Statistics

- Measure & Analyze the data (Structured Data)

Data

```
1. Stuctured Data
    (i) Cross Sectional Data : Data don't depend on time
            - Continous
            - Discrete
    (ii) Time Series Data  : Data depends on time

2. Unstructured Data
    (i) Images & Videos
    (ii) Text & Audio
```

Types of variables in structured data

Quantitative Data (Numerical Data):

```
- Data that measured in numbers. It deals with numbers that make sense to perso
rm some mathematical operations
```

1. **Continuous (values can be decimal)**

```
    - Refers to variables that can take on any numerical value
    - eg: length, height, volume, age (22.3), stock price etc.,
```

2. **Discrete Count (count data)**

```
    - Refers to variables that can only be measured in certain numbers
    - eg: strength of a class, sales of mobiles etc.,
```

Qualitative Data (Discrete Categorical Data or Text Data):

```
- Refers to the value that place things into different groups or categories
```

1. **Nominal** - no natuaral ordering to the values of a categorical variable

```
    - eg: species names, colors, brands
```

2. **Ordinal** - natural ordering to the values of a categorical variable

```
    - eg: Grade A+ > Grade A >....., ranks, (very likely, likely, .....)
```

Population

- Refers to set of all items or individuals of interest

- Ex: All voters in next election
- A Statistical Measure that describes the data from a population **- Parameter**
- Total Number of Population records = N

Sample

- A sample is a subset of population
- Ex: 1000 Voters selected
- A Statistical Measure that describes the data from a sample **- Statistic**
- Total Number of Sample records = n

Types of Statistics

- **Descriptive Statistics**
  - collecting, measuring , presenting and describing data
- **Inferential Statistics**
  - drawing conclusion and/or making decisions concerning a population based on sample data

In [1]:

```
import numpy as np
import pandas as pd
```


Measures of Central Tendency (1st Business Moment)

- **Mean , Median , Mode**
- Mean & Median are applied on continous data
- Mode is applied on discrete data

Mean

```
- Sum of all data values/Total Number of Data values
- Mean is applied on continous data
```

- Population mean (μ)= $\frac{\sum X}{N}$
- Sample mean ($\bar{x}$) = $\frac{\sum x}{n}$

In [2]:

```
df = pd.DataFrame({"X":[1,2,3,4,5]})
df
```

Out[2]:

	X
0	1
1	2
2	3
3	4
4	5

In [3]:

```
df["X"].mean()
```

Out[3]:

3.0

In [4]:

```
df["X"].sum()/len(df)
```

Out[4]:

3.0

Median

- Refers to the data value that is *positioned* in the middle of an ordered datas et
- Median is applied on continous data
- Median refers to centre value if you have odd number of data points
- Median refers to average of centre 2 values if you have even number of data p oints

In [5]:

```
df = pd.DataFrame({"X":[2,4,1,9,16,10,4,8,7]})
df
```

Out[5]:

	X
0	2
1	4
2	1
3	9
4	16
5	10
6	4
7	8
8	7

In [6]:

```
df["X"].median()
```

Out[6]:

```
7.0
```

In [7]:

```
df = pd.DataFrame({"Y":[2,4,1,9,16,10,4,8,7,5]})
df
```

Out[7]:

	Y
0	2
1	4
2	1
3	9
4	16
5	10
6	4
7	8
8	7
9	5

In [8]:

```
df["Y"].median()
```

Out[8]:

```
6.0
```

Mode

- Most repeated value/Most frequent value

- **Unimodal Data** (if the data have only 1 Mode value)
- **Bimodal Data** (if the data have 2 Mode values)
- **Multimodal Data** (if the data have >2 Mode values)

In [9]:

```
df = pd.DataFrame({"X":[1,1,2,3,4,5],"Y":[1,1,2,3,4,4],"Z":[1,1,2,2,3,3],"i":[1,2,3,4,5,
df
```

Out[9]:

	X	Y	Z	i
0	1	1	1	1
1	1	1	1	2
2	2	2	2	3
3	3	3	2	4
4	4	4	3	5
5	5	4	3	6

In [10]:

```
df["X"].mode()              # unimodal data
```

Out[10]:

```
0    1
Name: X, dtype: int64
```

In [11]:

```
df["Y"].mode()              # bimodal data
```

Out[11]:

```
0    1
1    4
Name: Y, dtype: int64
```

In [12]:

```
df["Z"].mode()              # multimodal data
```

Out[12]:

```
0    1
1    2
2    3
Name: Z, dtype: int64
```

In [13]:

```
df["i"].mode()
```

Out[13]:

```
0    1
1    2
2    3
3    4
4    5
5    6
Name: i, dtype: int64
```


Measures of Dispersion or Measures of Spread (2nd Business Moment)

- **Range**, **IQR**, **Variance**, **Std.deviation**
- all are applied on Contionous Variable only

In [14]:

```
df = pd.DataFrame({"X":[1,2,3,4,5]})
df
```

Out[14]:

	X
0	1
1	2
2	3
3	4
4	5

Minimum

In [15]:

```
df["X"].min()
```

Out[15]:

1

Maximum

In [16]:

```
df["X"].max()
```

Out[16]:

5

Range

- Range = Maximum value - Minimum value

In [17]:

```
df["X"].max() - df["X"].min()
```

Out[17]:

4

Deviation (X - μ)

- deviation = data deviated from the mean = how dispersed the data is from the central value

In [18]:

```
df["X-μ"] = df["X"] - df["X"].mean()
df
```

Out[18]:

	X	X-μ
0	1	-2.0
1	2	-1.0
2	3	0.0
3	4	1.0
4	5	2.0

Mean deviation $= \frac{\sum (x_i - \mu)}{N}$

In [19]:

```
df["X-μ"].mean()
```

Out[19]:

```
0.0
```

- for any given data set **mean deviation is always zero**

Population Variance $(\sigma^2) = \frac{\sum (x_i - \mu)^2}{N}$

In [20]:

```
df["X"].var(ddof=0)
```

Out[20]:

```
2.0
```

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

In [21]:

```
df["X"].var()                    #by default ddof =1 (means by default we have sample var
```

Out[21]:

```
2.5
```

Standard deviation:

- It is the **statistical measure of the dispersion of the dataset relative to its mean**.
- It tells about, How close the values in the data set are to the mean
- High std deviation means the values are largely deviated from the mean i.e., the spread is high
- Low std deviation means the all data values are close to the mean

Population Standard Deviation $(\sigma) = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$

In [22]:

```
df["X"].std(ddof=0)
```

Out[22]:

```
1.4142135623730951
```

Sample Std.deviation $S = \sqrt{\frac{\sum (x-\bar{x})^2}{n-1}}$

In [23]:

```
df["X"].std(ddof=1)
```

Out[23]:

```
1.5811388300841898
```

Coefficient of variation

- ratio of std. deviation and mean expressed in percentage = $\frac{\sigma}{\mu} * 100$
- It is the measure of variability of the dataset around its mean
- The higher the CV the greater the std. deviation to its mean

In [24]:

```
df["X"].std(ddof=0)/df["X"].mean()
```

Out[24]:

```
0.47140452079103173
```

Inter Quartile Range (IQR) = Q3-Q1

Quartile3 (Q3) -- 75 percentile : 75% of data is below to this value

Quartile1 (Q1) -- 25 percentile : 25% of data is below to this value

In [25]:

```
df = pd.DataFrame({"X":[10,11,12,25,25,27,31,33,34,34,36,36,43,50,59]})
df
```

Out[25]:

	X
0	10
1	11
2	12
3	25
4	25
5	27
6	31
7	33
8	34
9	34
10	36
11	36
12	43
13	50
14	59

In [26]:

```
Q1 = df["X"].quantile(0.25)
Q1
```

Out[26]:

```
25.0
```

In [27]:

```
Q3 = df["X"].quantile(0.75)
Q3
```

Out[27]:

```
36.0
```

In [28]:

```
df["X"].quantile(0.5)
```

Out[28]:

```
33.0
```

In [29]:

```
Q1 = df["X"].quantile(0.25)
Q2 = df["X"].quantile(0.5)
Q3 = df["X"].quantile(0.75)

IQR = Q3 - Q1
IQR
```

Out[29]:

```
11.0
```

Outlier

- A data value that is numerically distant from a data set
- Outliers will more affect the mean, variance, standard deviation
- Outliers will less affect the median & IQR

- A data value is considered to be an outlier,if

$$datavalue < Q1 - (1.5 * IQR)$$

$$datavalue > Q3 + (1.5 * IQR)$$

lower limit

In [30]:

```
ll= Q1 - (IQR * 1.5)
ll
```

Out[30]:

```
8.5
```

upper limit

In [31]:

```
ul = Q3 + (IQR * 1.5)
ul
```

Out[31]:

52.5

to extact outliers data

In [32]:

```
df[(df["X"]<ll) | (df["X"]>ul)]
```

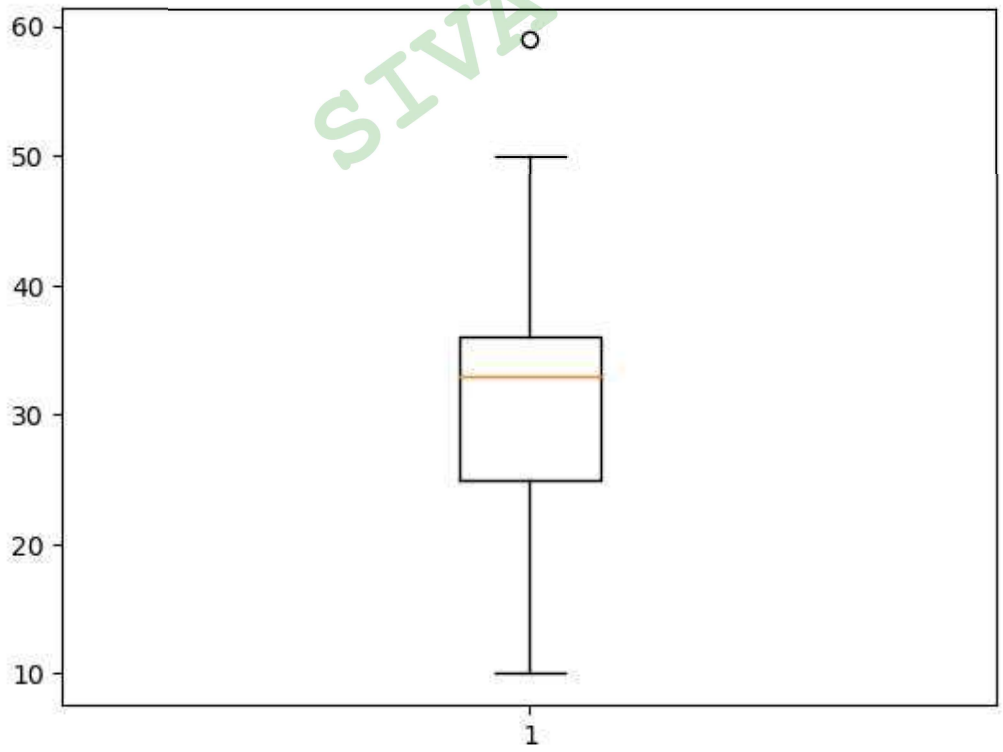
Out[32]:

	X
14	59

- **To check outliers** are available or not, we use **box plot**

In [33]:

```
import matplotlib.pyplot as plt
plt.boxplot(x=df["X"])
plt.show()
```



In [34]:

```
df = pd.DataFrame({"Gender":["F","F","F","F","M","M","F","M","F","F","F","F","M","M
                   "Marks":[30,41,42,51,52,53,61,62,68,69,77,78,79,88,89,100],
                   "no_of_assignments":[1,1,1,1,2,2,2,3,3,3,3,3,4,4,4,4]})

df
```

Out[34]:

	Gender	Marks	no_of_assignments
0	F	30	1
1	F	41	1
2	F	42	1
3	F	51	1
4	M	52	2
5	M	53	2
6	F	61	2
7	M	62	3
8	F	68	3
9	F	69	3
10	F	77	3
11	F	78	3
12	M	79	4
13	M	88	4
14	F	89	4
15	M	100	4

Frequency Distribution

- Graphical representation of variable with corresponding frequency.

Disrete Frequency Distribution : Graphical representation of discrete variable with corresponding frequency

In [35]:

```
df["Gender"].unique()
```

Out[35]:

```
array(['F', 'M'], dtype=object)
```

In [36]:

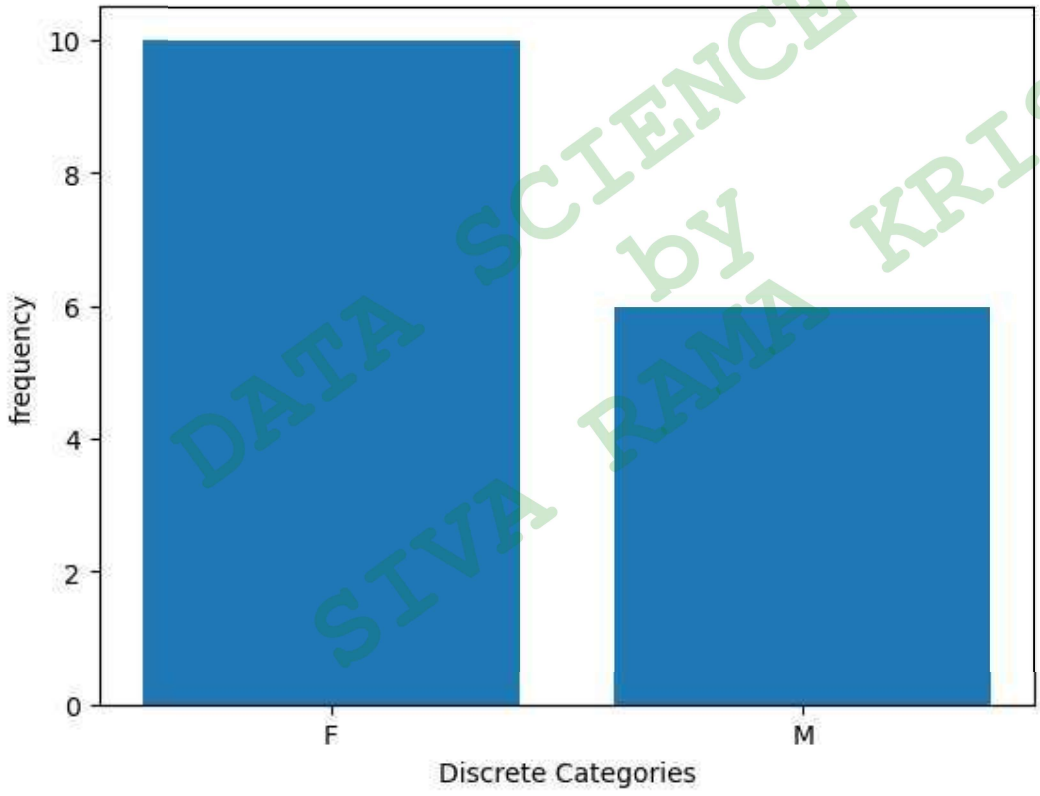
```
df["Gender"].value_counts()
```

Out[36]:

```
F    10
M     6
Name: Gender, dtype: int64
```

In [37]:

```
plt.bar(x=df["Gender"].unique(),height=df["Gender"].value_counts())
plt.xlabel("Discrete Categories")
plt.ylabel("frequency")
plt.show()
```



Continous Frequency Distribution : Graphical representation of continous variable with corresponding frequency.

In [38]:

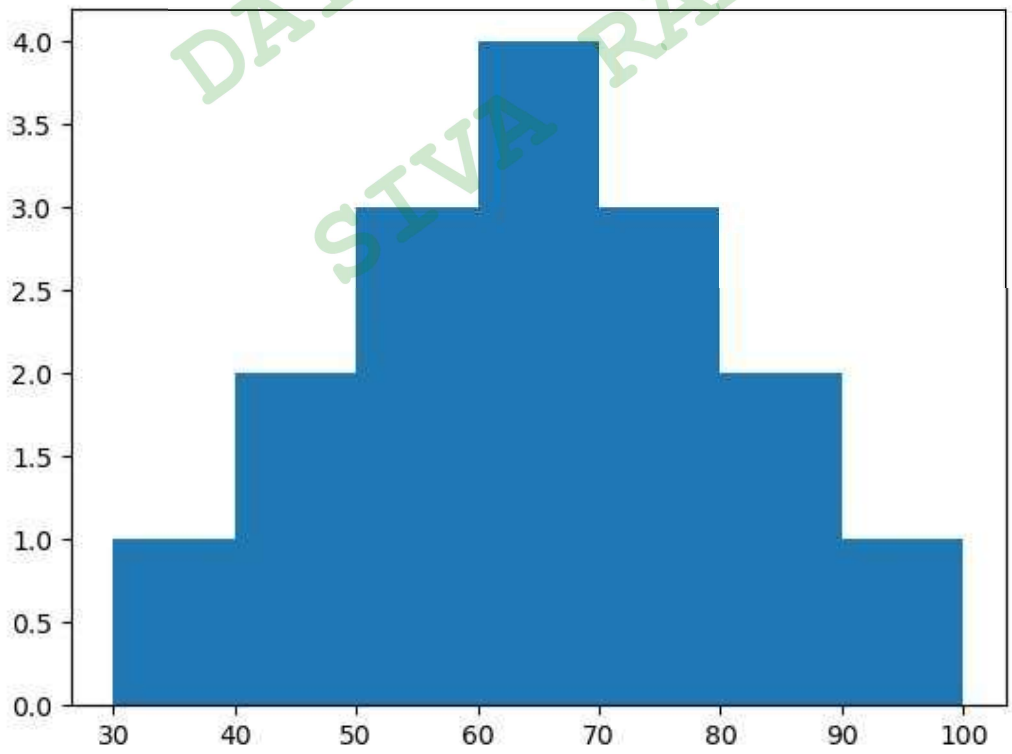
```
df["Marks"]
```

Out[38]:

```
0      30
1      41
2      42
3      51
4      52
5      53
6      61
7      62
8      68
9      69
10     77
11     78
12     79
13     88
14     89
15    100
Name: Marks, dtype: int64
```

In [39]:

```
plt.hist(df["Marks"],bins=7)
plt.show()
```



Probability

- Chance of occurrence

$$Probability(requirement) = \frac{No.of\ values\ satisfies\ your\ requirment}{total\ no.of\ values}$$

- **Example:** chance of occurence of head when tossing a coin
- Always probability value lies between 0 to1.
- Sum of all Probabilities =1

In [40]:

```
df["Gender"].value_counts()/len(df)
```

Out[40]:

```
F    0.625
M    0.375
Name: Gender, dtype: float64
```

In [41]:

```
df["Gender"].value_counts(normalize=True)
```

Out[41]:

```
F    0.625
M    0.375
Name: Gender, dtype: float64
```

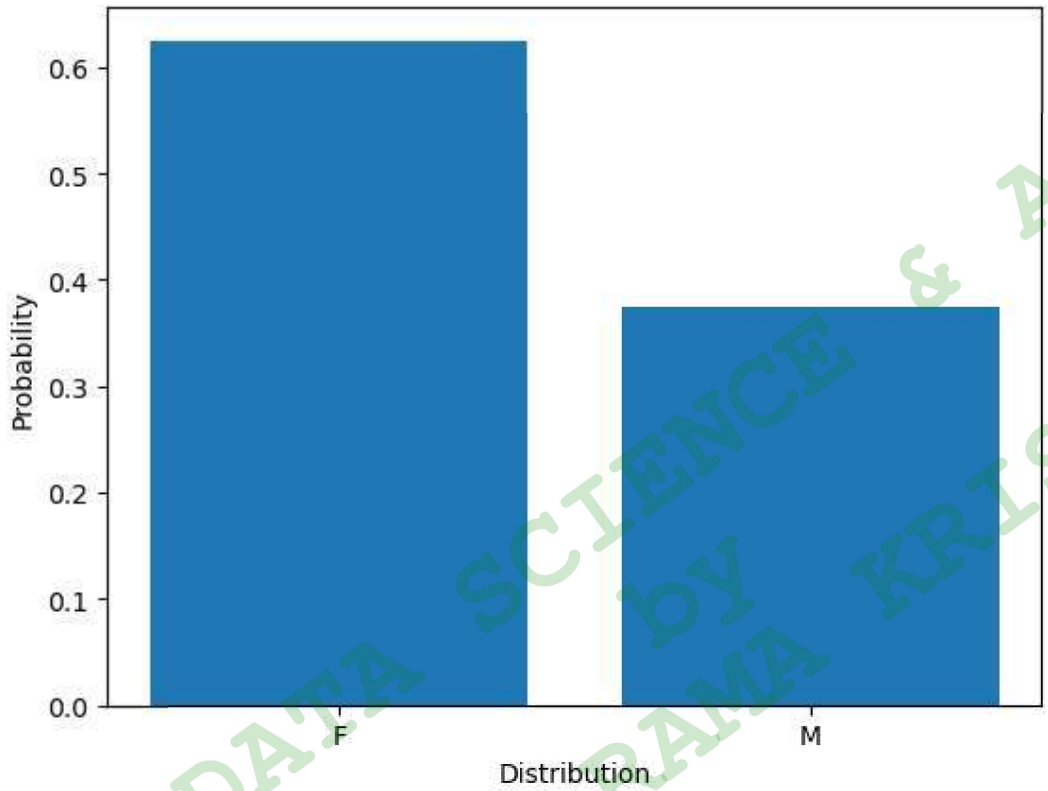

Probability Distribution

- Graphical representation of variable & respective probabilities of variable.

Disrete Probability Distribution : Graphical representation of discrete variable with corresponding probability

In [42]:

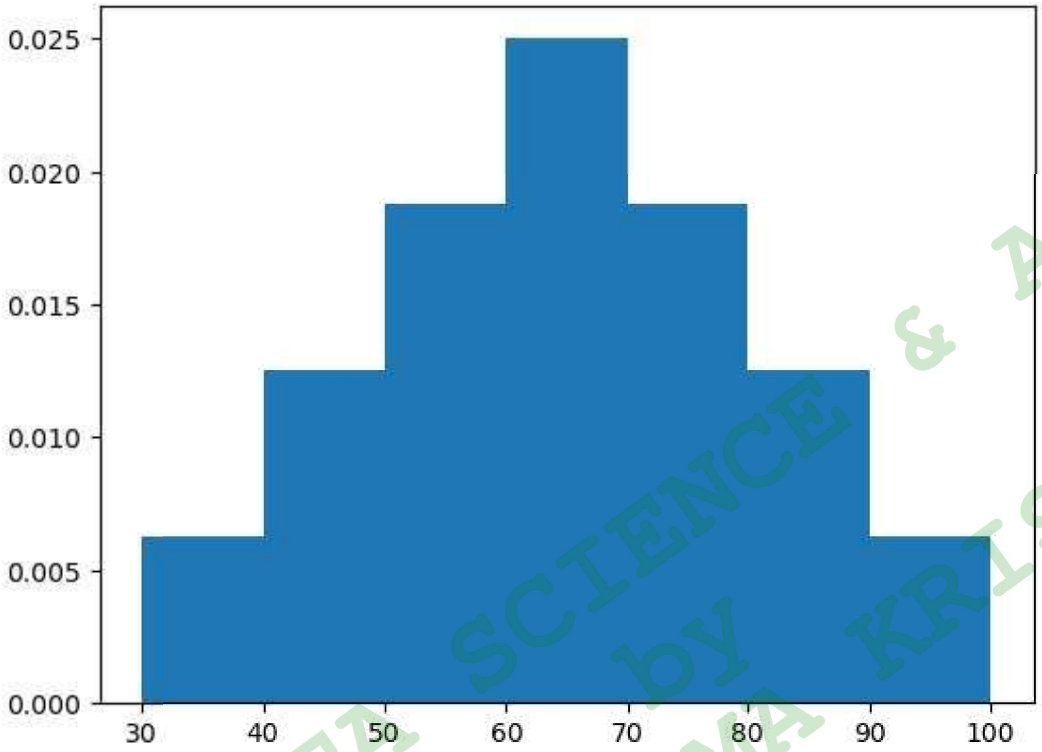
```
plt.bar(x=df["Gender"].unique(),height=df["Gender"].value_counts(normalize=True))
plt.xlabel("Distribution")
plt.ylabel("Probability")
plt.show()
```



Continous Probability Distribution : Graphical representation of continous variable with corresponding probability

In [43]:

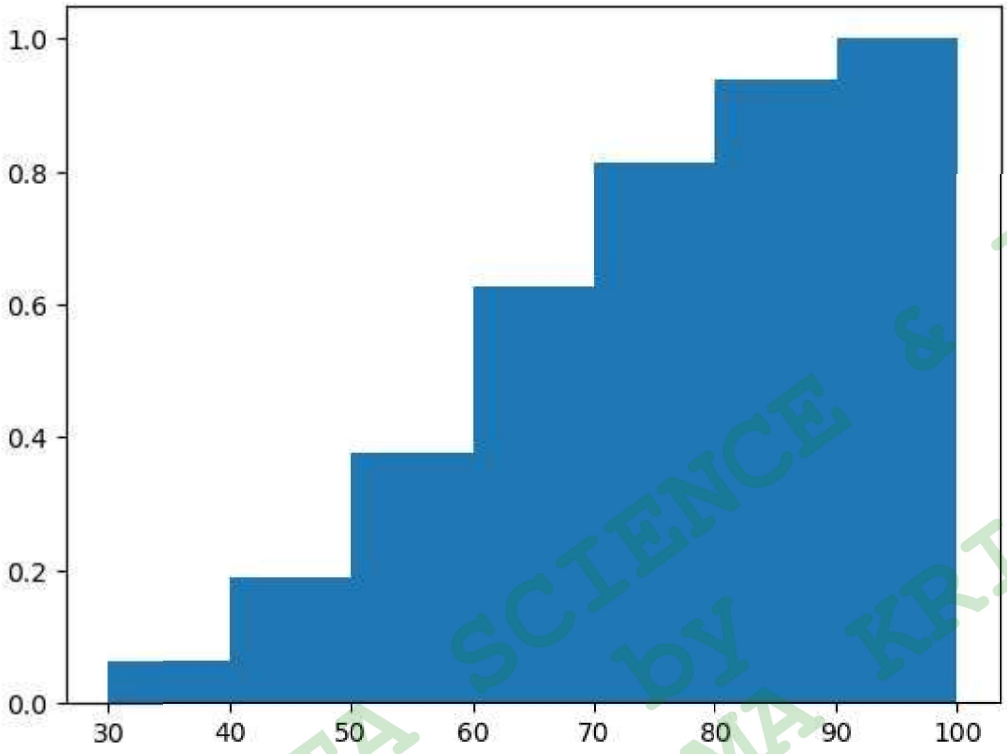
```
plt.hist(df["Marks"],bins=7,density=True)
plt.show()
```



cumulative probability distribution

In [44]:

```
plt.hist(df["Marks"],bins=7,density=True,cumulative=True)
plt.show()
```



Measure of shape (3rd Business Moment)

skewness (s)

- it tells, whether the data is symmetrical or unsymmetrical distribution
- it describes asymmetry from the normal distribution
- Symmetrical Distribution is called as **Normal Distribution**
- Unsymmetrical Distribution is known as **Skewed Distribution**

Skewness $= \frac{n}{(n-1)*(n-2)} \sum \frac{(x-\bar{x})^3}{s^3}$

In [45]:

```
df = pd.DataFrame({"X":[1,2,3,4,5]})
df
```

Out[45]:

	X
0	1
1	2
2	3
3	4
4	5

In [46]:

```
df['X'].skew()
```

Out[46]:

```
0.0
```

- If `Skewness=0` then it is **Perfect Symmetrical or Perfect Normal Distribution**
- if `Skewness<0` then it said to be a **Negative Skewed or Left Skewed Disrtibution**
- if `Skewness>0` then it said to be a **Positive Skewed or Right Skewed Disrtibution**
- $-1 < Skewness < 1$ then it is still considered to **Normal distribution**

In [47]:

```
df = pd.DataFrame({"X":[0,11,12,21,22,23,31,32,38,39,47,48,49,58,59,70],
                   "Y":[0,11,12,21,22,23,24,28,29,33,34,35,37,44,59,70],
                   "Z":[0,11,12,21,22,23,34,38,49,43,44,45,47,54,59,70]})

df
```

Out[47]:

	X	Y	Z
0	0	0	0
1	11	11	11
2	12	12	12
3	21	21	21
4	22	22	22
5	23	23	23
6	31	24	34
7	32	28	38
8	38	29	49
9	39	33	43
10	47	34	44
11	48	35	45
12	49	37	47
13	58	44	54
14	59	59	59
15	70	70	70

Symmetrical Distribution or **Normal Distribution**

$$Mean = Median$$

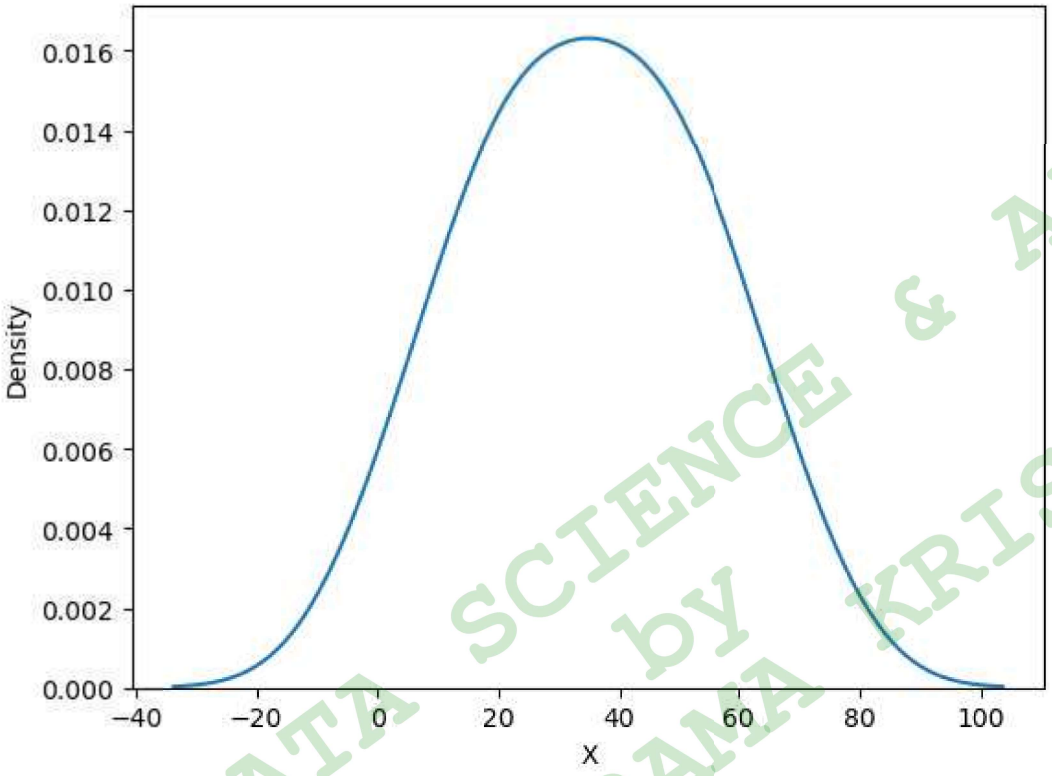
In [48]:

```
print("Mean of X:",df["X"].mean())
print("Median of X:",df["X"].median())
print("Skewness of X:",df['X'].skew())
```

```
Mean of X: 35.0
Median of X: 35.0
Skewness of X: 0.0
```

In [49]:

```
import seaborn as sns
sns.kdeplot(df['X'])
plt.show()
```



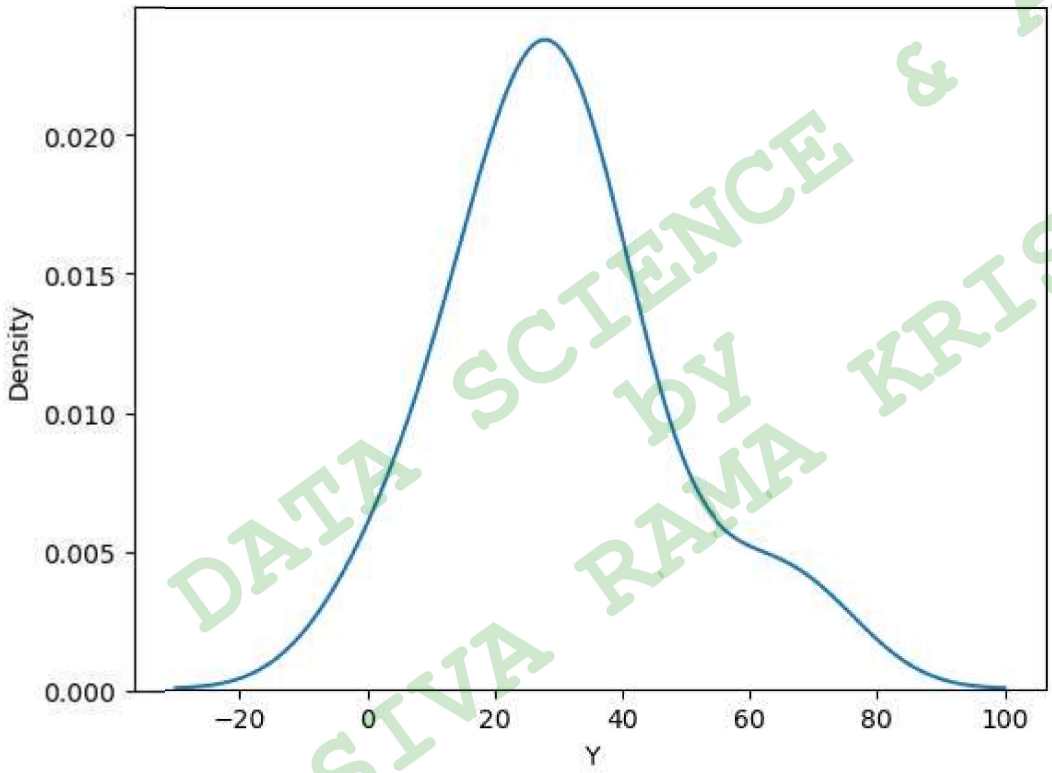
Right Skewed Distribution or **Positively Skewed Distribution**

$$Mean > Median$$

In [50]:

```
print("Mean of Y:",df["Y"].mean())
print("Median of Y:",df["Y"].median())
print("Skewness of Y:",df['Y'].skew())

sns.kdeplot(df['Y'])
plt.show()
```

```
Mean of Y: 30.125
Median of Y: 28.5
Skewness of Y: 0.6978985152470283
```



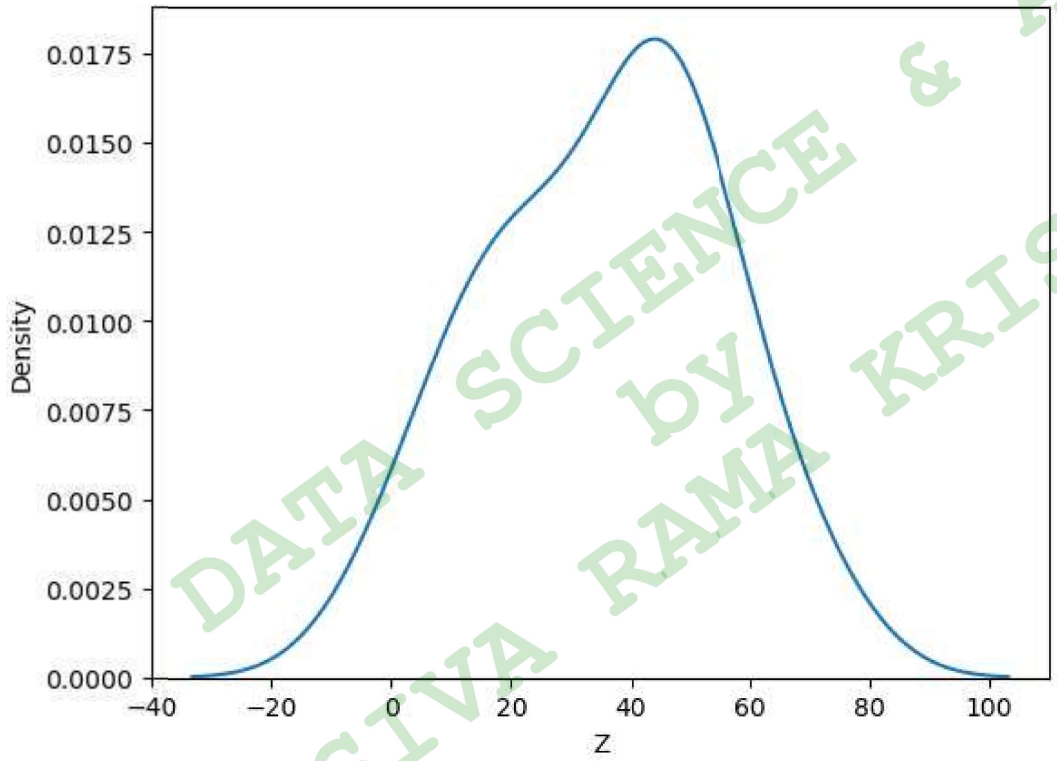
Left Skewed Distribution or **Negative Skewed Distribution**

$$Mean < Median$$

In [51]:

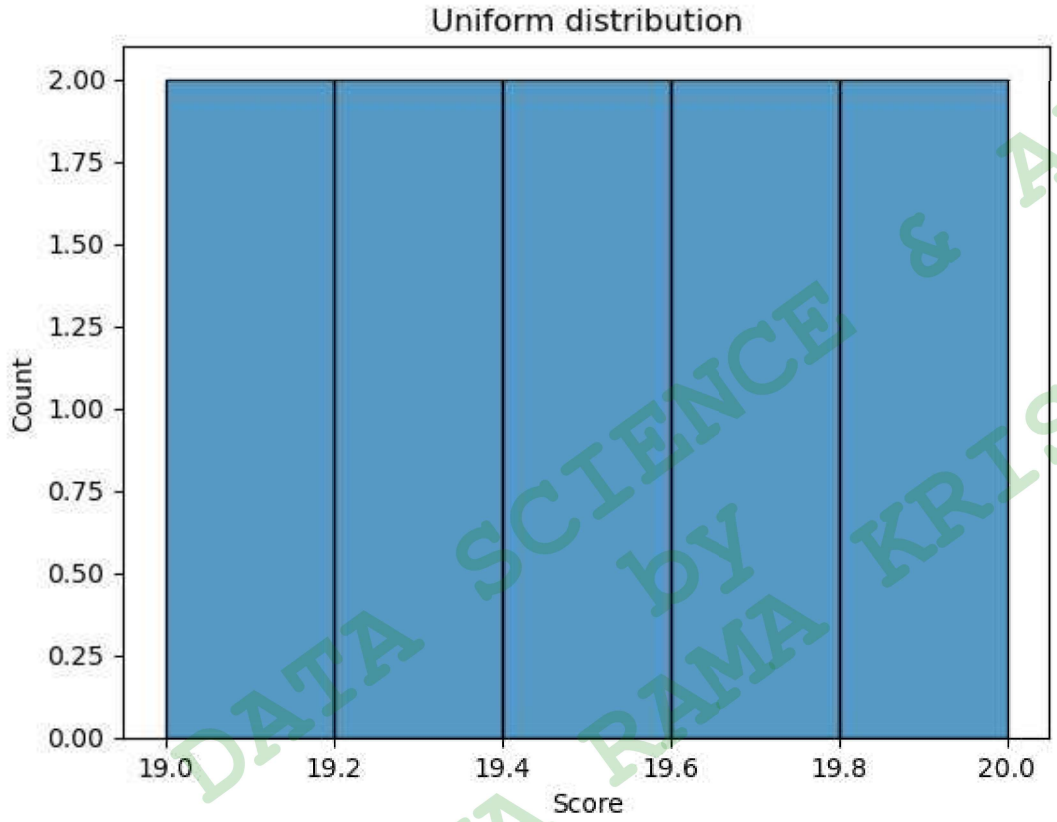
```
print("Skewness of Z:",df['Z'].skew())
print("Mean of Z:",df["Z"].mean())
print("Median of Z:",df["Z"].median())

sns.kdeplot(df['Z'])
plt.show()
```

```
Skewness of Z: -0.18882851815445098
Mean of Z: 35.75
Median of Z: 40.5
```

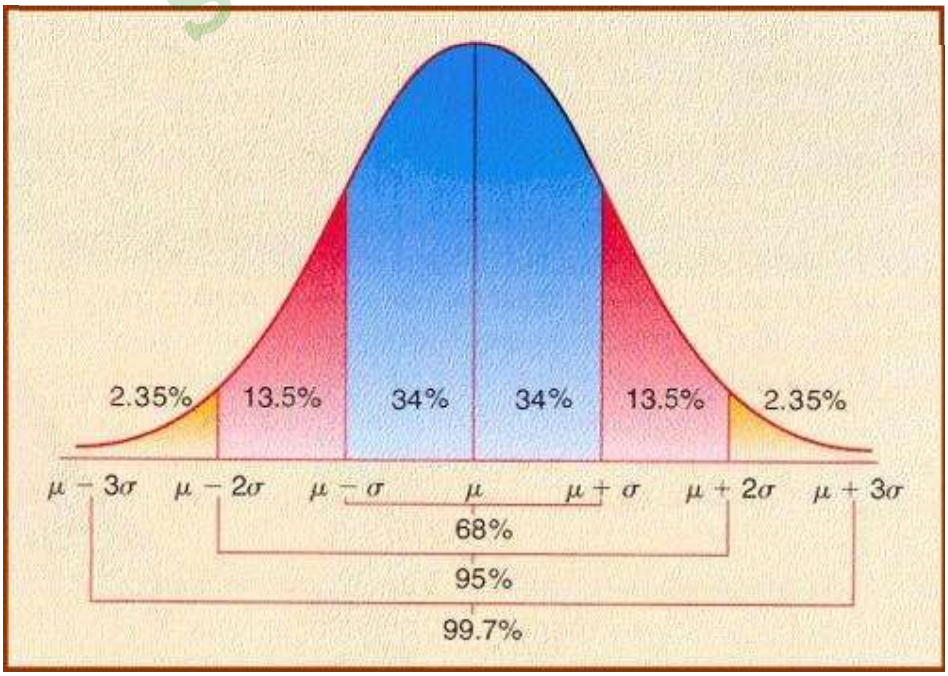


In [52]:

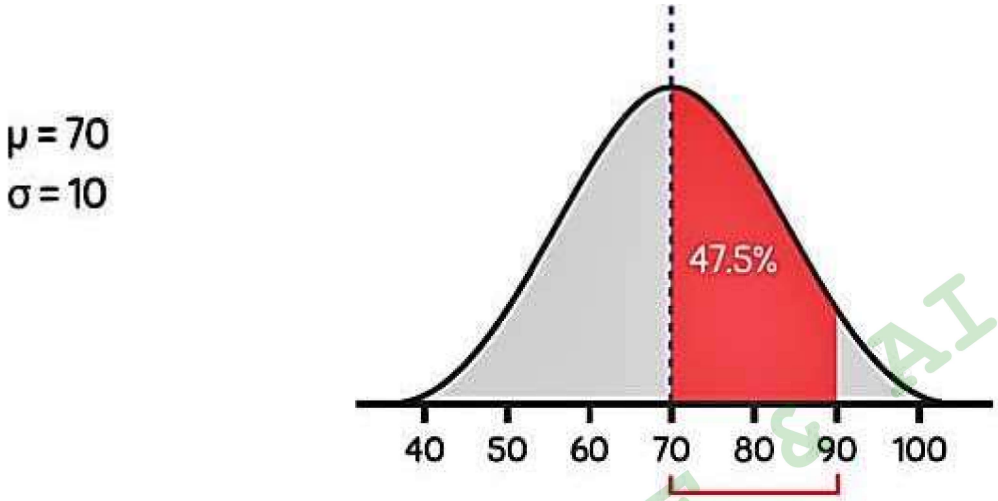
```
ds1 = pd.DataFrame({'Score':[19,19.1,19.2,19.3,19.4,19.5,19.6,19.7,19.8,20]})

sns.histplot(ds1['Score'])
plt.title('Uniform distribution')
plt.show()
```



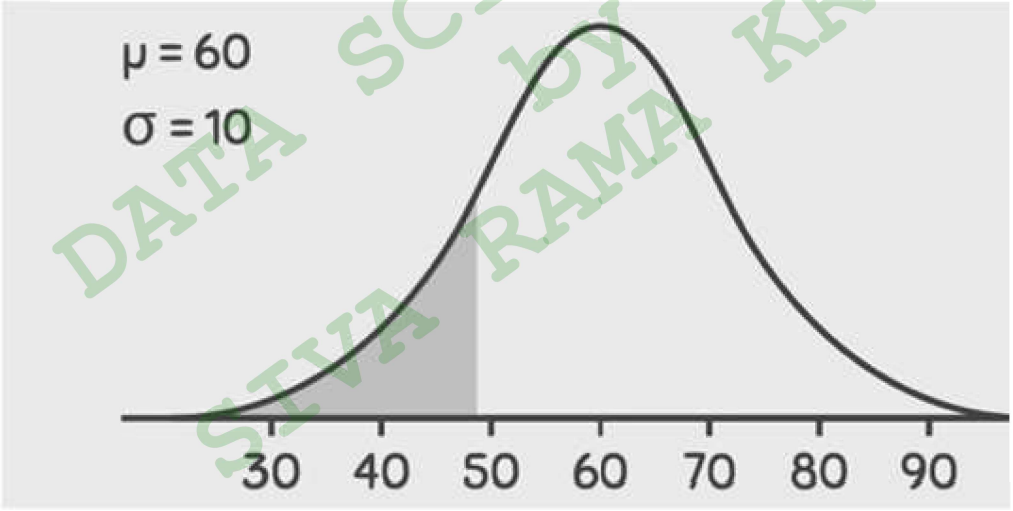
For **Normal Distribution** Data, we have **68-95-99.7% Rule**



Question1 : The normal distribution data with mean of 70 & standard deviation of 10. Approxima what area is contained between 70 and 90?



Question 2: Suppose rhat we gathered data from last mock test conducted at NareshIT and found thatit followed Normal Distribution with mean of 60 & Standard Deviation of 10. What proportion of students scored less than 49 in that exam?



Z-Score

$$Z = \frac{x - \mu}{\sigma}$$

- **Standardization** : Converting all X values to corresponding Zscores
- **Z-distribution** : Distribution of Zsores is called Z-distribution

Calculate probability using Z-score

In [53]:

```
Zvalue = (49-60)/10
Zvalue
```

Out[53]:

```
-1.1
```

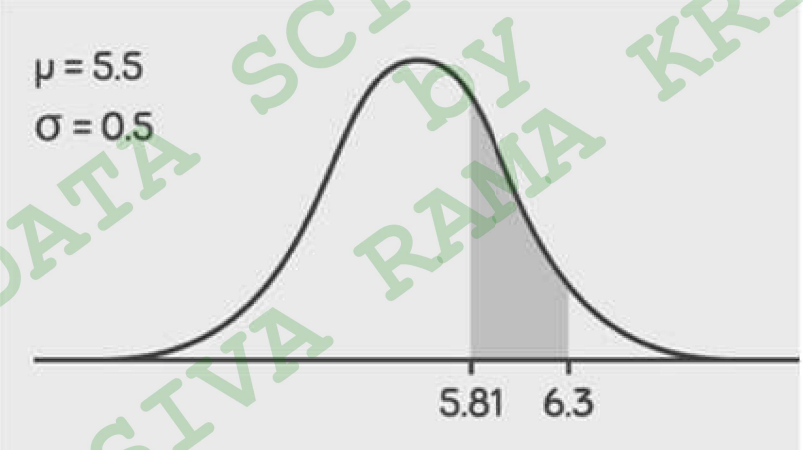
In [54]:

```
from scipy.stats import norm
norm.cdf(Zvalue)
```

Out[54]:

```
0.13566606094638267
```

Question3: When measuring the heights of all students at a local universty, it was found that it was normally distributed with a mean height of 5.5 feet and standard deviation of 0.5 feet . What proportion of studens are between 5.81 feet to 6.3 feet?



In [55]:

```
Z1 = (5.81-5.5)/0.5
p1 = norm.cdf(Z1)
print("probaility of students less than 5.81:",p1)

Z2 = (6.3-5.5)/0.5
p2 = norm.cdf(Z2)
print("probaility of students less than 6.3:",p2)

final = p2-p1
print("probaility of students between 5.81 to 6.3:",final)
```

```
probaility of students less than 5.81: 0.7323711065310168
probaility of students less than 6.3: 0.945200708300442
probaility of students between 5.81 to 6.3: 0.21282960176942523
```

Central Limit Theorem:

- For continous variable, Probability of a single value is **Zero**
- Since, Probability can' be calculated for a single value, we take a interval i.e., Point Estimate +- std. error
- **standard error** = $\frac{\sigma}{\sqrt{n}}$
- **The probability for the continuous variable is calculated on interval only for which CLT is used**

$$[\bar{X} - \frac{\sigma}{\sqrt{n}}, \bar{X} + \frac{\sigma}{\sqrt{n}}]$$

Confidence Interval (CI):

$$ConfidenceInterval = [\bar{X} - (Z_{1-\alpha})\frac{\sigma}{\sqrt{n}}, \bar{X} + (Z_{1-\alpha})\frac{\sigma}{\sqrt{n}}]$$

$$ConfidenceInterval = [\bar{X} - (Z_{1-\alpha/2})\frac{\sigma}{\sqrt{n}}, \bar{X} + (Z_{1-\alpha/2})\frac{\sigma}{\sqrt{n}}]$$

- α is the error
- if the confidence is 95% then the error is 5% so α is 0.05
- for 90% confidence $Z_{1-\alpha}$ = 1.281 & $Z_{1-\alpha/2}$ = 1.645
- for 95% confidence $Z_{1-\alpha}$ = 1.645 & $Z_{1-\alpha/2}$ = 1.96
- for 99% confidence $Z_{1-\alpha}$ = 2.326 & $Z_{1-\alpha/2}$ = 2.576
- **As confidence decreases then the interval range also increases**

Calculate zscore using probabilty

In [56]:

```
from scipy import stats
stats.norm.ppf(0.99)
```

Out[56]:

```
2.3263478740408408
```

Bivariate & Multivariate Analysis

In [57]:

```
df =  pd.DataFrame({"X":[1,2,3,4,5],"Y":[10,9,8,7,6],"Z":[11,12,13,14,15]})
df
```

Out[57]:

	X	Y	Z
0	1	10	11
1	2	9	12
2	3	8	13
3	4	7	14
4	5	6	15

Covariance:

- Covariance is used on two continous variables, unlike variance (σ^2) which is a univariant
- It is represented as cov, for x and y variable it can be rep as cov(x,y)

$$cov(x, y) = \frac{\sum(x - \bar{x})(y - \bar{y})}{n - 1}$$

- in univariant variance since it is apllied on a single variable that is var(x,x) which becomes $(x - \bar{x})(x - \bar{x})$
- The range of covariance values is $(-\infty, \infty)$
- **Zero** cov denotes no relation between two variables
- if cov is positive x and y are directly proportional, if it is negative then they are inversely proportional
- **In covariance sign is important not the value**
- it can be applied only on variables which have equal number of datapoints

In [58]:

```
df.cov()
```

Out[58]:

	X	Y	Z
X	2.5	-2.5	2.5
Y	-2.5	2.5	-2.5
Z	2.5	-2.5	2.5

Correlation:

- It measure the degree to which two variables are related to each other
- It is used to show how two variables are related, and is represented with **'r'**

$$r = \frac{cov(x, y)}{S_x * S_y} = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sqrt{\sum(x - \bar{x})^2}\sqrt{\sum(y - \bar{y})^2}}$$

- **The values of 'r' lies within [-1,1]**

- **In correlation value is important not the sign**, based on the value we can tell how related they a
- Higher the correlation values greater the correlation, the closer the datapoints the higher the correl
- The corr() function is used to calculate the correlation between **two or more independent variabl**
  Correlation is a statistical measure that shows how strongly two variables are related to each other
- |r| = 1 --> Perfect correlation
- |r| > 0.8 --> Strong correlation
- 0.5 <= |r| >= 0.8 --> Moderate
- |r| < 0.5 --> Weak correlation
- |r| = 0 --> NO correlation

In [59]:

```
df.corr()
```

Out[59]:

	X	Y	Z
X	1.0	-1.0	1.0
Y	-1.0	1.0	-1.0
Z	1.0	-1.0	1.0