Population Data :

often times we need to ask questions and collect data for an entire group , class or type of people or objets .

For examples:
height of all citizens,
longevity in hrs of all cfl bulbs,
educational attainment of all frequent startbucks customers,
weekly time spent online in hrs by all childrens

Population can be very very large
thats why, rately we can collect data on all members of population because of time and expenses.
but still we need to be able to make conclusion about the entire population.

Sample Data:

When we need to make conclusions or estimates about an entire population, we must always use a sample from out population of interest.

A small but well well chosen sample can accurately reflect the characteristics of the entire population from which it is chosen .

Sample Data ⊂ Populatin Data

Sample is a proper subset of Population Data.

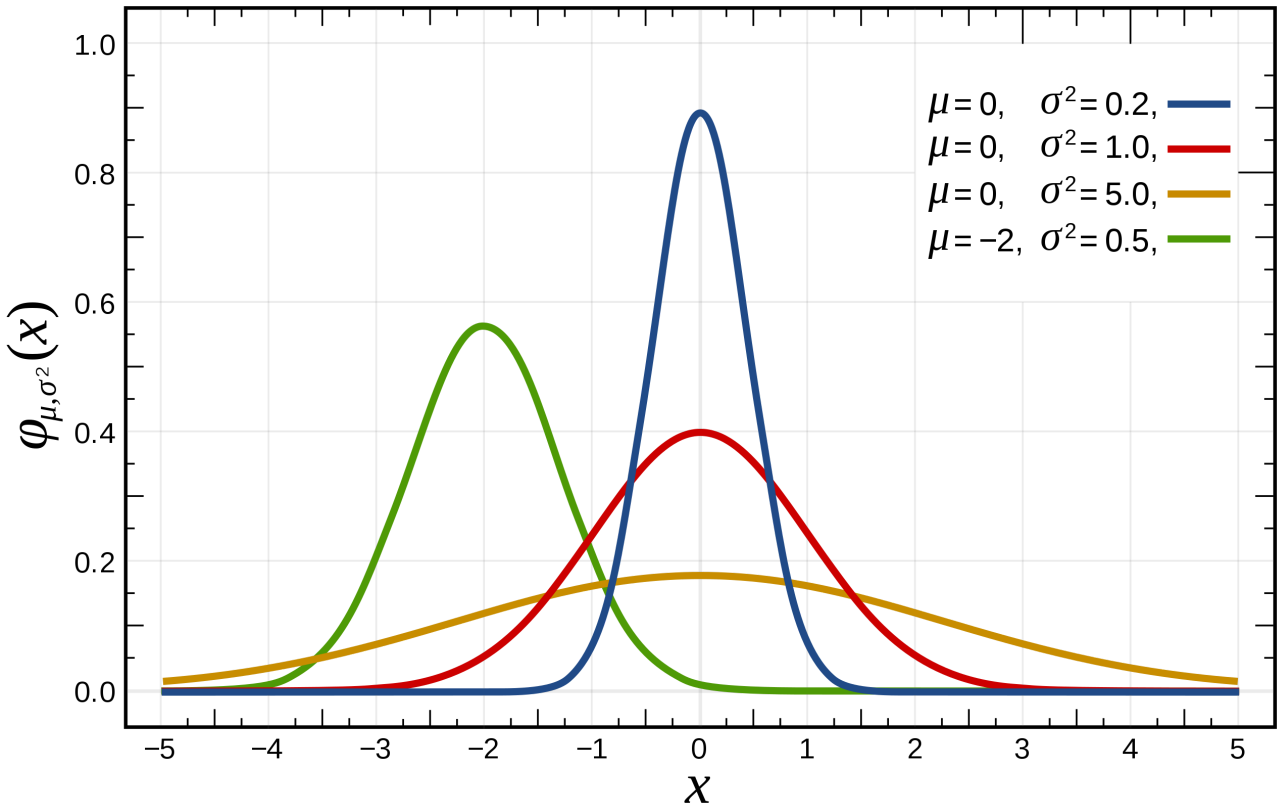
Gaussian Distribution :

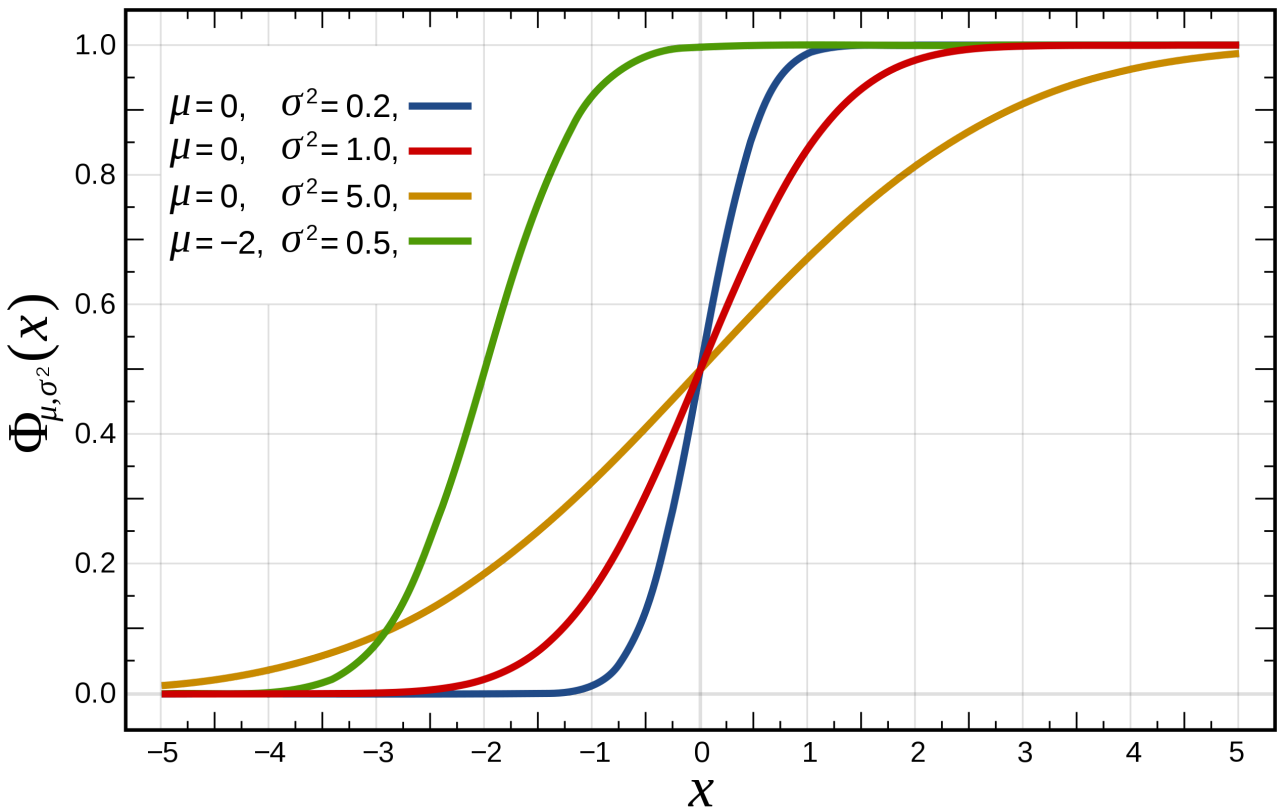
Normal Distribution :

- In statistics, a normal distribution (also known as Gaussian, Gauss, or Laplace–Gauss distribution) is a type of continuous probability distribution for a real-valued random variable. The general form of its probability density function is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

- The parameter mu is the mean or expectation of the distribution (and also its median and mode), while the parameter sigma is its standard deviation. The variance of the distribution is sigma^{2}.
- A random variable with a Gaussian distribution is said to be normally distributed, and is called a normal deviate.
- Normal distributions are important in statistics and are often used in the natural and social sciences to represent real-valued random variables whose distributions are not known.
- Their importance is partly due to the central limit theorem.
- It states that, under some conditions, the average of many samples (observations) of a random variable with finite mean and variance is itself a random variable—whose distribution converges to a normal distribution as the number of samples increases.
- Therefore, physical quantities that are expected to be the sum of many independent processes, such as measurement errors, often have distributions that are nearly normal.



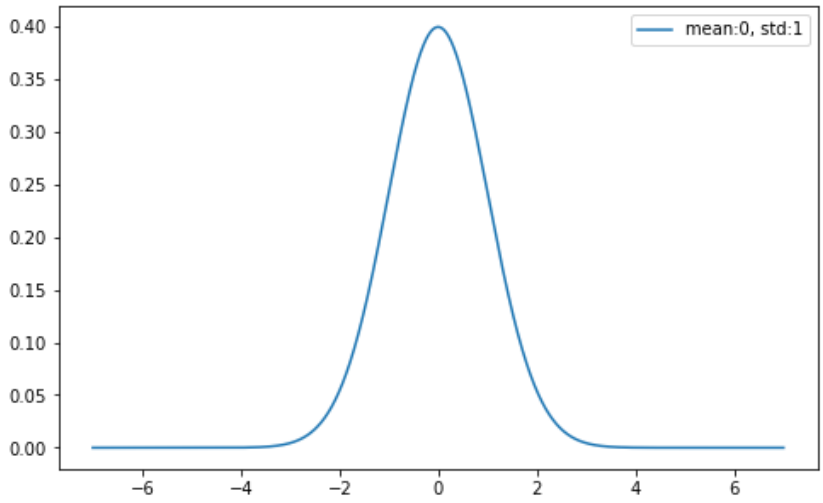


In [1]:

```
import numpy as np
import scipy as scipy
from scipy import stats
import matplotlib.pyplot as plt
from matplotlib import figure
```

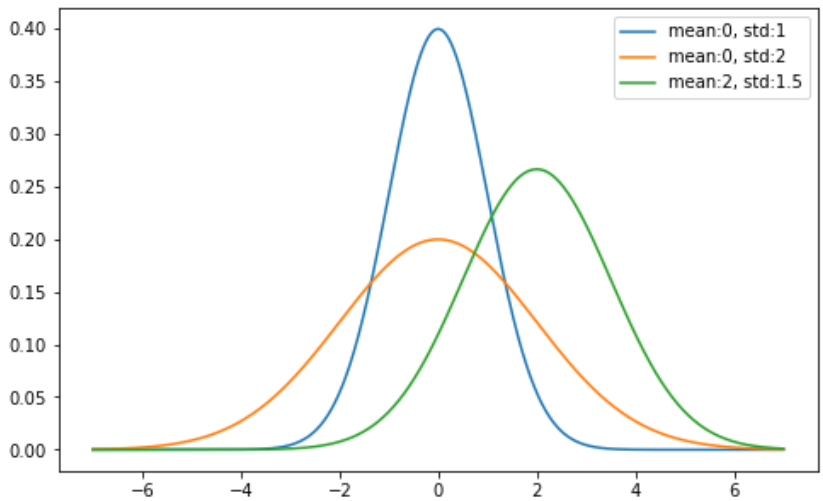
In [2]:

```
plt.figure(figsize=(8,5))

x = np.arange(-7,7.001,0.001)
y0 = stats.norm.pdf(x,0,1)
                # x :data , 0 : mean , 1 : standard deviation
plt.plot(x,y0,label = "mean:0, std:1")
plt.legend()
plt.show()
```



In [3]:

```
plt.figure(figsize=(8,5))

x = np.arange(-7,7.001,0.001)
y0 = stats.norm.pdf(x,0,1)
y1 = stats.norm.pdf(x,0,2)
y2 = stats.norm.pdf(x,2,1.5)
                # x :data , 0 : mean , 1 : standard deviation
plt.plot(x,y0,label = "mean:0, std:1")
plt.plot(x,y1,label = "mean:0, std:2")
plt.plot(x,y2,label = "mean:2, std:1.5")
plt.legend()
plt.show()
```



scipy.stats.norm = <*scipy.stats.distributions.norm_gen object at 0x4cdc250*> [source]

A normal continuous random variable.

The location (loc) keyword specifies the mean. The scale (scale) keyword specifies the standard deviation.

Continuous random variables are defined from a standard form and may require some shape parameters to complete its specification. Any optional keyword parameters can be passed to the methods of the RV object as given below:

Parameters :

x : *array_like*
quantiles
q : *array_like*
lower or upper tail probability
loc : *array_like, optional*
location parameter (default=0)
scale : *array_like, optional*
scale parameter (default=1)
size : *int or tuple of ints, optional*
shape of random variates (default computed from input arguments)
moments : *str, optional*
composed of letters ['mvsk'] specifying which moments to compute where 'm' = mean, 'v' = variance, 's' = (Fisher's) skew and 'k' = (Fisher's) kurtosis. (default='mv')

Alternatively, the object may be called (as a function) to fix the shape, location, and scale parameters returning a "frozen" continuous RV object:

rv = norm(loc=0, scale=1)

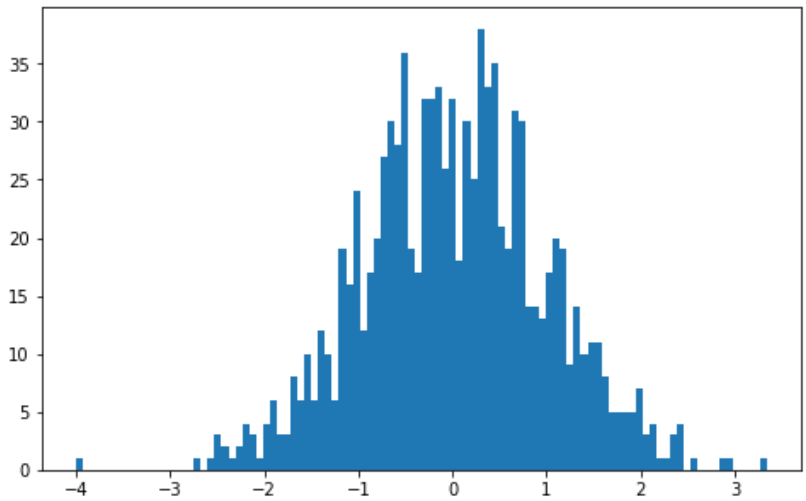
- Frozen RV object with the same methods but holding the given shape, location, and scale fixed.

data = np.random.randn(50)

```
randn(d0, d1, ..., dn)

Return a sample (or samples) from the "standard normal" distribution.
```

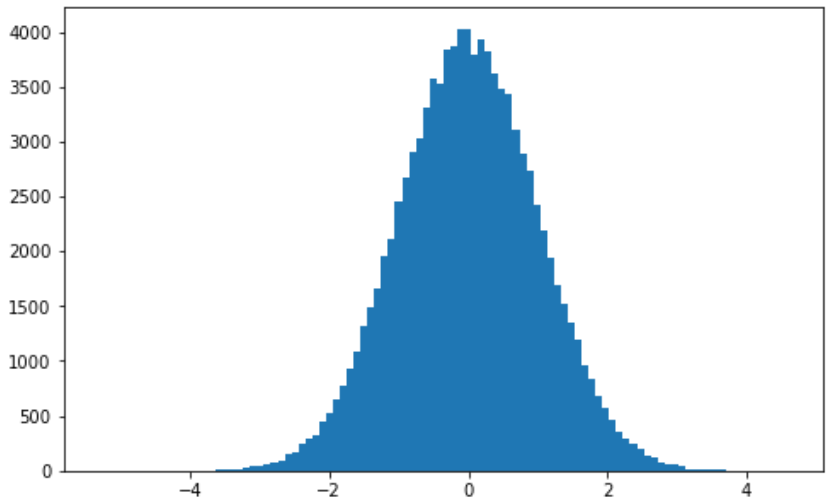
In [4]:

```
1 plt.figure(figsize=(8,5))
2
3 data = np.random.randn(1000)
4
5 plt.hist(data,bins = 100)
6 plt.show()
```



In [5]:

```
plt.figure(figsize=(8,5))

data = np.random.randn(100000)

plt.hist(data,bins = 100)
plt.show()

```



numpy.random.normal

`random.normal(loc=0.0, scale=1.0, size=None)`

Draw random samples from a normal (Gaussian) distribution.

The probability density function of the normal distribution, first derived by De Moivre and 200 years later by both Gauss and Laplace independently [2], is often called the bell curve because of its characteristic shape (see the example below).

The normal distributions occurs often in nature. For example, it describes the commonly occurring distribution of samples influenced by a large number of tiny, random disturbances, each with its own unique distribution [2].

> **Note**
>
> New code should use the `normal` method of a `default_rng()` instance instead; please see the Quick Start.

Parameters:

loc : *float or array_like of floats*

Mean ("centre") of the distribution.

scale : *float or array_like of floats*

Standard deviation (spread or "width") of the distribution. Must be non-negative.

size : *int or tuple of ints, optional*

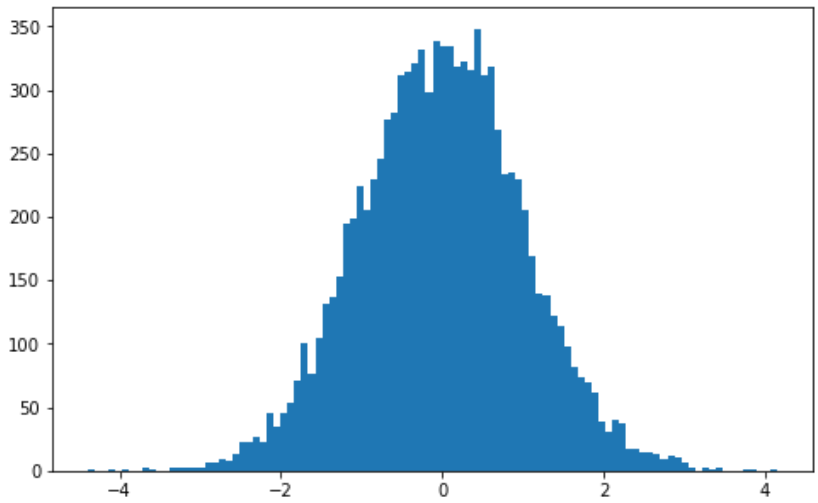
Output shape. If the given shape is, e.g., `(m, n, k)`, then `m * n * k` samples are drawn. If size is `None` (default), a single value is returned if `loc` and `scale` are both scalars. Otherwise, `np.broadcast(loc, scale).size` samples are drawn.

Sampling

In [6]:
```
1 np.random.normal(0,1,50)
```

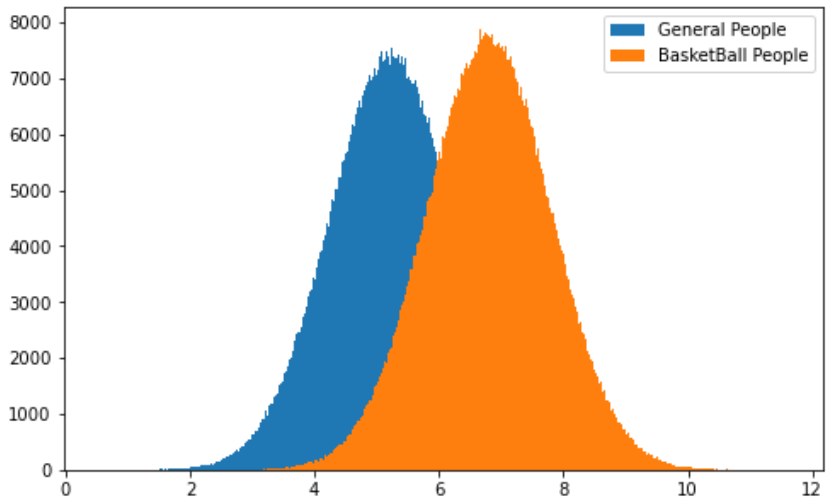
Out[6]:
```
array([-0.61469241, -1.23155815,  3.41326741,  0.15868811, -0.0894354 ,
       -0.67526716,  0.26454259, -0.43124391,  1.57773342,  0.85705707,
       -0.06765771,  1.25833843,  0.71912378, -0.49868666, -0.15762357,
        0.36085114,  2.2031449 , -0.39666518,  0.33160623, -2.5782371 ,
       -0.78711186,  0.20776785,  0.49933021,  0.76599117, -0.64476777,
        0.81453589, -0.90256792,  1.32406933, -1.03308474,  0.60621032,
        0.25264965, -1.27379272, -0.91467514, -0.82757126,  1.5678347 ,
        0.3597998 ,  1.49542146, -0.17277223, -0.11084299, -0.73743347,
       -0.60526139,  0.97075657, -0.67047765,  0.61755766,  0.60497321,
       -0.04005662,  0.12456561,  0.02364164, -1.05613207,  0.41503393])
```

In [7]:
```
plt.figure(figsize=(8,5))

data = np.random.normal(0,1,size = 10000)

plt.hist(data, bins = 100)
plt.show()
```



Example of General People height distribution and Basketball People height distribution

In [8]:
```
plt.figure(figsize=(8,5))


general_people_mean = 5.23 # height
basketball_people_mean = 6.8

general_people_samples = np.random.normal(general_people_mean,
                                          scale = 1.0,
                                          size = 1000000)

basketball_people_sample = np.random.normal(basketball_people_mean,
                                          scale = 1.0,
                                          size = 1000000)


plt.hist(general_people_samples, bins = 500,label = "General People")
plt.hist(basketball_people_sample, bins = 500, label = "BasketBall People")

plt.legend()
plt.show()
```



In [ ]:
```
1
```

What Is the Probability Density Function?

a function of a continuous random variable, whose integral across an interval gives the probability that the value of the variable lies within the same interval.

A function that defines the relationship between a random variable and its probability, such that you can find the probability of the variable using the function, is called a Probability Density Function (PDF) in statistics.

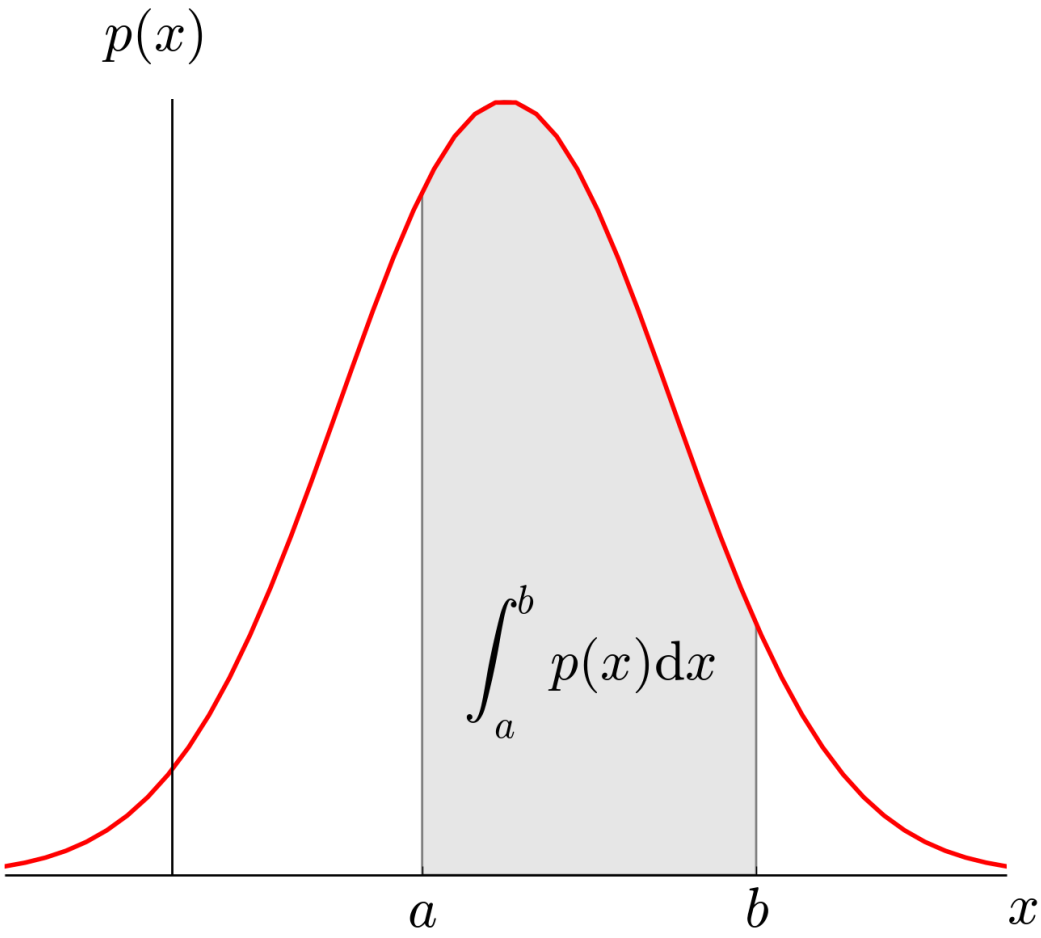
The different types of variables. They are mainly of two types:

Discrete Variable:

- A variable that can only take on a certain finite value within a specific range is called a discrete variable. It usually separates the values by a finite interval, e.g., a sum of two dice. On rolling two dice and adding up the resulting outcome, the result can only belong to a set of numbers not exceeding 12 (as the maximum result of a dice throw is 6). The values are also definite.

Continuous Variable:

- A continuous random variable can take on infinite different values within a range of values, e.g., amount of rainfall occurring in a month. The rain observed can be 1.7cm, but the exact value is not known. It can, in actuality, be 1.701, 1.7687, etc. As such, you can only define the range of values it falls into. Within this value, it can take on infinite different values.



Probability Density Function Properties

Let x be the continuous random variable with density function f(x), and the probability density function should satisfy the following conditions:

- For a continuous random variable that takes some value between certain limits, say a and b, the PDF is calculated by finding the area under its curve and the X-axis within the lower limit (a) and upper limit (b). Thus, the PDF is given by
$P(x) = \int_a^b f(x)\,dx$

- The probability density function is non-negative for all the possible values, i.e. **f(x)≥ 0**, for all x.
- The area between the density curve and horizontal X-axis is equal to 1, i.e.
$\int_{-\infty}^{\infty} f(x)\,dx = 1$

- Due to the property of continuous random variables, the density function curve is **continued** for all over the given range. Also, this defines itself over a range of continuous values or the domain of the variable.

In [ ]:
```
1
```

The probability density function (PDF) $P(x)$ of a continuous distribution is defined as the derivative of the (cumulative) distribution function $D(x)$,

$$D'(x) = [P(x)]_{-\infty}^{x} \tag{1}$$
$$= P(x) - P(-\infty) \tag{2}$$
$$= P(x), \tag{3}$$

so

$$D(x) = P(X \le x) \tag{4}$$
$$\equiv \int_{-\infty}^{x} P(\xi)\,d\xi. \tag{5}$$

A probability function satisfies

$$P(x \in B) = \int_B P(x)\,dx \tag{6}$$

and is constrained by the normalization condition,

$$P(-\infty < x < \infty) = \int_{-\infty}^{\infty} P(x)\,dx \tag{7}$$
$$= 1. \tag{8}$$

Special cases are

$$P(a \le x \le b) = \int_a^b P(x)\,dx \tag{9}$$
$$P(a \le x \le a + da) = \int_a^{a+da} P(x)\,dx \tag{10}$$
$$\approx P(a)\,da \tag{11}$$
$$P(x = a) = \int_a^a P(x)\,dx \tag{12}$$
$$= 0. \tag{13}$$

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

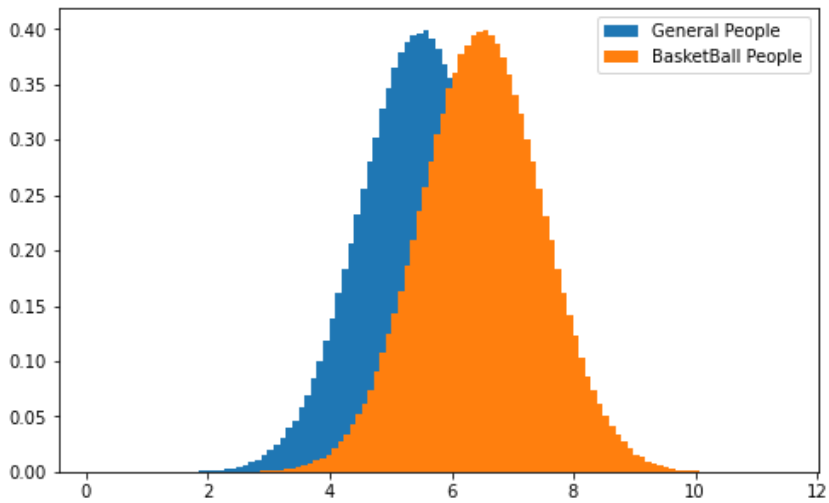
In [ ]:
```
1
```

Histogram - Pdf

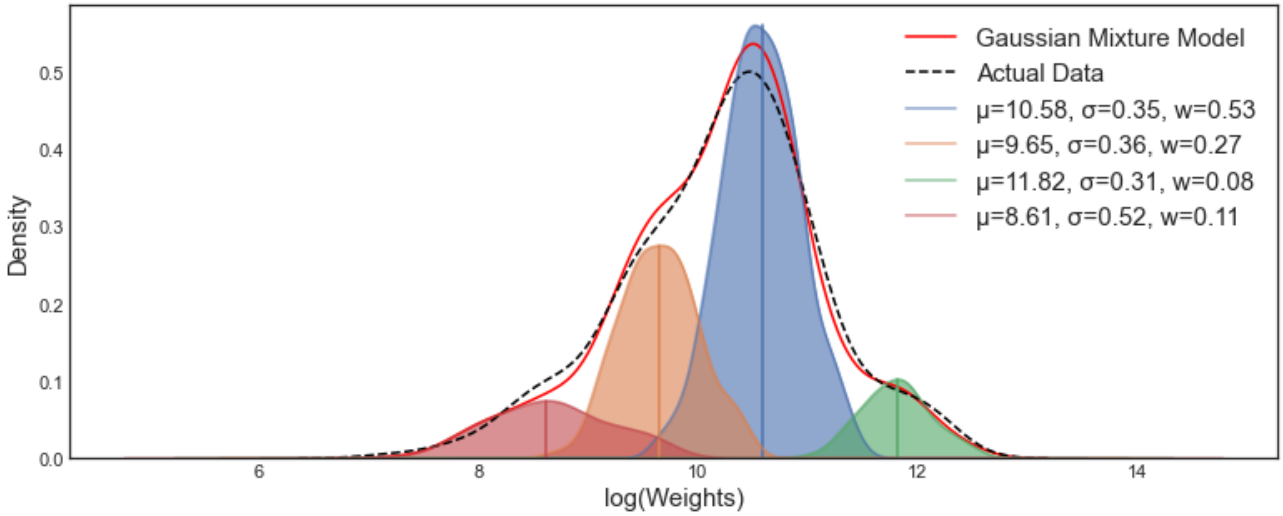
different means (for basketball and general people)

In [ ]:
```
1
```

In [9]:
```
plt.figure(figsize=(8,5))


general_people_mean = 5.5 # height
basketball_people_mean = 6.5

general_people_samples = np.random.normal(general_people_mean,     # mean
                                          scale = 1.0,             # std
                                          size = 1000000)          # sample size

basketball_people_sample = np.random.normal(basketball_people_mean,
                                          scale = 1.0,
                                          size = 1000000)


plt.hist(general_people_samples,
         bins = 100,
         label = "General People",density=True)
# denity devides all data points  with sample size

plt.hist(basketball_people_sample,
         bins = 100,
         label = "BasketBall People", density=True)

plt.legend()
plt.show()
```

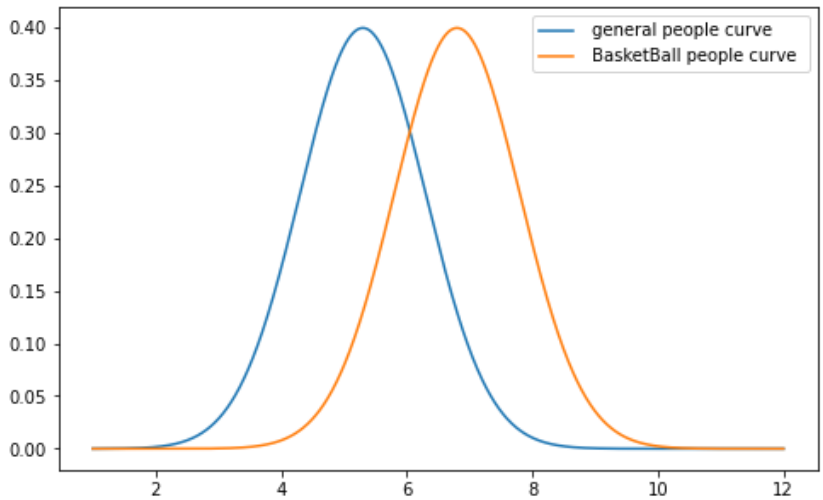


Gaussian Mixture Model



In [10]:

```
plt.figure(figsize=(8,5))

general_people_mean = 5.3
basketball_people_mean = 6.8
x = np.arange(1,12,0.001)

general_people_pdf = stats.norm.pdf(x,                          # range
                                    general_people_mean,        # mean
                                    1)                          # std

basketball_people_pdf = stats.norm.pdf(x,
                                       basketball_people_mean,
                                       1)

                # x :data , 0 : mean , 1 : standard deviation
plt.plot(x, general_people_pdf,    label = " general people curve ")
plt.plot(x, basketball_people_pdf ,label = " BasketBall people curve ")

plt.legend()
plt.show()
```



In [ ]:

```
1
```

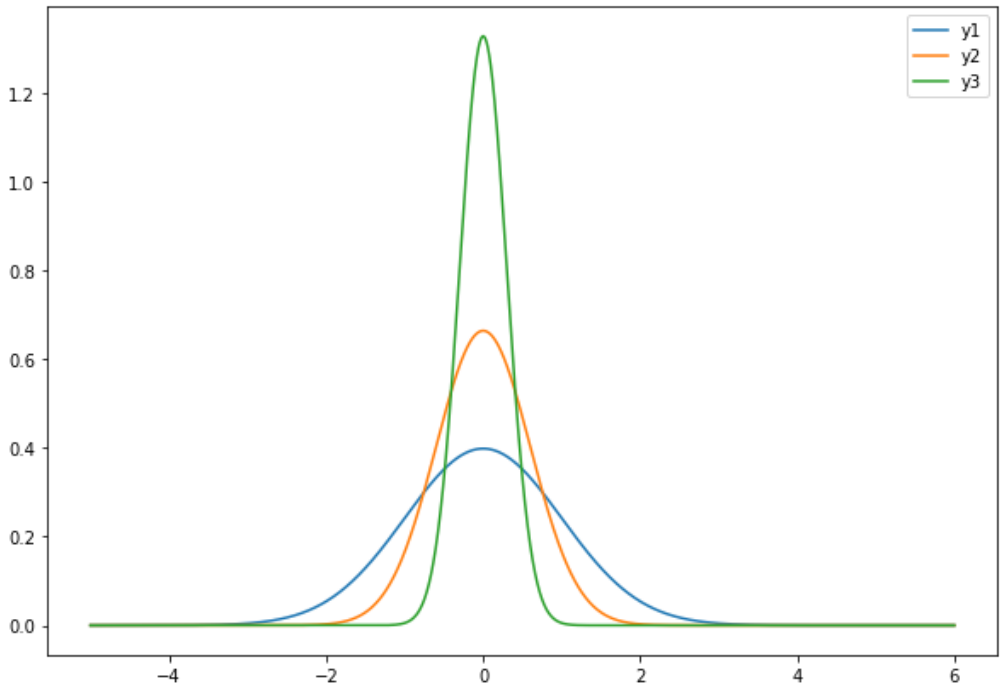
In [ ]:

```
1
```

pdf - different vairances :

In [11]:

```
# distributions with different Standard deviation (variablility)

plt.figure(figsize=(10,7))


x = np.arange(-5,6,0.001)
y1 = stats.norm.pdf(x ,
                    0,    # mean for y1
                    1)    # standard deviation for y1

y2 = stats.norm.pdf(x ,
                    0,      # mean for y1
                    0.6)    # standard deviation for y2
y3 = stats.norm.pdf(x ,
                    0,      # mean for y1
                    0.3)    # standard deviation for y2

plt.plot(x,y1,label = "y1")
plt.plot(x,y2,label = "y2")
plt.plot(x,y3,label = "y3")

plt.legend()
plt.show()

print()
print("Y3 data is more consistant (less variability ) than y2  data.")
print("Y2 data is more consistant (less variability ) than y1  data.")
```



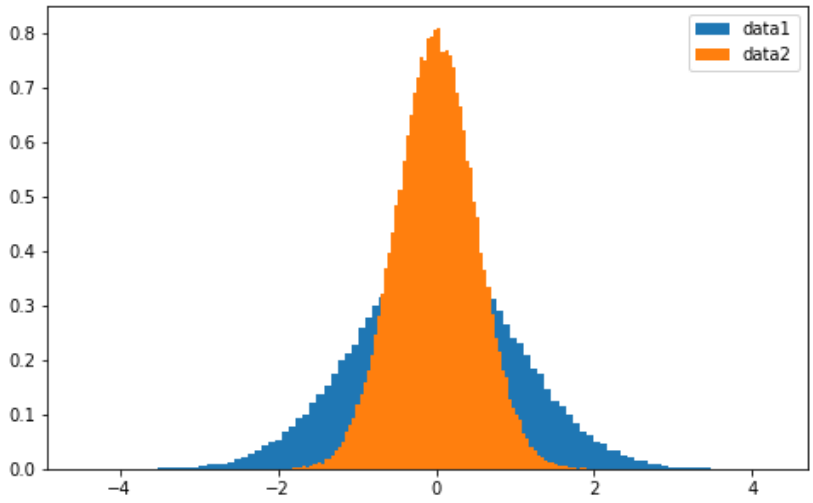

```
Y3 data is more consistant (less variability ) than y2  data.
Y2 data is more consistant (less variability ) than y1  data.
```

In [ ]:

```
1
```

In [12]:

```
plt.figure(figsize=(8,5))

sigma1 = 1
sigma2 = 0.5

data1 = np.random.normal(loc = 0.0,
                         scale = sigma1,
                         size = 100000)

data2 = np.random.normal(loc = 0.0,
                         scale = sigma2,
                         size = 100000)

plt.hist(data1 , bins = 100 , density= True , label = "data1")
plt.hist(data2 , bins = 100 , density= True , label = "data2")

plt.legend()
plt.show()
```



In [13]:

```
print("Variance of Data1 : ",np.var(data1))
print("Variance of Data2 : ",np.var(data2))

```

```
Variance of Data1 :  1.0106404591372906
Variance of Data2 :  0.2494414243636118
```

Sample Variance :

The variance reflects the variability of your dataset by taking the average of squared deviations from the mean.

Formula

$$S^2 = \frac{\sum (x_i - \bar{x})^2}{n - 1}$$

S^2 = sample variance

x_i = the value of the one observation

$\bar{x}$ = the mean value of all observations

n = the number of observations

In [ ]:
```
1
```

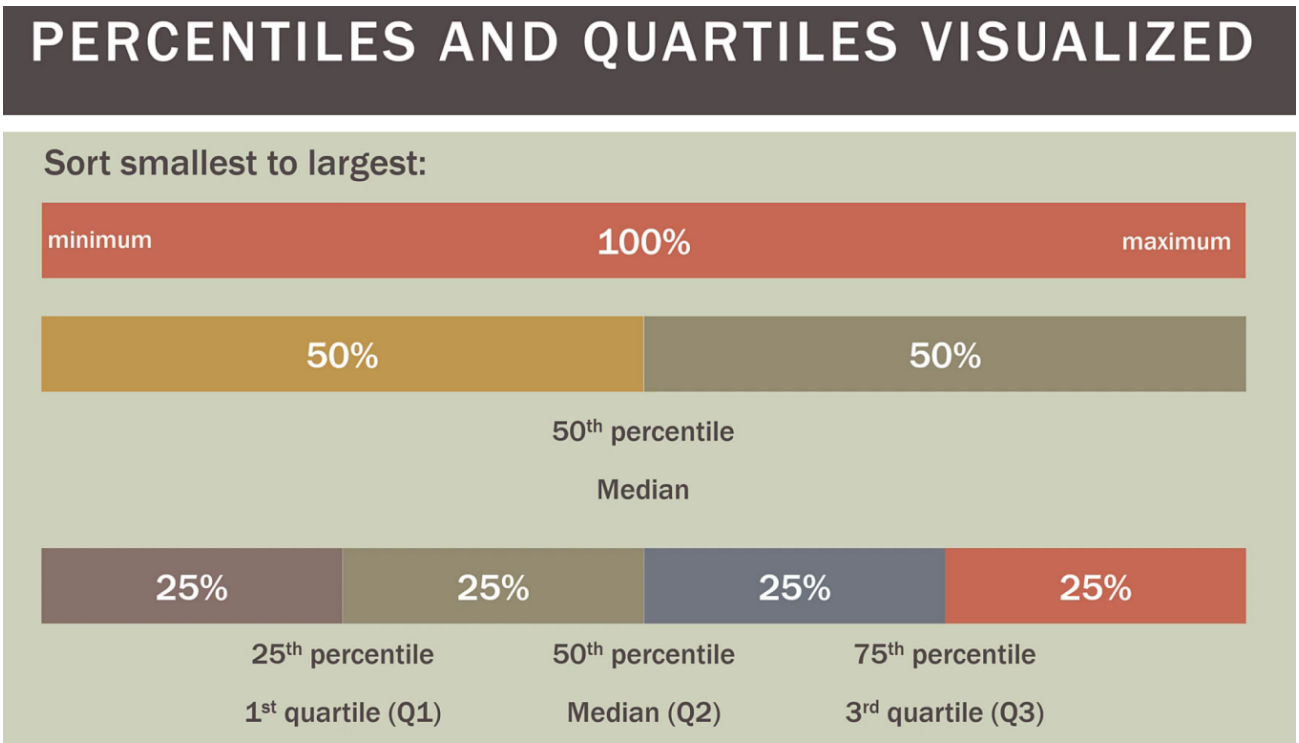
In [ ]:
```
1
```

In [ ]:
```
1
```

Descriptive Statistics:

Percentile

In [14]:
```
1 # snapshots taken from
2
3 # https://www.youtube.com/c/BrandonFoltz/playlists
```




```
1 ### Percentile :
```

```
- ##### is the locating an observation in dataset ; an address

### Percentile Rank :

data : 2,2,3,4,5,5,5,6,7,8,8,8,8,8,9,9,10,11,12  total n = 20
what is the percentile rank of 10 :  x = 10

Percentile rank of x = ( # number of values below x *100) / n

                       number of values below x is 16
                        = value 10 is at  80th percentile

                        mean 80% of the entire distrubtion is less than x , less than 10
```

In [15]:
```
(16/20)*100
```

Out[15]: 80.0

```
# what value exists at percentile ranking of 25%

value = (percentile  * (n+1)) / 100
      =  (25 * (20+1)) / 100                 # index position
```

In [16]:
```
(25 * (20+1)) / 100
```

Out[16]: 5.25

```
 at 75th percentile value :

 (75 * (20+1))/100
```

In [17]:
```
(75 * (20+1))/100
```

Out[17]: 15.75

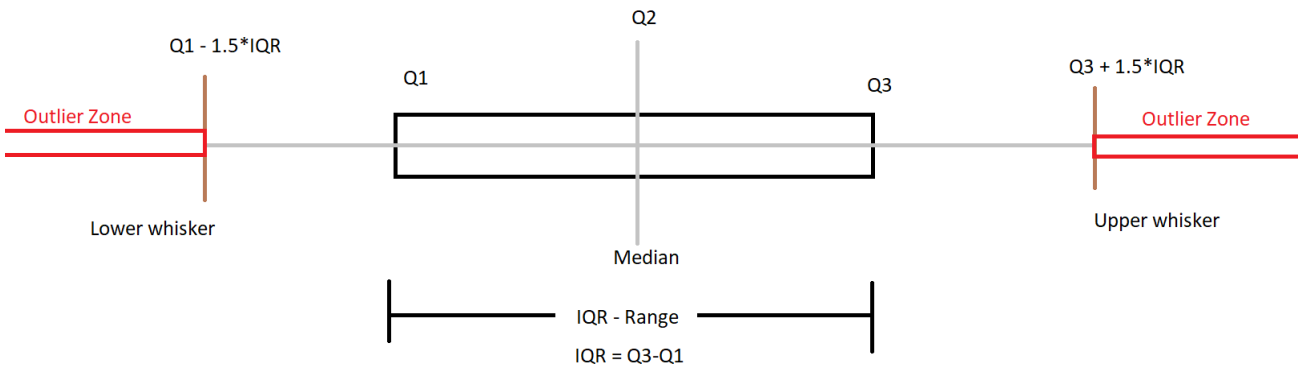
In [ ]:
```

```

```

```

In [ ]:
```

```

IQR - BoxPlots

```
Data :  1 2 2 2 3 3 4 5 5 5 6 6 6 6 7 8 8 9 27
```



In [ ]:
```

```

In [18]:
```
import pandas as pd
```

```
Inter Quartile Range : IQR = Q3 - Q1
```

Detect Outliers

In [20]:
```
def outliers(data):
    import numpy as np

    print("Data :",data)
    A =np.array(data)
    Q1 = np.percentile(A,25)
    Q3 = np.percentile(A,75)
    IQR = Q3-Q1
    print("IQR = ", IQR)
    print("Q1 = ",Q1)
    print("Median",np.median(A))
    print("Q3 = ",Q3)
    upper_bound = Q3 + (1.5*IQR)
    lower_bound = Q1 - (1.5*IQR)

    outliers = []

    for i in data:
        if i < lower_bound or i > upper_bound:
            outliers.append(i)
    print("Outliers : " , outliers)
```

In [ ]:
```

```

In [21]:
```
data = [10,8,9,6,5,11,7,1,9,929,100]
outliers(data)


```

```
Data : [10, 8, 9, 6, 5, 11, 7, 1, 9, 929, 100]
IQR =  4.0
Q1 =  6.5
Median 9.0
Q3 =  10.5
Outliers :  [929, 100]
```

In [ ]:
```

```

In [22]:
```
data = [int(x) for x in "-5 1 2 2 2 3 3 4 5 5 97 5 6 6 6 6 7 8 8 9 27 85 10".split()]
```

In [23]:
```
A = data
```

In [24]:
```
outliers(A)
```

```
Data : [-5, 1, 2, 2, 2, 3, 3, 4, 5, 5, 97, 5, 6, 6, 6, 6, 7, 8, 8, 9, 27, 85, 10]
IQR =  5.0
Q1 =  3.0
Median 6.0
Q3 =  8.0
Outliers :  [-5, 97, 27, 85]
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```
1
```

In [ ]:
```
1
```

==========

example: IQR Outliers

In [25]:
```
1 dravid = pd.read_html("https://stats.espncricinfo.com/ci/engine/player/28114.html?class=2;templa
2 sehwag = pd.read_html("https://stats.espncricinfo.com/ci/engine/player/35263.html?class=2;templa
3
```

In [26]:
```
1 # sehwag.head()
```

In [27]:
```
1 # dravid.head()
```

In [28]:
```
1 # dravid["Runs"].unique()
```

In [29]:
```
1 dravid = dravid[(dravid["Runs"]!="DNB") & (dravid["Runs"]!="TDNB")]
2 dravid_score = pd.to_numeric(dravid["Runs"].apply(lambda r:"".join(r.split("*"))))
```

In [30]:
```
1 sehwag = sehwag[(sehwag["Runs"]!="DNB") & (sehwag["Runs"]!="TDNB")]
2 sehwag_score = pd.to_numeric(sehwag["Runs"].apply(lambda r:"".join(r.split("*"))))
```

In [31]:
```
1 from scipy import stats
2
```

In [32]:
```
1 stats.iqr(dravid_score)
```

Out[32]:
```
44.0
```

In [ ]:
```
1
```

In [33]:
```
1 np.percentile(dravid_score,75) ,  np.percentile(dravid_score,25)
```

Out[33]:
```
(54.0, 10.0)
```

In [34]:
```
1 Dravid_upper_bound = np.percentile(dravid_score,75) + (1.5*  (np.percentile(dravid_score,75) -
```

In [35]:
```
1 dravid_score[dravid_score > Dravid_upper_bound]
```

Out[35]:
```
65     123
83     145
104    153
Name: Runs, dtype: int64
```

In [36]:
```
1 sum(dravid_score > Dravid_upper_bound)
```

Out[36]:
```
3
```

In [ ]:
```
1
2
3
4
```

In [37]:
```
np.percentile(sehwag_score,75) ,  np.percentile(sehwag_score,25)
```

Out[37]: (46.0, 8.0)

In [38]:
```
sehwag_upper_bound = np.percentile(sehwag_score,75) + (1.5*  (np.percentile(sehwag_score,75) -
```

In [39]:
```
sehwag_score[sehwag_score > sehwag_upper_bound]
```

Out[39]:
```
39     126
45     114
51     108
55     112
77     130
107    108
168    114
187    119
197    116
203    125
211    146
226    110
228    175
239    219
Name: Runs, dtype: int64
```

In [40]:
```
sum(sehwag_score > sehwag_upper_bound)
```

Out[40]: 14

==========

In [ ]:

In [ ]:

In [ ]:

Probability Terminologies , Rules and Formulas :

union , intesection , subset

A ∪ B ,   A ∩ B ,   A ⊆ B

In [ ]:

In [ ]:

In [ ]:

Sample space : S

Events : E

Intersection : A ∩ B

Union : A ∪ B Complement : A'

Mutually Exclusive : A ∩ B = {}

Mutually Exhaustive : A ∪ B = S

Conditional Probability :

```
P[A|B] = P(A ∩ B)/P(B)

P[B|A] = P(A ∩ B)/P(A)
```

```
Independence

P[A|B] = P[A]

P[B|A] = P[B]

P[A ∩ B] = P[A]*P[B]
```

```
Bayes Theorem

P[A|B] = P(A ∩ B)
           /P(B)

       = P[B|A]*P[A]
           / P[B]

       = P[B|A]*P[A]
        / P[B|A]*P[A] + P[B|A']*P[A']
```

```
Bernouli RV : p,(1-p)
```

```
Probability Mass function
```

- The Probability Mass function is defined on all the values of R, where it takes all the arguments of any real number. It doesn't belong to the value of X when the argument value equals to zero and when the argument belongs to x, the value of PMF should be positive.
- The probability mass function is usually the primary component of defining a discrete probability distribution, but it differs from the probability density function (PDF) where it produces distinct outcomes. This is the reason why probability mass function is used in computer programming and statistical modelling. In other words, probability mass function is a function that relates discrete events to the probabilities associated with those events occurring. The word "mass" indicates the probabilities that are concentrated on discrete events.

```
What is the difference between PMF and PDF?
```

- PMF

Solution ranges between numbers of discrete random variables

Uses discrete random variables

- PDF

The solution is in a range of continuous random variables

Uses continuous random variables

```
Binomial RV : n,p,(1-p)

nCr * P^r * (1-p)^(n-r)

Expected Value , Sample Mean E(x)

E(x) = sum(x * P(x)) = actual mean

Sample Variance:

Var(X) = E(X^2)-(E(x))^2
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```