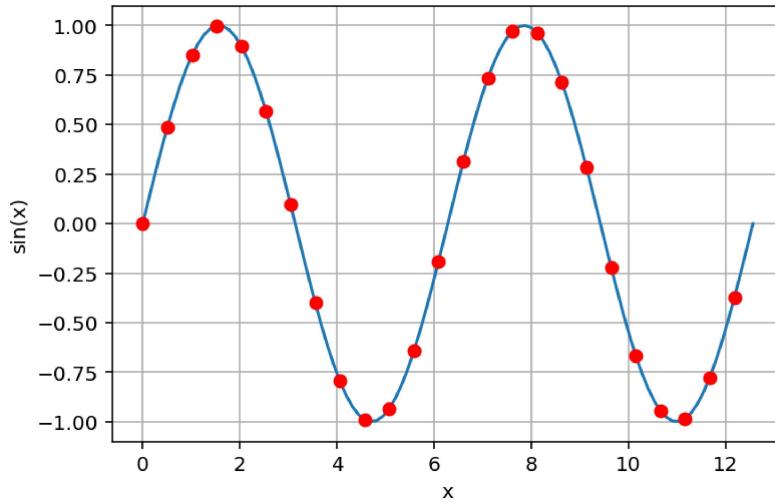
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

xmin = 0
xmax = 4*np.pi
nx = 100

xlist = np.linspace(xmin, xmax, nx)
ylist = np.sin(xlist)

plt.plot(xlist, ylist)
plt.plot(xlist[::4], ylist[::4], 'ro')
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.grid()
plt.show()
```

Out[98]:



```
In [99]: from numpy import sin, cos, sqrt, pi
import matplotlib.pyplot as plt
%matplotlib inline

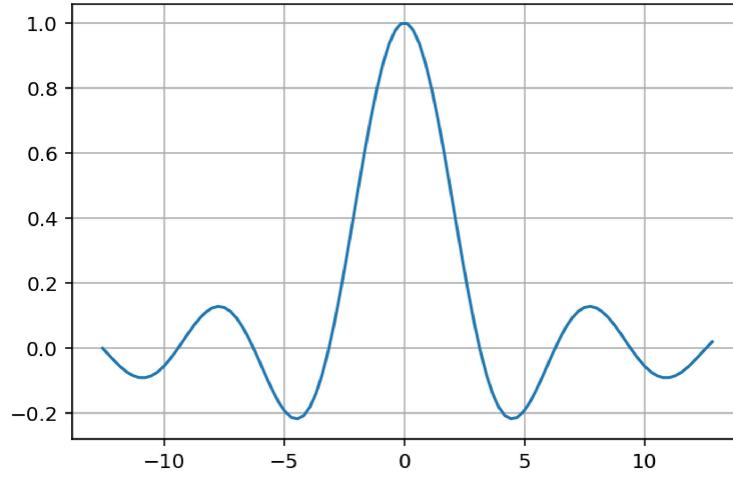
xmin = -4*pi
xmax = 4*pi
nx = 100
dx = (xmax - xmin)/(nx-1)

xlist = []
for i in range(nx+1):
    xlist = xlist + [xmin + i*dx]

ylist = []
for xi in xlist:
    ylist = ylist + [sin(xi)/xi]

plt.plot(xlist, ylist)
plt.grid()
plt.show()
```

Out[99]:



## Rastgele (Random) sayıları elde etme

```
In [100]: import random as rnd

# 0 ile 1 aralığında
# sözde rastgele sayı üretir
rnd.random()
```

Out[100]: 0.9848828557531956

```
In [101]: # 0 ile 10 aralığında rastgele
# sayı elde etme
10*rnd.random()
```

Out[101]: 9.351183993231606

```
In [102]: # 1 ile 100 aralığında rastgele
# tam sayı elde etme
rnd.randint(1, 100)
```

Out[102]: 91

## Monte Carlo yöntemiyle $\pi$ sayısı hesabı

[https://tr.wikipedia.org/wiki/Monte\\_Carlo\\_benzetimi](https://tr.wikipedia.org/wiki/Monte_Carlo_benzetimi) ([https://tr.wikipedia.org/wiki/Monte\\_Carlo\\_benzetimi](https://tr.wikipedia.org/wiki/Monte_Carlo_benzetimi))

Monte Carlo benzetimi. Rastgele üretilen sayılardan faydalılarak istatistiksel simülasyonlar Monte Carlo metoduyla yapılır. Monte-Carlo Nicholas Constantine Metropolis (1915-1999) tarafından bulunmuştur ve Atom bombasının geliştirildiği Los Alamos Ulusal Labratuvarında, bombanın patlamasından sonra dağılan nötronlara karşı kalkan modellemek için Stanislaw Ulam tarafından günümüzə taşınmıştır.

Deney girdileri belirli olmayan, kesin olmayan bir şekilde gelmesi bekleniyorsa ve dağılım bir fonksiyonla hesaplanabilecekse kullanılır. Monte Carlo, rastgele sayıları baz alarak tahmini sistemleri modeller. Hücre Simülasyonu, Borsa Modelleri, Dağılım Fonksiyonları, Sayısal Analiz, Doğal olayların simülasyonu, Atom ve Molekül Fiziği, Nükleer Fizik ve Yüksek Enerji Fiziği modellerini test eden simülasyonlar, Deneylerde kullanılan aletlerin simülasyonu (örneğin bir madde içerisinde x işinlarının dağılımı).

Yukarıdaki modellerde tahminler yapabilmek için; Rastgele sayı üretilir, bunun için programlama bilgisi gerekmektedir.

Programlama: Temel düzeyde Monte Carlo programları öğrenildikten sonra, Monte Carlo üreteçlerini oluşturabilmek gerekmektedir. Teorik eğitimin yanında örneğin C, C++ gibi bilimsel çalışmalarında sıkılıkla kullanılan programları iyi derecede bilmek gerekmektedir. Teorik: Programlama aşamasına geçmeden önce problemi ya da deney sistemimizin teorisini çok iyi oluşturmamız gerekmektedir. Aşağıda Pi sayısının Monte Carlo Yöntemi ile hesaplanması örneği C++ kodları ile verilmiştir.

```
In [103]: import numpy as np
import random as rnd
import matplotlib.pyplot as plt
%matplotlib inline

r = 2 # daire yarı çapı
atis = 1000

def mnc(x):
    return (-1)**rnd.randint(1,10) * x*rnd.random()

xlist = [mnc(r) for i in range(atis)]
ylist = [mnc(r) for i in range(atis)]

# dairenin içinde kalanları sayalım
daire = 0
for i in range(atis):
    ri = (xlist[i]**2 + ylist[i]**2)**.5
    if ri<=r:
        daire = daire + 1

pi_sayisi = 4*daire/atis

print(pi_sayisi)
```

3.104

```
In [104]: theta = np.linspace(0, 2*np.pi)
xd = r*np.cos(theta)
yd = r*np.sin(theta)

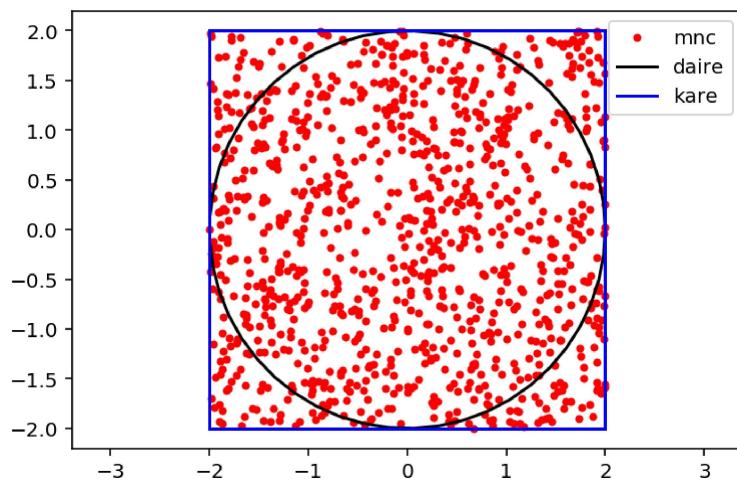
plt.plot(xlist, ylist, 'r.', label='mnc')

plt.plot(xd, yd, color="black", label='daire')

plt.plot([-r, -r], [-r, r], color="blue", label='kare')
plt.plot([-r, r], [r, r], color="blue")
plt.plot([r, r], [r, -r], color="blue")
plt.plot([r, -r], [-r, -r], color="blue")

plt.axis('equal')
plt.legend(loc='best')
plt.show()
```

Out[104]:



In [0]: