```python
# sürekli dağılımlar: tam sayı olmadığı için yani olasılık tam sayı ile ifade
#edilmedeiğinde bir çzigi şeklinde ifade edilir. sürekli düzgün dağılım adı verilir.
# olasılık altında kalan alanla hesaplanır!
# olasılık dağılımında altta kalan tüm alan 1'e eşittir
# bimodal dist: sürekli dağılımlar bazi değerlerinin daha yüksek olduğu tekdüze olmayan biçimler alabilir
# normal dağılım: en popüler. şekli çan eğrisi şeklindedir: tansiyon ve emeklilik yaşı. verilerin nprmal dağılıma uydurulmaya çalışılır.
#her dağılımın cdf fonksiyonu kullanılır latta kalan alan hesaplanırken

from scipy.stats import uniform #altta kalan. başka dağılımlar için de geçerlidir bu
uniform.cdf(7,0,12)#7den küçük olma olasılığı, 0'dan 12'ye kadar!
```

```
0.5833333333333334
```

```python
1 - uniform.cdf(7,0,12)
```

```
0.41666666666666663
```

```python
uniform.cdf(7,0,12) - uniform.cdf(7,4,12)
```

```
0.33333333333333337
```

```python
# uniform dağılımına göre rastgele sayılar üretme
uniform.rvs(0,5, size=10) #0 ile 5 arasından 10 rastgele sayı üret bir de random_state var her seferinde aynı sayıları random olarak vermeye
yarar
```

```
array([0.43291092, 4.02307956, 1.97213263, 1.01207317, 1.68885555,
       0.71401487, 3.36869825, 3.69077878, 0.39814114, 4.06739263])
```

```python
#binominal dist: bir bağımsız denemedeki başarı sayısının olasılığını tanımlar
# n= kaç defa p=başarılı olma olasılığı. dicrete bir
# yine alan hesaplanır! expected value = ortalama değerdir -> binomda n * p 'dir!

#random binom
binom.rvs(n,p, loc=0,size=1,random_state=None)

from scipy.stats import binom
binom.rvs(1, 0.5, size=1) # bir parayı atıyoruz, 0.5 olasılık var ve 1 defa atoıyoruz
```

```
array([1])
```

```python
binom.rvs(1, 0.5, size=8)
```

```
array([1, 0, 0, 0, 1, 0, 1, 0])

binom.rvs(8, 0.5, size=1) # 8 parayı havaya atıyoruz

array([6])

binom.rvs(3, 0.5, size=10) # 3 tane parayı 10 kere havaya attığında
kaç tane başarılı geldi?

array([0, 0, 2, 1, 1, 2, 2, 2, 2, 1])

#bir tarafı daha ağır para. yüzde 25 tura
binom.rvs(3,0.25,size=10) #tura gelme olasılığı, eğer yazı isteseydik
yüzde 75 olurdu

# olasılık kğtle fonksiyonu - kesikli dağılım. belli bir olasılıkla
kesikli değer alma ihtimalini açıklar. kesikli bir rastgele değişkenin
belirli bir değeri alma olasılığını hesaplar
# scipy.stats.pmf(k:başarı sayısı, n,p:başarı, loc)

# 10 jetondan 7sinin yazı gelme olasılığı
binom.pmf(7,10,0.5)

0.11718750000000004

# 7 ve daha az gelme olasılığı!
binom.cdf(7,10,0.5)

0.9453125

# 7'den fazla yazı gelme
1 - binom.cdf(7,10,0.5)

0.0546875

# normal dağılım: gerçek dünya verisi
# simetriktir.
# eğrinin altında kalan alan 1'e eşittik (tüm olasılık dağılımlarında)
# uçları öyle görünse de olasılık hiçbir zaman 0'a ulaşmaz
# orta çizgi ortalamayı (expected value)'yu gösterir
# m (mü) = ort , v = standart sapma
# verilerin yüzde 68'i bir standart sapma altında ve üstündedir
# iki ss üstünde ve altında yüzde 95
# yüzde 99.7 3 ss altında ve üstündedir
# 3 ss altının da altındaysa outline (?) iqr yerine ss de
kullanılabilir yani eğer veriler normal dağılıyorsa
# 68-95-99.7 -> 3ss ifade eder
# ort 0 ve ss 1 olan dağılıma standart-normal dağılım denir

# ort 161 cm ve ss 7 olan , 154den kısa kadınlar
from scipy.stats import norm
norm.cdf(154,161,7) #loc-> ort scale -> ss
```

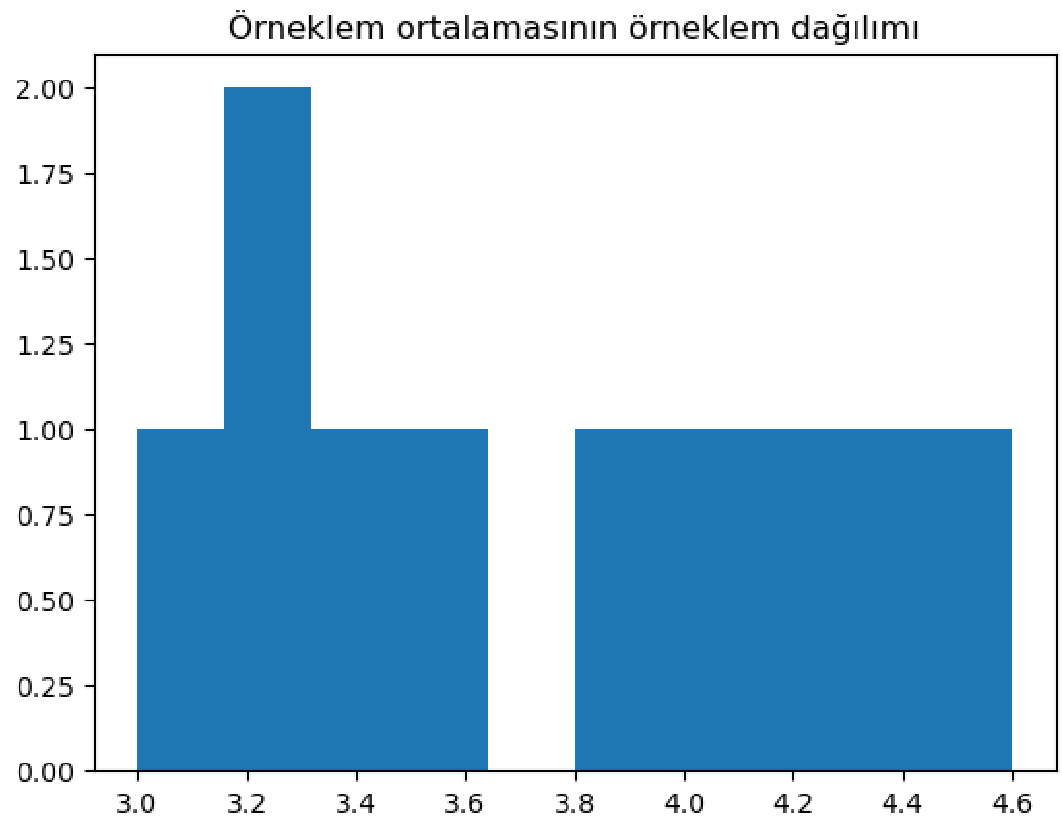
```
0.15865525393145707

#154den uzun
1 -norm.cdf(154,161,7)

0.8413447460685429

#154 157 arasındakiler
norm.cdf(157,161,7) - norm.cdf(154, 161,7)

0.1251993291672192

# ppf -> ters kümülatif dağılım
# belirli bir olasılığa karşılık gelen x'i bulmaya yarar. verilen bir
olasılığa karşılık gelen değeri bulur
norm.ppf(0.9,161,7) #kadınların yüzde 90ı 169.97den kısadır

169.9708609588122

#kadınların yüzden 90ı şu boydan uzundur
norm.ppf((1 - 0.9),161,7)

152.0291390411878

norm.rvs(161,7,size=10) #

array([168.76324661, 169.15291056, 171.86421587, 160.87714763,
       175.14265619, 161.2601973 , 150.97014182, 149.42614535,
       169.17858999, 161.53274865])

# çarpıklık _> veri simetrik değilse pozitif çarpık ya da negatif
çarpık
# sağdan tokat attıysam sağ çarpık -> pozitif çarpık. soldan tokat
atıysaç sol çarpık ->negatif çarpık
# basıklık -> değılımdakş aşırı değerleri açıklama. 3 çeşit

# merkezi limit teoremi
# bir zar atıldığında her yğzün değeri ile gelme olasılığı toplanırsa
-> 3.67 exprected value
# ortalama gibi bir özet istatistiğinin dağılımına örneklem dağılımı
denir. ne kadar çok yapılırsa o kadar normal dağılıma benzer. buna
merkezi limit teorisi denir.
# örneklem üyüklüğü arttıkça normal dağılıma yaklaşır -> merkezi limit
teoremi
# bu işi kod yazarak yapın dedi
# ortalama
# ss için de geçerlidir
# oran için de geçerlidir
# örnekleme dağılımları normsl oldupndan bir dağılımın ortalması ss ve
oranı hakkında bilgi edilebilir eğer yeterince fazla veri varsa ya da
örneklee güzel alındıysa
```

```python
import pandas as pd
import numpy as np
die = pd.Series([1,2,3,4,5,6])
samp_5 = die.sample(5,replace=True) #yerine koyarak örnekleme yapar
samp_5
```

```
4    5
0    1
0    1
3    4
0    1
dtype: int64
```

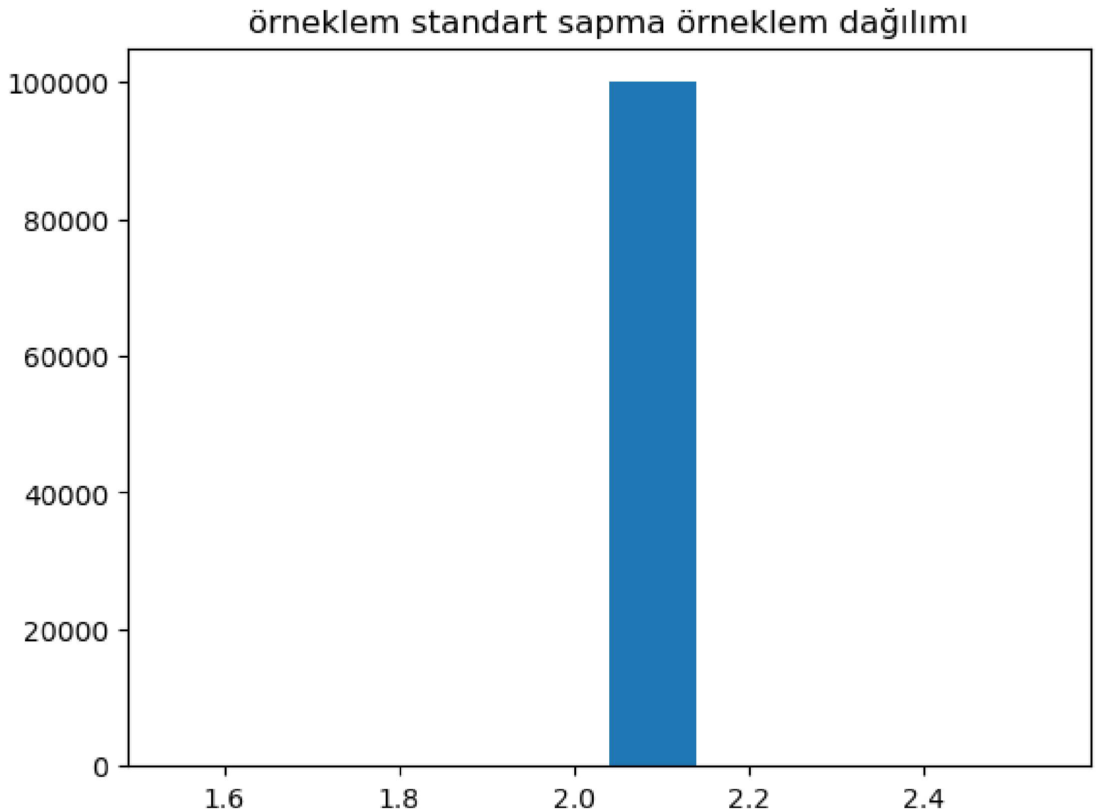
```python
# yukarıdaki işlem 10 defs tekrar edilirse
import matplotlib.pyplot as plt
sample_means = []
for i in range(10):
    samp_5 = die.sample(5, replace=True)
    sample_means.append(samp_5.mean())
plt.title("Örneklem ortalamasının örneklem dağılımı")
plt.hist(sample_means)
# böyle bir özet istatistiğinin dağılımına örneklem dağılımı denir.
```

```
(array([1., 2., 1., 1., 0., 1., 1., 1., 1., 1.]),
 array([3.  , 3.16, 3.32, 3.48, 3.64, 3.8 , 3.96, 4.12, 4.28, 4.44,
4.6 ]),
 <BarContainer object of 10 artists>)
```




```
sample_sts = []
for i in range(100000):
    smap_5 = die.sample(5, replace=True)
    sample_sts.append(np.std(samp_5))
plt.title("örneklem standart sapma örneklem dağılımı")
plt.hist(sample_sts)
```

```
(array([     0.,      0.,      0.,      0.,      0., 100000.,      0.,
             0.,      0.,      0.]),
 array([1.53960781, 1.63960781, 1.73960781, 1.83960781, 1.93960781,
        2.03960781, 2.13960781, 2.23960781, 2.33960781, 2.43960781,
        2.53960781]),
 <BarContainer object of 10 artists>)
```




```
# hiç gelmeme olasılığı oldupu için oranda try except var

# poisson süreci : belirli bir zaman dilimindeki ortalama olay
sayısının bilindiği ancakk olaylar arasındaki zaman veya boşluğun
rastgele olduğu bir süreçtir
# hangi aralıklarla geldiğini bilmezsiniz
# lambda ile ifade edilir
# zaman periyodu başına ortalama olay sayısı -> lambda
# dağılımın beklenen değeri -> ort !!
#discreed -> kesikli bir dağılımdır
# lamda dağılımın şeklini değiştirir -> basıklığını değiştirir yani.
# sample sayısı büyüdükçe poşsson dağılımı olarak normal dağılıma
benzer -Z merkezi limit teoremi

from scipy.stats import poisson
#haftada ort 8 sahiplenme olan bir yerde haftada 5 sahiplenme
poisson.pmf(5,8)
# 5 veya daha az
poisson.cdf(5,8) #ilk değer istenen ikinci lambda
#5ten fazla
1 - poisson.cdf(5,8)
```

```
0.8087639379203746
```

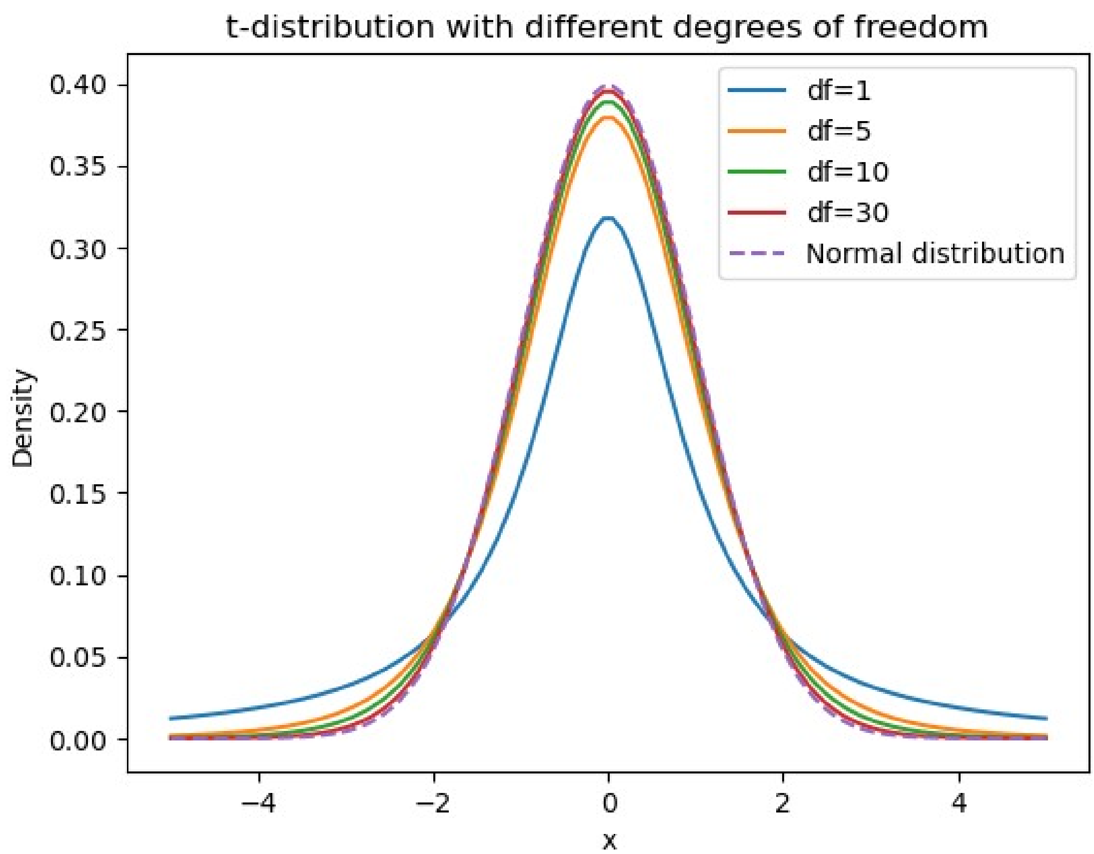
```
poisson.rvs(8, size=10) # 10 tane rastgele değer
```

```
array([ 7,  8, 10,  6,  9,  8,  4,  3,  7,  9])
```

```
# üstel dağılım -> poisson olaylarında belirli bir zaman geçme
olasılığını temsil eden dağılımdır
# lambda beklenen değerdir
# poissonun aksine zaman belirttiği için continous bir distrndır
# 2 dakikada 1 bilet ise periyot 0.5tir. posiondaki varsa 1/lambdadır
periyot

from scipy.stats import expon
#yeni bir istek için 1 dakikadan az bekleme olasılığı
expon.cdf(1, scale=2)
#4 dakikadan falza bekleme olasılığı
1 - expon.cdf(4, scale=2) #scale lambda
#1 ila 4 dk bekleme
expon.cdf(4, scale=2) - expon.cdf(1, scale=2)
```

```
0.4711953764760207
```

```
# t dağılımı -> küçük örneklem veya popülasyonun standart sapması
bilinmiyorsa kullanılır
# normale benzer.
# serbestlik derecesi -> istatistiksel bir parametreyi kullanılabilen
bağımsız bir değer veya bilgi parçalarının değeri. dağılımın
kuyrupunun ne kadar geniş ve düz olacağını.
# 30 veya daha yakın olduğunda normal dağılıma yaklaşır
# bğyğklük çok önemli (n-1) olur. n: örneklemin büyüklüğü

import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

x = np.linspace(-5, 5, 100)

# Plot for different degrees of freedom
for df in [1, 5, 10, 30]:
    plt.plot(x, stats.t.pdf(x, df), label=f'df={df}')

plt.plot(x, stats.norm.pdf(x), label='Normal distribution',
         linestyle='--')
plt.legend()
plt.title('t-distribution with different degrees of freedom')
plt.xlabel('x')
plt.ylabel('Density')
plt.show()
```




```
# lognormal dist -> onu izleyen değişkenlerin logarotması normal
dağılımı gösterir. ÖRNEK KOD!
```