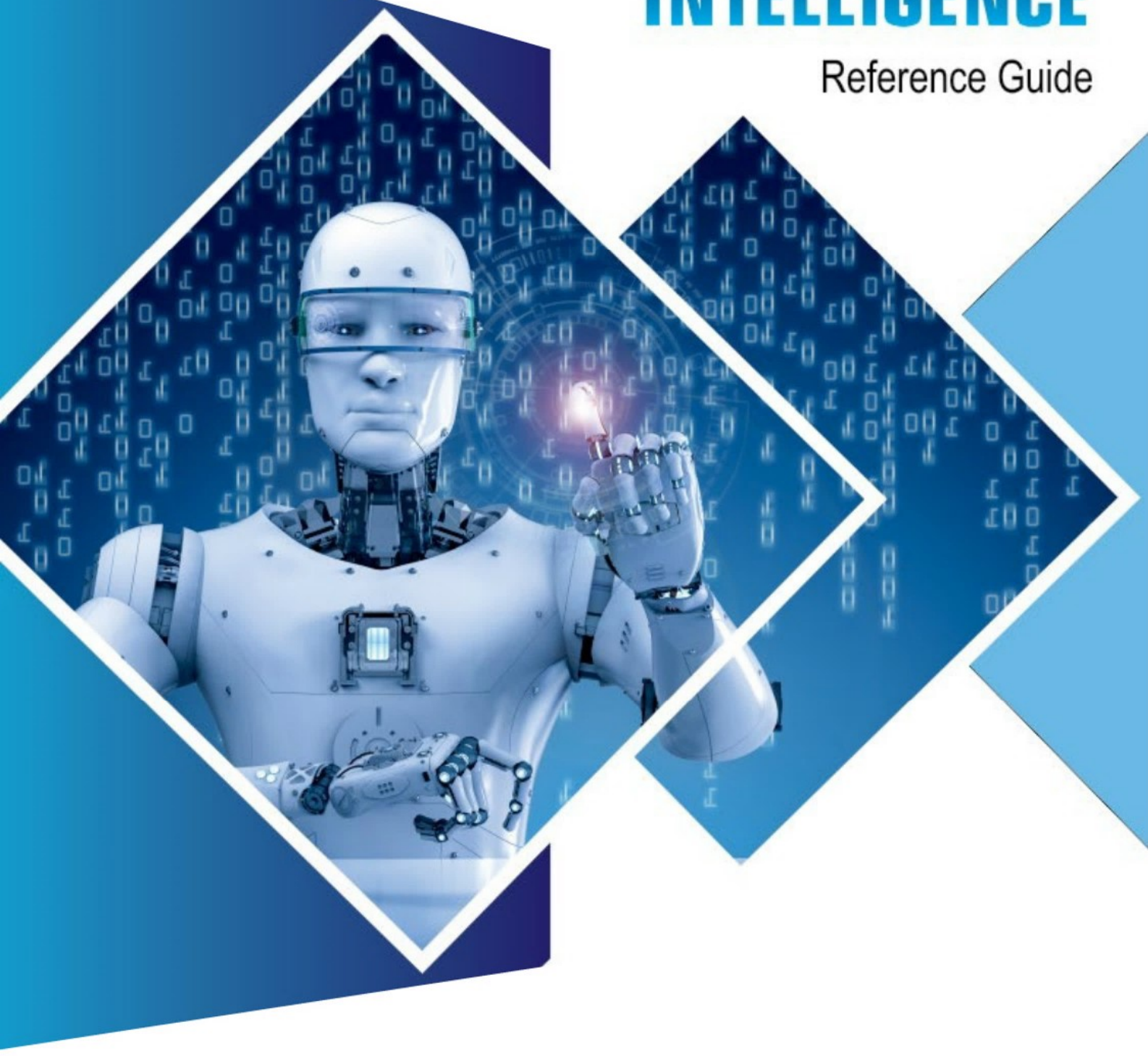
ARTIFICIAL INTELLIGENCE

Reference Guide



LIVEWIRE™

FOR LIVE CAREERS

Editor

Curriculum and Product Development Team

We appreciate your valuable feedback/suggestion on this courseware.
Kindly do mail it to us at: **cpd@livewireindia.com**

LIVEWIRE™
FOR LIVE CAREERS

CCTSPLV3082019

PREFACE

Welcome to the world of highly Intelligent System!!!

Artificial Intelligence (AI) is not only a technology it is a trend now which is used by each and every Industry as their primary requirement. AI is giving a very new era to the technologies in terms of intelligence. We have tried to cover almost complete scenario in the book for developing a robot starting from its first stage. This book is dealing with the traditional technology, mathematics, algorithm development and editing, perception, reasoning, learning and action. Some of the depth topics are covering in this book like working with Deep learning, Object segmentation etc. The name AI derived many years back but it took a lot of time to come in market as that time the algorithms are not efficient and mostly are time consuming.

LIVEWIRE's AI reference guide will help you in finding the ways making smart system with the capability of self learning without human involvement. The examples given in the book helps in getting the sight on the concept and making you build up your technical skills from scratch to high level and make your taste success in this intelligent world. We at LIVEWIRE, take every step necessary to ensure that the training that we provide and the courseware that we have developed are the best.

We have a balanced, holistic approach in viewing Artificial Intelligence from descriptive, predictive, and prescriptive that today defines the discipline.

So, enjoy learning and get into the new dimensions in your career!

S. Karaiadiselvan,
Managing Director,
LIVEWIRE
A division of CADD Centre Training Services Pvt Ltd.

CONTENTS

INTRODUCTION

Humans have the capacity of understanding something as humans are very strong with their mental power; this capacity is user for everyday live and our sense of self. So a number of things are require to elaborate, if we go with the artificial intelligence then it talks about inserting smartness into a system so that a system act like a human body. So in AI builds intelligent agents and allow to makes them understand and they are also they are interesting and useful for real applications. AI had already produced a number of attractive and significant applications till now. We cannot predict the future but human with machine intelligence will make a huge impact in market in terms of intelligence.

The study of intelligence is started past 2000 years ago, after that people are day to day progressing in making system more intelligent. For making system smart embedded system is use, but embedded only provides smartness in terms of taking decision and dnt make a system toh react like a human body. This is only possible using AI. At time AI turns to be more difficult then imagined, but implementation of new ideas makes it richer and acceptable in market.

AI currently works starting from the basic perception and reasoning to the making complex computation playing games, poetry and detecting diseases in the patient.

Let we understand the basic reasons because of which AI gets so much popularity in market.

Why AI?

Automated and centralized system:

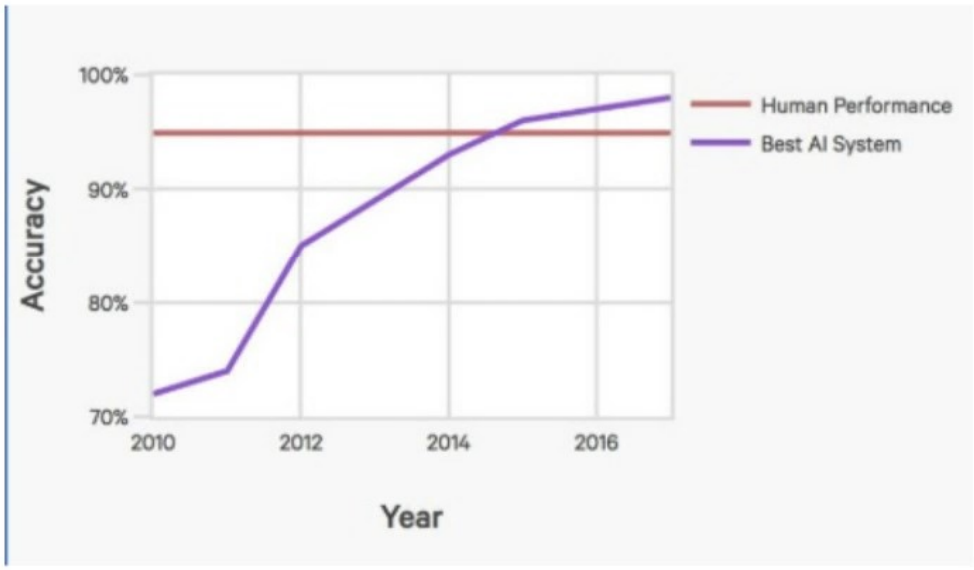
- We are growing at a very faster level and demands are also increasing and require an automated system.
- Every automated system requires some short of surveillance to find whether it is working fine or not. So an approach of centralization is fulfilling by AI.
- Automated systems are able to take decision and operate automatically without human involvement. Centralized system results in efficiently controlling multiple scenarios from a single unit.

Machine is as smart as human?

- Do you think we can replace humans with the machines?
- Do you think a machine can answer your all queries as a human do?
- Can a machine is as smart as a human being?

Is human can do something accurately?

- Can we do very difficult computations manually?
- How much accuracy we can provide while doing something?



What is AI?

- Is it a computational system which behaves intelligently?
- Is it a program which behaves like a human body internally?
- Is it a program which behaves like a human body externally?
- Is it a approach to make a machine act like a human body?
- Is it a system that act rationally(reasonable or logical way)

Turing Test

In 1950 Alan Turing gave the definition of intelligence. According to his definition peoples finds the good way between intelligent and non intelligent system.

The capabilities a machine should require for intelligence is:-

- NLP: a system should be capable to communicate through English language (or any other language)
- Knowledge representation: to store and process information provided at the time of interrogation.
- Automated reasoning: through the stored information a machine is capable of finding some solution and answers the question.
- Machine learning: to explore new patterns by understanding new circumstances.

Total Turing Test

It includes video processing to import a computer vision and robotics to make the system move. Turing test is not covering the computer vision power and the actuator part for a machine.

Approaches use to establish a pillar of AI:

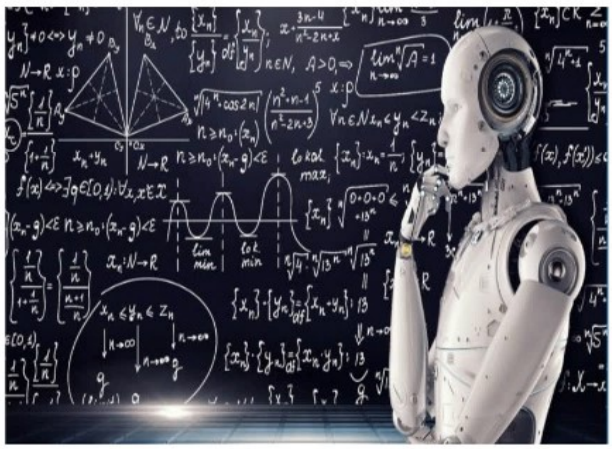
- Thinking Humanly: to understand a human two different process are adopted:
  - ➢ Through psychological experiments
  - ➢ Introspection by trying to batch our thoughts
- Thinking rationally: this provides the reasonable and logical approach. This will cover up the ways adopted to develop a process using any logic. The law of thought approach.

- Acting Rationally: it is acting one's approach to achieve a goal. The rational agent approach.

Foundation of AI:

In this part the main pillar on which AI is standing and still developing is discussed. Study of AI is from simple mathematics to complex algebra, from a myth to a reality, from a virtual to actual implementation. So one thing is not the base line of AI a no of scenarios is required as described below:

- Philosophy: This provides the number of ideas to implement for developing AI systems. So philosophy plays a very important role in AI.
- Mathematics: Machines have the capability of doing complex computations, probability and logic development, and because of it they requires mathematics.
- Psychology: To make a machine act like a human body. Understanding human brain is very important. So psychology cover the study of human mind, their way of processing information, their way of taking decision.
- Computer engineering: It is a branch of engineering which deals in the study of computer related application development and handling. For teaching a machine to act smartly this will provide the algorithm, computation, architecture and data handling.
- Linguistics: AI application should be capable of integrating with human being. It requires understanding of human language. So linguistics is the scientific study of language and its structure, including the study of grammar, syntax, and phonetics.



Img 1.1

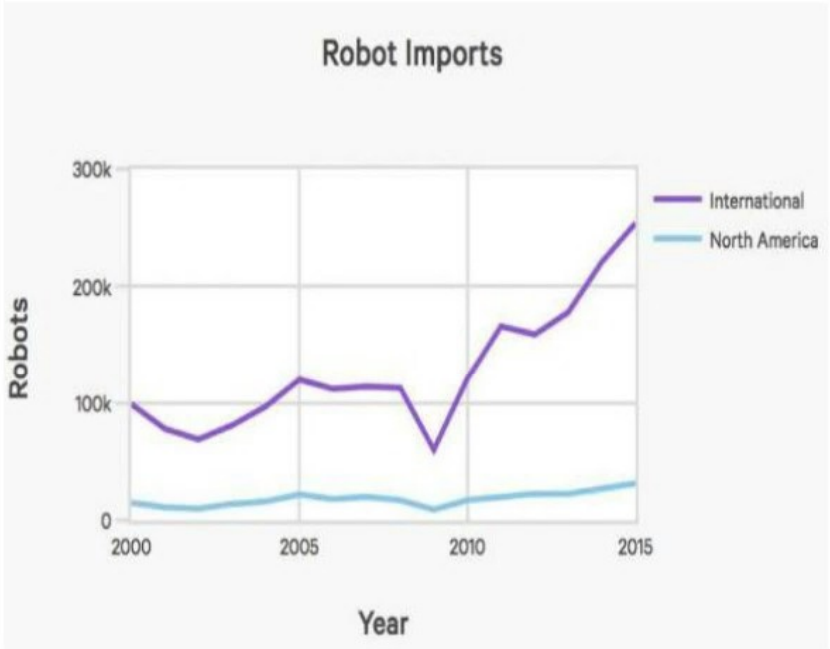
How to define AI?

In market different definition for AI comes in which many provides a good vision of writers to look for artificial intelligence. Some best suited definition if AI is stated below:-

- Artificial intelligence is the area which emphasizes the creation of intelligent machine that work and reacts like a human being.
- It is the process of simulation of human intelligence processes by machines, especially computer machines.
- A machine must produces responses indistinguishable to the human being

History of AI

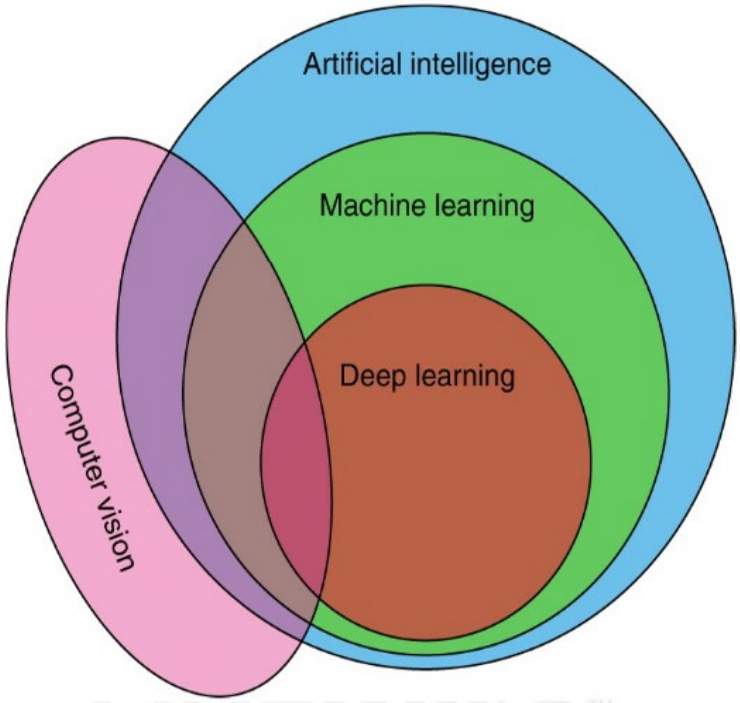
- Term AI coined by “John McCarthy” in 1955.
- After its development it didn’t get that much popularity because of not having good algorithm, good logic and fast computing hardware. So a number of things are still requiring making it popular in market.
- Modern AI was the creation of The Logic Theorist. Designed by Newell and Simon in 1955
- According to the Img 1.2 if we check out the latest trends then we can find one thing very easily that with time demand is increasing of AI.



Img 1.2

Goal of AI

- Creating expert system:- which exhibit intelligent behavior, learn, demonstrate, explain and advise
- Implementing human intelligence in machines:- To create a system which learn, think and understand like a human being.



Img 1.3

Capabilities of AI:

- Natural language processing:
  - ➢ This field deals out with the natural language, which a human can speak, write and understand.
  - ➢ Includes interactivity of a machine with the English language successfully
- Knowledge representation:
  - ➢ It is use to store knowledge gathering by the system through input devices
  - ➢ If the knowledge from input device is unstructured then the AI application should require a capability to convert it into a structured format.
- Automated reasoning:
  - ➢ Understanding of content/data is the key requirement for reasoning by a machine.
  - ➢ It analyze the information stored in the system and create new interfaces of answering
- Machine learning:

- It is a branch which deals out with the self learning power of a machine.
- It makes the machine automatically learn according to the experiences
- Their algorithms are the only pillar on which machine learning is dependent.

Applications of AI:

A number of concepts are already discussed about AI; these concepts are enough to state the applications of AI. Nowadays AI is at the top in market due to its capability to replace humans with machine. This concept reduces the involvement of humans and makes task execute with more accuracy and less time.

Some of the applications of AI is stated below:

- AI in business:
  - In business AI is use at this level because of its capability of making better decision in a business to make it grow.
  - AI makes sales and marketing automated.
- AI in vehicles
  - When it comes to vehicles the hottest topic people discuss is driverless car. A car without a driver is not just a dream it will add in the Indian list of cars soon.
  - Blind spot detection is again a good application of AI because at such places with analytic power computer eyes also require.
- AI in healthcare:
  - What happen when a human is handling some records of patients and due to their any mistake the records gets mix? Such faults are not acceptable when it is reflecting someone life.
  - AI is taking care of data management and maintaining records.
- AI in education:
  - In today world we are only progressing because our education system is getting update with new resources and new techniques. It ultimately result in development in skills and providing a good testing system.

- AI in security:
  - The definition of safety itself says nothing is safe in this world but still a level of safety is possible to insert into a system. Like using a pattern, face recognition in mobile phones. Face recognition is a part of AI.
  - Extracting some object from an image sometime required for security purpose for detecting some weapons. This part is known as image segmentation.
- AI for robotics(Discover life on other planet, chat bot):
  - Discovering life on other planet.
  - Chat-bots designing for reducing human involvement and making system automated.

STATISTICS

Data Types

In programming, we call a data type as a classification that states a variable has which type of value. A data type also specifies the type of operations (mathematical, logical, etc.,) that can be applied without receiving an error.

Basically, there are two types of data:

1. Qualitative

Qualitative data is said to be the data which can be observed subjectively. In other words, it connects with the characteristics and how an object can be described such as smell, shape, texture, colour, etc., when you try to classify something based on its characteristics, you are dealing with qualitative data.

These are Non-numerical and Categorical.

Example: colour, model, brand, type of defect, etc.

2. Quantitative

Quantitative data is something which you can measure based on the numbers (count). If you measure the dimensions of an object such as Length (L), width (W) and height (H), you obviously give a numeric value to it. Hence you are dealing with Quantitative data.

Example: counts, time, temperature, mass, weight, length, etc.

Quantitative Data can be further divided into:

a. Discrete

We call a data as Discrete when the count is more precise which involves integers. To understand better, the number of students in a class is a discrete data, because it will anyway be a whole number. i.e., it will always be 45, 57 students but it never can be 48.5 or 32.7 students.

Example: Counts, no. of people within a certain blood type, no. of dots on the top faces of a pair of dice,etc.

The number of Jujubes in a particular box is a discrete data.

b. Continuous

Continuous data is something which can be divided or reduced to have better or more accurate results. For better understanding, we can measure the height of an object with different scales to get more precise scales such as meters, centimeters, millimeters and so on.

Example: Height, Weight, Length, Speed, Temperature, etc.

The weight of each Jujube in a particular box or the weight of the entire box is a continuous data.

Four Levels of Data

The descending order of precision, the four different levels of measurement are:

1. Nominal
2. Ordinal
3. Interval
4. Ratio

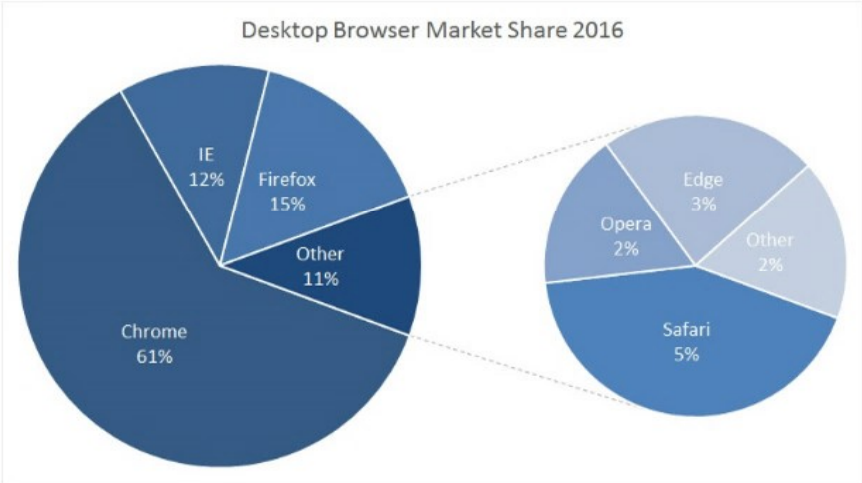
*First letters spell as **NOIR** = Nominal, Ordinal, Interval, Ratio*

Qualitative Levels

Nominal

The first level of measurement is called the nominal level of measurement. In this level, the numbers in the variable are used only to classify the data. In this level of measurement, words, letters, and alpha-numeric symbols can be used.

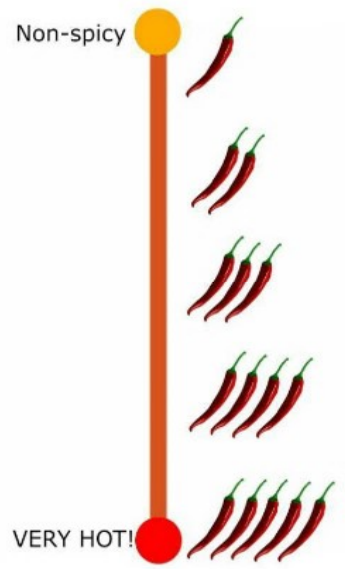
Suppose there are data about people belonging to three different gender categories. In this case, the person belonging to the female gender could be classified as F, the person belonging to the male gender could be classified as M, and transgendered classified as T. This type of assigning classification is the *nominal level of measurement.*



Example: colour, manufacturer, flavour, etc.

Ordinal

This is the second level of measurement which depicts some ordered relationship among the variable's observations. Suppose a student scores the highest grade of 100 in the class, he will be assigned the first rank. Then, another student scores the second highest grade of 92, will be assigned the second rank. A third student scores 81 will be assigned the third rank, and so on. The ordinal level of measurement indicates an ordering of the measurements.



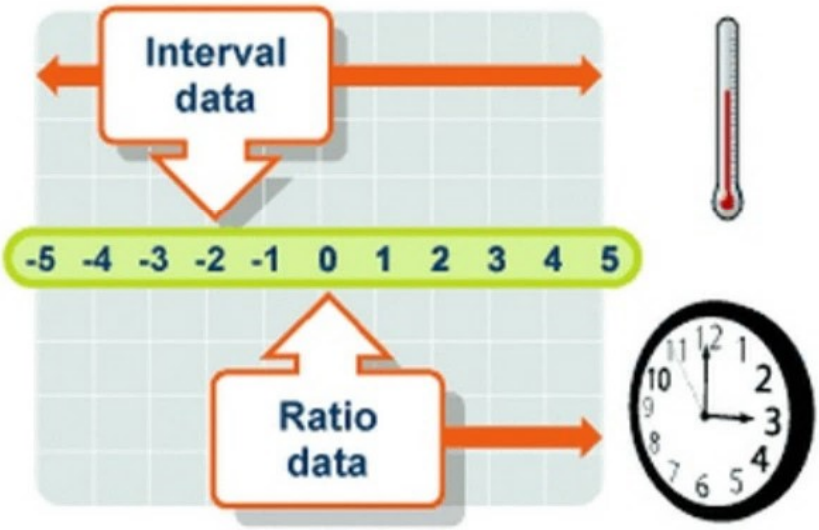
Example: Likert Type Scale (Strongly agree, Agree, Disagree, StronglyDisagree), Sizes (XS, S, M, L, XL), Places (First, Second, Third),etc.

Quantitative Levels

Interval

This is the third level of measurement and it not only classifies and orders the measurements, but it also specifies that the distances between each interval on the scale are equivalent along the scale from low interval to high interval. For example, an interval level of measurement could be the measurement of anxiety in a student between the score of 10 and 11, this interval is the same as that of a student who scores between 40 and 41.

Example: *Measurement of temperature in centigrade, where, for example, the difference between* $940℃\ and\ 960℃$ *is the same as the distance between* $1000℃\ and\ 1020℃$.



Ratio

In this level of measurement, the observations, in addition to having equal intervals, can have a value of zero as well. The zero in the scale makes this type of measurement unlike the other types of measurement, although the properties are similar to that of the interval level of measurement. In the ratio level of measurement, the divisions between the points on the scale have an equivalent distance between them.

Example:Counts, Weights, Height, etc.

Comparison

Provides:	Nominal	Ordinal	Interval	Ratio
The "order" of values is known		✔	✔	✔
"Counts," aka "Frequency of Distribution"	✔	✔	✔	✔
Mode	✔	✔	✔	✔
Median		✔	✔	✔
Mean			✔	✔
Can quantify the difference between each value			✔	✔
Can add or subtract values			✔	✔
Can multiple and divide values				✔
Has "true zero"				✔

Random Variable

A random variable is a function that associates a unique numerical value with every outcome of an experiment. It is variable because it can take one of the several possible values. It is random because there is some chance associated with each possiblevalue.

A random variable in probability is most commonly denoted by capital **X**, and the small letter **x** is then used to ascribe a value to the random variable.

For examples, given that you flip a coin twice, the sample space for thepossible outcomes is given by the following:

$$S = \{HH, HT, TH, TT\}$$

The random variable **X** can be given by thefollowing:

$$X = \begin{Bmatrix} HH \\ HT \\ TH \\ TT \end{Bmatrix}$$

There are two types of random variables:

1. Discrete variables and
2. Continuousrandom variables.

Discrete Random Variables

The word discrete means separate and individual. Thus, discrete random variables are those that take on integer values only. They never include fractions or decimals.

Discrete random variables give rise to discrete probability distributions.

Examples:

- total of roll of two dice: 2, 3, . . . ,12
- number of desktops sold: 0, 1, . ..
- customer count: 0, 1, . ..

Discrete Random Variable Application

- The number of aero planes taking off and landing during a given time in an airport.
- There are 2 PIA flights landing from Islamabad airport to King Abdul Aziz Airport at Jeddah and 2 PIA flights departing from King Abdul Aziz Airport at Jeddah every day
- So, 2 is a discrete number
- And can be denoted as X discrete random variable.
- In any business firm, there is a communication system with a certain number of lines communicating data and voice.
- If we need to know the probability of how many lines are working at one time we use discrete variables.

Continuous Random Variable

Continuous is the opposite of discrete. Continuous random variables are those that take on any value including fractions and decimals.

Continuous random variables give rise to continuous probability distributions.

Examples:

- interest rate: 3.25%, 6.125%, . ..
- task completion time: a non-negativevalue
- price of a stock: a non-negativevalue

Continuous Random Variable Application

- Error in the reaction temperature may be defined by a continuous random variablewith any probability density function
- Task completion time
- Suppose a construction project to be completed in 20 to 24 months and its probability is 0.05
- There is infinite sample space between 20 to 24 months.

Exercise:

Identify the set of possible values for each random variable. Classify each random variable as either discrete or continuous.

1. The number of arrivals at an emergency room between midnight and 6:00a.m.
2. The weight of a box of cereal labelled "18ounces".
3. The duration of the next outgoing telephone call from a businessoffice.
4. The number of kernels of popcorn in a 1- poundcontainer.
5. The number of applicants for ajob.

Answers:

1. discrete
2. continuous
3. continuous
4. discrete
5. discrete

Probability

Probability is the chance that something will happen - how likely it is that some event will occur. Sometimes you can measure a probability with a number like "10% chance", or you can use words such as impossible, unlikely, possible, even chance, likely and certain.

$$\text{Probability of an event happening} = \frac{\text{Number of ways it can happen}}{\text{Total number ofoutcomes}}$$

Example:

1. The chances of rolling a “4” with a die?

Solution:

Number of ways it can happen → 1 (there is only one face with a “4” on it)

Total number of outcomes → 6 (there are 6 faces altogether)

$$So, the\ probability = \frac{1}{6}$$

2. There are 5 marbles in a bag: 4 are blue and 1 is red. What is the probability that a blue marble gets picked?

Solution:

Number of ways it can happen → 4 (there are 4 blues)

Total number of outcomes → 5 (there are 5 marbles in total)

$$So, the\ probability = \frac{4}{5} = 0.8$$

Probability Line:

We can show probability on a probability line:

Discrete Probability Distribution

A discrete probability distribution lists the possible values of the random variable, with its probability.

Example: A survey asks a sample of families how many vehicles each owns.

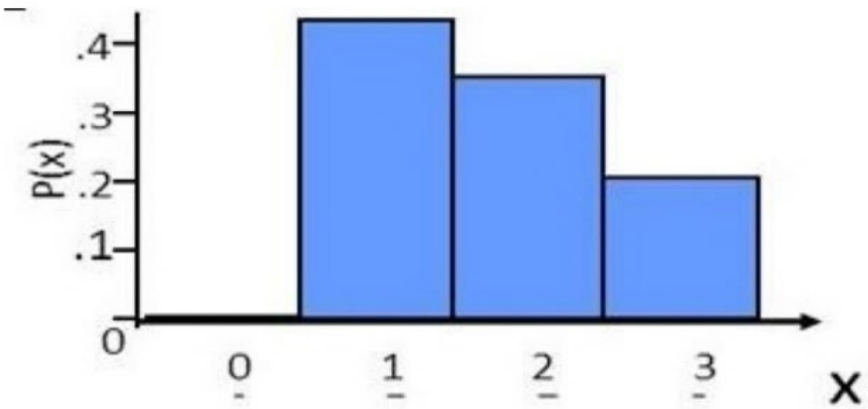
number of vehicles

x	P(x)
0	0.004
1	0.435
2	0.355
3	0.206

Number of Vehicles

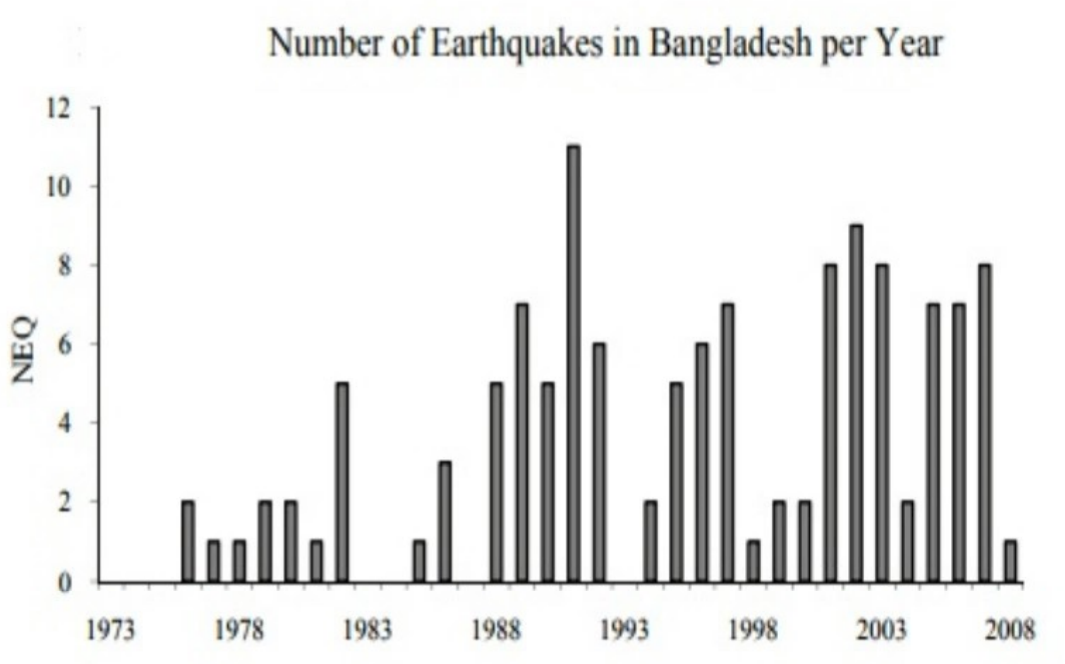
Conditions of a probability distribution:

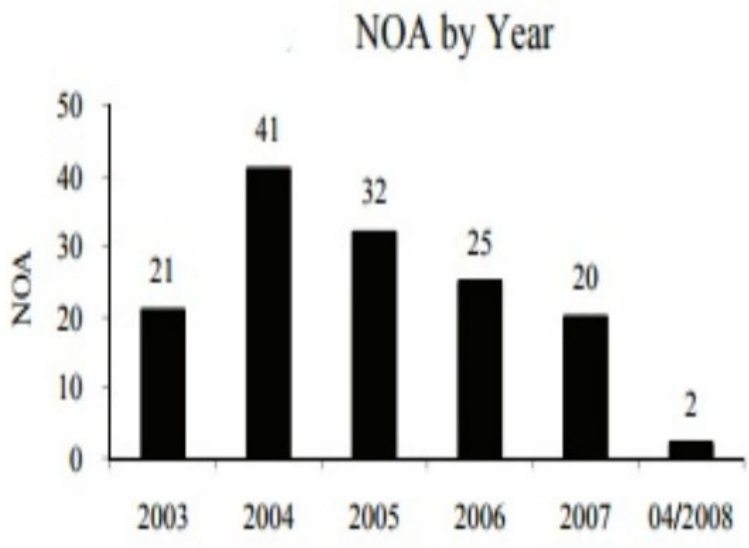
- Each probability must be between 0 and 1, inclusive
- The sum of all probabilities is 1

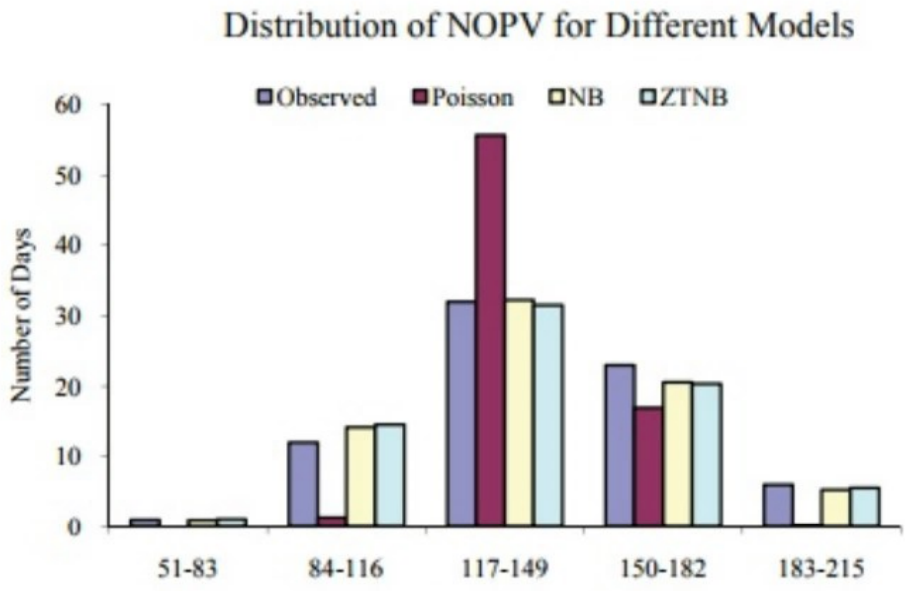


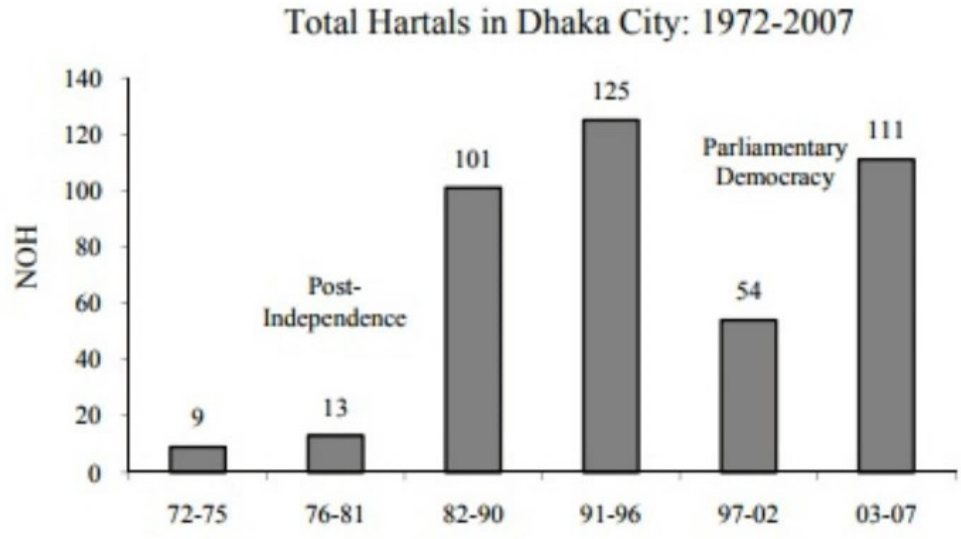
Applications of Discrete Probability

Type	Time period	n	Data Source
The number of traffic accidents (NOA) in the Dhaka district per month	Jan-2003 to April-2008	64 months	The daily star Newspaper
The number of people visiting (NOPV) Dhaka BMSSU per day	April-2007 to July-2007	74 days	BMSSU, Dhaka
The number of earthquakes (NEQ) in Bangladesh per year	1973 to 2008	37 years	http://neic.usgs.gov/cgi-bin/epic/epic.cgi
The number of hartals (NOH) in the city of Dhaka per month	Jan-1972 to Dec-2007	432 months	Dasgupta (2001) and the Daily Star Newspaper





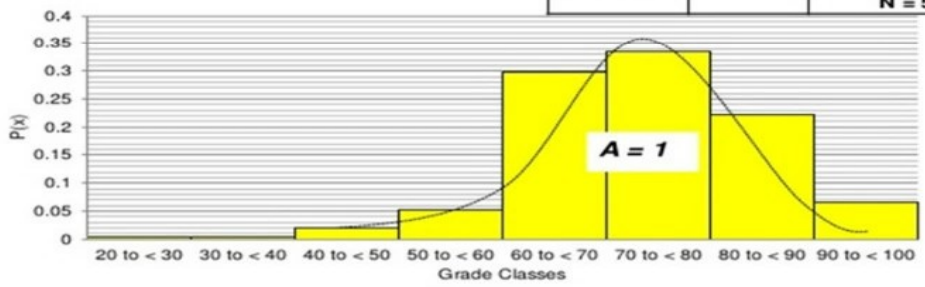




Continuous Probability Distribution

Example: Suppose 5000 students took a course on statistics in a college over the last 5 years. The categories of grades and corresponding frequencies are as shown in the Table below. Construct a probability distribution of student's grade.

Student grades (%)	Mid-point (x)	Number of students (frequency, f)	Relative frequency (RF)
20 to < 30	25	16	0.0032
30 to < 40	35	20	0.004
40 to < 50	45	98	0.0196
50 to < 60	55	256	0.0512
60 to < 70	65	1490	0.298
70 to < 80	75	1675	0.335
80 to < 90	85	1111	0.2222
90 to < 100	95	334	0.0668
		N = 5000	Sum = 1

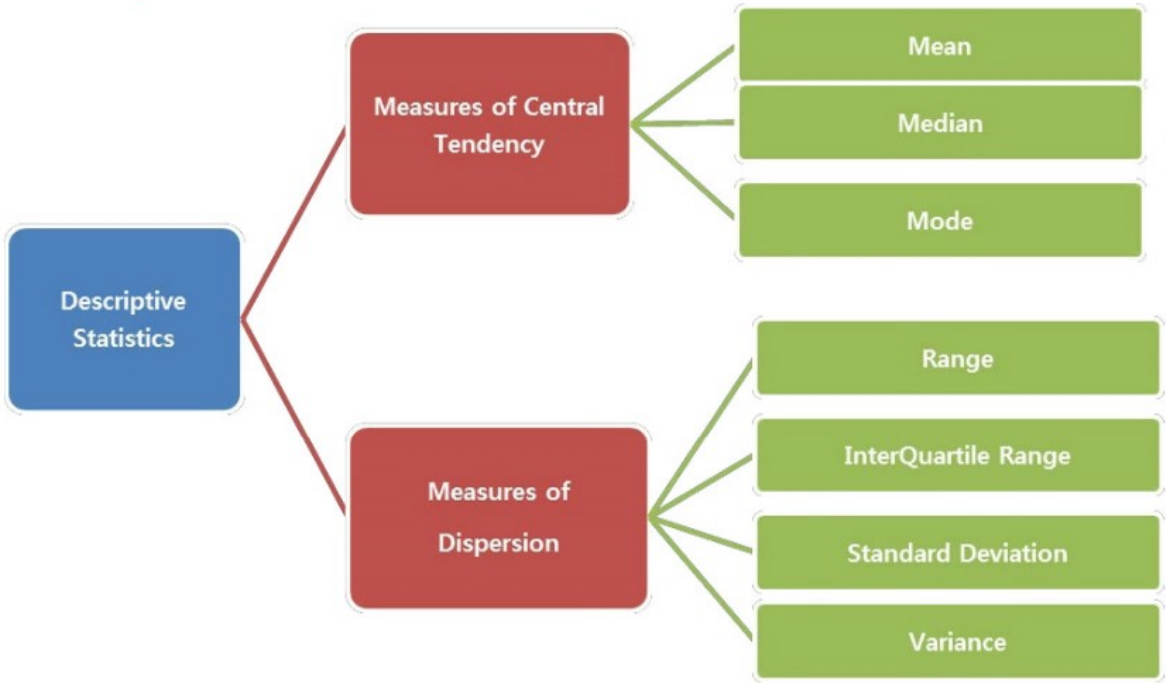


Applications of Continuous Probability

Sales Forecasting

One practical use in business analysis is to predict future levelssales. It is impossible to forecast precise value, however, using the forecasted values businesses can plan for future events.

Descriptive Statistics



First Moment Business Decision

Measures of Central Tendency

Mean:

The sum of all data values divided by the number of values.

For a population:

$$\mu = \frac{\sum x}{N}$$

For a sample:

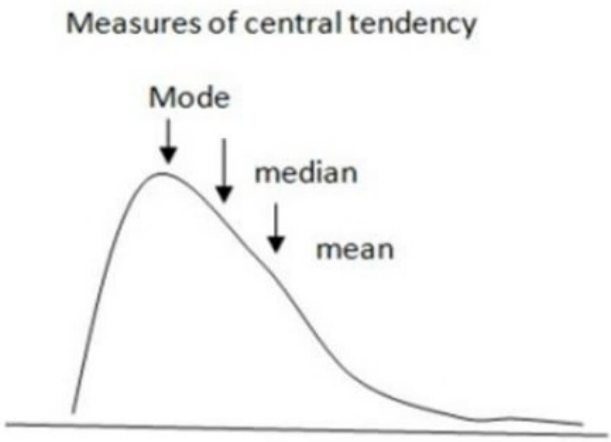
$$\bar{x} = \frac{\sum x}{n}$$

Median:

The point at which an equal number of values fall above and fall below.

Mode:

The value with the highest frequency.



Mean	Median	Mode
The sum of all values divided by the number of values	The middle number of an ordered data set	The number of item in a data set which appears most often
$Mean = \frac{Sum\ of\ Values}{Number\ of\ Values}$	If there are two middle numbers, find the mean of those numbers	A data set may have one mode, no mode or several modes

Second Moment Business Decision

Measures of Dispersion

Population Variance

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu^2)}{N}$$

Population Standard Deviation

$$\sigma = \sqrt{\frac{\sum(X-\mu)^2}{N}}$$

Sample Variance

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

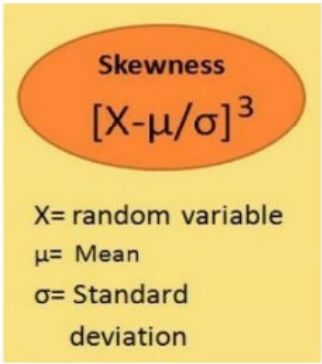
Sample Standard Deviation

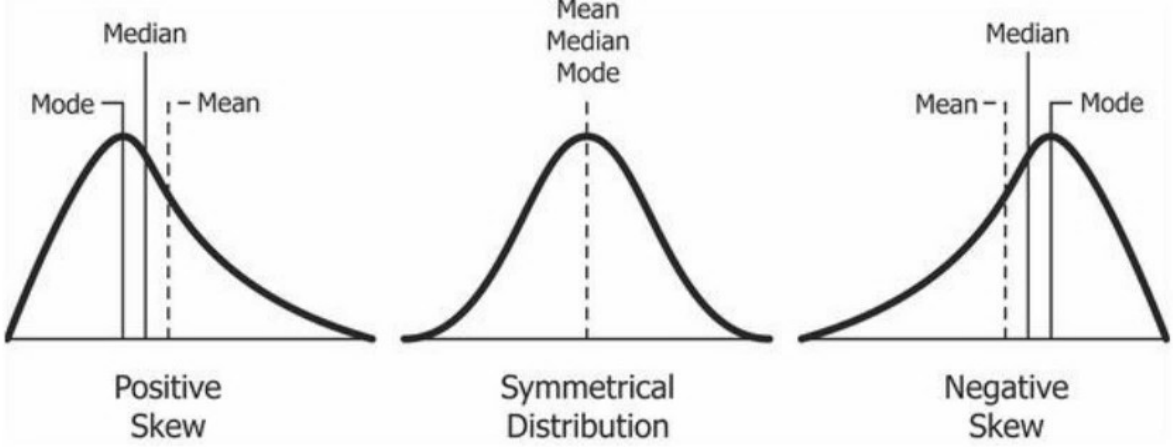
$$S = \sqrt{\frac{\sum(X-\bar{X})^2}{n-1}}$$

$$Range = max - min$$

Third Moment Business Decision - Skewness

It is a measure of symmetry. A distribution is symmetric if it looks the same to the left and right of the center point.





Fourth Moment Business Decision - Kurtosis

Kurtosis

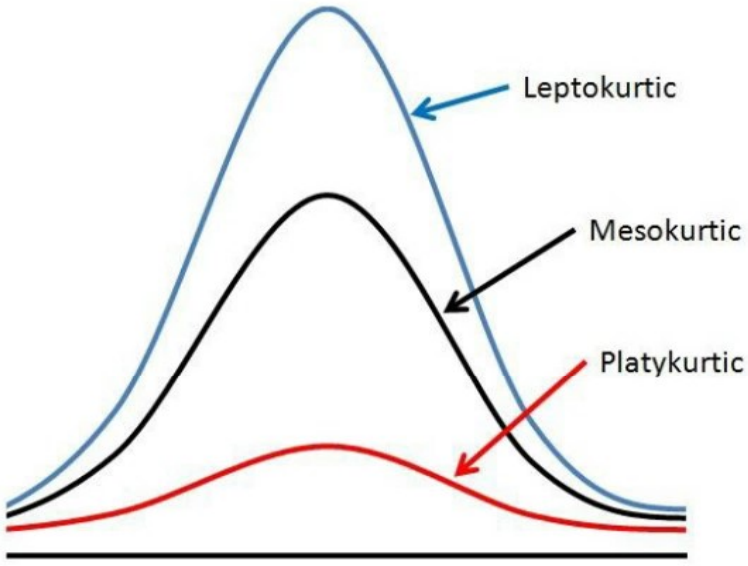
$$[X-\mu/\sigma]^4-3$$

X= random variable

μ= Mean

σ= Standard deviation

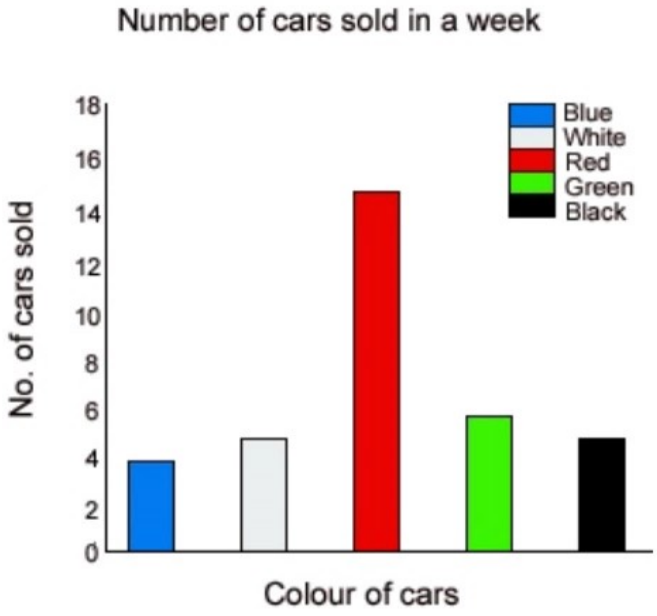
It is a measure of whether the data are peaked or flat relative to the rest of the data. Higher values indicate a higher, sharper peak; lower values indicate a lower, less distinct peak.



Graphical Representation

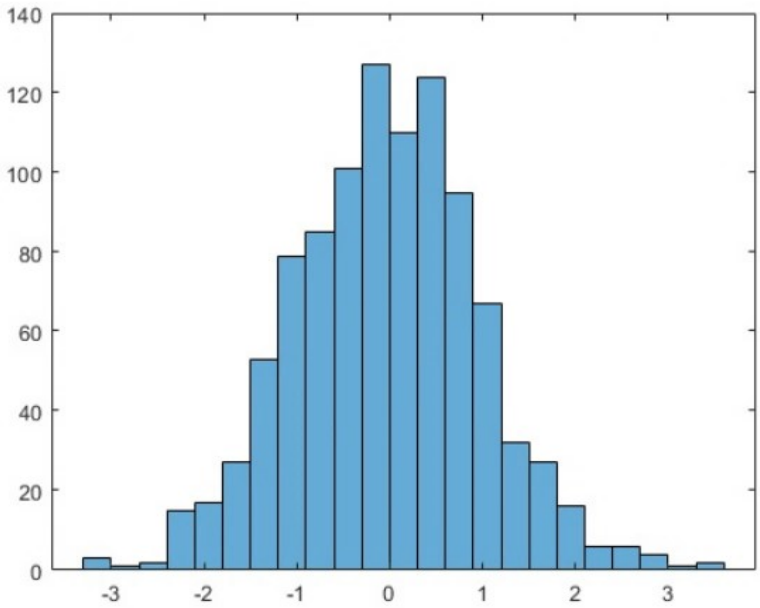
Bar Plot:

A bar plot is one of the most common types of plot. It shows the relationshipbetween a numerical variable and a categorical variable. For example, you can display the height of several individuals using a bar chart.



Histogram:

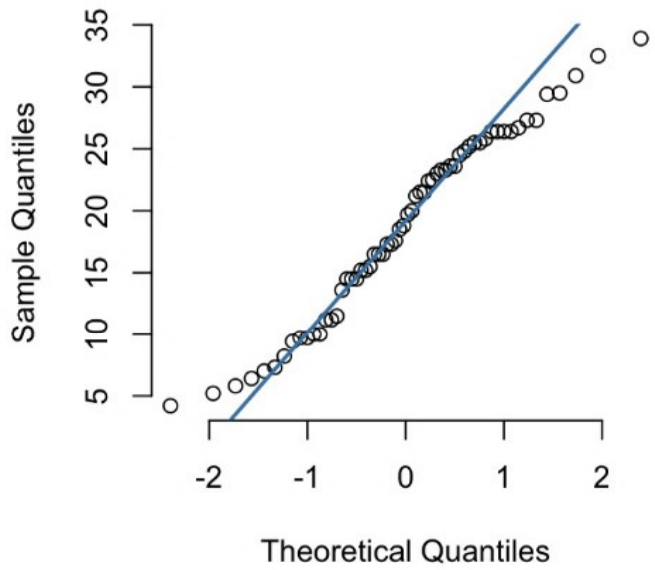
A histogram is a plot that lets you discover, and show, the underlying frequency distribution (shape) of a set of continuous data. This allows the inspection of thedata for its underlying distribution (e.g., normal distribution), outliers, skewness, etc. An example of a histogram and the raw data it was constructed from is shown below:



BoxPlot:

- The box plot (a.k.a. box and whisker diagram) is a standardized way of displaying the distribution of data based on the five-number summary: minimum, first quartile, median, third quartile, andmaximum.
- In the simplest box plot, the central rectangle spans the first quartile to the third quartile (the *interquartile range* or*IQR*).
- A segment inside the rectangle shows the median and "whiskers"above and below the box show the locations of the minimum and maximum.

Normal Quantile (Q-Q) plot:



Sampling Funnel

Sampling

Sampling is an important component of any piece of research because of the significant impact that it can have on the quality of yourresults/findings.

Major Sampling terms to be known are:

1. Population
2. Units
3. Sample
4. Sample Size
5. Sampling Frame
6. Sampling Techniques
7. Sampling Bias

Population:

In sampling, a population signifies the units that we are interested in studying. These units could be **people**, **cases** and **pieces of data**. Some examples of each of these types of the population are present below:

People

Nurses working at hospitals in the State of Texas Users of Facebook or Twitter

Cases (i.e., organisations, institutions, countries,etc.)

Recruitment agencies in Greater London, England Law firms in Manhattan, New York, UnitedStates

Pieces ofdata

The braking distances (in km/m) of a particular model ofcar University applications in the United States in 2011

Sample:

When we are interested in a population, it is often impractical and sometimes undesirable to try and study the entire population. For example, if the population we were interested in was frequent, male Facebook users in the United States, this could be millions of users. Instead, we choose to study just a sample of these Facebook users.

Sample Size:

The sample size is simply the number of units in your sample. In the example above, the sample size selected may be just 500 or 1000 of the Facebook users that are part of our population of frequent, male, Facebook users in the United States.

Sampling Frame:

The sampling frame is very similar to the population you are studying and may be exactly the same. When selecting units from the population to be included in your sample, it is sometimes desirable to get hold of a list of the population from which you select units. This is the case when using certain types of sampling technique (i.e., probability sampling techniques), which we discuss later in the article. This list can be referred to as the sampling frame.

Sampling Bias:

Sampling bias occurs when the units that are selected from the population for inclusion in your sample are not characteristic of (i.e., do not reflect) the population. This can lead to your sample being unrepresentative of the population you are interested in.

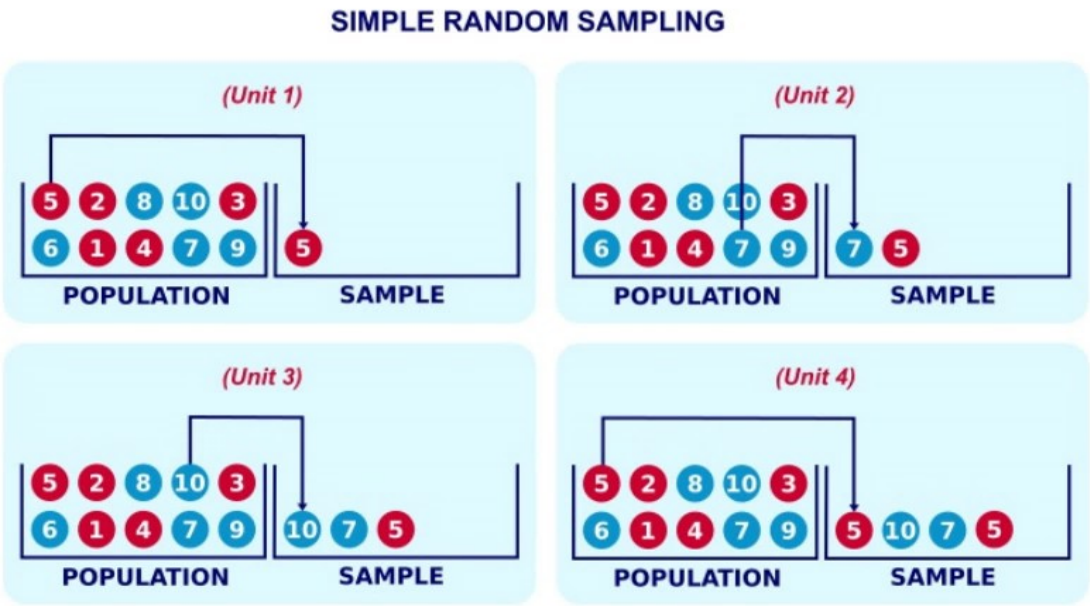
For example, you want to measure how often residents in New York go to a Broadway show in a given year. Clearly, standing along Broadway and asking people as they pass by how often they went to Broadway shows in a given year would not make sense because a higher proportion of those passing by, are likely to have just come out of a show. The sample would, therefore, be biased.

Simple Random Sampling

Simple random sampling is a type of probability sampling technique.

- With the simple random sample, there is an equal chance (probability) of selecting each unit from the population being studied when creating yoursample.

- Imagine that a researcher wants to understand more about the career goals of students at a singleuniversity.
- Let's say that the university has roughly 10,000students.
- In order to select a sample (*n*) of students from this population of 10,000 students, we could choose to use a simple random sample.
- With simple random sampling, there wouldan equal chance (probability) that each of the 10,000 students could be selected for inclusion in oursample.



Sampling Methods

The purpose of sampling techniques is to help you select units (e.g., Facebook users) to be included in your sample (e.g., of 500 Facebook users).

Broadly speaking, there are two groups of sampling technique:

1. Probability Sampling Techniques and
2. Non-probability Sampling Techniques.

Probability Sampling

Probability sampling techniques use random selection (i.e., probabilistic methods) to help you select units from your sampling frame (i.e., similar or exactly that same as your population) to be included in your sample.

Types:

- Simple Random Sampling,
- Stratified Sampling,
- Cluster Sampling.
- Multistage Sampling,
- Systematic Random Sampling

Non-probability Samples

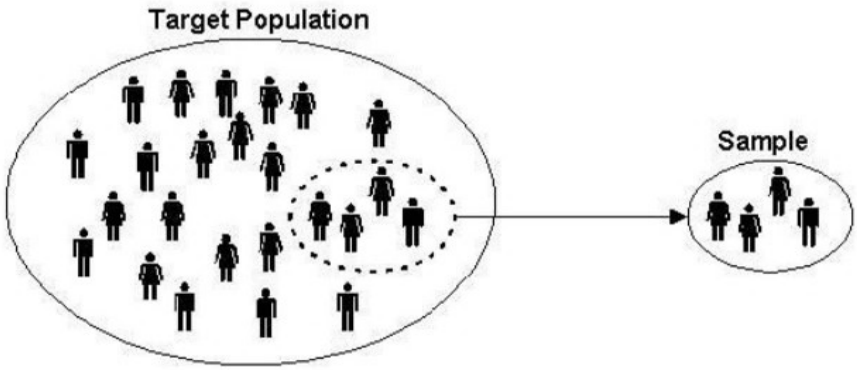
In such samples, one cannot be assured of having known probability of each population element. Non-probability sampling techniques refer to the subjective judgement of the
researcher when selecting units from the population to be included in the sample.

Types:

- Quota sampling,
- Purposive sampling,
- Convenience sampling,
- Snowball sampling and
- Self-selection sampling.

Inferential Statistics

- Often, however, you do not have access to the whole population you are interested in investigating, but only a limited number of data instead.



- For example, youmight be interested in the exam marks of all students in the UK.
- It is not feasible to measure all exam marks of all students in the whole of the UK so you have to measure a smaller sample of students (e.g., 100 students), which are used to represent the larger population of all UK students.

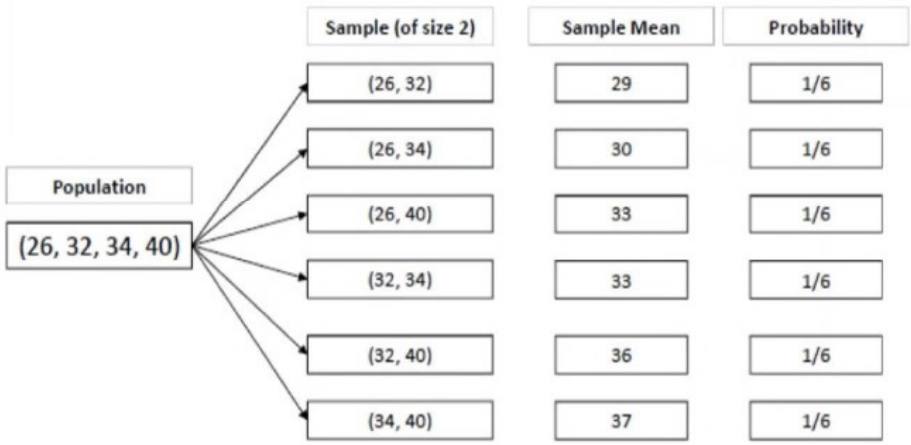
- Properties of samples, such as the mean or standard deviation, are not called parameters, but statistics.
- Inferential statistics are techniques that allow us to use these samples to make generalizations about the populations from which the samples weredrawn.
- It is, therefore, important that the sample accurately represents thepopulation.
- The process of achieving this is called sampling (sampling strategies are discussed in detail here on our sistersite).
- Inferential statistics arise out of the fact that sampling naturally incurs sampling error and thus a sample is not expected to perfectly represent thepopulation.

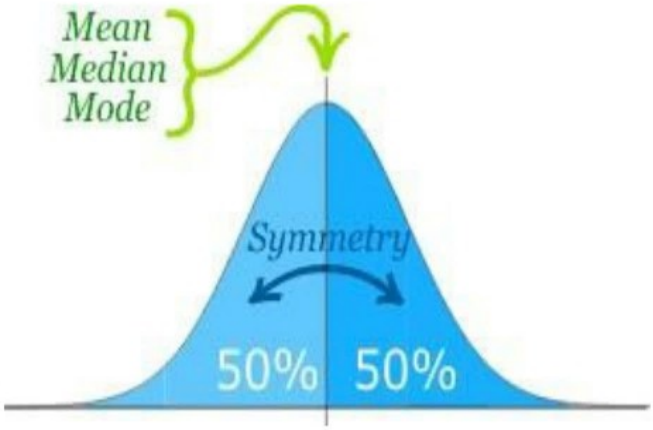
There are two main areas of inferentialstatistics:

1. **Estimating Parameters:** This means taking a statistic from your sample data (for example the sample mean) and using it to say something about a population parameter (i.e. the populationmean).
2. **Hypothesis Tests:** This is where you can use sample data to answer research questions. For example, you might be interested in knowing if a new cancer drug is effective. Or if breakfast helps children perform better inschools.

Sampling Variation

- Sample mean varies from one sample to another
- A sample mean can be (and most likely is) different from the population mean
- A sample mean is a random variable



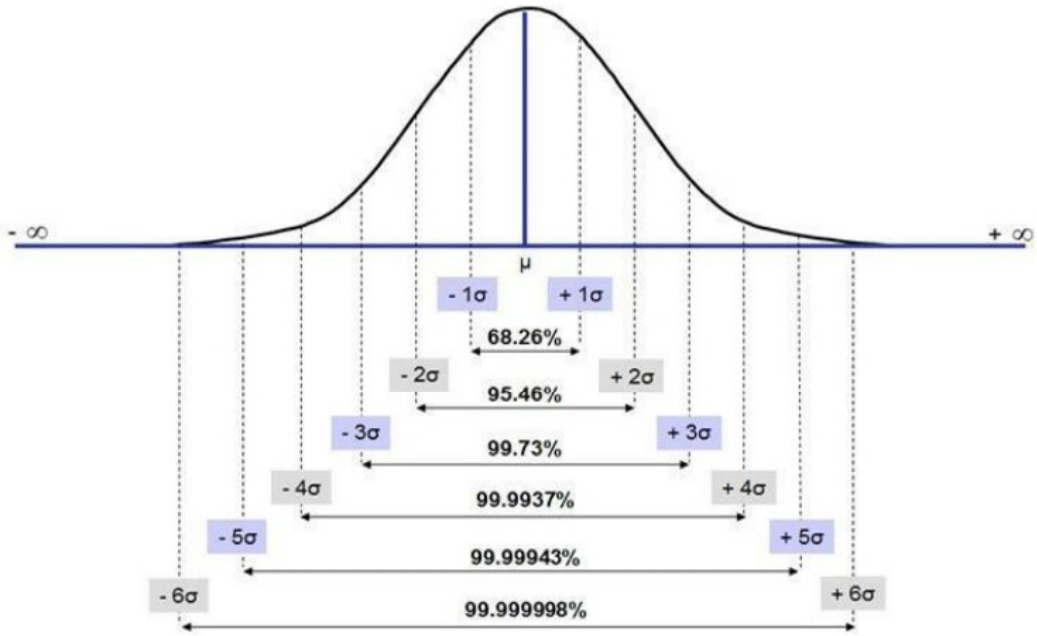


Normal Distribution

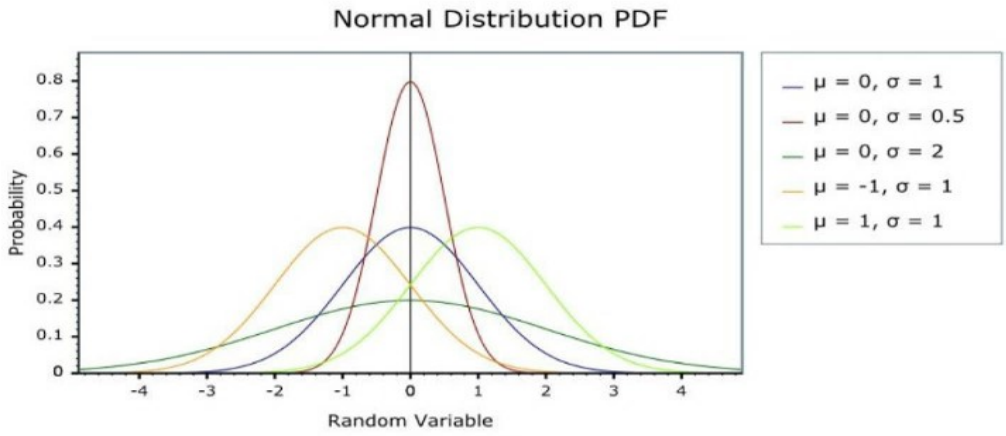
- The normal random variable takes values from -∞ to +∞
- The mean, mode and median are allequal.
- The curve is symmetric at the center (i.e. around the mean, μ).
- Exactly half of the values are to the left of center and exactly half the values are to the right.
- The Probability associated with any single value of a random variable is alwayszero
- The area under the entire curve is always equal to1.

Properties:

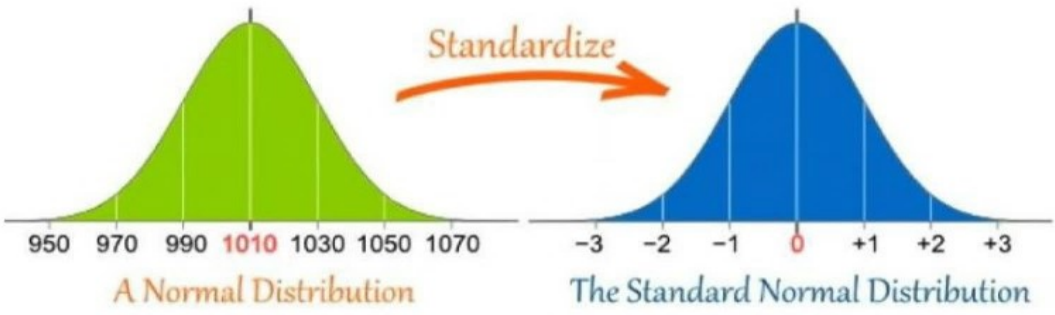
- 68.26% of values lie within ±1 σfrom themean
- 95.46% of the values lie within ±2σ from the mean
- 99.73% of the values lie within ± 3σfrom themean



Characterized by mean, μ, and standard deviation, σ



Standard Normal Distribution and Z-score



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

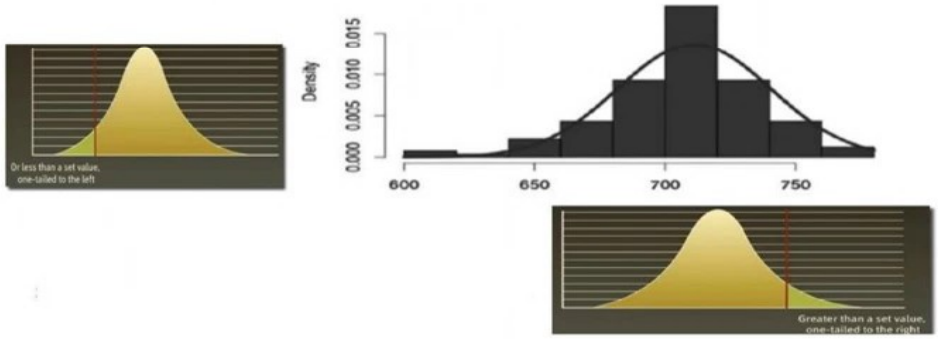
Why do we standardize?

Calculating Probability from Z distribution

Suppose GMAT scores can be reasonably modelled using a normal distribution

- μ= 711
- σ = 29

What is p(x ≤680)?



Step1: Calculate Z score corresponding to 680

z = (680-711)/29 = -1.06

Step2: Calculate the probabilities using Z - Tables

$P(z \leq -1 = 0.14$

Confidence Interval:

- What is the probability of tomorrow's temperature being 42 degrees?
- Can it be between [-50°*C* & 100 °*C*]?

Central Limit Theorem

The Distribution of the sample mean

-will be normal when the distribution of data in the population is normal

-will be approximately normal even if the distribution of data in the population is not normal if the "sample size" is fairly large

Mean $(\bar{X})=\mu$ (the same as the population mean of the raw data)

Standard Deviation $(X) = \frac{\sigma}{\sqrt{n}}$,

where,

σ is the population standard deviation

n is the sample size

-This is referred to as standard error of the mean

The standard error of me estimates the variability between samples whereas the standard deviation measures the variability within a single sample.

Case Study: Confidence Interval

- A University with 100,000 alumni is thinking of offering a new affinity credit card to its alumni.
- Profitability of the card depends on the average

balance maintained by the cardholders.

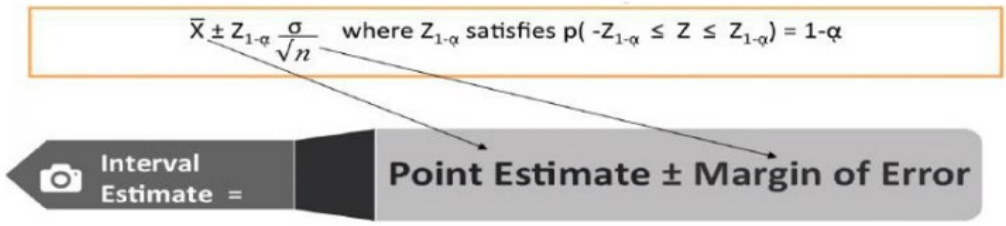
- A Market research campaign is launched, in which about 140 alumni accept the card in a pilot launch.
- Average balance maintained by these is $1990 and the standard deviation is $2833. Assume that the population standard deviation is $2500 from previous launches.
- What can we say about the average balance that will be held after a full-fledged market launch?

INTERVAL ESTIMATES OF PARAMETERS

- Based on sample data
  -The point estimate for mean balance = $1990
  -Can we trust this estimate?
- What do you think will happen if we took another random sample of 140 alumni?
- Because of this uncertainty, we prefer to provide the estimate as an interval (range) and associate a level of confidence with it.

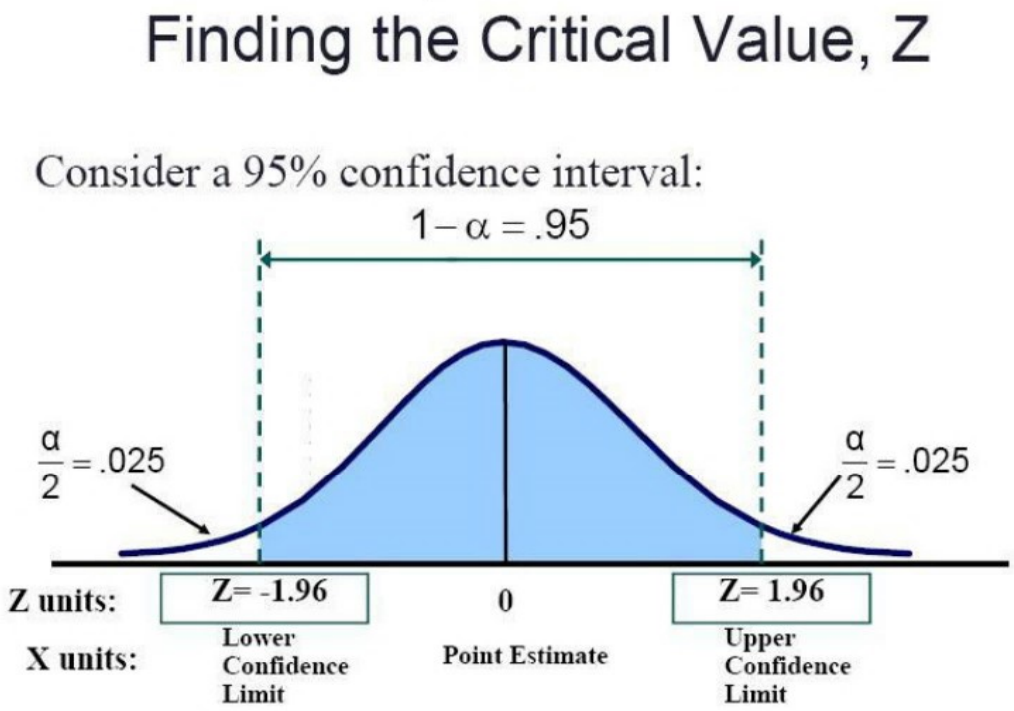
CONFIDENCE INTERVAL FOR THE POPULATION MEAN

Start by choosing a confidence level (1-cx) %
(e.g. 95%,99%,90%).
Then, the population mean will be within



Z-DISTRIBUTION

Calculate Z value – 90%, 95% & 99%



CONFIDENCE INTERVAL INTERPRETATION

Consider the 95% Confidence interval for the mean income: *[$1576, $2404]* Does this mean that

- The mean balance of the population lies in the range?
- The mean balance is in this range 95% of the time?
- 95% of the alumni have balance in this range?

Interpretation 1: Mean of the population has a 95% chance of being in this range for a random sample

Interpretation 2: Mean of the population will be in this range for 95% of the random samples

T-DISTRIBUTION

- Suppose, that the alumni of this university are very different and hence population standard deviation from previous launches cannot be used
- We replace a with our best guess (point estimate) s, which is the standard deviation of the sample:

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

Calculate, $t = \frac{\bar{x}-\mu}{s/\sqrt{n}}$

- If the underlying population is normally distributed, T is a random variable distributed according to a t-distribution with n-1degrees of freedom T_{n-1}
- Research has shown that the t-distribution is fairly robust to the deviation of the population of the normal model.

Confidence Interval for mean with unknown Sigma

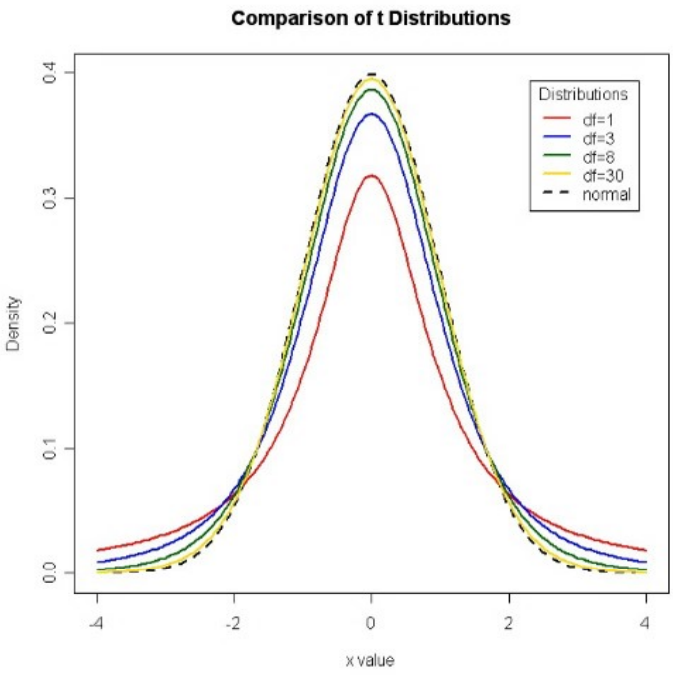
$$\bar{X} \pm t_{1-\alpha,n-1}\frac{s}{\sqrt{n}}$$

where, $t_{1-\alpha,n-1}\ satisfies\ P(-t_{1-\alpha,n-1} \leq T_{n-1} \leq t_{1-\alpha,n-1}) = 1-\alpha$

Confidence Interval for mean with known Sigma

$$\bar{X} \pm Z_{1-\alpha}\frac{\sigma}{\sqrt{n}}$$

where, $Z_{1-\alpha}\ satisfies\ P(-Z_{1-\alpha} \leq Z \leq Z_{1-\alpha}) = 1-\alpha$



Poisson Distribution

Suppose you work at a call center, approximately how many calls do you get in a day? It can be any number.

Now, the entire number of calls at a call centre in a day is modelled by Poisson distribution.

Some more examples are,

1. The number of emergency calls recorded at a hospital in aday.
2. The number of thefts reported in an area on aday.
3. The number of customers arriving at a salon in anhour.
4. The number of suicides reported in a particularcity.
5. The number of printing errors at each page of thebook.

You can now think of many examples following the same course. Poisson Distribution is applicable in situations where events occur at random points of time and space wherein our interest lies only in the number of occurrences of the event.

A distributionis called **Poisson distribution** when the following assumptions arevalid:

1. Any successful event should not influence the outcome of another successful event.
2. The probability of success over a short interval must equal the probability of success over a longer interval.
3. The probability of success in an interval approaches zero as the interval becomes smaller.

Now, if any distribution validates the above assumptions then it is a Poisson distribution.

Some notations used in Poisson distribution are:

- λ is the rate at which an event occurs
- t is the length of a time interval
- And X is the number of events in that time interval

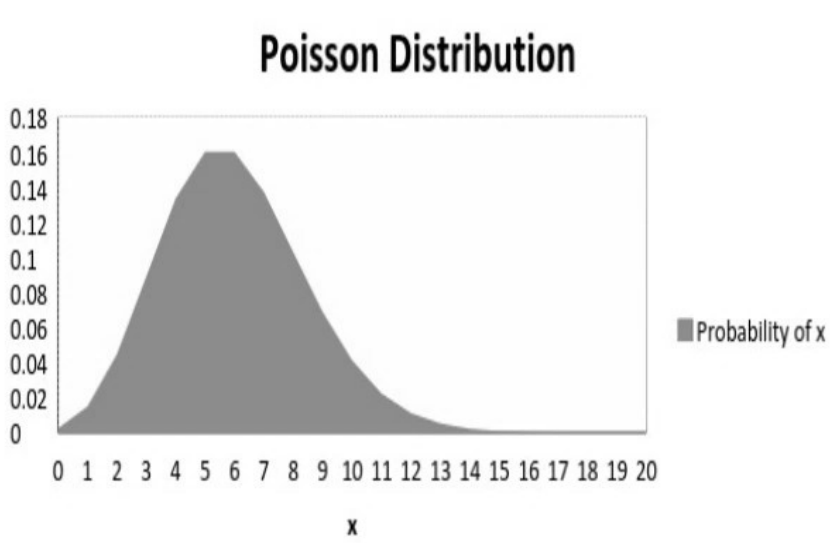
Here, X is called a Poisson Random Variable and the probability distribution of X is called Poisson distribution.

Let μ denote the mean number of events in an interval of length t.

Then, **μ = λ*t.**

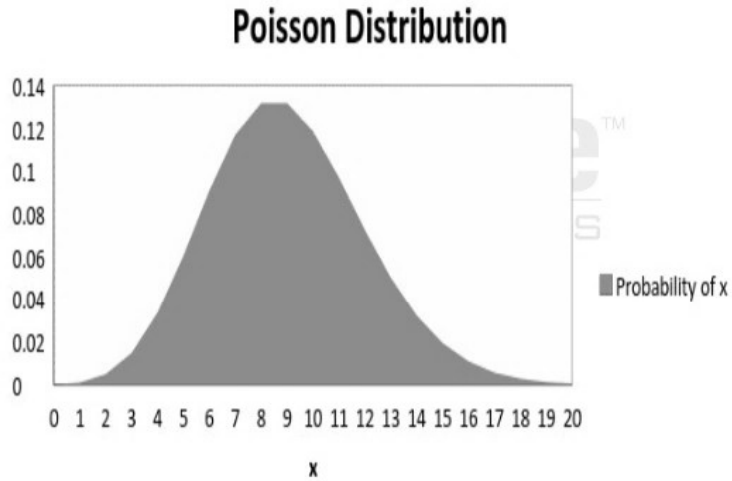
The PMF of X following a Poisson distribution is given by:

$$P(X = x) = e^{-\mu}\frac{\mu^x}{x!} \qquad for\ x = 0, 1, 2, \ldots\ldots$$



The mean μ is the parameter of this distribution. μ is also defined as the λ times length of that interval.

The graph of a Poisson distribution is shown above.
The graph shown below illustrates the shift in the curve due to increase in the mean.



It is perceptible that as the mean increases, the curve shifts to the right
The mean and variance of X following a Poisson distribution:

Mean → E(X) = μ
Variance → Var(X) = μ

Logarithmic distribution

The **logarithmic distribution** (also known as the **logarithmic series distribution** or the **log-series distribution**) is a discrete probability distribution derived from the Maclaurin series expansion,

$$-\ln(1-p) = p + \frac{p^2}{2} + \frac{p^2}{3} + \ldots,$$

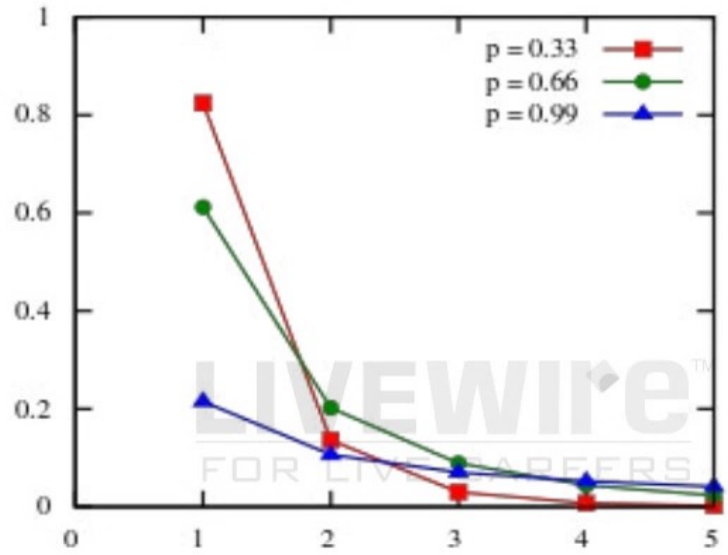
From this, we obtain the identity,

$$\sum_{k=1}^{\infty} \frac{-1}{\ln(1-p)} \frac{p^k}{k} = 1$$

This leads directly to the probability mass function of a Log(p)- distributed randomvariable:

$$f(k) = \frac{-1}{ln(1-p)} \frac{p^k}{k}$$

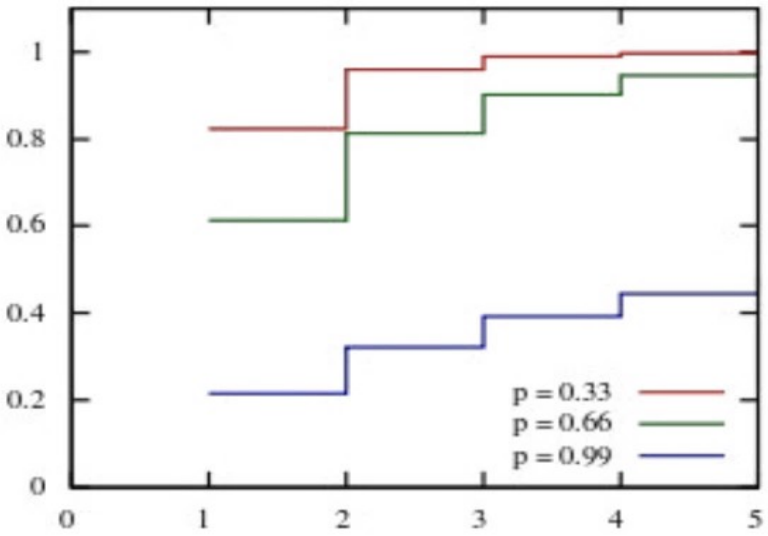
for $k \geq 1$, and where $0 < p < 1$.



The cumulative distribution function is,

$$f(k) = 1 + \frac{B(p; k+1, 0)}{\ln(1-p)}$$

where, B is the incomplete beta function.



Binomial Distribution

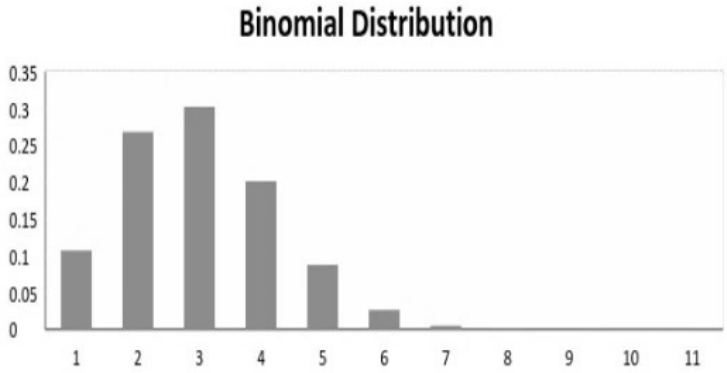
- A distribution where only two outcomes are possible, such as success or failure, gain or loss, win or lose and where the probability of success and failure is same for all the trials is called a Binomial Distribution.
- If the probability of success in an experiment is 0.2 then the probability of failure can be easily computed as $q = 1 – 0.2 = 0.8$.
- Each trial is independent since the outcome of the previous toss doesn't determine or affect the outcome of the current toss. An experiment with only two possible outcomes repeated n number of times is called binomial. The parameters of a binomial distribution are n and p where n is the total number of trials and p is the probability of success in each trial.

On the basis of the above explanation, the properties of a Binomial Distribution are,

1. Each trial isindependent.
2. Thereareonlytwopossibleoutcomesinatrial-either a, success or failure.
3. A total number of n identical trials areconducted.
4. The probability of success and failure is same for all trials. (Trials areidentical.)

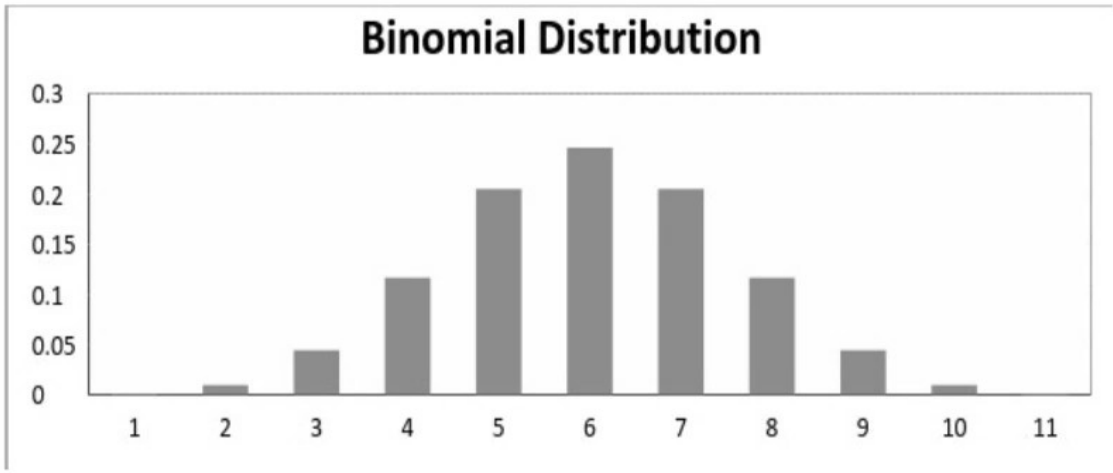
The mathematicalrepresentation of binomial distribution is given by:

$$P(x) = \frac{n!}{(n-x)!\,x!} p^x q^{n-x}$$



A binomial distribution graph where the probability of success does not equal the probability of failure looks like

Now, when ***probability of success = probability of failure***,
in such a situation the graph of binomial distribution looks like



The mean and variance of a binomial distribution are given by:

Mean → $\mu = n*p$
Variance → $Var(X) = n*p*q$

Chi-Square Test

There are **two types of chi-square tests**. Both use the chi-square statistic and distribution for different purposes:

- A chi-square goodness of fit test determines if a sample data matches a population. For more details on this type, see Goodness of Fit Test.

- A chi-square test for independence compares two variables in a contingency table to see if they are related. In a more general sense, it tests to see whether distributions of categorical variables differ from each another.
- A very small chi-square test statistic means that your observed data fits your expected data extremely well. In other words, there is a relationship.
- A very large chi-square test statistic means that the data does not fit very well. In other words, there isn't a relationship.

Chi-Square Statistic

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

The subscript "c" is the degrees of freedom. "O" is your observed value and E is your expected value.

The Chi-Squared Distribution

Let's say you have a random sample taken from a normal distribution.

The chi-square distribution is the distribution of the sum of the random samples **squared.** The degrees **of freedom (k)** is equal to the number of samples being summed.

The chi-squared distribution has many uses in statistics,including:

- Confidence intervalestimation for a population standard deviation of a normal distribution from a sample standard deviation.
- Independence of two criteria of classification of qualitative variables.
- Relationships between categorical variables (contingency tables).
- Sample variance study when the underlying distribution is normal.
- Tests of deviations of differences between expected and observed
- frequencies (one-way tables).

- The chi-square test (a goodness of fit test).

Case Study

256 visual artists were surveyed to find out their zodiac sign. The results were: Aries (29), Taurus (24), Gemini (22), Cancer (19), Leo (21), Virgo (18), Libra (19), Scorpio (20), Sagittarius (23), Capricorn (18), Aquarius (20), and Pisces (23). Test the hypothesis that zodiac signs are evenly distributed across visual artists.

- *Null Hypothesis* = *Ho*: zodiac signs are evenly distributed
- *Alternate Hypothesis* = *Ha:* zodiac signs are not evenly distributed

Make a table with columns for *"Categories," "Observed," "Expected," "Residual (Obs-Exp)", "(Obs-Exp)^2" and "Component (Obs-Exp)^2 / Exp."*

PYTHON

Python is an object-oriented, interpreted language which was developed by **Guido van Rossum** in 1991 at the National Research Institute for Mathematics and Computer Science, in the Netherlands. Python is an *easy to learn and write* program, which makes it attractive for an Application Developer. It has a design environment which emphasises code readability and a syntax which lets programmers to express concepts in fewer lines, when compared to other languages.

Python is an open source and platform independent software, which allows you to run on a wide variety of systems.

Python's *syntax* and *type casting* together with its interpreted nature, makes it an ideal language for scripting and application development in many areas.

It is well-known as a *multi-paradigm programming language* because it can be used with the web, enterprise and web services, mobile application, big data, cloud etc.

Python makes the development and debugging *fast*. Since there is no compilation process, edit-test-debug cycle is incredibly fast.

Features of Python

- ***Easy to learn and read*** – Python has a simple structure and keywords that allow students to pick up the language quickly.
- ***Portable and Cross platform compatible*** – Python has the ability to run on a wide variety of hardware platforms and maintain the same interface on all platforms.
- ***Interactive mode*** – Python supports an interactive mode that allows interactive testing and debugging of snippets of code.
- ***GUI Programming*** -- Python supports GUI applications; libraries for major operating systems.
- ***Scalable*** – Python offers a better structure and support for large programs, rather than shell scripting.
- ***Database Support*** --Python provides interfaces to all major commercial databases.

Python Interpreter and its Environment

Invoking the Interpreter

On Windows machines, the Python installation is usually placed in **C:\Python36**, though you can change his when you are running the installer.

Typing an end-of-file character (Control-Z on Windows) at the primary prompt, that causes the interpreter to exit with a zero exit status. If that does not work, you can exit the interpreter by typing the following command: **quit().**

The interpreter's line-editing features include interactive editing, history substitution and code completion on systems that support *read line.* Perhaps the quickest way to check and see whether the command line editing option is supported, is by typing Control-P to the first Python prompt you get. If it beeps, you have command line editing.

The interpreter operates somewhat like the Unix shell. When called with standard input connected to a tty device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a *script* from that file.

A second way of starting the interpreter is python-command[arg]..., which executes the statement(s) in *command*, analogous to the shell's-c option. Since Python statements often contain spaces or other characters that are special to the shell, it is usually advised to quote *command* in its entirety with single quotes.

Some Python modules are also useful as scripts. These can be invoked using python-module[arg]..., which executes the source file for the *module* as if you had spelled out its full name on the command line.

When a script file is used, it is sometimes useful to be able to run the script and enter interactive mode afterwards. This can be done by passing -i before the script.

Argument Passing

The Python sys module provides access to any command-line arguments via the sys.argv. This serves two purposes.

- **sys.argv** is the list of command-line arguments.
- **len(sys.argv)** is the number of command-line arguments.

Here **sys.argv[0]** represents the program name i.e. Script name.

Interactive Mode

Interactive mode is a command line shell which gives immediate feedback for each statement, while
running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole. In this mode, it prompts for the next command with the *primary prompt*, usually three greater-than signs (>>>); for continuation lines, it prompts with the *secondary prompt*, by default three dots (...). The interpreter prints a welcome message stating its version number and a copyright notice before printing the first prompt:

```
$Python 3.7.2 (v3.5.2:4def2a2901a5, Jan 25 2017, 22:01:18)
[MSC v.1900 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

Continuation lines are needed when entering a multi-line construct. As an example, take a look at this if statement:

```
>>>the_world_is_flat=True
>>>ifthe_world_is_flat:
... print("Be careful not to fall off!")
...
Be careful not to fall off!
```

Environment

Source Code Encoding

By default, Python source files are treated as encoded in UTF-8. In that encoding, characters of most languages in the world can be used simultaneously in string literals, identifiers and comments — although the standard library only uses ASCII characters for identifiers, a convention that any portable code should follow. To display all these characters properly, your editor must recognize that the file is a UTF-8, and it must use a font that supports all the characters in the file.

To declare an encoding other than the default one, a special comment line should be added as the *first* line of the file. The syntax is as follows:

 # -*- coding: encoding -*-

Where *encoding* is one of the valid codecs supported by Python.

For example, to declare that Windows-1252 encoding is to be used, the first line of your source code file should be:

 # -*- coding: cp-1252 -*-

Python installation and environment setup

1. Python installation

Step-1.Double click on python .exe file and select Install now.

Step-2.Select the check box option “Install Launcher for all users “.

If you want the path to be added automatically to system variables, select the check box option “Add Python to Path”.

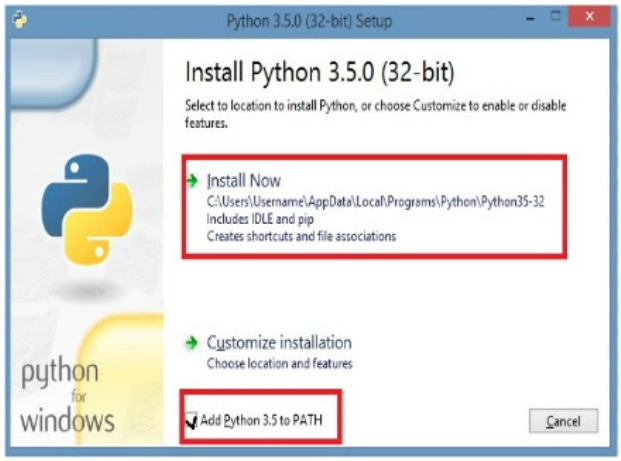


Figure-1-Python Installation first page.

Step-3.If you want to set the global path manually, go to system properties, click on Environment Variables, Select Path Variable and Click on Edit.

Step-4. Browse to your Python Installation directory, copy the path and paste it in Variable value in Environment Variables.

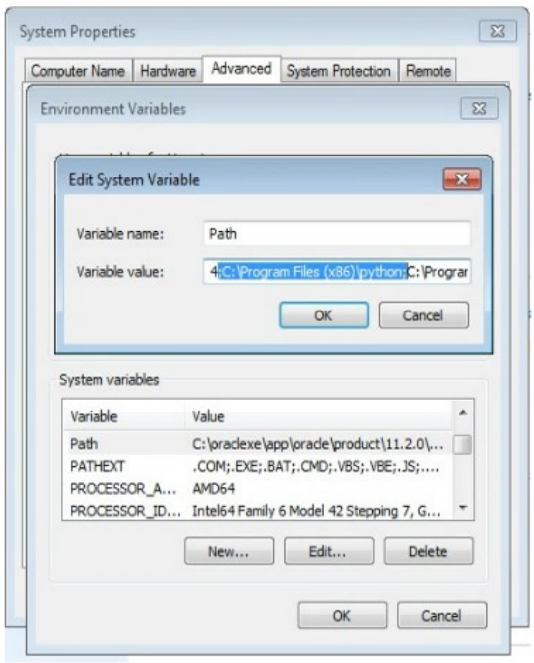


Figure-2-Python environment variable setting.

Now, you have completed your installation and path settings for Python.

Step 5. To open Python, go to the Start menu and select Python.

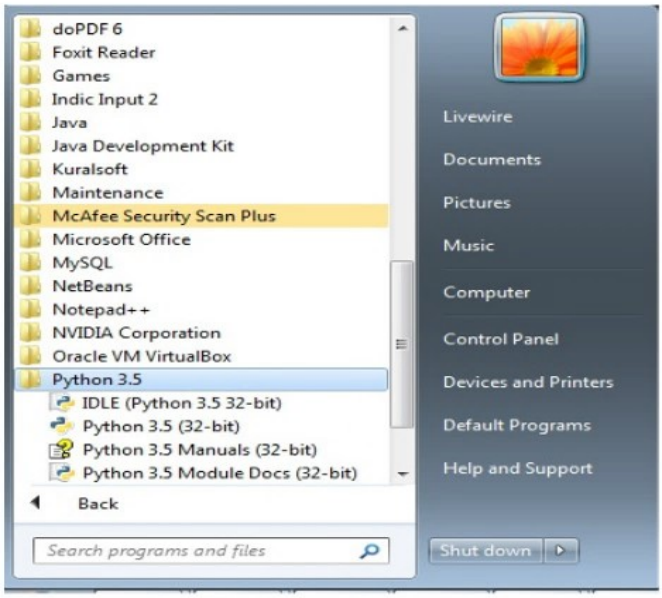


Figure-3-View of python on start menu.

Python can work in two modes:

Step 1. Script Mode (IDLE (Python 3.5 32-bit)).

Step 2. Shell mode (Python 3.5(32-bit).

#Script Mode (IDLE (Python 3.5 32-bit))

The scriptural mode allows you to write a **script** and then run. IDLE is very similar to a text editor. A script usually contains a sequence of statements. If there is more than one statement, the results appear one at a time as the statements are executed.

To use the Python Shell mode. Select IDLE(Python 3.5 32-bit) mode.

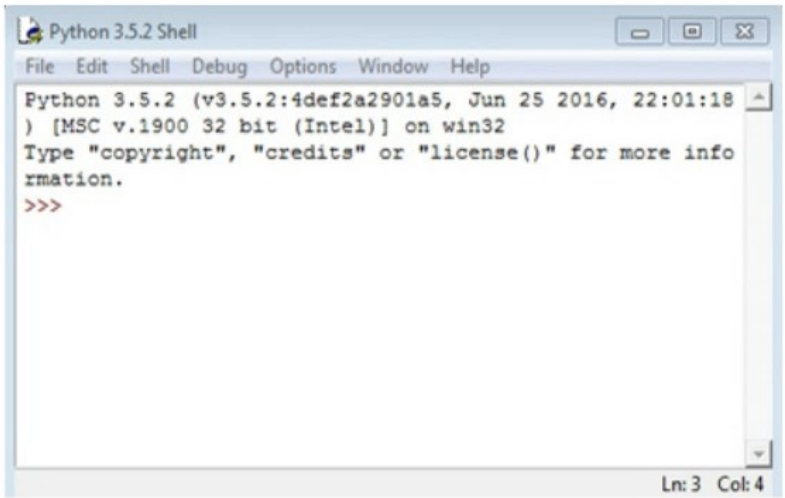


Figure-4-Python IDLE mode

Shell mode (Python 3.5(32-bit).

Shell **mode** immediately returns the results of commands you enter in the shell.

You can start Python from Unix, DOS, or any other system that provides you a command-line interpreter or shell window.
To use the Python Shell mode , Select Python 3.5 32-bit mode.

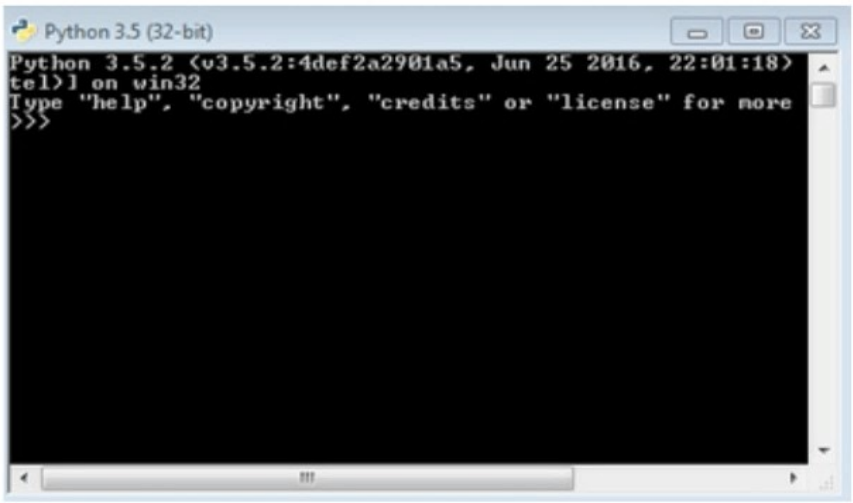


Figure-5-Python command mode

Python Program in Script Mode

Step 1.Open IDLE(Python 3.5 32-bit). Select File->New File

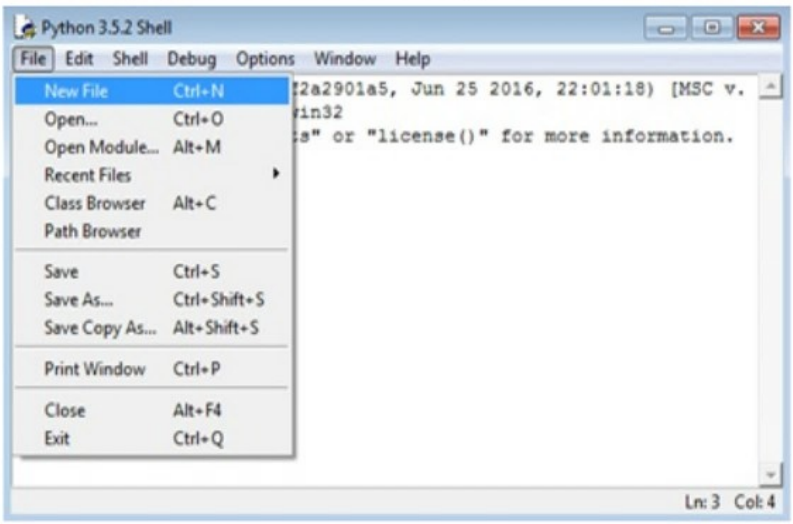


Figure-6-Creating new file on IDLE Mode.

Step 2. Type the Python code in the new window and save it.

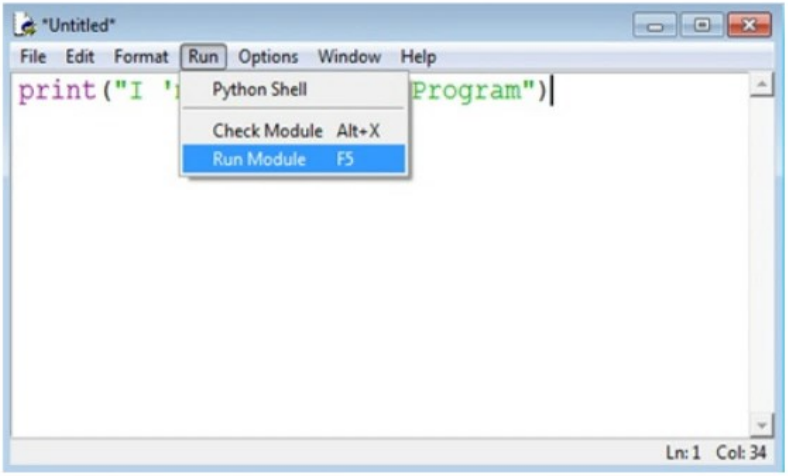


Figure-7-Running the program

Step 3. Select Run menu option and select Run Module or press F5.

Step 4. The output will be displayed as shown below.

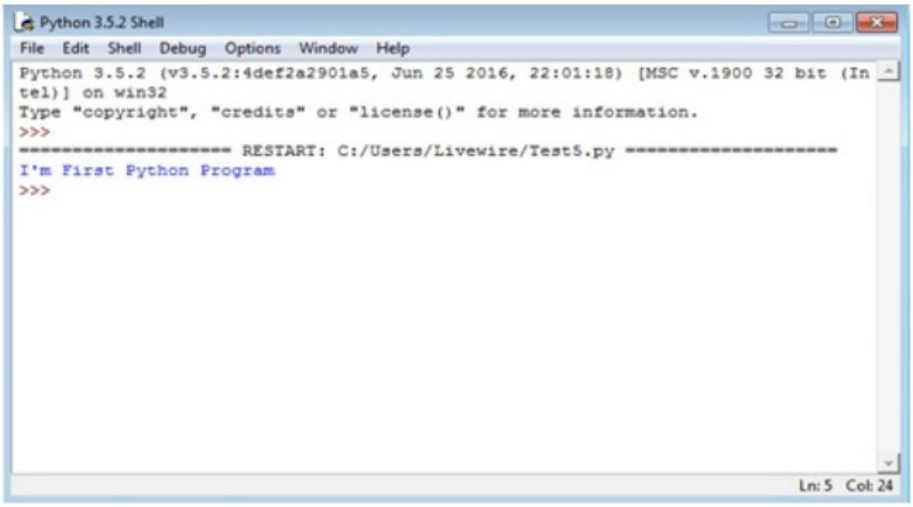


Figure-8-Output of your program.

Python Program in Shell Mode or Interactive mode

1. Open Python 3.5.2 32-bit.
2. In the Shell window, type the commands directly and press Enter.
3. The output will be displayed as shown below.

Example:

Here, a simple message is displayed in the shell mode by using ***print statement.***

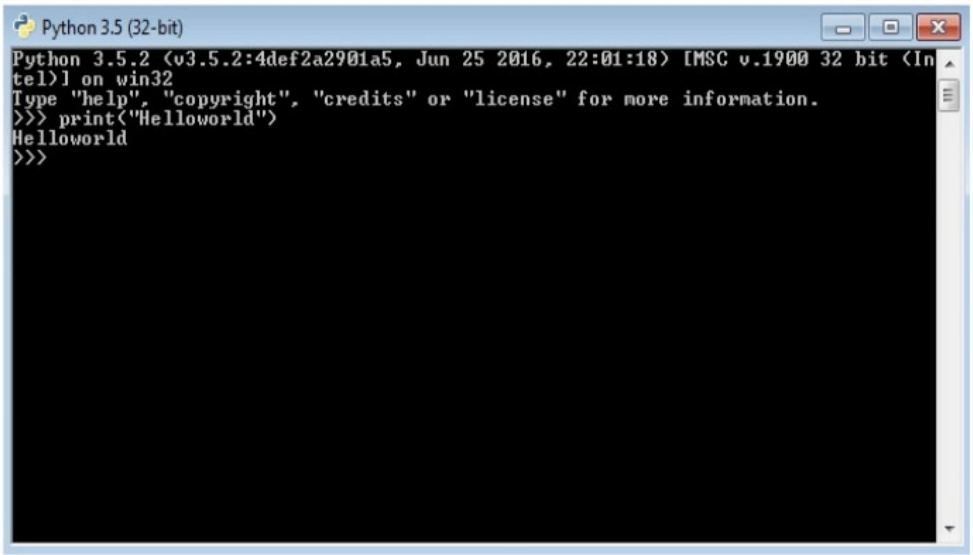


Figure-9-Hello world program in shell mode.

Python as a Calculator

The Python programming language is a great tool to use when working with numbers and evaluating mathematical expressions.

Numbers

The interpreter takes up the role of the calculator. You can type an expression and it will write the value. Expression syntax is straightforward. The operators **+, -,** * and / work just like in most other languages; parentheses **()** can be used for grouping.

Example:

```
>>>2+2
>>>50-5*6
>>>(50-5*6)
>>>8/5
```

The integer numbers have type *int*; the ones with a fractional part have type *float.*

Division **(/)** always returns a *float*. To do *floor division* and get an integer result, you can use the // operator; to calculate the remainder you can use **%:**

```
>>>17/3
5.6666666666
>>>17//3
5
>>>17%3
2
>>>5*3+2
```

With Python, it is possible to use the ** operator to calculate powers:

```
>>>5**2
>>>2**7
```

The equal sign **(=)** is used to assign a value to a variable. Afterwards, no result is displayed before the next interactive prompt:

```
>>>width=20
>>>height=5*9
>>>width*height
```

```
900
```

If a variable is not “defined”, trying to use it will give you an error:

```
>>>n# try to access an undefined variable
NameError: name 'n' is not defined
```

There is full support for floating point; operators with *mixed* type operands convert the integer operand to floating point:

```
>>>3*3.75 /1.5
7.5
>>>7.0/2
3.5
```

In interactive mode, the last printed expression is assigned to the variable _. This means that when you are using Python as a desk calculator, it is somewhat easier to continue calculations.

Example:

```
>>>t=12.5/100
>>>p =100.50
>>>p * t
12.56
>>>p + _
113.0625
>>>round(_,2)
```


This variable should be treated as read-only by the user. Do not explicitly assign a value to it — you would create an independent local variable with the same name, masking the built-in variable with its magic behaviour.

Strings

Python can also manipulate strings, which can be expressed in several ways. They can be enclosed in single quotes **('...')** or double quotes **("...")** with same result. \ can be used to escape quotes:

```
>>>’Hello livewire’  #single quotes
‘Hello livewire’
>>>’Hello livewire\’ # use \ to escape the single quotes
”Hello livewire”
```

```
>>>”livewire” #... or use double quotes instead
“livewire”
>>>’”livewire”,careers’
‘ “livewire”,careers’
```

In the interactive interpreter, the output string is enclosed in quotes and special characters are escaped with backslashes. While this might sometimes look different from the input, the two strings are equivalent. The string is enclosed in double quotes, if the string contains a single quote and no double quotes, otherwise it is enclosed in single quotes. The ***print()*** function produces a more readable output, by omitting the enclosing quotes and by printing escaped and special characters:

```
>>>’"Isn\'t," she said.'
’"Isn\'t," she said.'
>>>print('"Isn\'t," she said.')
“Isn’t, “ she said.
>>>s='First line.\nSecond line.' # \n means newline
>>>s
‘First line.\nSecond line.’
>>>print(s)
First line.
Second line.
```


If you do not want characters prefaced by **** to be interpreted as special characters, you can use *raw strings* by adding an **r** before the first quote:

```
>>>print(‘C:\live\n wire’) # \n means newline
C:\live
wire
>>>print(r’C:\live\wire’) # r means raw string
C:\live\wire
```

Strings can be concatenated with the **+** operator, and repeated with ***:**

```
>>>3 * ‘ab’ + ‘xyz’
abababxyz
```

Two or more *string literals* next to each other are automatically concatenated.

```
>>>'live' 'wire'
'Python'
```

This only works with two literals though, not with variables or expressions:

```
>>>P = 'live'
>>>P 'wire' #can't concatenate a variable and a string literal
...
Syntax Error: invalid syntax
```

If you want to concatenate variables or a variable and a literal, use **+**:

```
>>>P + 'wire'
livewire
```

Lists

Python knows a number of *compound* data types, used to group together other values. The most versatile is the *list*, which can be written as a list of comma-separated values between square brackets. Lists might contain items of different types, but usually the items all have the same type.

```
>>>squares=[1,4,9,16,25]
>>>squares
[1,4,9,16,25]
```

First Step towards Programming

We can use Python for more complicated tasks than adding two and two together. For instance, we can write an initial sub-sequence of the *Fibonacci* series, as follows:

```
>>>a,b=0,1
>>>while b<10:
    Print(b)
    A,b=b,a+b
>>>
0….9
```

This example introduces several new features.

The first line contains a *multiple assignment*: the variables ***a*** *and* ***b*** simultaneously get the new values 0 and 1. On the last line this is used again, demonstrating that the expressions on the right-hand side are all evaluated first, before any of the assignments take place. The right-hand side expressions are evaluated from the left to the right.

The *body* of the loop is *indented*: indentation is Python's way of grouping statements. At the interactive prompt, you have to type a tab or space for each indented line. In practice, you will prepare more complicated input for Python with a text editor. All decent text editors have an auto-indent facility. When a compound statement is entered interactively, it must be followed by a blank line to indicate completion. Note that each line within a basic block must be indented by the same amount.

The print() function writes the value of the argument it is given. It differs from just writing the expression you want to write in the way it handles multiple arguments, floating point quantities, and strings. Strings are printed without quotes, and a space is inserted between items, so you can format things nicely, like this:

```
>>>i=256*256
>>>print(i)
65536
```

The keyword argument *end* can be used to avoid the newline after the output, or end the output to a different string:

```
>>>a,b=0,1
>>>while b<10:
    Print(b,end=',')
    a,b=b,a+b
>>>
1,1
1,2
2,3
3,5
5,8
8,13
```

Keywords

Keywords are predefined words .The user can not specify the keyword as a variable name, function name or any other identifier for the program. They are defining the syntax and structure of the keywords of the Python language. These keywords are case sensitive. In Python totally 33 keywords are available.

Except True, False and None, which are in lowercase and the other must be written as follows.

Keywords in Python				
False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
And	del	global	not	with
As	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Identifiers

The Identifier is the name given to a class, a function, a variable etc. It helps differentiating one entity from another.

Rules for identifiers

- Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or numbers (0 to 9) or an underscore (_).
- An identifier cannot start with a digit.
- Keywords cannot be used as identifiers.

Statements

A simple statement is comprised within a single logical line. Several simple statements may occur on a single line separated by semicolons. The example for simple statement is:

```
x=1+2+3+ \
4+5+6+ \
7+8+9
```

The end of a statement is marked by a newline character **(\)**.
The line extension is implied inside parentheses ( ), brackets [ ] , braces { } and semicolon.

```
// Parentheses
x =(1+2+3+
4+5+6+
7+8+9)
// Brackets
monts=['January',
'February',
'March']

// Semicolon
a =1; b =2; c =3
```

Indentation

Indentation describes the block of code. In some programming languages, users have to write within the curly braces {}, but in Python, there is no syntax like this. In Python, the user can use the **indentation (space)** for separating a block of codes.
A block of code like the body of a function or loop, etc. starts with indentation and ends with the first un indented line. The line of indentation is up to the user, but it must be constant throughout that block.
Generally four **whitespace**s are used for indentation and is preferred over tabs.

Example: Test.py

```
for i in range(10,20):
    print(i)      #This line started after four whitespaces
if i == 15:
    break      # This line started after four whitespaces
```

Output

10,11,12,13,14

Comments

It describes what's going on inside a program so that a person looking at the source code does not have a hard time figuring it out.

Single line comments

Python uses the hash symbol (#) to start writing a single line comment.

```
#This is a single line comment in python
print('Hello Python world')
```

Multiline comments

Multiline comments doing this, is to use triple quotes, either '''or """". These triple quotes are generally used for multi-line strings comments.

```
"""This is also aExampleofMulti-line comments in python.
 If you are using python on enterprisesmode,
it is very high level support for transaction processing"""
```

Docstring

Docstring is a short form of documentation strings. Triple quotes are used while writing docstrings in python program.

Example: Test.py

```
Defmsg(name):
    """This is docstring demo"""

    print("Hi, " + name + ". Have a nice day!")

print(msg.__doc__)
print(msg('James'))
```

Output:

"""This is docstring demo"""

Hi James Have a nice day!

A Docstring message can be read by the attribute of __doc__ the function.

```
print(msg.__doc__)
```

Data Types Model in Python

Even though everything is an object in Python programming, data types are actually classes and variables are object of these classes or instances of the classes.

Python has the following standard data type models:

- Numbers
- String
- List
- Tuple
- Set
- Dictionary

Numbers

Integers, real numbers and complex numbers come under the numbers category. They are defined as *int, float* and *complex* class in Python.

Python can use the type() function to know which type a variable or a value belongs to, and the is instance() function to check whether the object belongs to a particular class.

```
>>>a = 50
>>>print(a, "is of type", type(a))
50 is of type <class 'int'>

>>>a = 20.0
>>>print(a, "is of type", type(a))
50 is of type <class 'float'>

>>>a = 15+25j
>>>print(a, "is complex number?",
isinstance(1+2j,complex))
15+25j is complex number? true
```

String

String is a sequence of Unicode characters. The user can use single quotes or double quotes to represent the strings. Multi-line strings like multiline comment can be denoted using triple quotes, ''' or """. Strings are **immutable** object in Python.

```
>>>s = 'Hello Livewire world!'

>>>print("s[4] = ", s[4])          # Hello

>>>print("s[6:11] = ", s[6:11])    # Live
```

List

The List is stored as an **ordered sequence** of items, these items are **mutable**. It is one of the most frequently used data types in Python and is very flexible. All the items in a list, need not be the same data type.

Declaring a list is very easy and straight forward. Items separated by commas are enclosed within **square brackets [ ].**The square bracket is representing the list. There is no need for any other classes or interfaces for list operation. The index values are starting at 0.

```
>>>a = [5, 10, 15, 20, 25, 30, 35, 40]

>>>print("a[2] = ", a[2])              #Output  15

>>>print("a[0:3] = ", a[0:3])        #Output 5,10,15,20

>>>print("a[5:] = ", a[5:])           #Output 30
```


Tuple

A Tuple is an **ordered sequence** of items, same as the list. The main difference is that tuples are i**mmutable**. Tuples once created cannot be modified.

Tuples are protecting the data and are usually faster than list, as it cannot change dynamically.

It is defined with **parentheses ()** where items are separated by commas. The index values are starting at 0.

```
>>>t = (5,'livewire', 1+3j)

>>>print("t[1] = ", t[1])#Output 5

>>>print("t[0:3] = ", t[0:3])# Output 5,'livewire', 1+3j
```

t[0] = 10// Re-initialization is not supported because this is an immutable object.

Set

The Set is an **unordered collection** of unique elements. A Set is defined by values separated by commas inside **curly braces {}.**

```
>>>a = {10, 2, 30, 400, 5}

>>>print("a = ", a)          # Output : 10,2,30,400,5

>>>print(type(a))
<class set>
```

Dictionary

The Dictionary is an **unordered collection** of key-value pairs. Generally python can use a huge amount of data. Dictionaries are optimized for retrieving data. The user must know the key to retrieve the value. The declaration of dictionary without any item, is written with just two **curly braces {}**.

Dictionaries are defined within curly braces {} with each item being a pair in the form key: value. The Key and Value can be of any data type.


```
>>>dict = {'Name': 'Peter', 'Age': 7}

>>>print(type(dict))
<classdict>

>>>print("dict[1] = ", dict[1]);                    #Output
Name:Peter

>>>print("dict['key'] = ", dict[key]);#Error

>>>print("dict[2] = ", dict[2]);          #Error
```

Properties of Dictionary

There are two important points to remember about keys:

(1) Should not allow duplicate keys.

(2) Keys are an immutable object.

Conversion between the data types

The below data types are working under the interpreter mode, or by print function scripting mode, and can get the output.

```
>>>float(10) # 10 will be converted to float
>>>int(10.6)# 10.6 will be converted to int
>>>str(50)# 50 will be converted to string
>>>set([1,2,3])# 1,2,3 will be converted to set
>>>tuple({5,6,7}) #5,6,7 will be converted to tuple
>>>list('hello')#hello will be converted to list
>>>dict([[1,2],[3,4]]) #values will be converted to dictionary value
```

Variables

A variable is a representation of the location in memory used to store the value of a given variable.

Python does not need to declare a variable before using it. In Python, one can directly assign a value to a variable and it will exist. Python does not have to declare any type of the variable. This is handled internally according to the type of value assigned to the variable.

```
>>>Name="Peter"        // Name act as a string type
>>>Age=10              // Age act as an integer type
```


Operators

Operators are special symbols that carry out arithmetic or logical or any other computation. Python can construct which symbol can be used to manipulate the value of the operands.

List of Operators
Arithmetic operators
Relational operators
Logical operators
Bitwise operators
Assignment operators
Special operators

Arithmetic Operators

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication etc.

```
>>>x = 100
>>>y = 50

# Addition
>>>print('x + y =',x+y)

# Subtraction
>>>print('x - y =',x-y)

# Multiplication
>>>print('x * y =',x*y)

# Division
>>>print('x / y =',x/y)

# Floor Division—just opposite of % operator
>>>print('x // y =',x//y)

# Exponent form
>>>print('x ** y =',x**y)
```


Comparison Operators

Comparison operators are used to compare the values. It either returns True or False according to the condition. These are also known as relational operators.

```
>>>x = 10
>>>y = 12

>>>print('x > y  is',x>y)# Output: x > y is False
>>>print('x < y  is',x<y)
# Output: x < y is True
>>>print('x == y is',x==y)
# Output: x == y is False

>>>print('x != y is',x!=y)
# Output: x != y is True
```

Logical Operators

The logical operators for AND, OR and NOT are used to combine simple relational statements into more complex expressions

```
>>>a = True
>>>b = False
>>>print('a and b is', a and b)          # Output: a and b is
False

>>>print('a or b is', a or b)            # Output: a or b is True

>>>print('not a is', not a)       # Output: not a is False
```

Bitwise operators

Bitwise operators act on the operands as if they were strings of binary digits. It operates bit by bit, hence the name. The bitwise operator works at bits and performs bit by bit operations in Python.

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
~	Bitwise NOT
^	Bitwise XOR
>>	Bitwise right shift
<<	Bitwise left shift

Assignment operators

Assignment operators are used in Python to assign values to variables.

Here, a = 5 is a simple assignment operator that assigns the value 5 on the right, to the variable *a* on the left.

Operator	Example	Equivatent to
=	x = 5	x = 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
//=	x //= 5	x = x // 5

**=	x **= 5	x = x ** 5
&=	x &= 5	x = x & 5
|=	x |= 5	x = x | 5
^=	x ^= 5	x = x ^ 5
>>=	x >>= 5	x = x >> 5
<<=	x <<= 5	x = x << 5

Special operators

Python language provides some special types of operators - identity operator and membership operator.

Identity operators

"is" and **"is not"** are the identity operators in Python. This is used to check if two variables are located on the same part of the memory. Two variables that are equal, do not imply that they are identical.

a=10;
b=20;

Operator	Meaning	Example
Is	If True, refer to the same object.	a is b
Is not	If True, do not refer to the same object.	a is not b

Example: Test.py

```
a1 = 5
b1 = 5
a2 = 'Hello'
b2 = 'Hello'
a3 = [1,2,3]
b3 = [1,2,3]
print(a1 is not b1)# Output: False
print(a2 is b2)# Output: True
print(a3 is b3)# Output: False
```

Membership operators

"in" and **"not in"** are the membership operators in Python. This is used to test whether a variable is found in a sequence or not.
In a dictionary, a user can only test for the key, not the value.

Operator	Meaning	Example
In	True if value is found in the sequence	5 in x
Not in	True if value is not found in the sequence	5 not in x

```
>>>x = 'Hello world'
>>>y = {1:'a',2:'b'}
>>>print('H' in x)# Output: True
>>>print('hello' not in x)# Output: True
>>>print(1 in y)# Output: True
>>>print('a' in y)# Output: False
```


Input, Output and Import

Output using print() function

print():

The print() function is used to *output* the data to the standard output device.

Syntax:

Print(statement or variable)

```
Example:Test.py
Print('Hello livewire')
a = 5
print('The value of a is',a)
```

Output:

The value of a is 5.

In the second print() statement, python adds the tring and the value of the variable *a in the above program*.

The actual syntax of the print() function is:

```
>>>print (*objects, sep='', end='\n', file=sys.stdout,
flush=False)
```

Here, objects are the values that are going to print.
The “sep” separator gives the space between the values.
All values are printed, then the end is going to print. It is used to print the new line by ‘\n’.

```
>>>print(1,2,3,4)
>>>print(1,2,3,4,sep='*')
>>>print(1,2,3,4,sep='#',end='&')
```

Output formatting

Python would like to format our output values to make it look attractive. The user can be do this by using the str.format() method. This method can be used for any string object.

```
>>>print('I like {0} and {1}'.format('Bread','Butter'))
I like Bread and Butter
```

Input function

Python has two input statements:

1. input (),
2. raw_input ().

1.input()

The input() function is used to get the input from the user.

Syntax

```
input([prompt])
```

Example: Test.py

```
name= input('Enter any Name: ')
print(name)
```

Output:

```
Enter any Name: Livewire
Livewire
```

Example: Test.py

```
first = input("Enter first number: ")
second = input("Enter second number: ")
print (first + second)
```

Output:

Enter first number:10
Enter second number:10
20

This prints the sum of first and second input.

Converts string to integer using input()

If there is a need to convert *string input* into *number format* for arithmetic progress, the user has to convert explicitly.

Example: Test.py

```
x=input("Enter a first number: ")
x1=int(x)
y=input("Enter a second number: ")
y1=int(y)
print (x1+y1)
```

Output:

Enter first number:102
Enter second number:102
204

Or

Example: Test.py

```
a=int(input("Enter a first  number: "))
b=int(input("Enter a second number: "))
print (a+b)
```

Output:

Enter first number:101
Enter second number:101
202

Here the input is converted to *int value.*

2.raw_input()

This is another function to read the user input and display back on the screen using print():

Example: Test.py

```
name=raw_input('Enter your name : ')
print("Hi %s, Let us be friends!" % name);
```

Output:

Enter your name: Livewire
Hi Livewire, Let us be friends! Livewire

Here for input, there is no need to specify the input string within double quotes.

Example: Test.py

```
first = raw_input("enter first number: ")
second = raw_input("enter second number: ")
print (first + second  )
```


Output:

Enter first number:201
Enter second number:301
502

This is going to print a concatenated of both strings.

Import statement

Using import keyword the definitions inside a module can be imported to another module or the interactive interpreter in Python.

Syntax:

import module1[, module2[,...]

Example

If the user wants to include the math module, he can import the math module by just typing the following:

```
>>>import math
>>>print(math.pi)
```

"import"* using *from.

Syntax: from module_alias_name import module_name

```
>>>from m import math
```

A user module_alias_name can act as a module_name. The module_alias_name can be used anywhere in the program.

"import*"* Statement using *from.

Syntax: from modulename import*

```
>>>from math import *
```

This import statement import whole classes and interfaces to the working environment.

3.2 CONTROL FLOW

The control flow is the ordering which the program's code executes. The control flow is regulated by conditional statements and loops.

The major divisions of control statements are:

- Decision making
- Looping structure

1. Decision making

Decision making is the list of conditions occurring during the execution of the program and specifying the actions taken according to the conditions. It produces TRUE or FALSE as the outcome.

If Statement

I*f statement* executes some statements only if some condition holds, or choose statements to execute depending on several mutually exclusive conditions.

a. Simple if

If statement is use for taking a decision on the basis of condition. If the condition is satisfied, the statement of ***true*** block is executed, otherwise it is moved out.

Syntax:

```
if(condition):
  statements
```

Example: Test.py

```
num = 3
ifnum> 0:
print(num, "is a positive number.")
print("This is always printed.")
```

Output:

3 is a positive number
This is always printed

b. if...else Statement

In *if...else* condition, if the condition is true, it will execute the ***true*** block; otherwise it will execute the ***false*** block.

Syntax:

```
if test expression:
Body of if
else:
Body of else
```

Example: Test.py

```
num = 3
ifnum>= 0:
print("Greater than Zero")
else:
print("Less  than Zero ")
```

Output:

Greater than Zero

c. if...elif...else

The *elif*is the short form of else if. It is used to check for multiple expressions.

If the condition is False, it moves to the condition of the next *elif* block. If whole conditions are False; body ofthe else is executed.

Only one block, along with several *if...elif...else* blocks, are executed according to the state.

The “if” block can contain only one “else” block. But it can have many “elif” blocks.

Syntax

```
if test expression:
Body of if
elif test expression:
Body of elif
else:
Body of else
```


Example: Test.py

```
num = 3.4
ifnum> 0:
print("Greater than Zero")
elifnum == 0:
print("Zero")
else:
print("Less than Zero ")
```

Output:

Greater than Zero

d. Nested if

In a *nested if* construct, you can have an *if...elif...else* construct inside another *if...elif...else* construct.

Syntax:

```
if test expression:
 if test expression:
```

```
 Body of if
 else: test expression:
  Body else
else:
Body of else
```

Example: Test.py

```
num=float(input("Enter a number: "))
ifnum>=0:
ifnum==0:
print("Zero")
else:
print("Greater than Zero")
else:
print("Less than Zero")
```

Output:

Enter a number: 20
Greater than Zero

2. Looping structure

A loop statement executes a statement or group of statements, multiple times.

a. for Loop

It executes a sequence of statements many times and abbreviates the code that manages the loop variable.

Syntax:

```
forval in sequence:
    Body of for
```

Example: Test.py

```
numbers = [6,5,3,8,4,2,5,4,11]
sum = 0
forval in numbers:
    sum = sum+val
print("The sum is",sum)
```

Output:
The sum is 48

b. While Statement

It repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.

Syntax:

```
whiletest_expression:
Body of while
```

Example: Test.py

```
n = 1
sum = 0
i = 1

while i <= n:
sum = sum + i
i = i+1    # increment the “i” value
print("The sum is", sum)
```

Output:
The sum is 1

While loop with else

In the While loop, the user can have an optional *else* block.

The else part is executed if the condition is moved to False. The while loop must be terminated with a break statement.

Example: Test.py

```
counter = 0
while counter < 3:
print("Inside loop")
counter = counter + 1
else:
print("Inside else")
```

Output:

Inside loop
Inside loop
Inside loop
Inside else

While with condition at middle:

Example: Test.py

```
vowels="aeiouAEIOU"
# infinite loop
whileTrue:
v =input("Enter a vowel: ")
if v in vowels:
break
print("That is not a vowel. Try again!")
print("Thank you!")
```

Output:

Enter a vowel: q
Enter a vowel: w
Enter a vowel: e
Thank you!

Loop control statement

a. Break

Terminates the loop statement and transfers execution to the statement immediately following the loop.

Syntax:

```
break
```

Example: Test.py

```
forval in "Livewire":
ifval == "i":
break
print(val)
print("The end")
```

Output:
The End

b. Continue

The *continue* statement is used to skip the rest of the code inside a loop for the current iteration only. The Loop does not terminate, but continues on to the next iteration.

Syntax:

```
continue
```

Example: Test.py

```
forval in "Livewire":
ifval == "i":
continue
print(val)

print("The end")
```

Output:
The End

Pass Statement

'pass' is a null statement. The *pass statement* is used when a statement is required syntactically, but you do not want any command or code to execute

However, nothing happens when the pass is executed. It results into **no** operation. This is like a "void" keyword, it does not return any value.

Example: Test.py

```
se = {'p', 'a', 's', 's'}
for val in se:
    pass
```

Output:
Empty places.

3.3 FUNCTIONS

A function is a block of organised, reusable code that is used to perform a single, related action. The keyword def introduces a function definition. It is followed by the function name and the parenthesized list of formal parameters. The body of the function starts at the next line and it must be indented.

Types of Functions

Python supports two types of functions:

1. Built-in functions
2. User-defined functions

Built-in functions

Built–in functions are the default functions provided by the Python library. The Python interpreter has a number of built-in functions, and types built into it that are always available.

		Built-in Functions		
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

User-defined functions

These functions are defined by the user to do certain specific tasks. User-defined functions help to break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organised and manageable. Furthermore, it avoids repetition and makes code reusable.

Function Definitions

Below are some rules to define a function in Python:

- Function blocks should begin the keyword **def** followed by the function name and parenthesis.
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The first statement of a function can be an optional statement - the documentation string of the function or docstring.
- A colon should be placed at the end of the function header.
- An optional return statement can be used to return a value from the function.

Syntax:

```
>>>deffunction_name(parameters):
    """docstring"""
    statement(s)
    return [expression]
```

Example:Test.py

```
deftestfunction(name):
    """This function greets to
    the candidate  passed in as
    parameter"""
    print("Hello, " + name + ". Welcome to livewire!")
```

Output:

Hello, James. Welcome to livewire!

Function Call

After defining a function, the user can execute it by calling from another function or directly from the Python prompt. To call a function, just type the function name with suitable parameters.

Try to run the following into the Python shell to see the output.

```
>>>testfunction('James')
Hello, James. Welcome to livewire!
```

Docstring

The string statement followed by the function name is called the docstring and is short for documentation string. It provides a convenient way of associating documentation with Python modules, functions, classes, and methods. It is specified in the source code that is used, like a comment, in order to document a specific segment of code.

In the below program, Python has a docstring right away below the function name. Python generally uses triple quotes, so that the docstring can extend up to multiple line comments. This string in Python can be accessed by ***__doc__*** attribute of the function.

```
>>>print(testfunction.__doc__)
“””Thisfunction greets to
    the candidate passed into the
    name parameter”””
```

Function return statement

The return statement is used to exit a function and passing back an expression to the caller. A return statement with no arguments is the same as return None.

Syntax

```
return [expression_list]
```

Example

```
>>>print(testfunction("James"))
Hello,JamesWelcome to livewire!
None
```

Here, none is the returned value.

Example :Test.py

```
def value(n):
    """This function returns the
    value of the entered number"""

    if n>= 0:
        return n
```

```
    else:
        return -n

print(value(2)) # Output: 2
print(value(-4)) # Output: 4
```

Output:
2
4

Scope and Lifetime of variables

Scope of a variable is the part of a program where the variable is recognised. Variables defined inside a function are not visible from outside. This is considered as a local scope.
Lifetime of a variable is how long the variable is loaded in the memory. The lifetime of variables inside a function, is until the function executes.

Example:Test.py

```
deftestfunc():
    x = 10
    print("Value inside function:",x)

x = 20
testfunc()    # calling function
print("Value outside function:",x)
```

Output:
Value inside function: 10
Value outside function: 20

Here, we can see that the value of *x* is 20 initially. Even then the function testfunc() changed the value of *x* to 10, and it did not affect the value outside the function.

On the other hand, variables outside of the function are visible from inside. They have a global scope. The local value is from inside the function, but it cannot change from outside of the function.

In order to modify the value of variables outside the function, they must be declared as global variables using the keyword ***"global".***

Function Arguments

Function arguments are also called as parameters. They are variables passed to the function which are used to provide different output. In the below example, the function prototype specifies the two parameters.

Example:Test.py

```
def hello(name,msg):
print("Hi",name + ', ' + msg)
hello("James","Good morning!") # this is calling function
```

Output:

Hi James, Good morning!

Variable Length Arguments

These arguments can be used to process a function for more arguments than you specified while defining the function.

Default Arguments

Default arguments assume a default value, if a value is not provided in the function call for that argument. A default value to an argument can be given using the assignment operator ***(=).***

Example:Test.py

```
deffunctiontest(name, msg = "Good morning!"):
print("Hello",name + ', ' + msg)
functiontest ("James")
functiontest ("James","How are you?")
```

Output:

Hello James, Good morning!
Hello James, How are you?

In this function, the parameter "name" does not have a default value and is required during a calling function.

On the other hand, the parameter “msg” has a default value of "Good morning!". So, it is optional during a calling function. If a value is provided, it *will overwrite* the default value.

Keyword Arguments

When we call a function with some values, these values get assigned to the arguments according to their position.

For example, in the above function functiontest(), when we call it as functiontest ("Anne","How are you?"), the value "Anne" gets assigned to the argument *name* and similarly "How are you?" to *msg*.

These arguments are related to the function calls. We use the name (keyword) instead of the position to specify the arguments to the function.

Python allows functions to be called using keyword arguments. When we call functions in this way, the position of the arguments can be changed. Subsequent calls to the above function are all valid and produce the same effect.

```
>>>functiontest(name = "Anne",msg = "How are you?")
```

Arbitrary Arguments

Sometimes, the user does not know in advance the number of arguments that will be passed into a function. Python allows us to handle these kinds of circumstances, through function calls with an arbitrary number of arguments.

In the function definition, use an asterisk ***(*)*** before the parameter name to denote this kind of argument.

Example:Test.py

```
deffunctiontest(*names):
for name in names:
print("Hi",name)
functiontest("James","Peter","Ram","Vishwa")
```

Output:

Hi James
Hi Peter
Hi Ram
Hi Vishwa

Here, Python has called the function with *several arguments*. These arguments get wrapped up into a tuple before being passed into the function. Inside of the function, we may use a “for” loop to retrieve all the arguments back.

RECURSION

Recursion is the process of defining something in terms of itself. It would mean to place two parallel mirrors facing each other. Any object in between them would be reflected recursively.

Recursive Function

In Python, a function can call other functions. It is even possible for the function to call itself. These types of construct are termed as recursive functions.

The factorial of a number is the product of all the integers from 1 to that number.

Example: Test.py

```
def factorial(x):
if x == 1:
return 1
else:
return (x * factorial(x-1))
num = 4
print("The factorial value of given number", num, "is", factorial(num))
```

Output:

The factorial value of given number 4 is 24

Anonymous or Lambda Function

An anonymous function is a function that is defined without a name.

Whereas common functions are defined using the "***def***" keyword, anonymous functions are defined using the "***lambda***" keyword. Thus, anonymous functions are also called *lambda functions*.

Lambda Functions

Syntax

```
Lambda arguments: expression
```

Lambda functions can have several numbers of arguments but only one expression. The expression is evaluated and returned. Lambda functions can be used anywhere, where function objects are required.

Example

Here, use of the lambda function that doubles the input value.

```
>>>double = lambda a: a * 20
>>>print(double(5))
#Output is 100.
```

In the above program, lambda a: a * 20 is the lambda function. Here *a* is the argument and a * 20 is the expression that gets evaluated and returned.

This function has no name. It returns a function object which is assigned to the identifier double. The Python programmer can now call it as a normal function.

```
>>>double = lambda a: a * 2
```

This is the same as the following function.

```
>>>def double(a):
return as * 2
```

Use of Lambda

Lambda functions assist when a user requires a nameless function for a small period of time.

In general, we use it as an argument to a higher-order. Lambda functions are worn along with built-in functions like ***filter (), map ()*** etc.

Lambda with filter()

The ***"filter ()"*** function takes in a function and a list, as arguments. The function is called with all the items in the list and a new list is returned which contains items for which the function evaluates to True.

Use of the ***filter()*** function to filter out only even numbers from a list.

```
>>>ls = [1, 5, 4, 6, 8, 11, 3, 12]
>>>ls1 = list(filter(lambda x: (x%2 == 0) , ls))
>>>print(ls1)
# Output: [4, 6, 8, 12]
```


Lambda with map()

The ***map()*** function takes in a function and a list.
The function is called with all the items in the list and a new list is returned which contains items returned by that function for each item.
Here, use of the "***map ()"***function to double all the items in a list.

```
# An input list.
>>>items = [1, 2, 3]
# Apply lambda to all elements with map.
>>>for r in map(lambda x: x + 1, items):
print(r)
```

Output

2
3
4

3.4 Data Structures

Data Structures

Data structures are basically structures which can hold some data. They are used to store a collection of related data.

Numbers

There are different types of numbers used in Python. This supports integers, floating point numbers and complex numbers. They are defined as **int**, **float** and **complex** class in Python.

Complex numbers are written in the form, **x + yj**, where ***x*** is the real part and ***y*** is the imaginary part.Here we can use the **type()** function to identify which class a variable or a value belongs to and **isinstance()** function to check if it belongs to a particular class.

Example:Test.py

```
a = 5
print(type(a))
print(type(5.0))
c = 5 + 3j
print(c + 3)
print(isinstance(c, complex))
```

Output:

```
<class 'int'>
<class 'float'>
(8+3j)
True
```

While integers can be of any length, a floating point number is accurate only up to 15 decimal places (the 16th place is inaccurate).Numbers we deal with everyday are of the decimal (base 10) number system. However, computer programmers (generally embedded programmer) need to work with binary (base 2), hexadecimal (base 16) and octal (base 8) number systems.

It can represent these numbers by appropriately placing a prefix before that number. The following table lists these prefix.

Number System	Prefix
Binary	'0b' or '0B'
Octal	'0o' or '0O'
Hexadecimal	'0x' or '0X'

Example:Test.py

```
print(0b1101011)
print(0xFB + 0b10)
print(0o15)
```

Output:

```
107
253
13
```


Type Conversion

It converts one type of number into another. This is also well-known as coercion.

Operations like addition, subtraction, coerce integers to float implicitly, if one of the operands is float.

```
>>>1+2.0
3.0
```

Here can see above that 1 (integer) is coerced into 1.0 (float) for addition, and the result is also a floating point number.

Here built-in functions like **int ()**, **float ()** and **complex ()** can also be used to convert between types, explicitly. These functions can even convert from strings.

```
>>>int(2.3)
2
>>>int(-2.8)
```

```
-2
>>>float(5)
5.0
>>>complex('3+5j')
(3+5j)
```

When converting from float to integer, the number gets truncated (integer that is closer to zero).

Decimal

Python built-in class float performs some calculations that might amaze us. We all know that the sum of 1.1 and 2.2 is 3.3, but Python seems to disagree.

```
>>>(1.1+2.2)==3.3
False
```

This will only approximate 0.1 but will never be equal. Hence, it is the limitation of our computer hardware and not an error in Python.

```
>>>1.1+2.2
3.3000000000000003
```


To overcome this issue, we can use the ***decimal module*** that comes with Python. While floating point numbers have precision up to 15 decimal places, the decimal module has a user settable precision.

Example:Test.py

```
import decimal
print(0.1)
print(decimal.Decimal(0.1))
```

Output:

```
0.1
0.1000000000000000055511151231257827021181583404541015625
```

This module is used when the user wants to carry out decimal calculations like we learned in school.

It also preserves significance. Well know 25.50 kg is more accurate than 25.5 kg as it has two significant decimal places compared to one.

Example:Test.py

```
from decimal import Decimal as D
print(D('1.1') + D('2.2'))
print(D('1.2') * D('2.50'))
```

Output:

```
3.3
3.000
```

Why not implement the decimal every time, as an alternative of float? The main reason is competence. Floating point operations are carried out faster than Decimal operations.

Generally we use the decimal in the following cases:

- When we are making financial applications that need exact decimal representation.
- When we want to control the level of precision required.
- When we want to implement the notion of significant decimal places.
- When we want the operations to be carried out like we did at school.

Fractions

Python provides operations concerning fractional numbers throughout its **fractions** module. A fraction has a numerator and a denominator, both of which are integers. This module has support for normal number arithmetic.

Python can create Fraction objects in a variety of ways.

Example:Test.py

```
import fractions
print(fractions.Fraction(1.5))
print(fractions.Fraction(5))
print(fractions.Fraction(1,3))
```

Output:

3/2
5
1/3

While we create Fraction from **float**, we might get some abnormal results. This is appropriate to the imperfect binary floating point number representation.

Example:Test.py

```
import fractions
print(fractions.Fraction(1.1))
print(fractions.Fraction('1.1'))
```

Output:

2476979795053773/2251799813685248
11/10

Example:Test.py

```
from fractions import Fraction as F
print(F(1,3) + F(1,3))
print(1 / F(5,6))
print(F(-3,10) > 0)
print(F(-3,10) < 0)
```

Output:

2/3
6/5
False
True

Math Functions

Python offers modules like *math* and *random* to hold out different mathematics like trigonometry, logarithms, probability and statistics, etc.

Example:Test.py

```
import math
print(math.pi)
print(math.cos(math.pi))
print(math.exp(10))
print(math.log10(1000))
```

```
print(math.sinh(1))
print(math.factorial(6))
```

Output:

3.141592653589793
-1.0
22026.465794806718
3.0
1.1752011936438014
720

Here is the full list of functions and attributes available in the Python math module.

Example:Test.py

```
import random
print(random.randrange(10,20))
x = ['a', 'b', 'c', 'd', 'e']
print(random.choice(x))
random.shuffle(x)
print(x)
print(random.random())
```


Output:

15
c
['e', 'd', 'b', 'a', 'c']
0.8852914263200635

List

List is one of the most frequently used and very flexible data types used in Python. List holds an ordered collection of items i.e. you can store a sequence of items in a list.

Creating list

In Python programming, a list is created by placing all the items inside a square bracket **[ ]**, separated by commas. It can have any number of items and they might be of different types.

A list may be able to even have another list as an item. These are called *nested lists*.

```
my_list=[]
my_list=[1,2,3]
my_list=[1,"Hello",3.4]
my_list=["mouse",[8,4,6]]
```

Accessing from a list

a.List Index

The list can use the **index operator []** to use an item in a list. Index starts from 0. So, a list having 5 elements will have an index from 0 to 4.

The index must be an integer. Python cannot use float or other types, this will result into **TypeError**. Nested lists can be accessed using nested indexing.

Example:Test.py

```
my_list = ['p','r','o','b','e']
print(my_list[0])
print(my_list[2])
print(my_list[4])
n_list = ["Happy", [2,0,1,5]]
print(n_list[0][1])
print(n_list[1][3])
```

Output:

```
p
o
e
a
5
```

b.Negative indexing

Python allows negative indexing to be used for its sequences. The index of -1 refers to the last item, -2 to the second last item and so on.

Example:Test.py

```
my_list = ['p','r','o','b','e']
print(my_list[-1])
```

```
print(my_list[-5])
```

Output:

e

p

c.Slice lists

The user can enter a range of items in a list by using the slicing operator (colon).

Example:Test.py

```
my_list = ['p','r','o','g','r','a','m']
print(my_list[2:5])
print(my_list[:-5])
print(my_list[5:])
print(my_list[:])
```

Output:

['o', 'g', 'r']

['p', 'r']

['a', 'm']

['p', 'r', 'o', 'g', 'r', 'a', 'm']

Slicing can be best visualised, by considering the index to be between the elements, as made known below. So if we want to access a range, we need two indexes that will slice that portion from the list.

P	R	O	G	R	A	M
0	1	2	3	4	5	6
-6	-5	-4	-3	-2	-1	

Changing elements in list

List are *mutable*, meaning, their elements can be changed not like string or tuple. Here one can use the assignment operator (=) to modify an item or a range of items.

Example:Test.py

```
odd = [2, 4, 6, 8]
odd[0] = 1
print(odd)
odd[1:4] = [3, 5, 7]
print(odd)
```

Output:

[1, 4, 6, 8]
[1, 3, 5, 7]

It can add one item to a list using the **append ()** method or add several items using the **extend ()** method.

Example:Test.py

```
odd=[1,3,5]
odd.append(7)
print(odd)
odd.extend([9,11,13])
print(odd)
```


Output:

[1, 3, 5, 7]
[1, 3, 5, 7, 9, 11, 13]

Python can also use "**+**" operator to combine two lists. This is also called concatenation. The " * " operator repeats a list for the given number of times.

Example:Test.py

```
odd=[1,3,5]
print(odd +[9,7,5])
print(["re"]*3)
```

Output:

[1, 3, 5, 9, 7, 5]
['re', 're', 're']

Python can insert one item at a desired location by use the method **insert ()** or insert multiple items by squeeze it into an empty slice of a list.

Example:Test.py

```
odd = [1, 9]
odd.insert(1,3)
print(odd)
odd[2:2] = [5, 7]
print(odd)
```

Output:

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

Deleting elements from a list

Python can delete one or more items from a list using the keyword ***del***. It can even delete the list entirely.

Example:Test.py

```
my_list = ['p','r','o','b','l','e','m']
delmy_list[2]
print(my_list)
delmy_list[1:5]
print(my_list)
delmy_list
print(my_list)
```

Output:

Traceback (most recent call last):
File "C:/Users/Livewire/d.py", line 7, in <module>

print(my_list)
NameError: name 'my_list' is not defined

Here one can use the **remove ()** method to remove a certain item or the **pop ()** method to remove an item at the given index.

The **pop ()** method removes and returns the last item if the index is not provided. This helps us to implement lists as stacks. User can moreover use the **clear ()** method to empty a list.

Example:Test.py

```
my_list = ['p','r','o','b','l','e','m']
my_list.remove('p')
print(my_list)
print(my_list.pop(1))
print(my_list)
print(my_list.pop())
print(my_list)
my_list.clear()
print(my_list)
```

Output:

```
['r', 'o', 'b', 'l', 'e', 'm']
o
['r', 'b', 'l', 'e', 'm']
m
['r', 'b', 'l', 'e']
[]
```


The user will be able to also delete items in a list by handing over an empty list to a slice of elements.

```
>>>my_list=['p','r','o','b','l','e','m']
>>>my_list[2:3]=[]
>>>my_list
['p','r','b','l','e','m']
>>>my_list[2:5]=[]
>>>my_list
['p','r','m']
```

List Methods

List Methods that are available with list objects in Python programming, are tabulated below. The language can access as **list.method()**. Some of the methods have previously been used above.

Method	Description
append(***x***)	Add item ***x*** at the end of the list
extend(***L***)	Add all items in given list ***L*** to the end
insert(***i***, ***x***)	Insert item ***x*** at position ***i***
remove(***x***)	Remove first item that is equal to ***x***, from the list
pop([***i***])	Remove and return item at position ***i*** (last item if ***i*** is not provided)
clear()	Remove all items and empty the list
index(***x***)	Return index of first item that is equal to ***x***
count(***x***)	Return the number of items that is equal to ***x***
sort()	Sort items in a list in ascending order
reverse()	Reverse the order of items in a list
copy()	Return a shallow copy of the list

Example:Test.py

```
my_list = [3, 8, 1, 6, 0, 8, 4]
print(my_list.index(8))
print(my_list.count(8))
my_list.sort()
print(my_list)
my_list.reverse()
print(my_list)
```

Output:

```
1
2
[0, 1, 3, 4, 6, 8, 8]
[8, 8, 6, 4, 3, 1, 0]
```

Create new List

List understanding is an elegant and short way to create new lists from an existing list in Python. List understanding consists of an expression followed by *for statement* inside square brackets.

Here is an example to make a list with every item being increased by the power of 2.

```
>>>pow2 = [2 ** x for x in range(10)]
>>>print(pow2)
```

This code is equal to:

Example:Test.py

```
pow2 =[]
for x in range(10):
pow2.append(2** x)
```

A list understanding can optionally contain more **for** or **if** statements. A possible **if** statement can filter out items for the new list. Here are some examples:

```
>>> pow2 =[2** x for x in range(10)if x >5]
>>> pow2
[64,128,256,512]
>>>odd=[x for x in range(20)if x %2==1]
>>>odd
[1,3,5,7,9,11,13,15,17,19]
>>>[x+yfor x in['Python ','C ']for y in['Language','Programming']]
['Python Language','PythonProgramming','CLanguage','C Programming']
```

Other List Operations

List Membership Test

Python will be able to test if an item exists in a list or not, using the keyword "***in***".

Example:Test.py

```
my_list = ['p','r','o','b','l','e','m']
print('p' in my_list)
```

```
print('a' in my_list)
print('c' not in my_list)
```

Output:

True
False
True

Iterating Through a List

Using a **for** loop we will be able to iterate through each item in a list.

Example:Test.py

```
forlang in ['java','python','scala']:
print("I like",lang)
```

Output:

I like java
I like python
I like scala

List Functions

Built-in functions like **all()**, **any()**, **enumerate()**, **len()**, **max()**, **min()**, **list()**, **sorted()** etc. are regularly used with list to perform different tasks.

Function	Description
all()	Return True if all elements of the list are true.
any()	Return True if any element of the list is true else return False.
enumerate()	Return an enumerate object.
len()	Return the length (the number of items) in the list.
list()	Convert an iterable (tuple, string, set, dictionary) to a list.
max()	Return the largest item in the list.

min()	Return the smallest item in the list
sorted()	Return a new sorted list (does not sort the list itself).
sum()	Return the sum of all elements in the list.

Stack

The list methods, make it very easy to use a list as a stack, where the last element added is the first element retrieved ("last-in, first-out"). To add an item to the top of the stack, use ***append().*** To retrieve an item from the top of the stack, use ***pop()*** without an explicit index.

For example:

```
>>> Stack=[3,4,5]
>>>stack.append(6)
>>>stack.append(7)
>>>stack
[3, 4, 5, 6, 7]
>>>stack.pop()
7
>>>stack.pop()
6
>>>stack
[3,4,5]
```

Queue

It is also possible to use a list as a queue, where the first element added is the first element retrieved ("first-in, first-out"); however, lists are not efficient for this purpose. While appends and pops from the ends of list are fast, doing inserts or pops from the beginning of a list is slow.

To implement a queue, use collections. *Deque* was designed to have fast appends and pops from both ends.

For example:

```
>>>from collectionsimportdeque
>>>queue=deque(["Arun","Akbar","Paul"])
>>>queue.append("Amrith");
>>>queue.append("Peter");
>>>queue.popleft()
>>>queue.popleft()
>>>queue
```

Tuple

Tuple is like a list. The variation between the two is that we cannot change the elements of a tuple once it is assigned while in a list, elements can be changed.

Features of Tuple

Since tuples are pretty similar to lists; both of them are second-hand in similar situations as well.

However, there are firm advantages of implementing a tuple over a list.

Below listed, are several of the main advantages:

- Python normally uses tuple for heterogeneous data types and list for homogeneous data types.
- Since tuples are immutable, iterating through tuple is more rapid than with list. So there is a slight performance increase.
- Tuples that include immutable elements know how to be used as keys for a dictionary. With list, this is not probable.
- It can contain data that does not change, implementing it as tuple resolve guarantees that it remains write-protected.

Creating a Tuple

A tuple is shaped by placing all the items inside a **parentheses (),** separated by commas. The parentheses are optional, but it is a good practice to write it. A tuple knows how to have any number of items and they can be of different types.

Example:Test.py

```
my_tuple = ()
```

```
print(my_tuple)

my_tuple = (1, 2, 3)
print(my_tuple)

my_tuple = (1, "Hello", 3.4)
print(my_tuple)

my_tuple = ("mouse", [8, 4, 6], (1, 2, 3))
print(my_tuple)

my_tuple = 3, 4.6, "dog"
print(my_tuple)

a, b, c = my_tuple
print(a)
print(b)
print(c)
```


Output:

```
()
(1, 2, 3)
(1, 'Hello', 3.4)
('mouse', [8, 4, 6], (1, 2, 3))
(3, 4.6, 'dog')
3
4.6
dog
```

Creating a tuple with one element is a bit difficult. Having one element within parentheses is not sufficient. We need a trailing comma to point to the fact that it is a tuple.

Example:Test.py

```
my_tuple = ("hello")
print(type(my_tuple))
my_tuple = ("hello",)
print(type(my_tuple))
```

```
my_tuple = "hello",
print(type(my_tuple))
```

Output:

```
<class 'str'>
<class 'tuple'>
<class 'tuple'>
```

Accessing Elements in a Tuple

1. Indexing

Python can use the index operator [] to access an item in a tuple where the index starts from 0.

So, a tuple having 6 elements will have an index from 0 to 5. Trying to access an element further than (6, 7,...) will raise an IndexError. The index may be required to be an integer, so we cannot apply float or other types. This resolve results into TypeError.

Similarly, nested tuples are accessed using nested indexing, as shown in the illustration below:

Example:Test.py

```
my_tuple = ['p','e','r','m','i','t']

# Output: 'p'
print(my_tuple[0])

# Output: 't'
print(my_tuple[5])

# index must be in range
# If you uncomment line 14,
# you will get an error.
# IndexError: list index out of range

#print(my_tuple[6])
```

```
# index must be an integer
# If you uncomment line 21,
# you will get an error.
# TypeError: list indices must be integers, not float

#my_tuple[2.0]

# nested tuple
n_tuple = ("mouse", [8, 4, 6], (1, 2, 3))

# nested index
# Output: 's'
print(n_tuple[0][3])

# nested index
# Output: 4
print(n_tuple[1][1])
```

Output:

p
t
s
4

2. Negative Indexing

Python allow negative indexing for its sequence.
The index of -1 refers to the last item, -2 to the second last point and so on.

Example:Test.py

```
my_tuple = ['p','e','r','m','i','t']
# Output: 't'
print(my_tuple[-1])
# Output: 'p'
print(my_tuple[-6])
```

Output:

t
p

3. Slicing

Python can have the right to use a range of items in a tuple by means of the slicing operator - colon ***":".***

Example:Test.py

```
my_tuple = ('p','r','o','g','r','a','m')
print(my_tuple[1:4])
print(my_tuple[:-4])
print(my_tuple[6:])
print(my_tuple[:])
```

Output:

```
('r', 'o', 'g')
('p', 'r', 'o')
('m',)
('p', 'r', 'o', 'g', 'r', 'a', 'm')
```

Slicing is able to be best visualised by considering the index that needs to be between the elements, as exposed below. Thus, if we want to right of entry to a range, we want the index that will slice the portion starting the tuple.

P	R	O	G	R	A	M
0	1	2	3	4	5	6
-6	-5	-4	-3	-2	-1	

Changing a Tuple

A tuple cannot be changed once it has been assigned. But, if the element is in itself a mutable data type similar to list, its nested items could be changed. Python can also assign a tuple to different values.

Example:Test.py

```
my_tuple = (4, 2, 3, [6, 5])
my_tuple[3][0] = 6
print(my_tuple)
my_tuple = ('p','r','o','g','r','a','m')
print(my_tuple)
```

Output:
(4, 2, 3, [6, 5])
('p', 'r', 'o', 'g', 'r', 'a', 'm')

Here know how to use + operator to combine two tuples. This is also call **concatenation**. Python would be able to also **repeat** the elements in a tuple for a given number of times using the * operator.
Both + and * operations effect into a new tuple.

Deleting a Tuple

As discussed above, we cannot modify the elements in a tuple. That as well means we cannot delete or remove items from a tuple. But deleting a tuple wholly is possible using the keyword ***del***.

```
>>>my_tuple = ('p','r','o','g','r','a','m')
>>>delmy_tuple
>>>my_tuple
```

Tuple Methods

Methods that add items or remove items are not obtainable with tuple. Only the following two methods are available.

Method	Description
count(***x***)	Return the number of items that is equal to ***x***
index(***x***)	Return index of first item that is equal to ***x***

Example:Test.py

```
my_tuple = ('a','p','p','l','e',)

# Count
# Output: 2
print(my_tuple.count('p'))
# Index
# Output: 3
print(my_tuple.index('l'))
```

Output:
2
3

Other Tuple Operations

1. Tuple Membership Test

Here, we can examine if an item exists in a tuple or not, by using the keyword **in**.

Example:Test.py

```
my_tuple = ('a','p','p','l','e',)
# In operation
# Output: True
print('a' in my_tuple)
# Output: False
print('b' in my_tuple)
# Not in operation
# Output: True
print('g' not in my_tuple)
```


Output:
True
False
True

2. Iterating Through a Tuple

With **for** loop we can iterate each item in a tuple.

Example:Test.py

```
# Output:
# Hello John
# Hello Kate
for name in ('John','Peter'):
print("Hello",name)
```

Output:
Hello Jhon
Hello Peter

Tuple Function

Built-in functions

like **all()**, **any()**, **enumerate()**, **len()**, **max()**, **min()**, **sorted()**, **tuple()** etc. are normally used with tuple to complete different tasks.

Function	Description
all()	Return **True** if all elements of the tuple are true (or if the tuple is empty).
any()	Return **True** if any element of the tuple is true. If the tuple is empty, return **False.**
enumerate()	Return an enumerate object. It contains the index and value of all the items of tuple as pairs.
len()	Return the length (the number of items) in the tuple.
max()	Return the largest item in the tuple.
min()	Return the smallest item in the tuple
sorted()	Take elements in the tuple and return a new sorted list (does not sort the tuple itself).
sum()	Retrun the sum of all elements in the tuple.
tuple()	Convert an iterable (list, string, set, dictionary) to a tuple.

String

String is a sequence of characters. Computers do not agree with characters, they deal with numbers. Even although you may see characters on your screen, internally it is stored and manipulated as a mixture of 0's and 1's.

This change of character to a number is called encoding, and the backward process is decoding. ASCII and Unicode are a number of the well-liked encoding used.

In Python, a string is a series of Unicode character. Unicode can be introduced to include each character in all languages and bring uniformity in encoding. You can study more about Unicode from here.

Creating string

Strings are able to be created by enclosing characters inside a single quote or double quotes. Still triple quotes can be used in Python but are usually used to signify multiline strings and docstrings.

Example:Test.py

```
# all of the following are equal
my_string = 'Hello'
print(my_string)
my_string = "Hello"
print(my_string)
my_string = '''Hello'''
print(my_string)
# triple quotes string can extend multiple lines
my_string = """Hello, welcome to
the world of Python"""
print(my_string)
```


Output:

```
Hello
Hello
Hello
Hello, welcome to
the world of Python
```

Accessing characters in a string

Python is able to use individual characters using indexing and a range of characters using slicing. Index starts from 0. Trying to enter a character out of an index range will raise an **IndexError**. The index has to be an integer. We cannot make use of float or other types, this will result into **TypeError**. Python allows negative indexing for its sequences.

The index of -1 refers to the last item, -2 to the second last item and so on. We can access a range of items in a string by using the slicing operator (colon).

```
>>str = 'program'
>>>print('str = ', str)
#first character
>>>print('str[0] = ', str[0])
#last character
>>>print('str[-1] = ', str[-1])
#slicing 2nd to 5th character
>>>print('str[1:5] = ', str[1:5])
#slicing 6th to 2nd last character
>>>print('str[5:-2] = ', str[5:-2]
```

If user try to way in index out of the range or use decimal number, we resolve get errors.

```
# index must be in range
>>>my_string[19]
...IndexError:string index out of range
# index must be an integer
>>>my_string[1.8]
...TypeError:string indices must be integers
```


Slicing is able to be best visualized by considering the index to be between the elements as shown below. If you wish to access a range, the user needs the index that will slice the portion from the string.

P	R	O	G	R	A	M
0	1	2	3	4	5	6
-6	-5	-4	-3	-2	-1	

Delete a string

Strings are immutable. This resource, that elements of a string cannot be alive, changed once it has been assigned. Here one can simply reassign different strings to the same name.

```
>>>my_string='program'
>>>my_string[5]='a'
...TypeError:'str'object does not support item assignment
>>>my_string='Python'
>>>my_string
'Python'
```

Here, one cannot delete or remove characters from a string. But deleting the string entirely is probable using the keyword **del**.

```
>>>delmy_string[1]
TypeError:'str'object doesn't support item deletion
>>>delmy_string
>>>my_string
NameError: name 'my_string' is not defined
```

String Operations

There are various operations that are performed with strings which make it one of the most used datatypes in Python.

Concatenation of Two or More Strings

Joining of two or other strings into a single one is called concatenation.

The **+** operator does this in Python. Simply writing two string literals jointly also concatenates them.

The * operator can be used to do again the string for a given number of times.

Example:Test.py

```
str1 = 'Hello'
str2 ='World!'
# using +
print('str1 + str2 = ', str1 + str2)
# using *
print('str1 * 3 =', str1 * 3)
```

Output:

```
str1 + str2 =  HelloWorld!
str1 * 3 = HelloHelloHello
```

Writing two string literals jointly would concatenate them like + operator.

But if one wants to concatenate strings in different lines, then we can use parentheses.

```
>>>'Hello "World!'
'Hello World!'
```

```
>>># using parentheses
>>> s =('Hello '
...'World')
>>>s
'Hello World'
```

Iterating Through String

By means of *for loop* we will be able to iterate through a string. Here is an illustration to count the number of 'l' in a string.

Example:Test.py

```
count = 0
for letter in 'Hello World':
if(letter == 'l'):
count += 1
print(count,'letters found')
```

Output:

```
1 letters found
2 letters found
3 letters found
```


String Membership Test

The user can test if a sub string exists within a string or not, by means of the keyword **in**.

```
>>>'a'in'program'
True
>>>'at'notin'battle'
False
```

Built-in functions to Work

A variety of built-in functions that work with series, works with string as well.

A number of the commonly used ones are **enumerate ()** and **len()**. The **enumerate ()** function returns an enumerate object. It contains the index and value of all the items in the string as pair. This is of use for iteration. Likewise, **len()** returns the length of the string.

Example:Test.py

```
str = 'cold'
# enumerate()
list_enumerate = list(enumerate(str))
print('list(enumerate(str) = ', list_enumerate)
#character count
print('len(str) = ', len(str))
```

Output:

```
list(enumerate(str) =  [(0, 'c'), (1, 'o'), (2, 'l'), (3, 'd')]
len(str) =  4
```

String Formatting

Escape Sequence

If one hopes to print a text like -He said, "What's this?"- The user can neither use single quotes or double quotes. This will result into **SyntaxError** as the text itself contains both single and double quotes.

```
>>>print("He said, "What's this?"")
SyntaxError: invalid syntax
>>>print('Hesaid,"What's this"')
SyntaxError: invalid syntax
```

One approach to get around this problem is to make use of triple quotes. Alternatively, we will be able to use the escape sequences.

A run away sequence starts with a backslash and is interpreted in a different way. If we make use of single quotes to represent a string, all the single quotes inside the string must be escaped. Similarly is the case with double quotes. Here is how it can be ended to represent the above text.

```
# using triple quotes
>>>print('''He said, "What's this"''')
# escaping single quotes
>>>print('He said, "What\'s this?"')
# escaping double quotes
>>>print("He said, \"What's this?\"")
```

Here is a list of all the escape sequences supported by Python.

Escape Sequence	Description
\newline	Backslash and newline ignored
\\	Backslash
\'	Single quote
\"	Double quote
\a	ASCII Bell
\b	ASCII Backspace
\f	ASCII Formfeed
\n	ASCII Linefeed
\r	ASCII Carriage Return
\t	ASCII Horizontal Tab
\v	ASCII Vertical Tab
\ooo	Character with octal value ooo
\xHH	Character with hexadecimal value HH

Some examples here:

```
>>>print("C:\\Python32\\Lib")
C:\Python32\Lib
>>>print("This is printed\nin two lines")
Thisis printed
in two lines
>>>print("This is \x48\x45\x58 representation")
Thisis HEX representation
```

Raw String to ignore escape sequence

From time to time we may wish to ignore the escape sequences inside a string. To do this we will be able to place **r** or **R** in front of the string. This resolve implies that it is a raw string and any escape sequence inside it will be ignored.

```
>>>print("This is \x61 \ngood example")
Thisis a
good example
>>>print(r"This is \x61 \ngood example")
Thisis \x61 \ngood example
```

The format () Method for Formatting Strings

The **format ()** method that is available with the string object is very flexible and powerful in formatting strings. Format strings contains curly braces **{}** as replacement fields which gets replaced.
Python is able to use positional arguments or keyword arguments to state the order.

Example:Test.py

```
# default(implicit) order
default_order = "{}, {} and {}".format('Amar','Ram','Ragu')
print('\n--- Default Order ---')
print(default_order)
# order using positional argument
positional_order = "{1}, {0} and
{2}".format('Amar','Ram','Ragu')
print('\n--- Positional Order ---')
print(positional_order)

# order using keyword argument
keyword_order = "{s}, {b} and
{j}".format(j='Amar',b='Ram',s='Ragu')
print('\n--- Keyword Order ---')
print(keyword_order)
```

Output:

```
--- Default Order ---
Amar, Ram and Ragu
```

```
--- Positional Order ---
Ram, Amar and Ragu

--- Keyword Order ---
Ragu, Ram and Amar
```

The ***format()*** method is able to have optional format specifications. They are separated from field names using colons. For example, we will be able to left-justify **<**, right-justify **>** or center **^** a string in the known space. We will be able to format integers as binary, hexadecimal etc. and floats which can be round or displayed in the exponent format. There are tons of formatting you will be able to use. Visit here for all the string formatting accessible with the ***format()*** method.

```
>>># formatting integers
>>>"Binary representation of {0} is {0:b}".format(12)
'Binary representation of 12 is 1100'
>>># formatting floats
>>>"Exponent representation: {0:e}".format(1566.345)
'Exponent representation: 1.566345e+03'
>>># round off
>>>"One third is: {0:.3f}".format(1/3)
'One third is: 0.333'
>>># string alignment
>>>"|{:<10}|{:^10}|{:>10}|".format('butter','bread','ham')
'|butter    |  bread   |       ham|'
```

Old style formatting

The user will be able to even format strings like the old **sprintf()** style to make use of in the C programming language. We make use of the **%** operator to accomplish this.

```
>>> x =12.3456789
>>>print('The value of x is %3.2f'%x)
The value of x is12.35
>>>print('The value of x is %3.4f'%x)
The value of x is12.3457
```

Common String Methods

There are numerous methods presented with the string object. The **format()** method that we mentioned is one of them. Some of the commonly use methods
are **lower()**, **upper()**, **join()**, **split()**, **find()**, **replace()** etc.

```
>>>"PrOgRaM".lower()
'program'
>>>"PrOgRaM".upper()
'PROGRAM'
>>>"This will split all words into a list".split()
['This','will','split','all','words','into','a','list']
>>>''.join(['This','will','join','all','words','into','a','string'])
'This will join all words into a string'
>>>'Happy New Year'.find('ew')
7
>>>'Happy New Year'.replace('Happy','Brilliant')
'Brilliant New Year'
```

Set

A set is an unordered collection of items. Every element is ***unique*** and has to be ***immutable***. However, the set itself is mutable. We are able to add or remove items from it. Sets are able to be used to perform mathematical set operations like union, intersection, symmetric difference etc.

Creating a Set

A set is shaped by placing all the items inside curly braces **{}**, separated by commas or by using the built-in function **set()**.
It knows how to have any number of items and they might be of different types. But a set cannot have a mutable element, like list, set or dictionary, as its element.

Example:Test.py

```
# set of integers
my_set = {1, 2, 3}
print(my_set)
# set of mixed datatypes
my_set = {1.0, "Hello", (1, 2, 3)}
print(my_set)
```

```
The following examples as well.
# set do not have duplicates
# Output: {1, 2, 3, 4}
my_set = {1,2,3,4,3,2}
print(my_set)
# set cannot have mutable items
# here [3, 4] is a mutable list
# If you uncomment line #12,
# this will cause an error.
# TypeError: unhashable type: 'list'
#my_set = {1, 2, [3, 4]}
# we can make set from a list
# Output: {1, 2, 3}
my_set = set([1,2,3,2])
print(my_set)
```

Output:

```
{1, 2, 3}
{'Hello', 1.0, (1, 2, 3)}
{1, 2, 3, 4}
{1, 2, 3}
```


Create an empty set

Empty curly braces {} force make an empty dictionary in Python. To make a set without any elements we make use of the **set()** function without any argument.

Example:Test.py

```
# initialize a with {}
a = {}
# check data type of a
# Output: <class 'dict'>
print(type(a))
# initialize a with set()
a = set()
# check data type of a
# Output: <class 'set'>
print(type(a))
```

Output:

<class 'dict'>

<class 'set'>

Changing a Set

Sets are mutable. But while they are unordered, indexing has no meaning. We cannot right of entry or change an element of set using indexing or slicing. Set does not sustain it. The user has to add a single element using the **add ()** method and multiple elements using the **update ()** method. The **update ()** method is be able to take tuples, lists, strings or other sets as its argument. In all cases, duplicates are avoided.

Example:Test.py

```
# initializemy_set
my_set = {1,3}
print(my_set)
# if you uncomment line 9,
# you will get an error
# TypeError: 'set' object does not support indexing
#my_set[0]
# add an element
# Output: {1, 2, 3}
my_set.add(2)
print(my_set)
# add multiple elements
# Output: {1, 2, 3, 4}
my_set.update([2,3,4])
print(my_set)
# add list and set
# Output: {1, 2, 3, 4, 5, 6, 8}
my_set.update([4,5], {1,6,8})
print(my_set)
```

Output:

{1, 3}

{1, 2, 3}

{1, 2, 3, 4}

{1, 2, 3, 4, 5, 6, 8}

Removing Elements from a Set

A particular item can be removed from set using methods like, **discard ()** and **remove ()**.

The only difference between the two is that, even using **discard(),** if the item does not exist in the set, its remnants remain unchanged. But **remove()** will raise an error in such a condition.

Example:Test.py

```
# initializemy_set
my_set = {1, 3, 4, 5, 6}
print(my_set)
# discard an element
# Output: {1, 3, 5, 6}
my_set.discard(4)
print(my_set)
# remove an element
# Output: {1, 3, 5}
my_set.remove(6)
print(my_set)
# discard an element
# not present in my_set
# Output: {1, 3, 5}
my_set.discard(2)
print(my_set)
```

Output:

{1, 3, 4, 5, 6}
{1, 3, 5, 6}
{1, 3, 5}
{1, 3, 5}

Similarly, Python can remove and return an item by the **pop()** method.

Set being unordered, there is no way of determining which item will be pop. It is completely random. The user will be able to also remove all items from a set using **clear()**.

Example:Test.py

```
# initializemy_set
# Output: set of unique elements
my_set = set("HelloWorld")
print(my_set)

# pop an element
# Output: random element
print(my_set.pop())

# pop another element
# Output: random element
my_set.pop()
print(my_set)

# clearmy_set
Output: set()
my_set.clear()
print(my_set)
```


Output:

```
{'H', 'W', 'd', 'l', 'r', 'e', 'o'}
H
{'d', 'l', 'r', 'e', 'o'}
set()
```

Set Operation

Sets are to be used to carry out mathematical set operations like union, intersection, difference and symmetric difference. Here, one can perform this with operators or methods. Accede to us; consider the following two sets for the following operation:

```
>>> A ={1,2,3,4,5}
>>> B ={4,5,6,7,8}
```

Set Union

Union of ***A*** and ***B*** is a set of all elements from both sets.

Union is performed using "**|**" operator. Same is impossible using the method **union()**.

```
# initialize A and B
>>>A = {1, 2, 3, 4, 5}
>>>B = {4, 5, 6, 7, 8}
# use | operator
# Output: {1, 2, 3, 4, 5, 6, 7, 8}
>>>print(A | B)
```

Try the following examples.

```
>>>A.union(B)
{1,2,3,4,5,6,7,8}
>>>B.union(A)
{1,2,3,4,5,6,7,8}
```

Set Intersection

Intersection of ***A*** and ***B*** is a set of elements that are regular in equal sets.

Intersection is performed using **"&"** operator. Same is possible using the method **intersection()**.

```
# initialize A and B
>>>A = {1, 2, 3, 4, 5}
>>>B = {4, 5, 6, 7, 8}

# use& operator
# Output: {4, 5}
>>>print(A & B)
```

Try the following examples.

```
>>>A.intersection(B)
{4,5}
>>>B.intersection(A)
{4,5}
```

Set Difference

Difference of ***A*** and ***B*** (***A*** - ***B***) is a set of elements that are only in ***A*** but not in ***B***. Similarly, ***B*** – ***A*** is a set of element in ***B*** but not in ***A***.

Diversity is performed using " –"operator. Same can be accomplished using the method **difference()**.

```
# initialize A and B
>>>A = {1, 2, 3, 4, 5}
>>>B = {4, 5, 6, 7, 8}
# use - operator on A
# Output: {1, 2, 3}
>>>print(A - B)
```

Try the following on Python shell.

```
>>>A.difference(B)
{1,2,3}
>>> B - A
{8,6,7}
>>>B.difference(A)
{8,6,7}
```

Set Symmetric Difference

Symmetric distinction of ***A*** and ***B*** is a set of elements in both ***A*** and ***B*** except those that are general in both. Symmetric divergence is performed using **^** operator. Same can be accomplished using the method **symmetric_difference()**.

```
# initialize A and B
>>>A = {1, 2, 3, 4, 5}
>>>B = {4, 5, 6, 7, 8}
# use ^ operator
# Output: {1, 2, 3, 6, 7, 8}
>>>print(A ^ B)
Try the following on Python shell.
>>>A.symmetric_difference(B)
{1,2,3,6,7,8}
>>>B.symmetric_difference(A)
{1,2,3,6,7,8}
```

Set Methods

There are several set methods, some of which contain those already used above. Here is a list of all the methods that are offered with set objects.

Method	Description
add()	Add an element to a set
clear()	Remove all elements form a set
copy()	Return a shallow copy of a set
difference()	Return the difference of two or more sets as a new set
difference_update()	Remove all elements of another set from this set
discard()	Remove an element from set if it is a member.
intersection()	Return the intersection of two sets as a new set
intersection_update()	Update the set with the intersection of itself and another
isdisjoint()	Return **True** if two sets have a null intersection
issubset()	Return **True** if another set contains this set
issuperset()	Return **True** if this set contains another set
pop()	Remove and return an arbitrary set element. Raise **KeyError** if the set is empty
remove()	Remove an element from a set. It the element is not a member, raise a **KeyError**

symmetric_difference()	Return the symmetric difference of two sets as a new set
symmetric_difference_update()	Update a set with the symmetric difference of itself and another
union()	Return the union of sets in a new set
update()	Update a set with the union of itself and others

Other Set Operations

Set Membership Test

Here one will be able to test if an item exists in a set or not, using the keyword **in**.

Example:Test.py

```
# initializemy_set
my_set = set("apple")
# check if 'a' is present
# Output: True
print('a' in my_set)
# check if 'p' is present
# Output: False
print('p' not in my_set)
```

Output:

True
False

Iterating Through a Set

Using *for loop*, it can iterate though each item in a set.

```
>>>for letter inset("apple"):
...print(letter)
...
a
p
e
l
```

Set Functions

Built-in functions
like **all()**, **any()**, **enumerate()**, **len()**, **max()**, **min()**, **sorted()**, **sum()** etc. are generally used with set to execute different tasks.

Function	Description
all()	Return **True** if all elements of the set are true (or if the set is empty).
any()	Return **True** if any element of the set is true. If the set is empty, return **False.**
enumerate()	Return an enumerate object. It contains the index and value of all the items of set as a pair.
len()	Return the length (the number of items) in the set.
max()	Return the largest item in the set.
min()	Return the smallest item in the set.
sorted()	Return a new sorted list from elements in the set(does not sort the set itself).
sum()	Retrun the sum of all elements in the set.

Frozen set

Frozen set is a new class that has the characteristics of a set, although its elements cannot be changed once assigned. While tuples are immutable lists, frozen sets are immutable sets.

Sets being mutable are *a hashable*, so it cannot be used as dictionary keys. On the other hand, frozen sets are *hashable* and can be used as keys to a dictionary.

Frozensets can be created using the function **frozenset()**.
This datatype supports methods like:
copy(), **difference()**, **intersection()**, **isdisjoint()**, **issubset()**, **issuperset()**, **symmetric_difference()** and **union()**.

Being immutable, it does not have methods that add or remove elements.

```
# initialize A and B
>>>A = frozenset([1, 2, 3, 4])
>>>B = frozenset([3, 4, 5, 6])

Try these examples on Python shell.
>>>A.isdisjoint(B)
False
>>>A.difference(B)
frozenset({1, 2})
>>> A | B
frozenset({1, 2, 3, 4, 5, 6})
>>>A.add(3)
AttributeError: 'frozenset'object has no attribute 'add'
```

Dictionary

Dictionary is an unordered collection of items. Whereas other compound data types have only value as an element, a dictionary has a **key: value** pair. Dictionaries are optimising to retrieve values when the key is well-known.

Creating dictionary

Dictionary creation is as simple as placing items inside **curly braces {}** separated by comma. Items have a key and the corresponding value expressed as a pair, **key: value**.

Whereas values can be of any data type and can repeat; keys must be of immutable type and must be unique. Python will be able to create a dictionary using the built-in function **dict ()**.

Example: Test.py

```
# empty dictionary
my_dict={}
# dictionary with integer keys
my_dict={1:'apple',2:'ball'}
# dictionary with mixed keys
my_dict={'name':'Aadhithyan',1:[2,4,3]}
```

```
# using dict()
my_dict=dict({1:'apple',2:'ball'})
# from sequence having each item as a pair
my_dict=dict([(1,'apple'),(2,'ball')])
```

Output:
Empty

Access elements from a dictionary

Whereas indexing is used with other container types to access values; dictionary uses keys. Key can be used either surrounded by square brackets or by the **get()** method.

The dissimilarity while using **get()** is that it returns **None** instead of **KeyError**, if the key is not found.

Example:Test.py

```
my_dict = {'name':Peter', 'age': 10}
print(my_dict['name'])
print(my_dict.get('age'))
```


Output:
Peter
10

Add elements in a dictionary

Dictionary is mutable. Python is able to add new items or change the value of existing items by assignment of an operator. If the key is previously present, the value gets updated, as well as a new key: value pair is added to the dictionary.

Example:Test.py

```
my_dict = {'name':'John', 'age': 10}
# update value
my_dict['age'] = 27
#Output: {'age': 10, 'name': 'John'}
print(my_dict)
```

```
# add item
my_dict['address'] = 'New Delhi'
# Output: {'address': 'New Delhi', 'age': 10, 'name': 'John'}
    print(my_dict)
```

Output:

{'age': 27, 'name': 'John'}
{'address': 'Chennai', 'age': 27, 'name': John'}

Delete or remove elements from a dictionary

Python will be able to remove an exacting item in a dictionary by using the method **pop()**. This method removes an item with the provided key and returns the value.

The method, **popitem()** can be used to remove and return key value from the dictionary. All the items can be removed at once using the **clear()** method. Python is able to also use the **del** keyword to remove individual items or the entire dictionary itself.


```
# create a dictionary
>>>squares = {1:1, 2:4, 3:9, 4:16, 5:25}
# remove a particular item
# Output: 16
>>>print(squares.pop(4))
# Output: {1: 1, 2: 4, 3: 9, 5: 25}
>>>print(squares)
# remove an arbitrary item
# Output: (1, 1)
>>>print(squares.popitem())
# Output: {2: 4, 3: 9, 5: 25}
>>>print(squares)
# delete a particular item
>>>del squares[5]
# Output: {2: 4, 3: 9}
>>>print(squares)
# remove all items
>>>squares.clear()
```

```
# Output: {}
>>>print(squares)
# delete the dictionary itself
>>>del squares
# Throws Error
# print(squares)
```

Dictionary Methods

Methods that are presented with dictionary are tabulated below. Some of them have already been used in the above examples.

Method	Description
clear()	Remove all items form the dictionary.
copy()	Return a shallow copy of the dictionary.
fromkeys(***seq***[, ***v***])	Return a new dictionary with keys from ***seq*** and value equal to ***v***(defaults to **None**).
get(***key***[,***d***])	Return the value of ***key***. If ***key*** doesnot exit, return ***d*** (defaults to **None**).
items()	Return a new view of the dictionary's items (key, value).
keys()	Return a new view of the dictionary's keys.
pop(***key***[,***d***])	Remove the item with ***key*** and return its value or ***d*** if ***key*** is not found. If ***d*** is not provided and ***key*** is not found, raises **KeyError**.
popitem()	Remove and return an arbitary item (key, value). Raises **KeyError**if the dictionary is empty.
setdefault(***key***[,***d***])	If ***key*** is in the dictionary, return its value. If not, insert ***key*** with a value of ***d*** and return ***d*** (defaults to **None**).
update([***other***])	Update the dictionary with the key/value

	pairs from ***other***, overwriting existing keys.
values()	Return a new view of the dictionary's values

Here are a few example uses of these methods:

Example:Test.py

```
marks = {}.fromkeys(['Math','English','Science'], 0)
# Output: {'English': 0, 'Math': 0, 'Science': 0}
print(marks)
for item in marks.items():
print(item)
# Output: ['English', 'Math', 'Science']
list(sorted(marks.keys()))
```

Output:

```
{'English': 0, 'Math': 0, 'Science': 0}
('English', 0)
('Math', 0)
('Science', 0)
```


Dictionary Comprehension

Dictionary comprehension is an elegant and brief way to create a new dictionary from an iterable. Dictionary comprehension consists of an expression pair (key: value) followed by the **for** statement inside curly braces {}.

Here is an example to create a dictionary with each item being a pair of a number and its square.

Example:Test.py

```
squares = {x: x*x for x in range(6)}
# Output: {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
print(squares)
```

Output

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

This code is equivalent to:

Example:Test.py

```
squares={}
for x in range(6):
squares[x]= x*x
```

A dictionary comprehension is able to optionally include more *for* or *if* statements. An optional **if** statement can filter out items to form the new dictionary. Here is an example to make a dictionary with only odd items.

Example:Test.py

```
odd_squares = {x: x*x for x in range(11) if x%2 == 1}
# Output: {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
print(odd_squares)
```

Output

{1: 1, 3: 9, 9: 81, 5: 25, 7: 49}

Other Dictionary Operations

Dictionary Membership Test

Here one can test if a key is in a dictionary or not by the keyword **in**. Note that the membership test is for keys only, not for values.

Example:Test.py

```
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

# Output: True
print(1 in squares)

# Output: True
print(2 not in squares)

# Output: False
print(49 in squares)
```

Output:

True
True
False

Creating Through a Dictionary

Using **for** loop we can iterate though each key in a dictionary.

Example:Test.py

```
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
for i in squares:
print(squares[i])
```

Output:

```
1
9
81
25
49
```

Functions with Dictionary

Built-in functions like **all ()**, **any ()**, **len ()**, **cmp ()**, **sorted ()** etc. are commonly used with dictionary to perform different tasks.

Function	Description
all()	Return **True** if all keys of the dictionary are true.
any()	Return **True** if any key of the dictionary is true. If the dictionary is empty, return **False.**
len()	Return the length in the dictionary.
cmp()	Compares items of two dictionaries.
sorted()	Return a new sorted list of keys in the dictionary.

Here are some examples that use built-in functions to work with dictionary.

Example:Test.py

```
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

# Output: 5
print(len(squares))
```

```
# Output: [1, 3, 5, 7, 9]
print(sorted(squares))
```

Output

[1, 3, 5, 7, 9]

3.5 Modules and Packages

Modules

A module is a file containing Python definitions and statements. A file name is the module name with the suffix .py appended. The definitions from a module can be imported into other modules or into the main modules.

Python makes use of modules to break down large programs into small, convenient and organised files. Additionally, modules afford reusability of code. It can describe our most used functions in a module and import it, as an alternative of copying their definitions into different programs.

Let us create a module and save it as Test.py.

Example:Test.py

```
def add(a, b):
    res= a + b
    return res
add(10.2)
```

Output

12.2

Here, we have defined a function **add()** inside a module named Test. The function takes two numbers as input and returns their sum.

Import modules

Python can import the definitions inside a module to another module or to the interactive interpreter.

It has to use the “import” keyword to do this. To import our previously defined module, type the following in the prompt.

```
>>> import Test
```

This does not enter the names of the functions defined in Test directly in the current symbol table. It only enters the module name Test here.

Using the module name, Python can access the function using ***dot (.)*** operation. For example:

```
>>>Test.add(1,2)
3
```

Standard modules can be imported the same way as Python imports our user-defined modules. There are various ways to import modules.

Import pre defined module

The user can import a module using the import statement and access the definitions inside it, by using the dot operator as described above.

```
>>>import math
>>>print("The value of pi is", math.pi)
```

Import using renaming modules

Python can preserve and import a module by renaming it as follows.

```
>>>import math as ma
>>>print("The value of pi is", ma.pi)
```

Python has renamed the math module as *ma*. This can save the typing time in some cases.

Import using ‘from’

The user can import definite names from a module without importing the module as an entire.

```
>>>from math import pi
>>>print ("The value of pi is", pi)
```

Import only an attribute of pi from the MATH.
In this case do not use the dot operator. We could have imported multiple attributes as follows.

```
>>>from math import pi, e
>>>pi
3.141592653589793
>>>e
2.718281828459045
```

Import all names

Python is able to import all names from a module using the following:.

```
>>>from math import *
>>>print ("The value of pi is", pi)
```

The ***asterisk (*)*** symbol is not a recommended programming practice. This can direct to duplicate definitions for an identifier.

Module Search Path

When a module 'Test' is imported, an interpreter first looks for a built-in module with that name. If not found, it then looks into a list of directories defined in **sys.path. sys.path** is initialised from the following locations and the search is in this order:

- The directory containing the input script or the current directory when no file is specified.
- Python Path variable.
- The installation-dependentdefault.

```
>>> import sys
>>>sys.path
['',
'C: \Programfiles\Python3.5\\Lib\\idlelib',
```

Reloading a module

The interpreter imports a module only once during a session. This makes things more efficient. Suppose the user has the following code in a module named my_module_test.

```
>>>print("This code got executed")
```

The effect of multiple imports:

```
>>> import my_module_test
>>> import my_module_test
>>> import my_module_test
```

If our module changed during the program, users have to reload it. One way to do this is to re- invoke the interpreter. This provides a neat way of doing this. The user can use the ***reload()*** function inside the *imp* module to reload a module. This is how it is done. Notice the following:

```
>>> import imp
>>> import my_module
This will be executed

>>> import my_module_test
>>>imp.reload(my_module_test)
This will be executed
<module 'my_module_test' from '.\\my_module_test.py'>
```


dir() built-in function

Python can use the ***dir()*** function to find out names that are defined inside a module.

Example, we have defined a function **add()** in the module "Test" that we had in the beginning.

```
>>>dir(Test)
['__builtins__',
'__cached__',
'__doc__',
'__file__',
'__initializing__',
'__loader__',
'__name__',
'__package__',
'add']
```

Here can see a sorted list of names. All other names that begin with an underscore are default in Python attributes associated with the module. For example, the**__*name*__**attribute contains the name of the module.

```
>>> import Test
>>>Test.__name__
'Test'
```

All the names defined in our current namespace can be created using the **dir()** function without any arguments.

```
>>> a = 1
>>> b = "hello"
>>> import math
>>>dir()
['__builtins__', '__doc__', '__name__', 'a', 'b', 'math',
'pyscripter']
```

Packages

Packages are a way of structuring Python's module namespace by using "dotted module names".A package is a hierarchical file structure that defines a single Python application environment that consists of modules and sub packages in Python. The directory must contain a file named ***__init__.py***in order to consider it as a package.

Programmers do not usually store all of our files in our computer in the similar location. The user uses an ordered hierarchy of directories for easier access.

As our application program grows bigger in size with a lot of modules, we place similar modules in one package and different modules in different packages. This makes a project easy to manage and abstractly clear.

A directory must contain a file named ***__init__.py*** in order to consider it as a package. This file can be left empty, but we normally place the initialization code for that package in this file.

```
classBox:
def area(self):
    returnself.width * self.height
```

```
def__init__(self,width,height):
    self.width = width
    self.height = height

# Create an instance of Box.
x = Box(10, 2)
# Print area.
print(x.area())
```

Output

20

3.6 File Input and Output

File Operation

File is a name of the location on a disk to store related information. It is used to permanently store data in a hard disk. We use files for future use of the data.

When users want to read or write a file, we need to open it first. When the opening tasks are completed, it needs to be closed, so that content that is tied with the file is freed.

Hence, a file operation takes the following order:

1. Open a file
2. Read or write a file
3. Close the file

Opening a file

It has a built-in function ***open ()*** to open a file. This function returns a file object, also called a handle, as it is used to read or modify the file accordingly.

```
>>> f =open("test.txt")# open file in current directory
>>>  f  =open("C:/Python33/README.txt")#  specifying
full path of directory
```

Here, we can specify the mode while opening a file. The mode ***'r'*** can be used when the file will only be read; the mode ***'w'*** can be used for only writing or the mode **'a'** can be used for appending the

file. Users can specify if they want to open the file in text mode or binary mode. The default is reading in text mode. In this mode, we can read from the file. Further, binary mode returns bytes and this is the mode to be used when dealing with non-text files like image or exe files.

Mode	Description
'r'	Open a file for reading.
'w'	Open a file for writing. Creates a new file if it does not exist or truncates the file if it exists.
'x'	Open a file for exclusive creation. If the file already exists, the operation fails.
'a'	Open for appending at the end of the file without truncating it. Creates a new file if it does not exist.
't'	Open in text mode.
'b'	Open in binary mode.
'+'	Open a file for updating.

```
>>>f =open("test.txt",'r')# equivalent to 'r' or 'rt'
>>>f =open("test.txt",'w')# write in text mode
>>>f =open("img.bmp",'r+b')# read and write in binary mode
```

Unlike other languages, the character 'a' does not imply the number 97 until it is encoded using **ASCII**. Furthermore, the default encoding is platform dependent. In windows, it is 'cp1252' but in Linux 'utf-8'.Hence, the user must not rely too heavily on the default encoding, or else our code will act differently in different platforms. When working with files in text mode, it is highly recommended to indicate the encoding type.

```
>>>f =open("test.txt",mode='r',encoding='cp1252')
```

Closing a File

After reading from a file, we have to properly close it by the ***close()***method . Python has a garbage collector to clean up unreferenced objects but, we must not rely on it to close the file.

```
>>>f =open("test.txt",encoding='utf-8')
>>>f.close()
```

This method is not completely safe. If an exception occurs when users are performing some operation with the file, the code exits without closing the file. A safer way is to use a ***try...finally*** block.

Example:Test.py

```
try:
f =open ("test.txt",encoding='utf-8')
Print ("File opened successfully")
finally:
f.close()
```

Output:

File opened successfully

In this way, it is guaranteed that the file is correctly closed even if an exception is raised, causing the program flow to end. The most excellent way to do this, is by using the **with** statement. This ensures that the file is closed when the block inside with is exited.

Here we do not want to explicitly call the **close ()** method. It is done by itself.

```
with open("test.txt",encoding='utf-8')as f:
```

Writing to a File

In order to write into a file we want to open it in write ***'w'***, append 'a' or exclusive creation 'x' mode. The user needs to be careful with the 'w' mode, as it will overwrite into the file if it already exists. All previous data will be erased.

Writing a string or sequence of bytes is done using ***write ()*** method. This method returns the number of characters written to the file.

```
>>>withopen("test.txt",'w',encoding='utf-8')as f:
f.write("my first file\n")
f.write("This is a file\n\n")
f.write("File contains three lines\n")
```

Output:

My first file
This is a file
File contains three lines

This program will create a new file named 'test.txt' if it does not exist. If it does exist, it is overwritten.

Reading from a File

To read the recourses of a file, the user must open the file in reading mode.

There are various methods available for this purpose. The user can use the **read (size)** method to read in ***size*** number of data. If ***size*** parameter is not specified, it reads and returns up to the end of the file.

```
>>> f =open("test.txt",'r',encoding='utf-8')
>>>f.read(4)# read the first 4 characters including space…
    'This'
>>>f.read(4)# read the next 4 characters including space…
    ' is '
>>>f.read()# read in the rest till end of file
    'my first file\nThis file\ncontains three lines\n'
>>>f.read()# further reading returns empty sting"
```

User will be able to see, that the ***read()*** method returns newline as '\n'. This has reached the end of the file, and we get an empty string on further reading.

The user will be able to change our current file cursor position using the ***seek()*** method. Also, *the* ***tell()*** method returns our current position in number of bytes.

```
>>>f.tell()# get the current file position
56
>>>f.seek(0)# bring file cursor to initial position
0
>>>print(f.read())# read the entire file
Thisismy first file
This file
Contains three lines
```

Here we will be able to read a file line-by-line using *for loop*. This is efficient and fast.

```
>>>for line in f:
...print(line,end='')
...
Thisismy first file
This file
Contains three lines
```


The lines in the file itself have a newline character '\n'.
In addition, the **print()** content is to avoid newlines when printing.
Alternately, we will be able to use the **readline()** method to read individual lines of a file. This method reads a file till the newline, including the newline character.

```
>>>f.readline()
'This is my first file\n'
>>>f.readline()
'This file\n'
>>>f.readline()
'contains three lines\n'
>>>f.readline()
''
```

Lastly, the **readlines()** method returns a list of whole lines of the entire file. All these reading methods return empty values when the end of file is reached.

```
>>>f.readlines()
['This is my first file\n','This file\n','contains three lines\n']
```

File Methods

There are various methods available for operating on a file object. Here is the complete list of methods in text mode with a brief description.

Method	Description
close()	Close an open file. It has no effect if the file is already closed.
detach()	Separate the underlying binary buffer from the **TextIOBase** and return it.
fileno()	Return an integer number (file descriptor) of the file.
flush()	Flush the write buffer of the file stream.
isatty()	Return **True** if the file stream is interactive.
read(***n***)	Read at most ***n*** characters form the file. Reads till end of file if it is negative or **None**.
readable()	Returns **True** if the file stream can be read from.
readline(***n***=-1)	Read and return one line from the file. Reads in at most ***n*** bytes if specified.
readlines(***n***=-1)	Read and return a list of lines from the file. Reads in at most ***n*** bytes/characters if specified.
seek(*offset*, *from*=SEEK_SET)	Change the file position to ***offset*** bytes, in reference to ***from***.
seekable()	Returns **True** if the file stream supports random access.

tell()	Returns the current file location.
truncate(*size*=None)	Resize the file stream to ***size*** bytes. If ***size*** is not specified, resize to current location.
writable()	Returns **True** if the file stream can be written to.
write(***s***)	Write string ***s*** to the file and return the number of characters written.
writelines(*lines*)	Write a list of ***lines*** to the file.

Directory

A directory is a collection of files and sub directories. Python has the "**os**" module, which provides us with many useful methods to work with directories and files as well.

Get Current Directory

Here the user can get the current working directory using the **getcwd()** method.

This method returns the current working directory in the form of a string. We can also use the **getcwdb()** method to get it as bytes object.

```
>>>importos
>>>os.getcwd()
'C:\\Program Files\\PyScripter'
>>>os.getcwdb()
b'C:\\Program Files\\PyScripter'
```

The extra backslash implies an escape sequence. The **print()** function will render this properly.

```
>>>print(os.getcwd())
C:\Program Files\PyScripter
```

Changing Directory

The user will be able to change the current working directory using the **chdir()** method.

The new path that we would like to change to, must be supplied as a string to this method. We can use either forward slash (/) or the backward slash (\) to split path elements.

It is safer way to use escape sequence when using the backward slash(\).

```
>>>os.chdir('C:\\Python33')
>>>print(os.getcwd())
C:\Python33
```

List Directories and Files

All files and sub directories are inside a directory we have to use the **listdir()** method.

This method takes in a path and returns a list of sub directories and files in a specified path. If no path is specified, it returns from the current working directory.

```
>>>print(os.getcwd())
C:\Python33
>>>os.listdir()
['DLLs',
'Doc',
'include',
'Lib',
'libs',
'LICENSE.txt',
'NEWS.txt',
'python.exe',
'pythonw.exe',
'README.txt',
'Scripts',
'tcl',
```

```
'Tools']

>>>os.listdir('G:\\')
['$RECYCLE.BIN',
'Videos',
'Audios',
'Photos',
'Others']

>>>print(os.getcwd())
C:\Python33

>>>os.listdir()
['DLLs',
'Doc',
'include',
'Lib',
'libs',
'LICENSE.txt',
'NEWS.txt',
'python.exe',
'pythonw.exe',
'README.txt',
'Scripts',
'tcl',
'Tools']

>>>os.listdir('G:\\')
['$RECYCLE.BIN',
'Videos',
'Audios',
'Photos',
'Others']
```


Making a New Directory

The user wants to make a new directory by the **mkdir()** method.

This method takes in the path of the new directory. If the full path is not specified, the new directory is created in the current working directory.

```
>>>os.mkdir('test')
>>>os.listdir()
['test']
```

Renaming a Directory or a File

The **rename ()** method can be used to rename a directory or a file. The first argument is the old name and the new name must be supplied as the second argument.

```
>>>os.listdir()
['test']
>>>os.rename('test','new_test')
>>>os.listdir()
['new_test']
```

Removing Directory or a File

A file can be removed by using the **remove()** method. Likewise, the **rmdir()** method removes an empty directory.

```
>>>os.listdir()
['new_test','old.txt']
>>>os.remove('old.txt')
>>>os.listdir()
['new_test']
>>>os.rmdir('new_test')
>>>os.listdir()
[]
```

However, note that the **rmdir ()** method can only remove **empty** directories.

If a programmer wants to remove a non-empty directory we can use the **rmtree()** method inside the **shutil** module.

```
>>>os.listdir()
['test']
>>>os.rmdir('test')
Traceback(most recent call last):
...
```

```
OSError:[WinError145]The directory isnotempty:'test'
>>>importshutil
>>>shutil.rmtree('test')
>>>os.listdir()
[]
```

3.7 Errors and Exceptions

Errors

Errors are common in programming. Errors occur due to an improper usage of program statements. Python interpreter raises exceptions when an error occurs.

Syntax errors

These errors are also known as parsing errors. This error will occur when the correct structure of the program is not followed.

```
>>>if a <3
File"<interactive input>", line 1
if a <3
SyntaxError: invalid syntax
```


Python will be able to notice here, that a colon is missing in the “if” statement.

Exceptions

Even if a statement is syntactically correct, it may cause an error during execution. Errors that occur at runtime are called exceptions. Below are some of the exceptions in Python.

FileNotFoundError - File we try to open does not exist
ZeroDivisionError-Dividing a number by zero
ImportError - Module we try to import is not found

Whenever this type of runtime error occurs, Python creates an exception object. If not handled properly, it prints a *trace back* to that error along with some details of what error occurred.

```
>>>1/0
Traceback(most recent call last):
File"<string>", line 301,inruncode
File"<interactive input>", line 1,in<module>
ZeroDivisionError: division by zero

>>>open("emp.txt")
Traceback(most recent call last):
File"<string>", line 301,inruncode
File"<interactive input>", line 1,in<module>
FileNotFoundError:[Errno2]No such file or directory:'emp.txt'
```

Assertions

An assertion is a sanity-check that you can turn on or turn off when you are done with your testing of the program.

The easiest way to think of an assertion is to liken it to a **raise-if** statement. An expression is tested, and if the result comes up False, an exception is raised.
Assertions are carried out by the **assert** statement
Programmers often place assertions at the start of a function to check for a valid input, and after a function call to check for a valid output.

The *assert* Statement

When it encounters an assert statement, Python evaluates the accompanying expression, which is hopefully True. If the expression is False, Python raises an *AssertionError* exception.

Syntax

```
assertExpression[,Arguments]
```

If the assertion fails, Python uses *ArgumentExpression* as the argument for the AssertionError. AssertionError exceptions can be caught and handled like any other exception using the *try-except* statement, but if not handled, they will terminate the program and produce a *traceback*.

Here is a function that converts a temperature from degrees Kelvin to degrees Fahrenheit. Since zero degrees Kelvin is as cold as it gets, the function bails out if it sees a negative temperature.

Example:Test.py

```
defKelvinToFahrenheit(Temperature):
assert (Temperature >= 0),"Colder than absolute zero!"
return ((Temperature-273)*1.8)+32
print (KelvinToFahrenheit(273))
print (int(KelvinToFahrenheit(505.78)))
print (KelvinToFahrenheit(-5))
```

It produces the following result.

```
32.0
451
Traceback (most recent call last):
File "test.py", line 9, in
printKelvinToFahrenheit(-5)
File "test.py", line 4, in KelvinToFahrenheit
assert (Temperature >= 0),"Colder than absolute zero!"
AssertionError: Colder than absolute zero!
```

Exception Handling

Built-in Exceptions

There are lots of built-in exceptions in Python that are raised when corresponding errors occur. We are capable of viewing all the built-in exceptions using the local () built-in functions as follows:

```
>>>locals()['__builtins__']
```

This will return us a dictionary of built-in exceptions, functions and attributes.

Some of the common built-in exceptions in Python along with the cause of error, are given below:

Exception	Cause of Error
AssertionError	Raised when **assert** statement fails.
AttributeError	Raised when attribute assignment or reference fails.
EOFError	Raised when the **input()** functions hits end-of-file condition.
FloatingPointError	Raised when a floating point operation fails.
GeneratorExit	Raise when a generator's **close()** method is called.
ImportError	Raised when the imported module is not found.
IndexError	Raised when index of a sequence is out of range.
KeyError	Raised when a key is not found in a dictionary.
KeyboardInterrupt	Raised when the user hits interrupt key (Ctrl+c or delete).
MemoryError	Raised when an operation runs out of memory.
NameError	Raised when a variable is not found in local or global scope.
NotImplementedError	Raised by abstract methods.
OSError	Raised when system operation causes system related error.
OverflowError	Raised when result of an arithmetic operation is too large to be represented.
ReferenceError	Raised when a weak reference proxy is used to access a garbage collected referent.

RuntimeError	Raised when an error does not fall under any other category.
StopIteration	Raised by **next()** function to indicate that there is no further item to be returned by iterator.
SyntaxError	Raised by parser when syntax error is encountered.
IndentationError	Raised when there is incorrect indentation.
TabError	Raised when indentation consists of inconsistent tabs and spaces.
SystemError	Raised when interpreter detects internal error.
SystemExit	Raised by **sys.exit()** function.
TypeError	Raised when a function or operation is applied to an object of incorrect type.
UnboundLocalError	Raised when a reference is made to a local variable in a function or method, but no value has been bound to that variable.
UnicodeError	Raised when a Unicode-related encoding or decoding error occurs.
UnicodeEncodeError	Raised when a Unicode-related error occurs during encoding.
UnicodeDecodeError	Raised when a Unicode-related error occurs during decoding.
UnicodeTranslateError	Raised when a Unicode-related error occurs during translating.
ValueError	Raised when a function gets argument of correct type but improper value.
ZeroDivisionError	Raised when second operand of division or modulo operation is zero.

We can as well define our own exception in Python. We can handle these built-in and user-defined exceptions by using *try, except* and *finally* statements.

Handling Exception with– *try, except and finally*

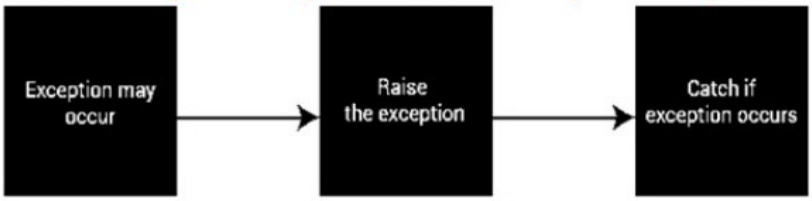


Figure-16-Exception Flow

Built-in exceptions, can force your program to output an error when something in it goes wrong. When these exceptions occur, it causes the current process to stop and passes it to the calling process, until it is handled. Otherwise, our program will crash. If the user is not handled, an error message will appear and our program comes to a sudden, unexpected halt.

Catching Exceptions

Exceptions can be handled using a *try block*.

A critical operation which can raise an exception is placed inside the **try** clause and the code that handles exception is written in **except** clause.

Example:Test.py

```
import sys
randomList = ['a', 0, 2]
for entry in randomList:
try:
print("The entry is", entry)
r = 1/int(entry)
break
except:
print("Oops!",sys.exc_info()[0],"occured.")
print("Next entry.")
print()
print("The reciprocal of",entry,"is",r)
```

Output

```
The entry is
Oops! <class 'ValueError'>occured.
Next entry.

The entry is 0
Oops! <class 'ZeroDivisionError'>occured.
Next entry.
The entry is 2
The reciprocal of 2 is 0.5
```

3.8 Standard Library-I

Python's standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to the system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python.

The Python Standard Library contains a huge number of useful modules and is part of every standard Python installation. Here, we will explore some of the commonly used modules in this library.

Operating System Interface

The ***OS*** module provides many functions for interacting with the operating system:

```
>>>import os
>>>os.getcwd()
>>>os.chdir('/server/accesslogs')
>>>os.system('mkdir today')
```

Be sure to use the import OS style instead of from OS import *. This will keep os.open() from shadowing the built-in ***open()*** function which operates much differently. The built-in ***dir ()*** and ***help ()*** functions are useful a interactive aids for working with large modules like OS:

```
>>>import os
>>>dir(os)
>>>help(os)
```

For daily file and directory management tasks, *the* ***shutil*** module provides a higher level interface that is easier to use:

```
>>>import shutil
>>>shutil.copyfile('data.db','archive.db')
'archive.db'
>>>shutil.move('/build/executables','installdir')
'installer'
```

File Wildcard

The ***glob*** module provides a function for making file lists from directory wildcard searches:

```
>>>import glob
>>>glob.glob('*.py')
['primes.py','randam.py','quote.py']
```

Command Line Arguments

Common utility scripts often need to process command line arguments. These arguments are stored in the ***sys*** module's *argv* attribute as a list. For instance the following output results from running python demo.py one two three at the command line:

```
>>>import sys
>>>print(sys.argv)
['demo.py', 'one.py']
```

The ***getopt*** module processes *sys.argv* using the conventions of the Unix ***getopt()*** function. More powerful and flexible command line processing is provided by the ***argparse*** module.

Error Output Redirection and Program Termination

The ***sys*** module also has attributes for *stdin*, *stdout*, and *stderr*. The latter is useful for emitting warnings and error messages to make them visible even when *stdout* has been redirected:

```
>>>Sys. stderr. writes ('Warning, log file not found starting a new one\n')
Warning, log file not found starting a new one
```

The most direct way to terminate a script is to use ***sys.exit().***

String Pattern Matching

The ***re*** module provides regular expression tools for advanced string processing. For complex matching and manipulation, regular expressions offer succinct, optimised solutions:

```
>>>import re
>>>re.findall(r'\bf[a-z]*', 'Which foot or hand fell fastest')
['foot','fell','fastest']
>>>re.sub(r'(\b[a-z]+) \1', r'\1', 'cat in the the hat')
'cat in the cat'
```

When only simple capabilities are needed, string methods are preferred because they are easier to read and debug:

```
>>>'java  two' .replace('too','2')
'java 2'
```


Mathematics

The ***math*** module gives access to the underlying C library functions for floating point math:

```
>>>import math
>>>math.cos(math.pi /4)
0.70710678118654757
>>>math.log(1024,2)
10.0
```

The ***random*** module provides tools for making random selections:

```
>>>import randam
>>>randam.choice(['apple','pear','banana'])
'apple'
>>>randam.sample(range(100),10)
[30,83,16,4,8,81,41,50,18,33]
>>>randam.randam()
0.1797098769370
>>>Random. randrange (6)
```

The ***statistics*** module calculates basic statistical properties of numeric data:

```
>>>import statistics
>>>data=[2.75,1.75,1.25,0.25,0.5,1.25,3.5]
>>>statistics.mean(data)
1.60714285714285 72
>>>statistics.variance(data)
1.3720238095238095
```

Dates and Times

The ***datetime*** module supplies classes for manipulating dates and times in both simple and complex ways. While date and time arithmetic is supported, the focus of the implementation is on efficient member extraction for output formatting and manipulation. The module also supports objects that are timezone aware.

```
>>>from datatime import date
>>>now =date.today()
>>>now datetime.date(2003,12,2)
>>>now.strftime("%m-%d-%y. %d %b %y is a %A on the %d day of %B.")
>>>birthday=date(1964,7,31)
>>>age=now – birthday
>>>age.days
```

Data Compression

Common data archiving and compression formats are directly supported by modules
including: zlib, gzip, bz2, lzma, zipfile and tarfile.

```
>>>import zlib
>>>s=b'This is the demo for data compression'
>>>len(s)
>>>t=zlib.compress(s)
>>>len(t)
>>>zlib.decompress(t)
>>>zlib.crc32(s)
```

Performance Measurement

Some Python users develop a deep interest in knowing the relative performance of different approaches to the same problem. Python provides a measurement tool that answers those questions immediately. For example, it may be tempting to use the tuple packing and unpacking feature instead of the traditional approach to swapping arguments. The ***timeit*** module quickly demonstrates a modest performance advantage:

```
>>>from timeit import Timer
>>>Timer('t=a; a=b; b=t','a=1; b=2').timeit()
>>>Timer('a,b=b,a', 'a=1; b=2').timeit()
```

In contrast to *timeit's* fine level of granularity, the ***profile*** and ***pstats*** modules provide tools for identifying time critical sections in larger blocks of code.

Standard Library –II

This second library II gives more advanced modules that support professional programming needs. These modules rarely occur in small scripts.

Output Formatting

The ***reprlib*** module provides a version of ***repr()*** customised for abbreviated displays of large or deeply nested containers:

```
>>>import reprlib
>>>reprlib.repr(set('asdfghjklqwertyuiomnbvcxz'))
```

The ***pprint*** module offers more sophisticated control over printing both built-in and user defined objects in a way that is readable by the interpreter. When the result is longer than one line, the "pretty printer" adds line breaks and indentation more clearly reveals data structure:

```
>>>import pprint
>>>t=[[[['black','cyan'],'white',['green','red']],[['magenta','yell
ow'],'blue']]]
>>>pprint.pprint(t,width=30)
```

The ***textwrap*** module formats paragraphs of text to fit a given screen width:

```
>>>import textwrap
>>>doc="""" The wrapping document.""""
>>>print(textwrap.fill(doc,width=40))
```

The ***locale*** module accesses a database of culture specific data formats. The grouping attribute of the locale's format function provides a direct way of formatting numbers with group separators:

```
>>>import locale
>>>locale.setlocale(locale.LC_ALL,'E
>>>locale.setlocale(locale.LC_ALL, 'English_United
States.1252')
>>>conv = locale.localeconv()
>>>x = 1234567.8
>>>locale.format("%d", x, grouping=True)
>>>locale.format_string("%s%.*f",
(conv['currency_symbol'], conv['frac_digits'], x),
grouping=True)
```


Templating

The ***string*** module includes a versatile ***Template*** class with a simplified syntax suitable for editing by end-users. This allows users to customise their applications without having to alter the application. The format uses placeholder names formed by **$** with valid Python identifiers. Surrounding the placeholder with braces allows it to be followed by more alphanumeric letters with no intervening spaces. Writing $$ creates a single escaped $:

```
>>>from string import Template
>>> t = Template('${village}folk send $$10 to $cause.')
>>>t. substitute(village='ChinnaKallery', cause='the ditch
fund')
```

The ***substitute ()*** method raises a ***KeyError*** when a placeholder is not supplied in a dictionary or a keyword argument. For mail-merge style applications, user supplied data may be incomplete and the ***safe substitute ()*** method may be more appropriate — it will leave placeholders unchanged if data is missing:

```
>>>t= Template('Return the $item to $owner.')
>>>d= dict(item='unladen swallow')
>>>t.substitute(d)
>>>t.safe_substitute(d)
```

Template subclasses can specify a custom delimiter. For example, a batch renaming utility for a photo browser may elect to use percent signs for placeholders such as the current date, image sequence number, or file format:

```
>>>import time, os.path
>>>photofiles = ['img_1074.jpg', 'img_1076.jpg',
'img_1077.jpg']
>>>class BatchRename(Template):
    delimiter = '%'
>>>fmt = input('Enter rename style (%d-date %n-seqnum
%f-format):  ')
>>>t = BatchRename(fmt)
>>>date = time.strftime('%d%b%y')
>>>for i, filename in enumerate(photofiles):
    base,ext=os.splitext(filename)
    newname=t.substitute(d=date,n=I,f=ext)
    print('{0}-->{1}'.format(filename,newname))
```

Working with Binary Data Record Layouts

The ***struct*** module provides ***pack()*** and ***unpack()*** functions for working with variable length binary record formats. The following example shows how to loop through header information in a ZIP file without using the ***zipfile*** module. Pack codes "H" and "I" represent two and four byte unsigned numbers respectively. The "<" indicates that they are standard size and in little-endian byte order:

```
import struct
with open('myfile.zip','rb') as f:
    data =f.read()
start =0
for I in range(3):
    start +=14
    fields=struct.unpack('<IIIHH',data[start:start+16])
    crc32,comp_size,uncomp_size,filenamesize,extra_size
=fields

    start +=16
    filename=data[start:start+filenamessize]
    start+=filenamesize
    print(filename,hex(crc32),comp_size,uncomp_size)
    start+=extra_size+comp_size
```

MACHINE LEARNING

Introduction to ML

In 1959 Arthur Samuel, coined the term ML. Arthur Samuel was an American pioneer in the field of computer vision and artificial intelligence. After this the progress on machine learning is increasing day by day in terms of performance, algorithm, computation, accuracy and many more. Machine learning sometimes use as a leader in system design and sometime use as a technology which is handling the algorithm part in the backend of the system.

Some key points defining the introductory part of ML are:

- Learning is the conversion of experience into expertise or knowledge.
- It is one of the most growing branches of computer science.
- It's an algorithm based technology to solve any problem.
- We find how we can transform theoretical ideas, mathematics and algorithm into practical algorithm.
- Its powerfulness is in predicting the future by continuously learning from the experiences that is why it is use in improving many scenarios.
- Automated learning/Machine learning.

Working with ML involves working with different technologies and fields also like:

- Working with data require the data analysis part
- Extracting key information of data requires computation part and algorithm part.
- If the computations are very heavy then basic algorithms gets fail so deep learning results in a big savior.

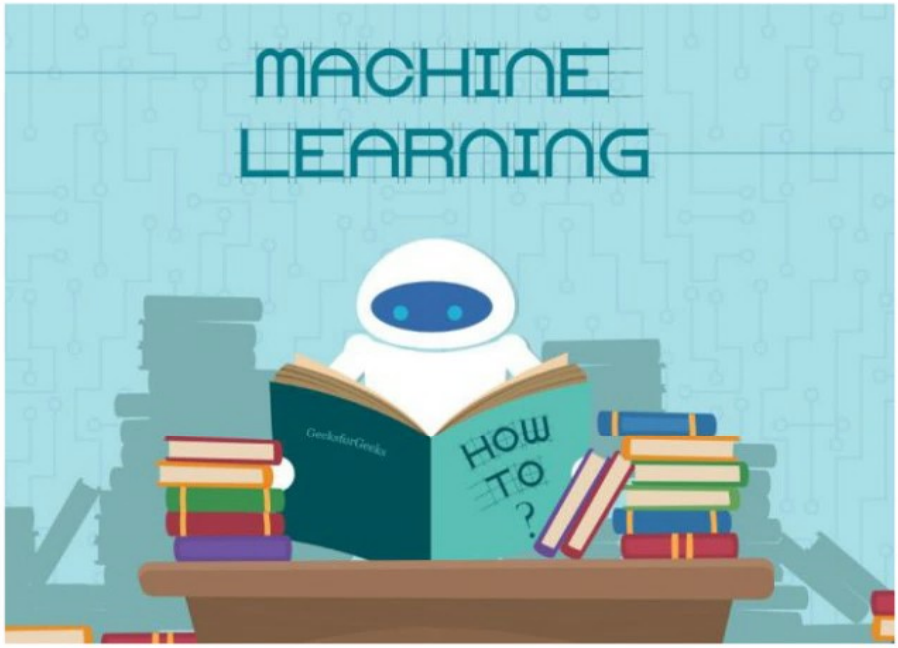
Why ML is required?

- It is possible to solve a number of problem by writing a long size code using any programming language still the reason of using ML is:
  - Programming too complex task
    - Tasks performed by humans: whatever task we do in routine are not that easy to elaborate well to transform it into a program. For example driving a car.
    - Tasks beyond human capacities: contains the tasks which are highly complex because of their dataset.

- Adaptivity
  - programming is limited to rigidity that is if programmed installed then no changes are possible, we know with time the data gets change so machine learning provides the solutions for the same that is if some modification in the input data we will have the future prediction gets change.

What is ML?

ML is a branch of computer science and a subset of artificial intelligence, there are many definition of ML is there which defines the ML.

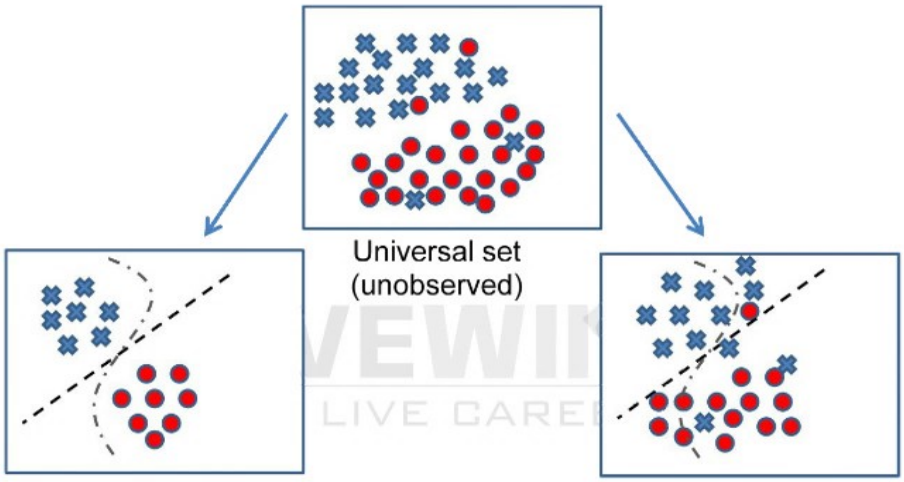


Some of the definitions which define ML are:-

- It is a branch of artificial intelligence, focus on the design and development of algorithm that allow computers to evolve behaviours based on empirical data.
- It is the ability of a machine to automatically learn through data in place of explicit programming.
- Example: transaction on the basis of customer behaviors
- The person who interested in buying a T-shirt can also buy jeans(**www.amazon.in**)

Training and testing set

- In machine learning the dataset is require to divide in major two categories that is training set and testing set.
- Training set is use to train our model. It provides the relationship between independent and dependent variables. So that a machine can self learn on the experience we have provided through train set
- Testing set is use to validate the training set. It also helps in finding the accuracy of overall system. In ML algorithm only independent variable of test set is pass and the result is get which is further compared by the actual set to find out the accuracy.



Training set (observed) Testing set (unobserved)

Over-fitting and Under-fitting

- Under-fitting and over-fitting of data is responsible for bad results. So they are not acceptable.
- Over-fitting is a problem in which system takes input and output relationship at an extent that it considers noise and exceptional cases in the categorization and makes them classify. Over-fitting of data is showing through a graphical model.
- Under-fitting occurs when an algorithm do not fit a problem well enough. This occurs when algorithm show low variance and high bias.

- Appropriate-fitting is an acceptable fitting of dataset. To achieve it we have to set the boundary conditions to remove over-fitting and under-fitting issues.

Types of learning strategy

Machine learning is divided into multi categories:

- Supervised learning: this learning strategy is use for the dataset whose prior information we will have. It means the dataset requires both independent and depended variables.

  For example: we will have a basket of different colour balls and we know their colour. Now to make them extract we have to apply some properties and model like colour size etc. So such problem is categorized in supervised learning.

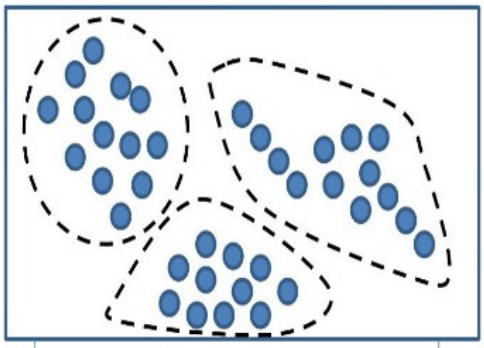
- **Classification:** these categories of algorithm are use for the dataset whose dependent variable is a categorical value.

  For example: On the basis of property of a customer we have to find out whether bank loan is possible to give or not.

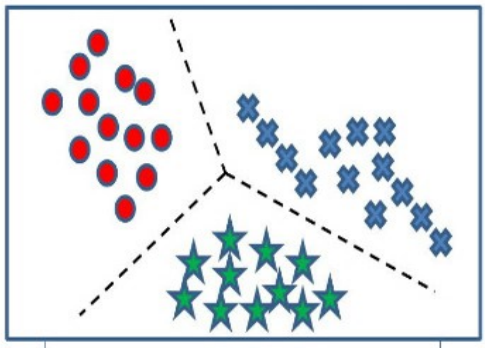
- **Regression:** these categories of algorithm are use for the dataset whose dependent variable is a continuous value.

  For example: On the basis of the properties of a person we have to find out their age.

- Unsupervised learning: this learning strategy is use for the data set whose prior information we were not having that is their dependent variable is missing.



Unsupervised learning



Supervised learning

DATA PREPROCESSING

- Before implementing something on algorithm a lot of changes are require in the data set.
- These changes make the data set suitable for algorithm.
- It also makes the system computationally fast as we standardize our data also.
- Python contain predefined library for data preprocessing that is preprocessing in sklearn
- There is few steps use for data preprocessing.
    - Data collection
    - Converting strings into labels
    - Dealing with missing values
    - Making dummies of labeled data
    - Standardize the data
    - Splitting the data into training and testing set

Data collection:

- It contains collection of data from some file.
- The type of data is supported by pandas is data frame.
- To collect the data from excel or CSV format it is require to use pandas library with xlrd in backend.

❑ Library to import for data collection:

- Import pandas as pd(pd is the alias name given to pandas and it is use to collect data from excel or csv files)

- Functions to use:
  - Read_excel/read_csv: these two functions are use to read the content of excel or csv file.
  - Open: this function is use to open the file according to the permission assigned.
  - Iloc/loc: this is use to extract the content row and column wise.
  - Dataframe: this function convert the content into data frame format

Converting strings into labels

- String values in the dataset are not possible to use further for computation. So it is require converting them into an integer label. For doing it there will be a library in python that is LabelEncoder
- Library to import for data labelling:
  - from sklearn.preprocessing import LabelEncoder
- Functions to use:
  - Fit_transform: this function is use to pass the data to label and also it makes the label of the string attributes

Dealing with missing values

- Dataset may contain many missing places where data is not inputted. These missing places create an error in the algorithm so it is require making them fill or remove the entire tuple.
- For filling the missing value Imputer library is use.
- Library to import for filling value:
  - From sklearn.preprocessing import Imputer

- For dropping the row of missing value:
    - From pandas.DataFrame import dropna

Making dummies of labelled data

- Labels are assigned to the string data in alphabetic order. Now it may be possible that the data whose labels is let 0 can have the highest priority and the data having 4 have the least one. When algorithm works it will take the weight age of 4 as more because of maths. So overall accuracy of the system get disturb. So making the content distinguishable and not transform into wrong weight dummy variable are made.
- Dummy variables increase the attributes as it convert a single attribute into multi attribute depend on the number of categories it can have.
- The data whose label is not made such data is not selected for making dummies.
- Library to import for dummies:
    - from sklearn.preprocessing import OneHotEncoder
- Functions to use:
    - Fit_transform: this function is use to pass the labelled data to create their dummies.
    - Categorical_features: this is use to set the attributes who dummies are require to make.
    - Toarray: the return type of dummies function is an object so it is require converting them into array so this function is use.

Standardize data

- The data set contain many small and very large values.
- These values increases the computation cost of overall system.
- Standardize process reduces them and put them from few negative to few positive values.
- Only input variables are required to standardize.
- It transforms the data in such a way that the distribution contains mean value 0 and standard deviation 1.

- Library to import for standardize:
    - from sklearn.preprocessing import StandardScaler
- Functions to use:
    - Fit_transform: this function is use to pass the dataset and make them standardize

Splitting the data into training and testing set

- Splitting of data set contain diving the dataset into small modules.
- Mostly the dataset is split into training and testing set.
- Training set if use to train our system. This set tells the system that how input is depending on output. After this the system gets ready for predicting for future data.
- Testing set is use to validate the training set. This set is use just to check that how good system understand the relationship between input and output.
- Library to import for splitting dataset:
    - from sklearn.model_selection import train_test_split
- Functions to use:
    - Test_size: this function is use to set the test size of data

Problem:

The data set we are using for pre-processing is a dataset of salary of a person by considering many properties like their age, their credit score etc. Some of the entries of data set is shown below

Age	Workclass	fnlwgt	education	educ	status	occupation	relationship	race	sex	capital_gai	capital_los	hoursperw	native country	salary exce
39	State-gov	77516	Bachelors	13	Never-married	Adm-cleric	Not-in-family	White	Male	2174	0	40	United-States	<=50K
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-mana	Husband	White	Male	0	0	13	United-States	<=50K
38	Private	215646	HS-grad	9	Divorced	Handlers-c	Not-in-family	White	Male	0	0	40	United-States	<=50K
53	Private	234721	11th	7	Married-civ-spouse	Handlers-c	Husband	Black	Male	0	0	40	United-States	<=50K
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-speci	Wife	Black	Female	0	0	40	Cuba	<=50K
37	Private	284582	Masters	14	Married-civ-spouse	Exec-mana	Wife	White	Female	0	0	40	United-States	<=50K
49	Private	160187	9th	5	Married-spouse-absen	Other-serv	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K
52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-mana	Husband	White	Male	0	0	45	United-States	>50K
31	Private	45781	Masters	14	Never-married	Prof-speci	Not-in-family	White	Female	14084	0	50	United-States	>50K
42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-mana	Husband	White	Male	5178	0	40	United-States	>50K
37	Private	280464	Some-colle	10	Married-civ-spouse	Exec-mana	Husband	Black	Male	0	0	80	United-States	>50K
30	State-gov	141297	Bachelors	13	Married-civ-spouse	Prof-speci	Husband	Asian-Pac-	Male	0	0	40	India	>50K
23	Private	122272	Bachelors	13	Never-married	Adm-cleric	Own-child	White	Female	0	0	30	United-States	<=50K
32	Private	205019	Assoc-acdm	12	Never-married	Sales	Not-in-family	Black	Male	0	0	50	United-States	<=50K
40	Private	121772	Assoc-voc	11	Married-civ-spouse	Craft-repa	Husband	Asian-Pac-	Male	0	0	40	?	>50K
34	Private	245487	7th-8th	4	Married-civ-spouse	Transport-	Husband	Amer-Indi	Male	0	0	45	Mexico	<=50K
25	Self-emp-not-inc	176756	HS-grad	9	Never-married	Farming-fi	Own-child	White	Male	0	0	35	United-States	<=50K
32	Private	186824	HS-grad	9	Never-married	Machine-c	Unmarried	White	Male	0	0	40	United-States	<=50K
38	Private	28887	11th	7	Married-civ-spouse	Sales	Husband	White	Male	0	0	50	United-States	<=50K
43	Self-emp-not-inc	292175	Masters	14	Divorced	Exec-mana	Unmarried	White	Female	0	0	45	United-States	>50K
40	Private	193524	Doctorate	16	Married-civ-spouse	Prof-speci	Husband	White	Male	0	0	60	United-States	>50K
54	Private	302146	HS-grad	9	Separated	Other-serv	Unmarried	Black	Female	0	0	20	United-States	<=50K

Programming steps:

- Data collection:
  - import pandas as pd
  - obj=pd.read_excel(open('adult_salary_dataset.xlsx','rb'))
  - x=obj.iloc[:,:-1]
  - y=obj.iloc[:,-1]
- The received data will be of data frame format. So require to convert them into array using numpy library
  - import numpy as np
  - x=np.array(x)
  - y=np.array(y)
- Converting strings into labels
  - from sklearn.preprocessing import LabelEncoder
  - lb=LabelEncoder()
  - x[:,1]=lb.fit_transform(x[:,1])
  - x[:,3]=lb.fit_transform(x[:,3])
  - x[:,5]=lb.fit_transform(x[:,5])
  - x[:,6]=lb.fit_transform(x[:,6])
  - x[:,7]=lb.fit_transform(x[:,7])
  - x[:,8]=lb.fit_transform(x[:,8])
  - x[:,9]=lb.fit_transform(x[:,9])
  - x[:,13]=lb.fit_transform(x[:,13])
  - y=lb.fit_transform(y)
- Dealing with missing values
  - from sklearn.preprocessing import Imputer
  - im=Imputer()
  - x=im.fit_transform(x)
- Making dummies of labeled data
  - from sklearn.preprocessing import OneHotEncoder
  - one=OneHotEncoder(categorical_features=[1,3,5,6,7,8,9,13])
  - x=one.fit_transform(x).toarray()

- Standardize the data
  - from sklearn.preprocessing import StandardScaler

- sc=StandardScaler()
- x=sc.fit_transform(x)

- Splitting the data into training and testing set
  - from sklearn.model_selection import train_test_split
  - xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)

Results:

These are the results of pre-processing:

Xtrain:

```
array([[-0.24382934, -0.17456198, -0.26424389, ..., -0.14583994,
        -0.2155646 , -0.03347295],
       [-0.24382934, -0.17456198, -0.26424389, ..., -0.14583994,
        -0.2155646 ,  0.77959971],
       [ 4.10122919, -0.17456198, -0.26424389, ..., -0.14583994,
        -0.2155646 , -0.03347295],
       ...,
       [-0.24382934, -0.17456198, -0.26424389, ..., -0.14583994,
        -0.2155646 , -0.03347295],
       [-0.24382934, -0.17456198, -0.26424389, ..., -0.14583994,
        -0.2155646 , -2.39138367],
       [-0.24382934, -0.17456198,  3.78438274, ..., -0.14583994,
        -0.2155646 , -0.03347295]])
```

Ytrain:

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

Xtest:

```
array([[-0.24382934, -0.17456198, -0.26424389, ..., -0.14583994,
        -0.2155646 , -0.03347295],
       [-0.24382934, -0.17456198, -0.26424389, ..., -0.14583994,
```

-0.2155646 , -0.84654561],
[-0.24382934, -0.17456198, -0.26424389, ..., -0.14583994,
-0.2155646 , -0.03347295],
...,
[-0.24382934, -0.17456198, 3.78438274, ..., 0.84381276,
-0.2155646 , 0.77959971],
[-0.24382934, -0.17456198, -0.26424389, ..., 1.85530974,
-0.2155646 , -0.03347295],
[-0.24382934, -0.17456198, -0.26424389, ..., -0.14583994,
-0.2155646 , -0.35870202]])

Ytest:

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

This processed data is suitable to implement through algorithm.

MACHINE LEARNING ALGORITHM

Supervised Algorithm

Supervised learning algorithm: Before applying any learning strategy it is require to doing data pre-processing to converting our data into an acceptable format by an algorithm. The pre-processing part we have already covered in the last chapter. So we are assuming that the data we are having is already pre-processed and ready to implement on algorithm.

Classification:

- Classification: There are many algorithms in case of supervised learning but the algorithm we are using is Gaussian and Multinomial Naïve bayes.
- Gaussian Naïve Bayes is use when we have continuous data in features that is their independent variable is a continuous value.
- Multinomial Naïve Bayes is use when we have discrete data in features that is their independent variable is a discrete value.
- Python contain a predefined library of machine learning in which all algorithms are predefined. So we have to import sklearn.naive_bayes and use the algorithm.

Bayesian Classifiers: Understanding probability

The probability of an event is estimated from the observed data by dividing the number of trials in which the event occurred by the total number of trials

For instance, if it rained 3 out of 10 days with similar conditions as today, the probability of rain today can be estimated as 3 / 10 = 0.30 or 30 percent.

Similarly, if 10 out of 50 prior email messages were spam, then the probability of an incoming message being spam can be estimated as 10 / 50 = 0.20 or 20 percent.

For example, given the value *P(spam) = 0.20, we can calculate P(ham) = 1 - 0.20 = 0.80*

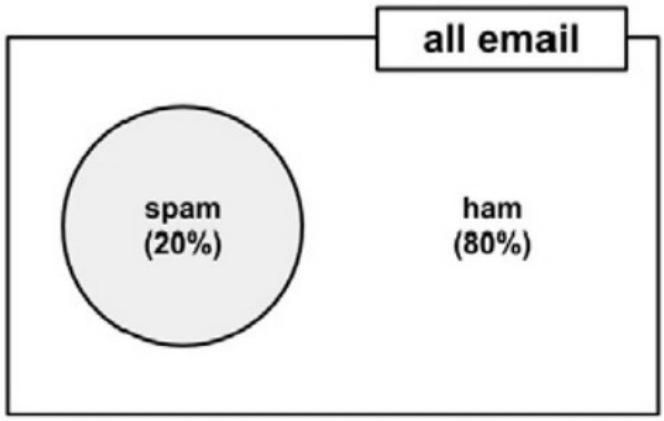
Note: The probability of all the possible outcomes of a trial must always sum to 1

Because an event cannot simultaneously happen and not happen, an event is always mutually exclusive and exhaustive with its complement

The complement of event A is typically denoted Ac or A'.

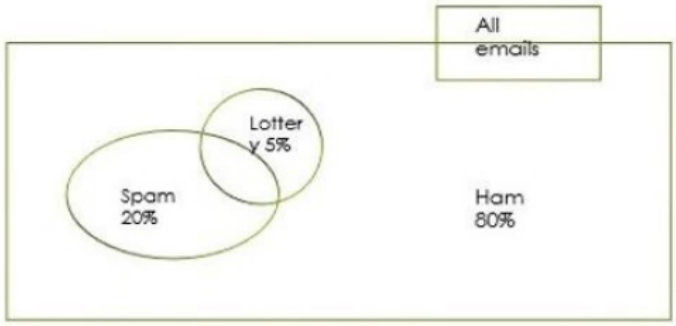
Additionally, the shorthand notation P(A) can be used to denote the probability of event A not occurring, as in P(spam) = 0.80.

This notation is equivalent to P(Ac).



Understanding joint probability

Often, we are interested in monitoring several non-mutually exclusive events for the same trial.



Estimate the probability that both P(spam) and P (Spam) occur, which can be written as P(spam ∩ Lottery). The notation A II B refers to the event in which both A and B occur.

Calculating P(spam ∩ Lottery) depends on the joint probability of the two events or how the probability of one event is related to the probability of the other.

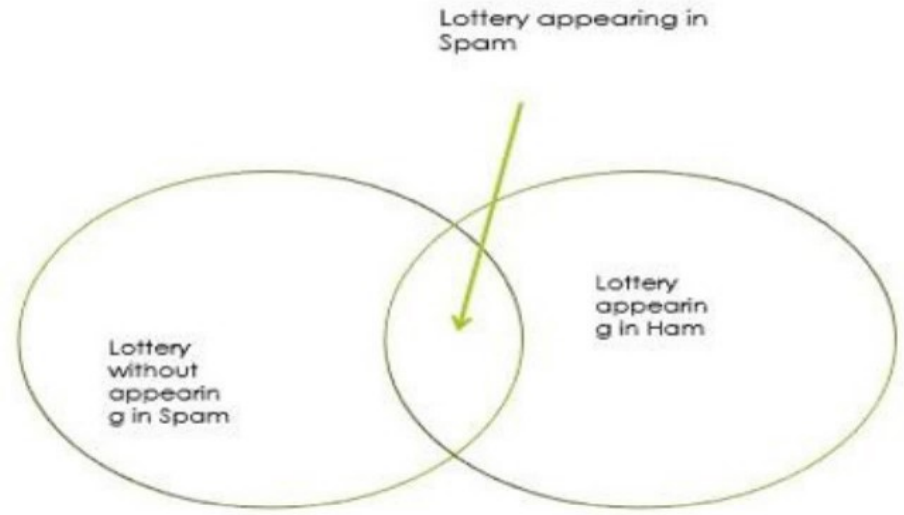
If the two events are totally unrelated, they are called independent events

If P (spam) and P (Lottery) were independent, we could easily calculate P (spam ∩ Lottery), the probability of both events happening at the same time.

Because 20 percent of all the messages are spam, and 5 percent of all the e-mails contain the word Lottery, we could assume that 1% of all messages are spam with the term Lottery.

More generally, for independent events A and B, the probability of both happening can be expressed as P (A ∩ B) = P (A) * P (B).

0.05 * 0.20 = 0.01



Bayes Rule

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

- **Bayes Rule: The most important Equation in ML!!**

$$P(Class|Data) = \frac{P(Class)\,P(Data|Class)}{P(Data)}$$

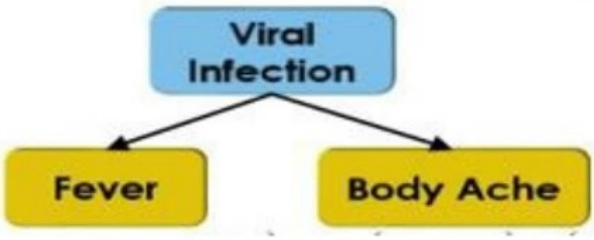
Class Prior — Data Likelihood given Class

Posterior Probability (Probability of class AFTER seeing the data)

Data Prior (Marginal)

Naïve Bayes Classifier: Conditional Independence

$$P(Fever, BodyAche|Viral) = P(Fever|Viral)P(BodyAche|Viral)$$



$$P(Fever, BodyAche) \neq P(Fever)P(BodyAche)$$

Simple Independence between two variables:

$$P(X_1, X_2) = P(X_1)P(X_2)$$

Class Conditional Independence assumption:

$$P(X_1, X_2) \neq P(X_1)P(X_2)$$

$$P(X_1, X_2|C) = P(X_1|C)P(X_2|C)$$

Conditional Independence among variables given Classes:

$$P(C|X_1,X_2,\ldots,X_D)=\frac{P(C)P(X_1,X_2,\ldots,X_D|C)}{\sum_{C'}P(X_1,X_2,\ldots,X_D|C')}=\frac{P(C)\prod_{d=1}^{D}P(X_d|C)}{\sum_{C'}\prod_{d=1}^{D}P(X_d|C')}$$

- Simplifying assumption
- Baseline model especially when a large number of features are involved.
- Taking log and ignoring denominator:

$$\log\left(P(C|X_1,X_2,\ldots,X_D)\right)\propto\log\left(P(C)\right)+\sum_{d=1}^{D}\log\left(P(X_d|C)\right)$$

Naïve Bayes Classifier for Categorical Valued Variables

Let's Naïve Bayes!

$$\log\left(P(C|X_1,X_2,\ldots,X_D)\right)\propto\log\left(P(C)\right)+\sum_{d=1}^{D}\log\left(P(X_d|C)\right)$$

Class Prior Parameters:

$P(\text{Like}=\text{Y})=???$

$P(\text{Like}=\text{N})=???$

Class Conditional Likelihoods

$P(\text{Color}=\text{Red}|\text{Like}=\text{Y})=????$

$P(\text{Color}=\text{Red}|\text{Like}=\text{N})=????$

...

$P(\text{Shape}=\text{Triangle}|\text{Like}=\text{N})=????$

#EXMPLS	COLOR	SHAPE	LIKE
20	Red	Square	Y
10	Red	Circle	Y
10	Red	Triangle	N
10	Green	Square	N
5	Green	Circle	Y
5	Green	Triangle	N
10	Blue	Square	N
10	Blue	Circle	N
20	Blue	Triangle	Y

Parameter Estimation

$$\log\left(P(C|X_1,X_2,\ldots,X_D)\right)\propto\log\left(P(C)\right)+\sum_{d=1}^{D}\log\left(P(X_d|C)\right)$$

- What / How many Parameters?
- Class Priors: $P(c)=\frac{N(c)}{N}\approx\frac{N(c)+\lambda}{N(c)+\lambda\times|\mathbf{C}|}$
- Conditional Probabilities:

$$X_d=\left\{v_1^{(d)},v_2^{(d)},\ldots,v_{M_d}^{(d)}\right\}$$

$$P\left(v_m^{(d)}|c\right)=\frac{N\left(v_m^{(d)},c\right)}{N(c)}\approx\frac{N\left(v_m^{(d)},c\right)+\lambda}{N(c)+M_d\lambda}$$

Problem

For getting more clarity with the Naive bayes let me take a problem to solve:

Dataset to choose: IRIS plant dataset. It contains 4 columns in the input i.e sepal length/sepal width/petal length/petal width. So according to these we have to define the category of plant.

Sepal_length	Sepal_width	petal_length	petal_width	plant
5.1	3.5	1.4	0.2	Iris-setosa
4.9		1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
5.4	3.7	1.5	0.2	Iris-setosa
4.8	3.4	1.6	0.2	Iris-setosa
4.8	3	1.4	0.1	Iris-setosa
4.3	3	1.1	0.1	Iris-setosa
5.8	4	1.2	0.2	Iris-setosa
5.7	4.4	1.5	0.4	Iris-setosa
5.4	3.9	1.3	0.4	Iris-setosa
5.1	3.5	1.4	0.3	Iris-setosa
5.7	3.8	1.7	0.3	Iris-setosa
5.1	3.8	1.5	0.3	Iris-setosa
5.4	3.4	1.7	0.2	Iris-setosa

Importing libraries using import keyword. Pandas is a library use to collect data from excel sheet. Data is collected using read_excel and open method. Using iloc the content of received data is split into input and output variables.

```
import pandas as pd

dat=pd.read_excel(open('irisdataset.xlsx','rb'))

x=dat.iloc[:,:-1]

y=dat.iloc[:,-1]
```

The dataset contain a missing value so imputer is use to fill the missing value. Imputer object is created first and then using fit_transform the data is inputted on it.

Note: For imputer only non integer values are acceptable.

```
from sklearn.preprocessing import Imputer
im=Imputer()
x=im.fit_transform(x)
```

The data set is requires to split into training and testing set. The training set is further use to train our system and testing set is use to validate our system. Here the train size is taken as 80% of complete data.

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
```

It is better to do normalization of data to reduce the computation cost of the system. Here minmaxscaler is used as this scaled the data in the range of 0 and 1. It doesn't transform the data into negative values while doing standardization. For minmaxscaler it is require to made their object and using fit_transform pass the input variables

```
from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler()
xtrain=mms.fit_transform(xtrain)
xtest=mms.fit_transform(xtest)
```

Importing GaussianNB, it is a predefined library of machine learning. Gnb is the name of object created. Using fit method the train data is fitted and then using predict method the system is predicted result as per the training.

```
from sklearn.naive_bayes import GaussianNB
gnb=GaussianNB()
```

gnb.fit(xtrain,ytrain)

ypred=gnb.predict(xtest)

Now to find out the accuracy and accuracy matrix import confusion_matrix and accuracy_score. Finally print the accuracy got from GaussianNB.

from sklearn.metrics import confusion_matrix

cm=confusion_matrix(ytest,ypred)

from sklearn.metrics import accuracy_score

acc=accuracy_score(ytest,ypred)

print("accuracy given by Gaussian is:-",acc)

Importing MultinomialNB, it is a predefined library of machine learning. mnb is the name of object created. Using fit method the train data is fitted and then using predict method the system is predicted result as per the training.

from sklearn.naive_bayes import MultinomialNB

mnb=MultinomialNB()

mnb.fit(xtrain,ytrain)

ypredm=mnb.predict(xtest)

Now to find out the accuracy and accuracy matrix import confusion_matrix and accuracy_score. Finally print the accuracy got from MultinomialNB.

from sklearn.metrics import confusion_matrix

cmm=confusion_matrix(ytest,ypredm)

from sklearn.metrics import accuracy_score

accm=accuracy_score(ytest,ypredm)

print("accuracy given by MultiNomial is:-",accm)

Result:

accuracy given by Gaussian is:- 0.8666666666666667

accuracy given by MultiNomial is:- 0.6666666666666666

Accuracy of GaussianNB is more compared to MultiNomial because of the dataset independent variables. The independent variables of dataset are continuous values.

Regression:

- Regression: There are many algorithms in case of supervised learning but the algorithm we are using is Linear Regression.
- Linear regression can be applied on the data set when the relation between independent and dependent variable is linear.
- Python contain a predefined library of machine learning in which all algorithms are predefined. So we have to import sklearn.linear_model and use the algorithm.

Problem:

To get more clarity with regression algorithm let we take a regression data set of stock index price containing 4 independent variable and one dependent variable as shown below.

Year	Month	Interest_Rate	Unemployment_Rate	Stock_Index_Price
2017	12	2.75	5.3	1464
2017	11	2.5	5.3	1394
2017	10	2.5	5.3	1357
2017	9	2.5	5.3	1293
2017	8	2.5	5.4	1256
2017	7	2.5	5.6	1254
2017	6	2.5	5.5	1234
2017	5	2.25	5.5	1195
2017	4	2.25	5.5	1159
2017	3	2.25	5.6	1167
2017	2	2	5.7	1130
2017	1	2	5.9	1075
2016	12	2	6	1047
2016	11	1.75	5.9	965
2016	10	1.75	5.8	943
2016	9	1.75	6.1	958
2016	8	1.75	6.2	971
2016	7	1.75	6.1	949
2016	6	1.75	6.1	884

Importing libraries using import keyword. Pandas is a library use to collect data from excel sheet. Data is collected using read_excel and open method. Collected data is further converted into dataframe format using pandas dataframe. Matplotlib is a library imported for plotting support. Using iloc the content of received data is split into input and output variables.

```
import pandas as pd

from pandas import DataFrame

import matplotlib.pyplot as plt

Stock_Market = pd.read_excel(open('multireg.xlsx','rb'))

dff =
DataFrame(Stock_Market,columns=['Year','Month','Interest_Rate','Unemployment_Rate','Stock_Index_Price'])
```

Plotting the input variable and output variables to find out whether their relation is linearly defined or not. If their relation is not linearly defined then the dataset is not relevant for linear algorithm.

The linear plots are shown below:

plt.scatter(dff['Interest_Rate'], dff['Stock_Index_Price'], color='green')

plt.title('Stock Index Price Vs Interest Rate')

plt.xlabel('Interest Rate')

plt.ylabel('Stock Index Price')

plt.show()

plt.scatter(dff['Unemployment_Rate'], dff['Stock_Index_Price'], color='blue')

plt.title('Stock Index Price Vs Unemployment Rate')

plt.xlabel('Unemployment Rate')

plt.ylabel('Stock Index Price')

plt.show()

Now split the data into input and output variables.

X = dff[['Interest_Rate','Unemployment_Rate']]

Y = dff['Stock_Index_Price']

The data set is requires to split into training and testing set. The training set is further use to train our system and testing set is use to validate our system. Here the train size is taken as 80% of complete data.

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(X,Y,test_size=0.2)
```

Import library for linear regression. Create an object of linear regression and fit train set of input and output. Predict the values by inserting the test set of input.

```
from sklearn import linear_model
regr = linear_model.LinearRegression()
regr.fit(xtrain, ytrain)
ypred=regr.predict(xtest)
print ('Predicted Stock Index Price: \n',ypred)
```


The resultant data we will have is of regression type of we cannot make confusion matrix, in place of it we can plot their result to make them compare with actual and predicted scenario. For making them plot it is require some values for X-axis so linspace method is use from numpy.

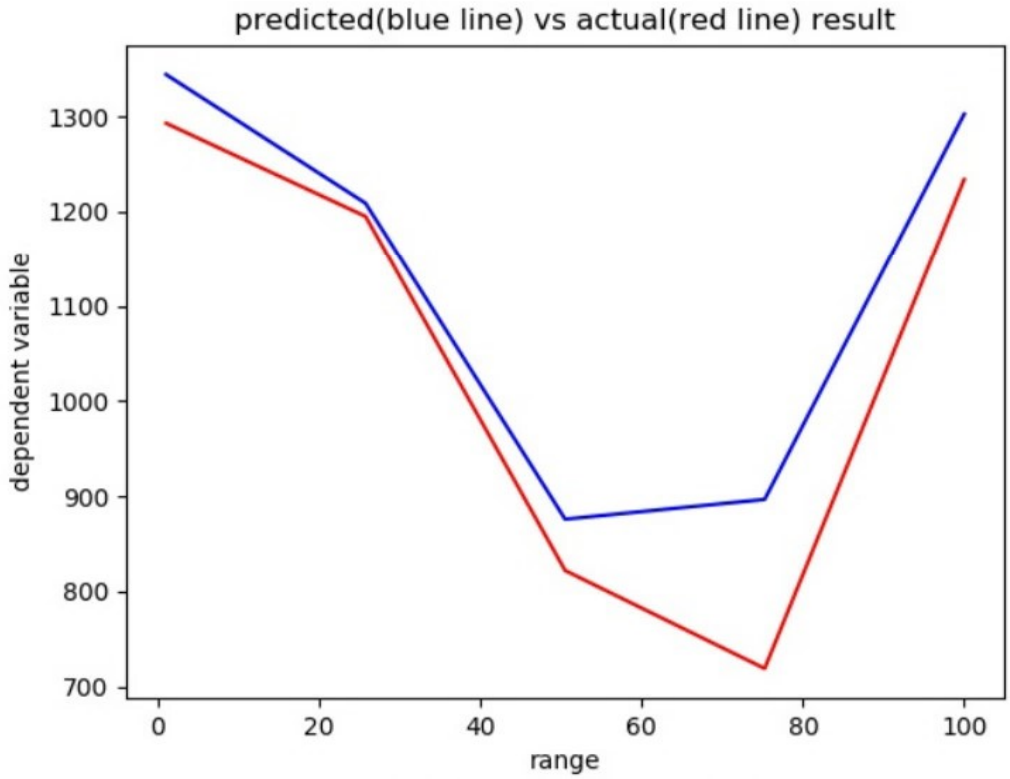
```
a=np.linspace(1,100,len(ytest))
plt.title('predicted(blue line) vs actual(red line) result')
plt.xlabel('range')
plt.ylabel('dependent variable')
plt.plot(a,ytest,color='red')
plt.plot(a,ypred,color='blue')
plt.show()
```

Result:

Predicted Stock Index Price:

[882.4461377 882.4461377 1330.19582972 1015.61700963
1330.19582972]

Plot with predicted and actual result:



This plot is indicating how the predicted and actual outputs are related to each other. It means the algorithm is not completely matching them but still the trends are relevant and set up some accepted accuracy.

OPEN – CV

OPEN-CV

Introduction

This is a computer vision library use for setting the vision power of a machine. Through this library machine is capable of looking out something through camera. Open-CV supports pixel processing, image processing, video processing, etc.

History and basics of Open-CV:

- OPEN-CV was started by Gary Bradsky at Intel in 1999, in 2000 first version of OPEV-CV came in market.
- OPEN-CV is first use in 2005 on Stanley, the vehicle that won the 2005 DARPA grand challenge.
- OPEN-CV support a wide variety of programming language such as c++, java, python etc. and available on different platform like linux, windows and mac etc.
- OPEN-CV-python is python API for OPEN-CV use for solving computer vision problems.
- OPEN-CV python makes use of numpy, which is a highly optimized library for numerical operations.

Library setup

- Open-CV library is required to download through command prompt.
    - For anaconda write: conda/pip install opencv-contrib-python
    - For python write: pip install opencv-contrib-python in cmd prompt.
    - Download cmake using pip/conda install cmake
- To import library in python write. Import cv2

Modules in OPEN-CV

- **Image**
    - Image is the collection of pixels placed in the matrix form.
    - Image quality depends on the row and column of a mtrix.

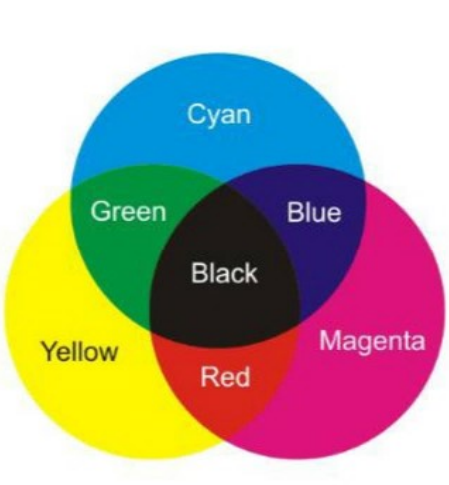
    There are two types of images:

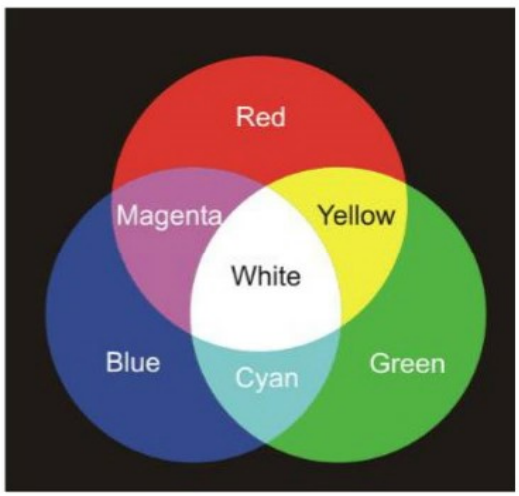
    1) Grayscale
    2) Color Image

Gray scale: It is an image containing two colors only that is black and white. The pixel value of color is vary from 0 to 255. The values which are near to zero are dark pixels, and the values which are near to 255 are bright pixels. So according to this the brightness and darkness of an image is defined. So it mean if we are having a image then just making its pixel value increment and decrement we can set the brightness and darkness. Grayscale image is represented by 2D matrix. For example: 480x640, 1200x1280

Color image: It is an image containing three frames that is red, blue and green. Color image is represented by 3D matrix. For example 480x640x3, 1200x128x3.These three colors are combined responsible for creating a color image as shown in figure.

Creation of image color is shown by using the venn diagram. The venn diagram is showing that how the combination of different colors makes a new color.





- Image processing in Open-CV utilize some image functions.

Functions in image processing:

Imread: this functions is use to read the content of an image. It read the content in the format of a matrix. Two arguments are require to pass in it. The first argument contain the name of the image file. The second argument will be an integer value with value zero or one. Zero for black and white image and one for colour image.

For example: variable=cv2.imread('image path',type)

Imshow: this function is use to show the image read from imread. It converts the readied pixel into an image. We have to pass two argument in it first is the name of the image window and second argument is the image variable name.

For example: cv2.imshow('window name', variable of image)

Imwrite: this function writes the pixel back into the image. We have to pass two arguments in it first is the name of the image in which we have to make it save and second is the image variable name.

For example: cv2.imwrite('image save name', variable of image)

WaitKey: this function is use to hold the screen of image display and video till some defined time completes or some key will get press from pc.

For example: variable=cv2.waitKey(time in msec)

If the value of time taken in waitkey is zero it means the system will wait for some key press from keyboard, otherwise other number will represent the time in milli sec.

DestroyAllWindows: this function is use to destroy all opened windows associated with Open-CV. This will be required to make them destroy otherwise the process gets hang.

For example: cv2.destroyAllWindows()

Ord: this method is use to convert the variable value into their equivalent ASCII value. This is require in Open-CV because the waitKey method returns value of a key pressed in ASCII.

- **Video:**
  - Video is the collection of images placed in a back to back format.
  - Video have more significance in AI as making intelligent system continuous monitoring is require.
  - OPEN-CV contains a number of functions for video processing.
  - The quality of a video depends on the number of images use in a sec and pixels of an image.
  - Image processing in Open-CV utilize some image functions.

    Read: this function is use to read the frames from a video. No arguments are required in this function but we got two values in the return. The first value defines the current status of source of video that is it is working or not, if true received then video is on and if false received it mean video is not on.

    For example: StatusVariable, frameVariable=videoObject.read()

    Write: this function is use to write the frames into a video. One argument is require to pass that is frame of images.

    For example: videoObject.write(frame of image)

    VideoWriter: this function is use to make the video. In video writer we have to pass multi argument like: output video, four code conversion, frames per second, matrix composition.

    For example: videoOutput=cv2.videoWriter(output video, codec, frames per second, matrix pixel)

VideoWriter_fourcc: this function is use to set the four code conversion while working with video. Fourcc reflect the encoding process in case of video. Some fourcc are XVID,MPEG etc.
For example: fourCC
object=cv2.videoWriter_fourcc(*'XVID')
VideoCapture: this function is use to capture the content of video from a source. There are frequently use two sources like webcam and some already prepared video.
For example: objectOfVideo=cv2.videoCapture(source Id)
isOpened: this function is use to check whether our source is opened or not. If the source is not open it returns false, otherwise it returns true.
Release: this function is use to release the source which is utilized for a video. It is always mandatory to release the source otherwise the source gets busy and we didn't make it utilize and a restart will be require.

Applications:

- **Pixel processing & transformation**
  - Let we take an example in which we extract the pixels of an image and then will use them to set the brightness.
  - Library to import
    Import cv2
  - Read content of an image
    Img=cv2.imread('image.jpg',0)
  - Apply loops to increase the pixel of an image by 10 value

```
For i in range(len(img)):
    For j in range(len(img[0][:])):
        Img[i][j]=Img[i][j]+10
```

  - Convert the incremented pixel into an image again

    cv2.imwrite('newImg.jpg',img)

NLP

Installing NLP library NLTK (natural language tool kit)

Install NLTK with Python 2.x using:

```
pip install nltk
```

Install NLTK with Python 3.x using:

```
pip3 install nltk
```

Installation is not complete after these commands. Open python and type:


```
import nltk
nltk.download()
```

Natural language processing (NLP) and text mining:-

Agenda of it:

Evolution of human language

What is text mining?

Text mining and natural language processing?

Application of NLP.

NLP components and demo.

Evolution of human language:

Success among us is because we have capability to communicate using a language.

We have alphabets whose combination makes a word and combination of word makes sentences.

Set of rules use to makes the sentences is **grammer**.

21st Centuary:

If we go with industrial estimates we have only 20% structured data. Rest data is unstructured. We can extract information from structured data only

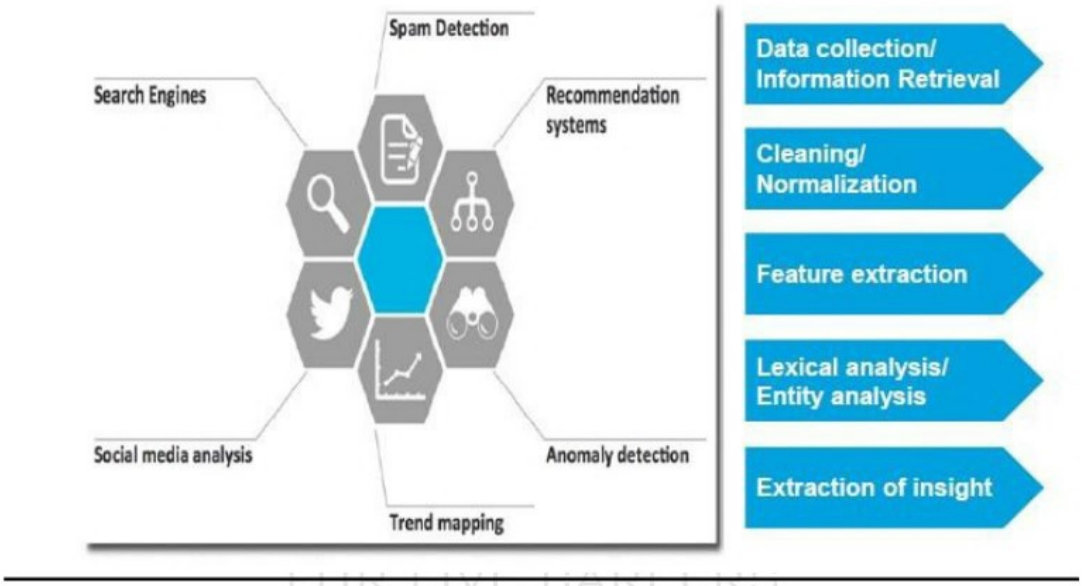
This data may generate by:

We speak, Fb,wssp,text msg, camera images etc

Text mining: it is the process of extracting important information from a natural text. It is unstructured, amorphous and difficult to deal with algorithmically.

Goal: turn the text into data for analysis using natural language processing.

NLP: this is the part of artificial intelligence and computer science which deals with human language.



Applications of NLP:

1)Sentimental analysis

2)chatbot(chatting with a robot)

3)speech recognition

4)machine translation

Application of NLP and text mining:

1)Spell checking

2)keyword search

3)Information extraction(from website or document)

4)Advertising machine(recommendation of adds as per history)

Component of NLP:

1)NLU(natural language understanding)

2)NLG(natural language generation)

3)NLU:

Matching input into useful representation

Analyzing different aspects of the language

4)NLG:

Text planning

Sentence planning

Text realization

Problems in understanding any language:

Ambiguity: Lexical ambiguity, Syntactic ambiguity and referential ambiguity

Lexical: possibility of two or more meaning of a word(some time also known as syntactic ambiguity).

Eg: she is looking for a match

Syntactic: two or more meaning of a sentence.

Eg: The chicken is ready to eat

Referential: referencing something as a pronoun.

Eg: the boy told his father the theft. **He** was very upset

Working with a text few features are required to be known:-

1. Tokenization:steps of breaking string into tokens or small modules.

   Steps

   - Break the sentence into words
   - Understand the importance of each of the words with respect to the sentence
   - Produce a structured description on an input sentence

To do this import some of the files:

```
Import os
Import nltk
```

Document

Each row is called as a 'Document' & even an empty row is considered as a ‘Document’

Corpus

Collection of all these documents is called as 'Corpus'

```
Import nltk.corpus
```

Download data using

```
Nltk.download(‘gutenberg’)
```

To check this file writes:

```
Os.listdir(nltk.data.find(‘corpora’))
#this will return the file
```

To check the fields associated with the file write:

```
Print(nltk.corpus.gutenberg.fileids())
```

To check the words associated in the files in Gutenberg:

```
Variable=nltk.corpus.gutenberg.words(‘file
name’)
```

Printing it using loop:

```
For word in variable[0:100]:
    Print(word,sep=” ”,end=’ ‘)
```

To make tokenize of a word write

```
from nltk.tokenize import
word_tokenize as wt
```

#need to download a module for it i.e nltk.download(‘punkt’)

```
variable=word_tokenize("string
which we want to tokenize")
```

to findout the frequency of distinct elements in NLP

```
from nltk.probability import FreqDist
variable=FreqDist(tokenize output)
```

to find the highest frequency elements in the values

```
variable=variable.most_common(which
common no we have to find)
```

blankline tokenizer is use to find the number of paragraph a string is having

```
from nltk.tokenize import
blankline_tokenize
variable=blankline_tokenize(string)
```

Now further some key terms in tokenization are:

Bigrams(tokens of two consecutive word**)**

Trigrams(tokens of three consecutive word**)**

Ngrams(tokens of n consecutive word**)**

```
from nltk.util import
bigrams,trigrams,ngrams
a="this is a string i am using to
check bigrams, trigrams and
ngrams."
bi=list(bigrams(a))
tri=list(trigrams(a))
ng=list(ngrams(a,5))
```

2. Stemming:

Normalize the word into their base form or root form.

for example we have word like affection affecting etc……..root word is affect

one thing to put in mind always that it was no assured that the return word will always be a root word(because it contain cutting of word from its prefix and suffix and this may not always helps in creating root word)

```
from nltk.stem import
PorterStemmer as ps
obj=ps()
obj.stem('word')
```

Next stemmer is LancasterStemmer(more powerfull)

```
From nltk.stem import
LancasterStemmer
Lst=LancasterStemmer()
A=['given','giving','give','gave']
For I in a:
    Print(lst.stem(i))
```

3. Lemmanization

It groups together different infected forms of a word, called lemma

Somehow similar to stemming, as it maps several words into one common root

Output of a lemmatization is a proper word

Like if we have "gone,go,gone" it lemmatize it into "go"

```
From nltk.stem import wordnet
#for it download
nltk.download(wordnet)
From nltk.stem import
WordNetLemmatizer
```

```
Lemm=WordNetLemmatizer()
dat=["given","giving","give","gave"]
for i in dat:
    print(lemm.lemmatize(i))
```

this return as it is value as we dnt define the pos(parts of speech) tags and it assumes all the word as noun pos tags tells you what exactly the given word is i.e noun part of speech etc.

Stop Words: in NLP stop words are not provide any significance but they are helpful in making sentences which we don't require.

For download: nltk.download(stopwords)

```
From nltk.corpus import stopwords
Print(stopwords.words('english'))
#it returns all stopwords of
English grammer
len(stopwords.words('english'))
```

Next we will remove stop words and punctuation from a paragraph using stopwords and regular expression matching.

```
Import re
s = re.sub(r'[^\w\s]','',string)
```

This will return a string free of punctuations next we will remove the stop words.

4. POS

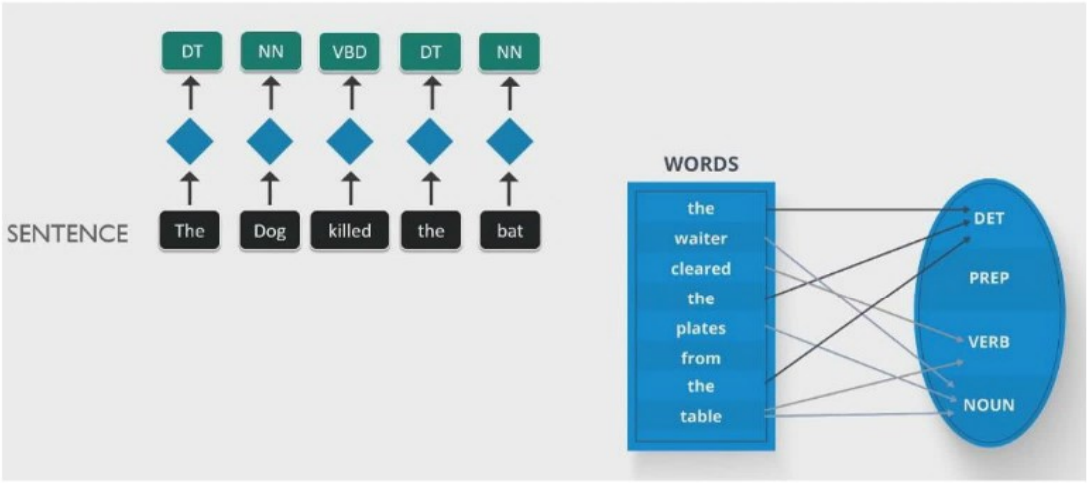
Parts of speech: use it as statistical NLP task, helpful in text realization, easy to evaluate how many tags are correct and we can also get the semantic information from the given text.

POS tags and descriptions.

Tags	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign Word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item maker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun,plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBR	Adverb, superlative
RP	Particle
SYM	Symbol
TO	To
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non 3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Whdeterminer
WP	Whpronoun

WP$	Possesive whpronoun
WRB	Whadverb

POS_example:



To do the same we have to select a string and make their word tokenize and then using pos_tag from nltk make their pos tags. For pos_tags download from nltk:

nltk.download('averaged_perceptron_tagger')

```
from nltk.tokenize import
word_tokenize as wt

String=”hello how are you sir I am
fine”

tkn=wt(String)

for token in tkn:

    print(nltk.pos_tag([token]))
```

5. Name entity Recognition(using ne_chunk)

Name entity recognition: detection of company name, person name, qualities, location name.

Extracted noun frames are classified into their respective categories. Sometimes the entities are misclassified so it is better to make a validation on top of the results, it will be very useful. The use of knowledge graph can be exploited for this purpose, the

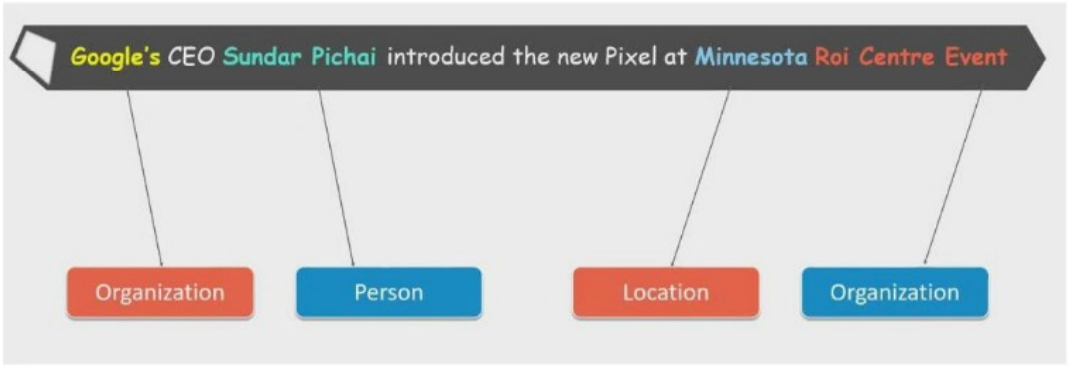
popular knowledge graph are google knowledge graph, IBM Watson, Wikipedia.

Download ne_chunk using nltk:

nltk.download('maxent_ne_chunker')

nltk.download('words')

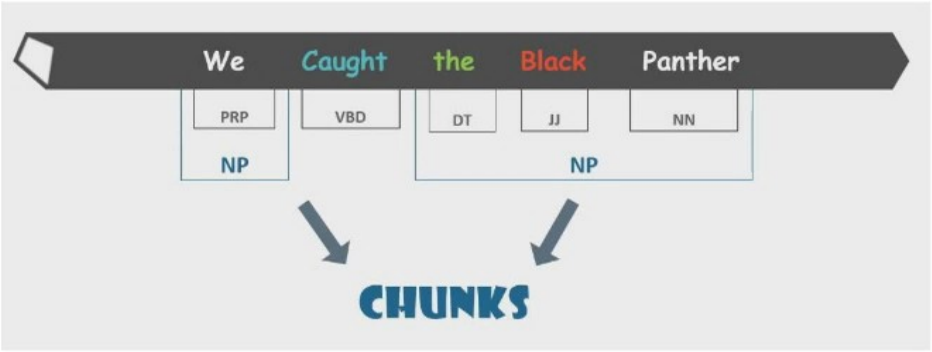
For example:




```
From nltk import ne_chunk
From nltk.tokenize import word_tokenize as wt
Import nltk
String=" India were tentatively scheduled to reach the Caribbean island immediately after the World Cup final on 14 July"
Tkn=wt(String)
Pos=nltk.pos_tag(Tkn)
Chunk=ne_chunk(Pos)
Print(Chunk)
```

6. Chunking

It is picking up individual information and grouping them into bigger pieces. These bigger pieces are known as chunk.




```
From nltk.tokenize import
word_tokenize as wt

Import nltk

String="The big cat ate the little
mouse who was after fresh
cheese."

Tkn=wt(String)

Pos=nltk.pos_tag(Tkn)
```

Now we have to create a grammer from noun phrase and will mention the tags which we want in chunk phrase with in curley braces as shown

```
Grammer_np=r"NP:
{<DT>?<JJ>*<NN>}"
```

Our grammer will be now:

```
Chunk_Parser=nltk.RegexpParser(Grammer_np)
```

Now we have to pass the chunk so parse will be used as shown below:-

```
Chunk_result=Chunk_Parser.parse(pos_tag
result we have to pass here)
```

We will get certain error because we dnt use ghost script and we dnt draw the syntactical tree.

But at the end we will get a tree which is not in tree structure but still we can read it.

Working on project:

We have done a number of concepts to deal with a text. Now in corpora movie review dataset is there, let use NLP for machine learning classifier.

Library to import

```
Import pandas as pd
Import numpy as np
From
sklearn.feature_extraction.text
import CountVectorizer
```

Will check different element of corpora

```
print(os.listdir(nltk.data.find("corpora")))
```

A list of directory opens in it. Select movie_reviews dataset.

```
From nltk.corpus import
movie_reviews
```

Checking the categories in movie_reviews(it contain two categories i.e pos and neg)

```
Print(movie_reviews.categories())
```

Checking the fields of pos and neg in movie_reviews

```
Print(movie_reviews.fileids('pos'))
Print(movie_reviews.fileids('neg'))
```

This will return 1000 pos and neg text files

Now import any text file

```
filename=nltk.corpus.movie_reviews.words('pos/cv000_29590.txt')
```

Join the tokens returns by it into a string using join methods because having already tokenize string increase work and there will be a problem in using CountVectorizer because it accept string in place of token.

Take an empty list and remove all comma spaces from it using loop.

```
lisEmpty=[]
negi=nltk.corpus.movie_reviews.fileids('neg')
for I in negi:
    rev_text=I=nltk.corpus.movie_reviews.words(I)
    rev_one=" ".join(rev_text)
    rev_one=rev_one.replace(' ,',',')
    rev_one=rev_one.replace(' .','.')
    rev_one=rev_one.replace('\'',"'")
    rev_one=rev_one.replace(' \'',"'")
    lisEmpty.append(rev_one)
```

Take an empty list and remove all comma spaces from it using loop.

```
posi=nltk.corpus.movie_reviews.fileids('pos')
for I in posi:
    rev_text=I=nltk.corpus.movie_reviews.words(I)
    rev_one=rev_one.replace(' ,',',')
    rev_one=rev_one.replace(' .','.')
    rev_one=rev_one.replace('\'',"'")
    rev_one=rev_one.replace(' \'',"'")
    lisEmpty.append(rev_one)
```

Now create targets before creating features for classifier

```
Neg_tar=np.zeros((1000,),dtype=np.int)
Pos_tar=np.ones((1000,),dtype=np.int)
Target_list=[]
For i in Neg_tar:
```

```
    Target_list.append(i)


For i in Pos_tar:
    Target_list.append(i)
```

Now create pandas series for target_list. i.e convert the list type into pandas series type

```
Y=pd.Series(Target_list)
```

Now lets check some starting series of resultant.

```
Y.head()
```

Now start creating feature using count Vectorization

Import count vectorization and set its features(Word Counts with CountVectorizer. The CountVectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary. You can use it as follows: Create an instance of the CountVectorizer class. Min_df: removes the words which are infrequent.)

```
From sklearn.feature_extraction.text import CountVectorizer
Count_vect=CountVectorizer(lowercase=True,stop_words='english',min_df=2)
X_count_vect=Count_vect.fit_transform(lisEmpty)
X_count_vect.shape  #this gives the dimension of the data
```

Now will create a list with all the names by tapping the vectorization name

```
X_names=Count_vect.get_feature_names()
```

Now will create a pandas dataframe by passing the scipy as the values and feature name as the columns

```
Xcount_vect=pd.DataFrame(X_count_vect.toarray(),columns=X_names) #Still dimension remain same
```

Now lets check the first few entries of X_count_vect

```
Xcount_vect.head()
```

Here all the values are zero

Now split the data frame into training and testing set and examine the test set.

```
From sklearn.model_selection import train_test_split

From sklearn.metrics import confusion_matrix

From sklearn import metrics

Xtrain,xtest,ytrain,ytest=train_test_split(X_count_vect,y,test_size=0
.25,random_state=5)
```

Here check the dimension of x test and x train

Now use naïve bayes classifier for text classification, over the training and testing set.

Bernoulli Naive Bayes : It assumes that all our features are binary such that they take only two values. Means **0s** can represent “word does not occur in the document” and **1s** as "word occurs in the document".

Multinomial Naive Bayes: Its is used when we have **discrete data** (e.g. movie ratings ranging 1 and 5 as each rating will have certain **frequency** to represent). In text learning we have the count of each word to predict the class or label.

Gaussian Naive Bayes: Because of the assumption of the **normal distribution**, Gaussian Naive Bayes is used in cases when all our features are **continuous**. For example in **Iris dataset** features are sepal width, petal width, sepal length, petal length. So its features can have different values in data set as width and length can vary. We can’t represent features in terms of their occurrences. This means data is continuous. Hence we use Gaussian Naive Bayes here.

```
From sklearn.naive_bayes import
GaussianNB

Gnb=GaussianNB()

Ypred=Gnb.fit(Xtrain,ytrain).predict(xtest)
```

Import multinomial classifier to check the accuracy with this algorithm also

```
From skelarn.naive_bayes import
MultinomialNB

Clf=MultinomialNB()

Clf.fit(Xtrain,ytrain)

Ypredclf=Clf.predict(Xtest)
```

Now check the accuracy score

```
Print(metrics.accuracy_score(ytest,Ypred))
```

Check confusion matrix also to find the accuracy and relation between actual and predicted results.

```
cm=confusion_matrix(ytest,Ypred)
```

DEEP LEARNING

Introduction to deep learning

- Deep learning is the subset of machine learning(application wise), it is growing popularity due to its capability of making complex computations.
- Today's real time problems have high dimensions, so they are very complex to handle and process. Like image data. ML is not capable of solving these use-case so deep learning came for rescue.
- In deep learning we design neural network containing neurons to process information
- Keras, Tensorflow and Theano are the main API of DL(NN)

Applications of deep learning

- Future predictions
- Chat-bots
- Self driving cars
- Google eye doctor
- AI based music composer

How deep learning works?

- Deep learning studies the basic unit of a brain called a neuron. On the basis of these neurons artificial neurons(perceptron) are developed.
- Biological neuron contain
  - Dendrite: it receive signal from other neurons
  - Cell body: it sums all the inputs
  - Axon: it is used to transmit signals to the other cells
- Perceptron also receives multi input then apply various transformation and then produces results

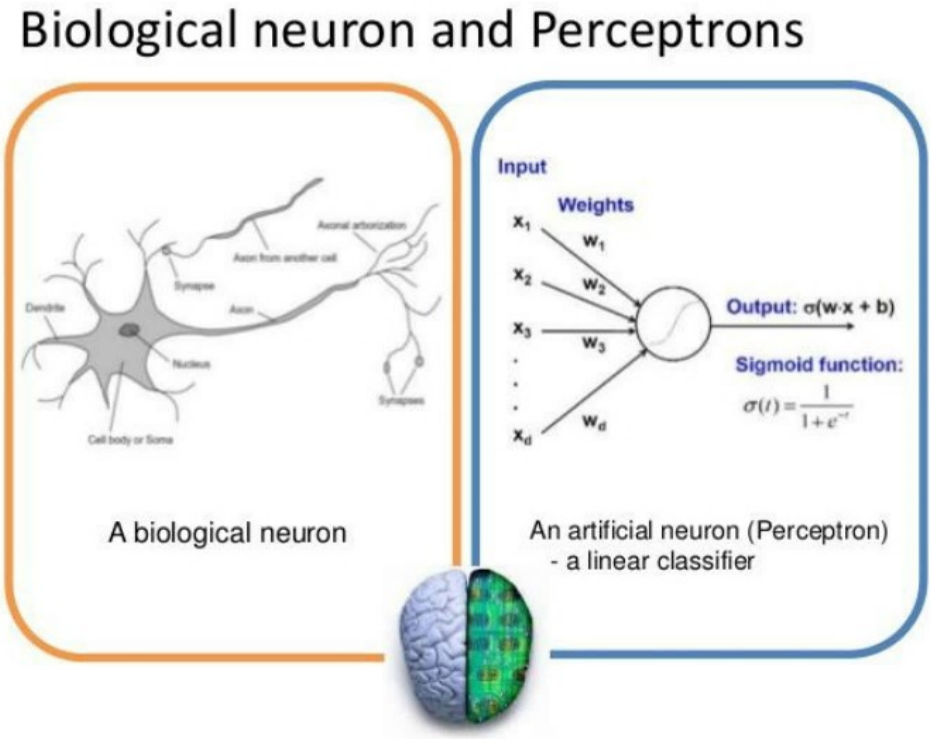
Introduction to Neural network

An Artificial Neural Network (ANN) models the relationship between a set of input signals and an output signal using a model derived from our understanding of how a biological brain responds to stimuli from sensory inputs. Just as a brain uses a network of interconnected cells called neurons to create a massive parallel processor, ANN uses a network of artificial neurons or nodes to solve learning problems

The human brain is made up of about 85 billion neurons, resulting in a network capable of representing a tremendous amount of knowledge

For instance, a cat has roughly a billion neurons, a mouse has about 75 million neurons, and a cockroach has only about a million neurons. In contrast, many ANNs contain far fewer neurons, typically only several hundred, so were in no danger of creating an artificial brain anytime in the near future

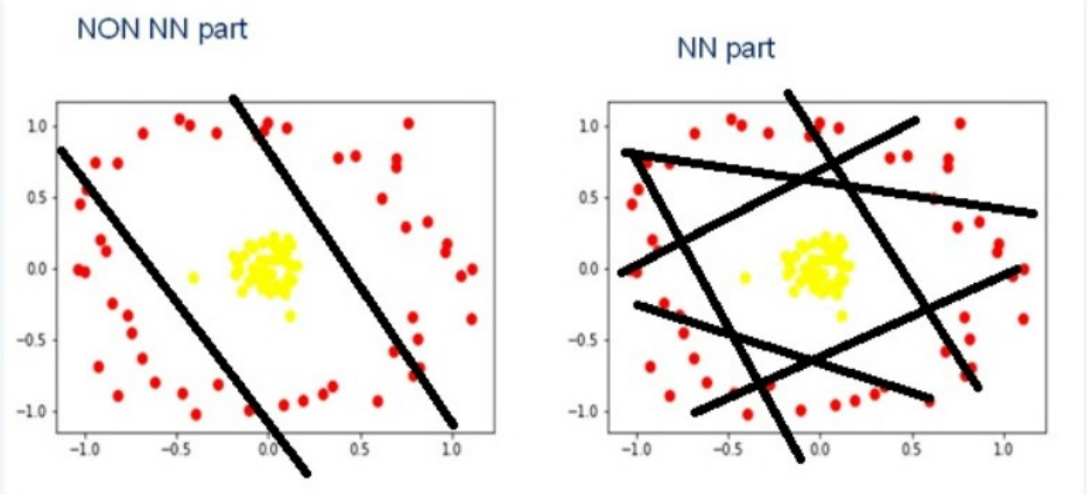
Biological to artificial neurons



Incoming signals are received by the cell's dendrites through a biochemical process. The process allows the impulse to be weighted according to its relative importance or frequency. As the cell body begins accumulating the incoming signals, a threshold is reached at which the cell fires and the output signal is transmitted via an electrochemical process down the axon. At the axon's terminals, the electric signal is again processed as a chemical signal to be passed to the neighbouring neurons.

- Works like a human brain to process an information
- They are computational algorithm
- It is inspired by the nervous system of animal
- Applicable for continuous and regression problems
- Applicable for linear and non linear problems
- **Types:**
  - ANN
  - CNN
  - RNN

How neural network works?

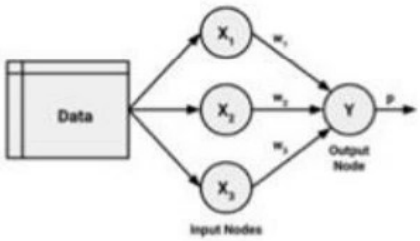


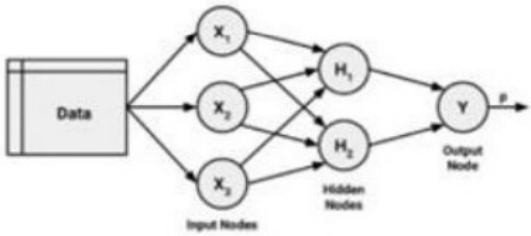
Components of neural network

- **Layers**

The input and output nodes are arranged in groups known as **layers.**

Input nodes process the incoming data exactly as it is received, the network has only one set of connection weights (labelled here as wl, w2, and w3). It is therefore termed a single-layer network.

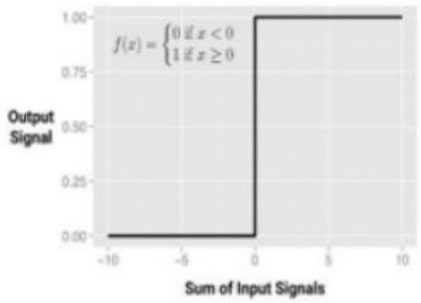


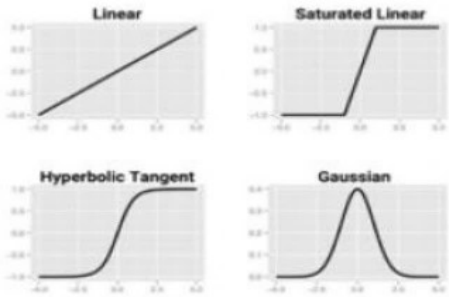


- INPUT
- HIDDEN
- OUTPUT

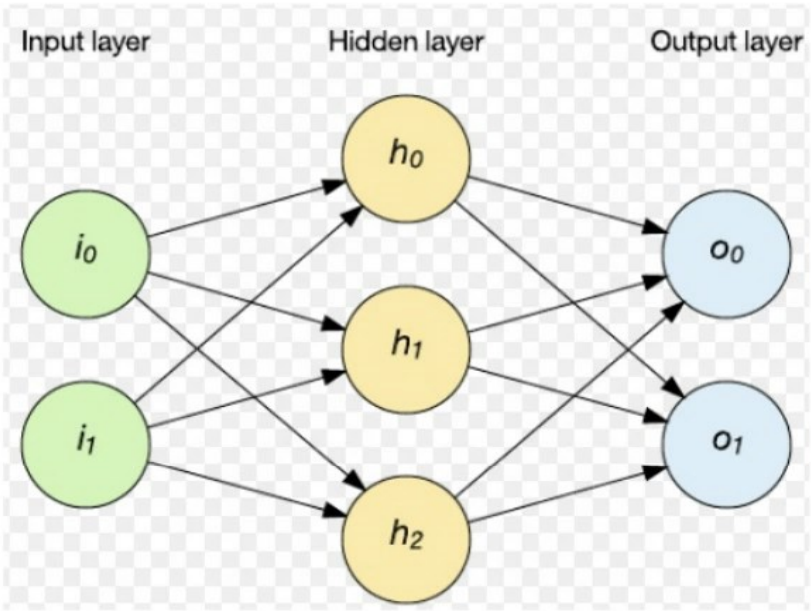
Activation function

The following figure depicts a typical threshold function; in this case, the neuron fires when the sum of input signals is at least zero. Because its shape resembles a stair, it is sometimes called a unit step activation function.





- Relu
- TanH
- Sigmoid
- Linear



Introduction to Keras

- Keras was developed and is maintained by Francois Chollet
- It's a high level API run at the top of tensorflow and theano.
- It runs on both CPU and GPU.
- It performs a number of tasks like initialize neural network, adding layers, etc.

Introduction to Tensorflow

- It is developed by google brain team of google in 2015 for computation purpose. Uses to make the algorithm fast.

- It's an open source machine learning framework for dataflow programming.
- Tensorflow is make up of two word:
  - Tensor: multi dimensional array
  - Flow: data process
- It uses some of the concept of numpy for computation.

Introduction to Theano

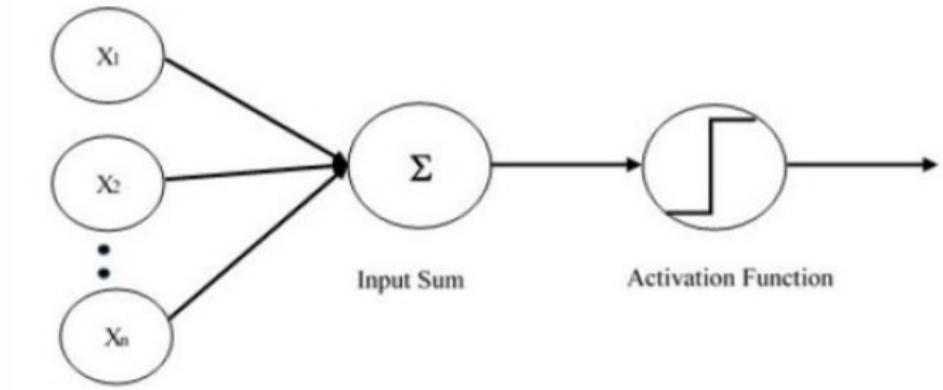
- It is developed by LISA(now MILA) group at the university of Montreal, Quebec, Canada (home of YoshuaBengio)
- It's a python library that allows us to define, optimize and evaluate mathematical expression involving multi dimensional array efficiently.
- It uses some of the concept of numpy for computation.

Functionality of Tensorflow

- It's a great library of deep learning and performs a lot of operations related to computation and optimization.
- It's make the overall system fast. Speeding up a process in terms of computation was the main requirement, it is doing it smoothly.
- Due to its advance feature making of deep neural network is very easy.
- We can find out the weight of neurons using this framework.

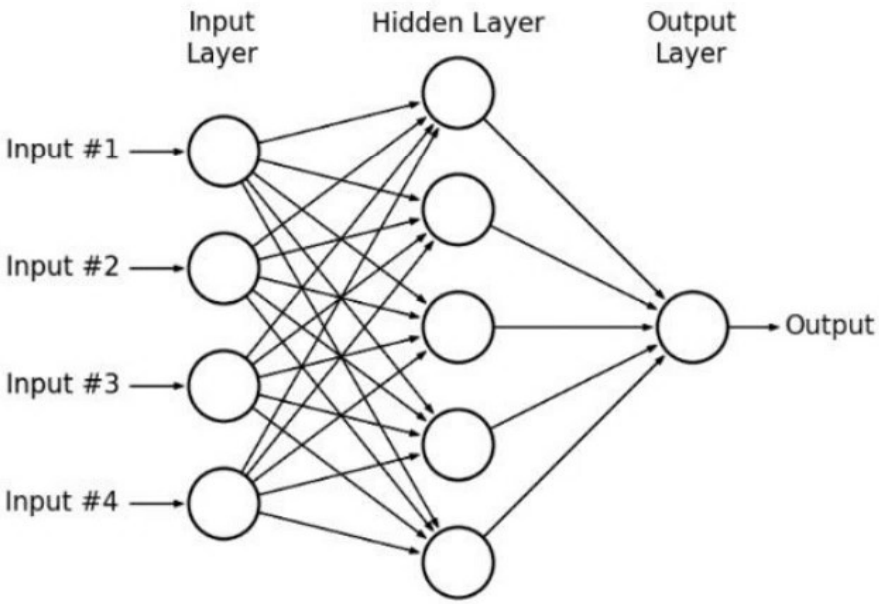
Single layer perceptron

- It is the simplest neural network design with a single layer, in which there is only one layer of input nodes that send the weighted input to the subsequent node.
- It handles the problems having less attributes. Otherwise if the attributes get increase the accuracy gets worst.



Multi layer perceptron

- It is a class of feedforward artificial neural network.
- It can have minimum three layers(input, hidden, output)
- Except input nodes, each node in it uses non linear activation function.
- Its uses back propagation for training.
- This is mainly use for extreme complex problems.



Pros and cons of Single, multi layer perceptron

- Single layer can be use for only simple problems but multi layer can use for complex problems
- Single layer requires less training time but multi layer require more training time
- Single layer contain one layer of perceptron but multi layer have minimum three layers.

- Single layer accuracy remains low but multi layer accuracy is far better.

Convolutional neural network

Why not fully connected network?

- Let consider and image 20*20*3=1200, so at the first layer we will have 1200 weight.
- Let consider one more image 400*400*3=480000, so here you can think how many neurons were required and this case is not practically possible also.
- So connected network is not use for such problems.

Introduction to CNN

- It can handle classification and image recognition type of problems.
- It is a regularization of multi layer perceptron due to high dimensionality of data
- It requires little pre-processing compared to other image recognition algortihm.
- CNN requires attributes:
    - Input tensor with a shape
    - No of convolutional kernels
        - Width and height of kernal are hyper-parameters.
        - Depth of kernal must be equal to the image depth.

Concept of an image

- Image is the collection of pixels placed in the matrix form.
- Image quality depends on the row and column of a mtrix.
- There are two types of images:
    - Grayscale
    - Color Image

- Gray scale: It is an image containing two colors only that is black and white. The pixel value of color is vary from 0 to 255. The values which are near to zero are dark pixels, and the values which are near to 255 are bright pixels. So according to this the brightness and darkness of an image is defined. So it mean if we are having a image then just making its pixel value increment and decrement we can set the brightness and darkness. Grayscale image is represented by 2D matrix. For example: 480x640, 1200x1280
- Color image: It is an image containing three frames that is red, blue and green. Color image is represented by 3D matrix. For example 480x640x3, 1200x128x3.These three colors are combined responsible for creating a color image as shown in figure.

CNN

- These are like neural network and they are made of the neurons with learnable weight and biases.
- In it neurons learns to convert the input signal into output signal.
- If we are modelling a neuron to find something from an image and if image gets match and noticed by the neuron then that will be labelled by the neuron. This is the concept CNN use for finding objects from an image or video.

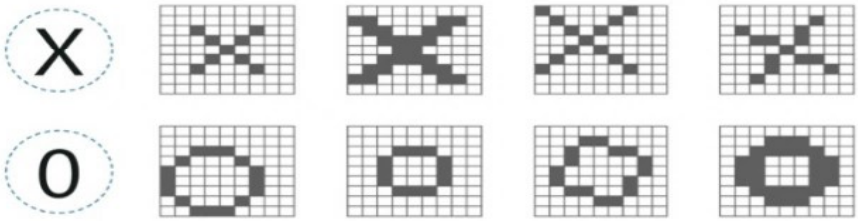
History of CNN

- In 1960 this term is coined by Rosenblatt.

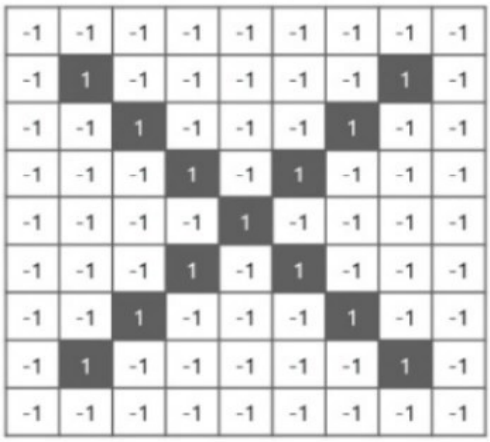
- In late 2000 deep learning use neural network.
- In july 2012 researcher of Google exposed an advanced neural network concept.
- They applied it to detect the cat from an image.

Working of CNN

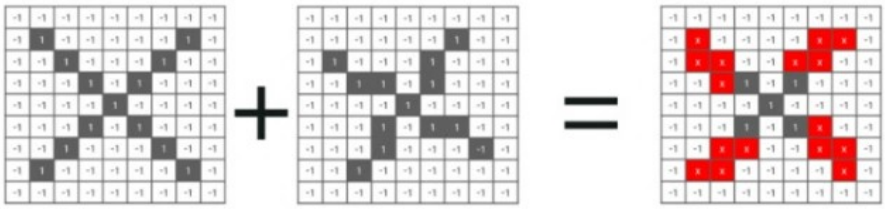
- Let take an example to understand the working of CNN



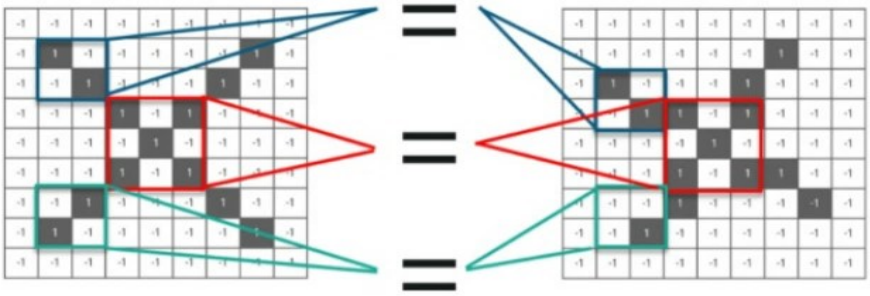
- In this bright pixel are taken as -1 and, dark pixel are taken as 1 so that calculus part gets understand easily.



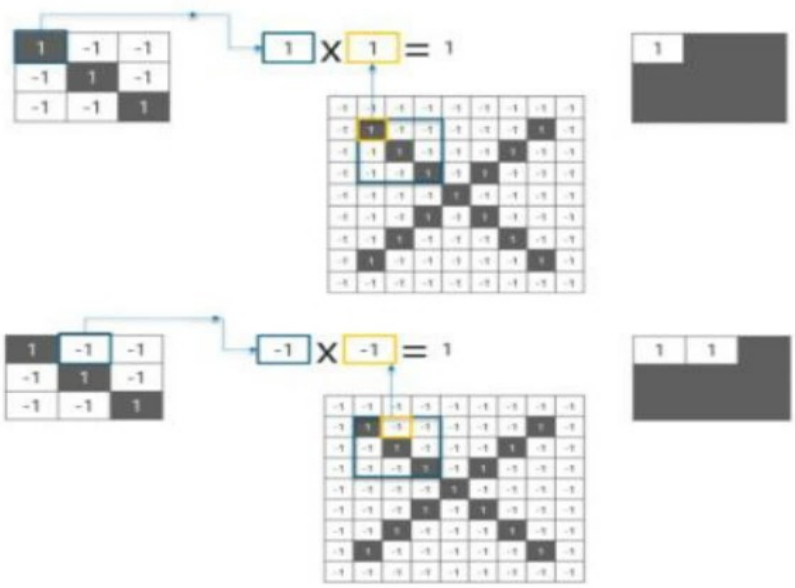
- While making comparison of sample image and test image then pixel are misaligned and are not making the match even the shape are representing same symbol. This is happening because of the misalignment of pixels.



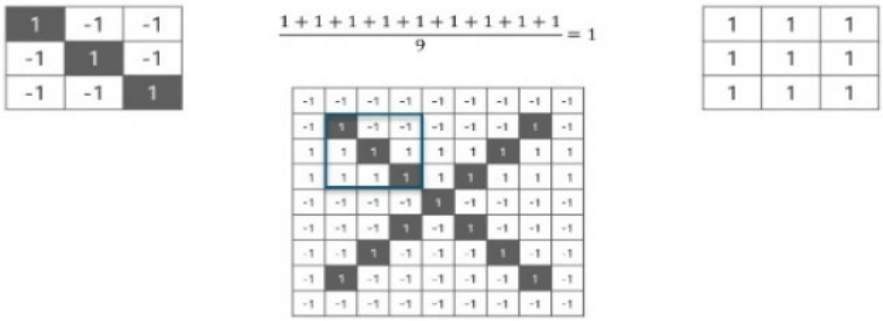
- Fix the misaligned pixels by taking small patches and then compare them. These small patches are also known as windows.



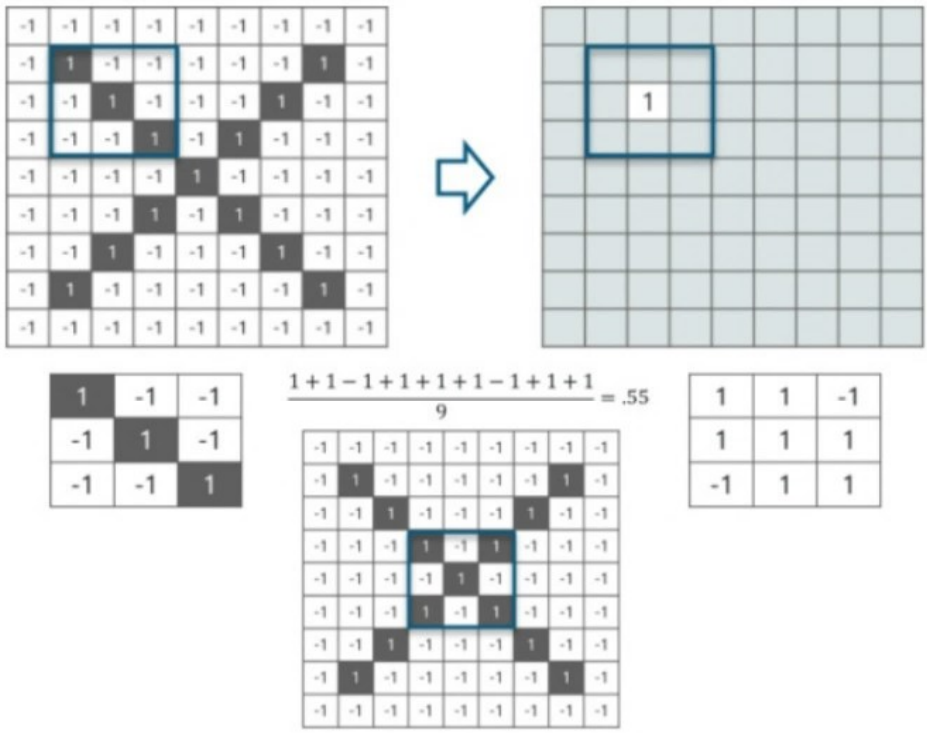
- Now let go through some processes which are also known as layers of CNN.
  - **Convolution**
  - **Relu**
  - **Pooling**
  - **Fully connected layer**
- **Convolution**
  - Steps
    - Line up: image and feature
    - Multiply: each image pixel with the corresponding feature pixel.



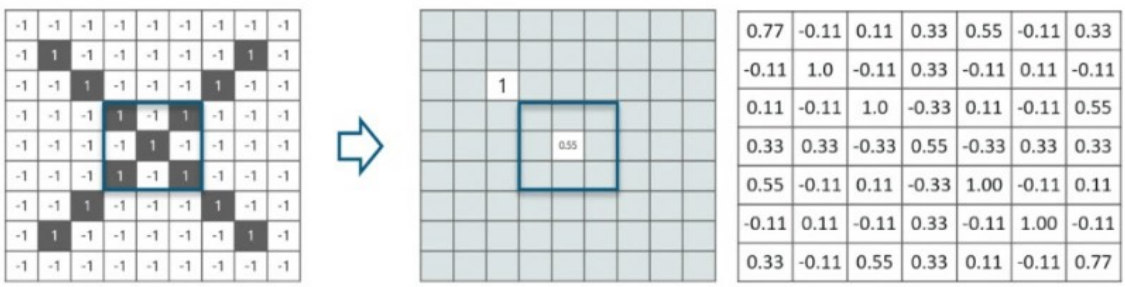
- Add the values and find the sum
- Divide the sum with total no of pixels



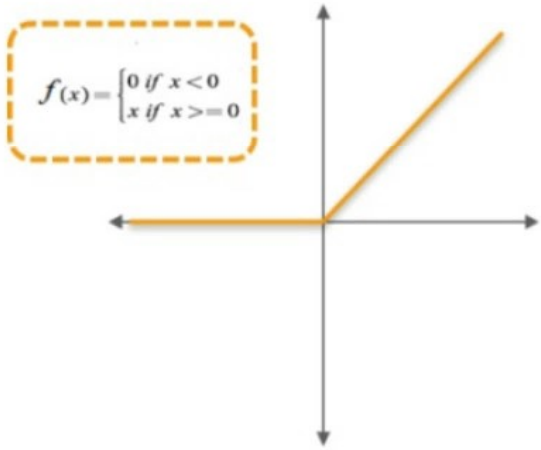
- Now the pixel get fill it at the centre
- Now move this filter around and repeat the same process



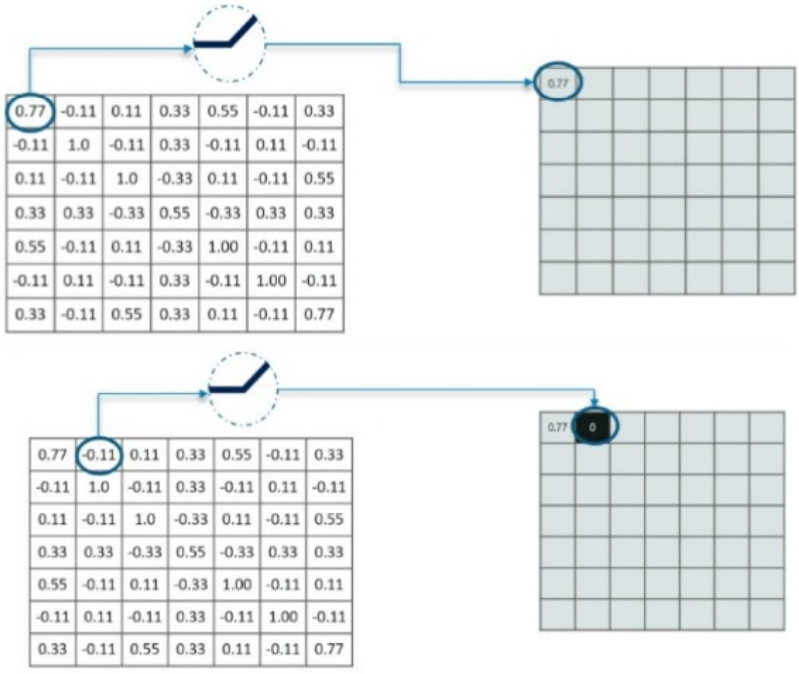
- Now the pixel get fill it at the centre
- Now move this filter around and repeat the same process



- **Relu**
  - It is an activation function it makes the node activate when an output is above a certain threshold value. Otherwise it makes the output zero. So a filter kind of work is performing by the activation function.



  - It specially removes thenegative values and make it zero.

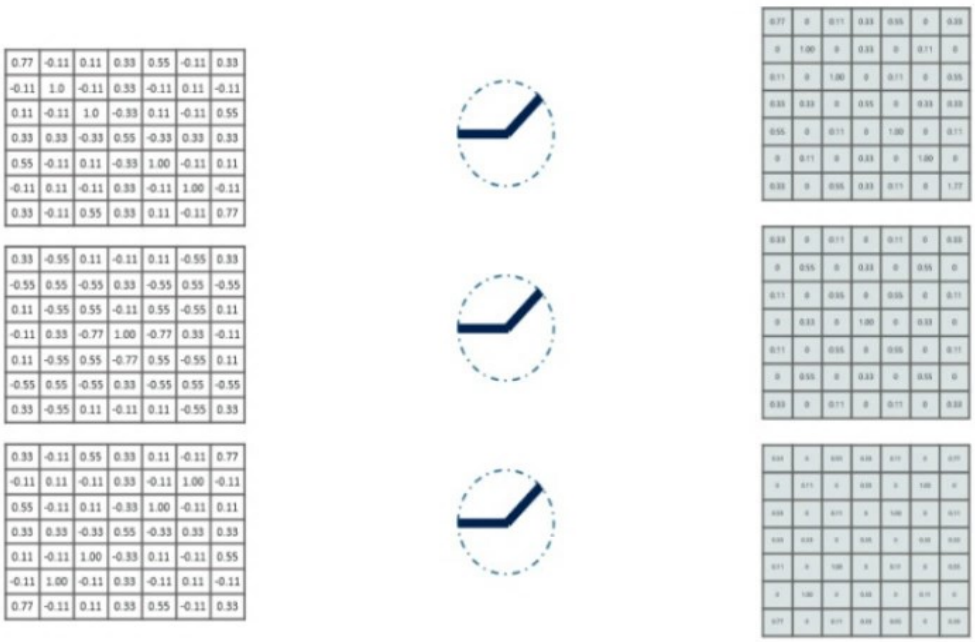


- This is the matrix we will get after removing all values less than 0

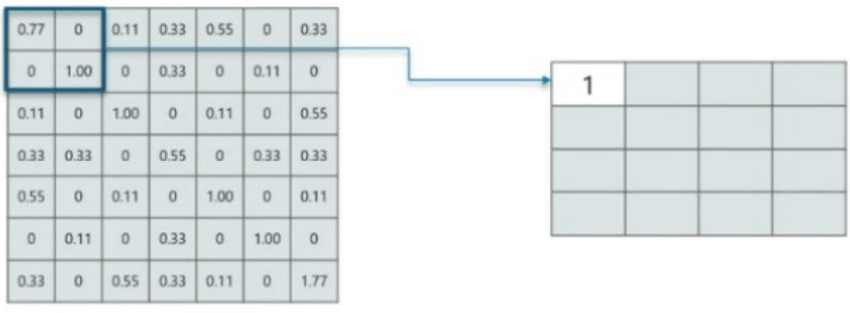
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.0	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.0	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	1.77

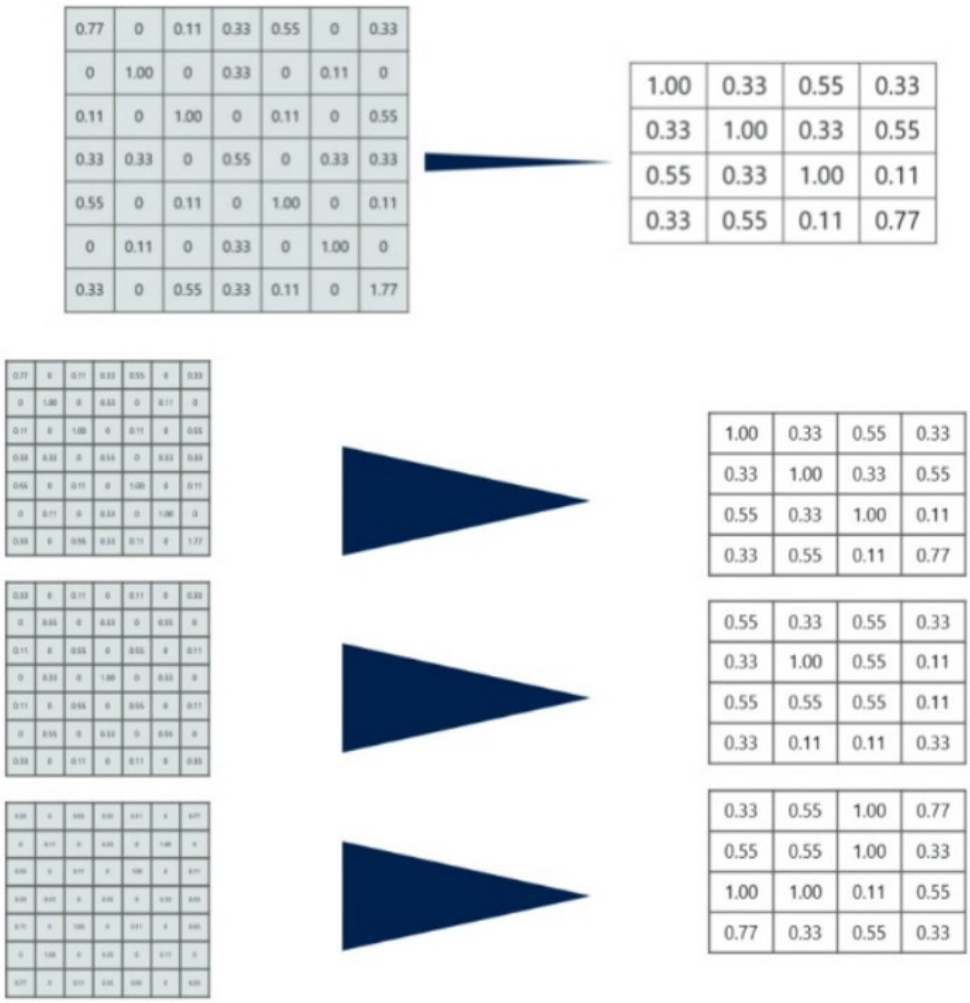
- Let insert the values for other feature of an image



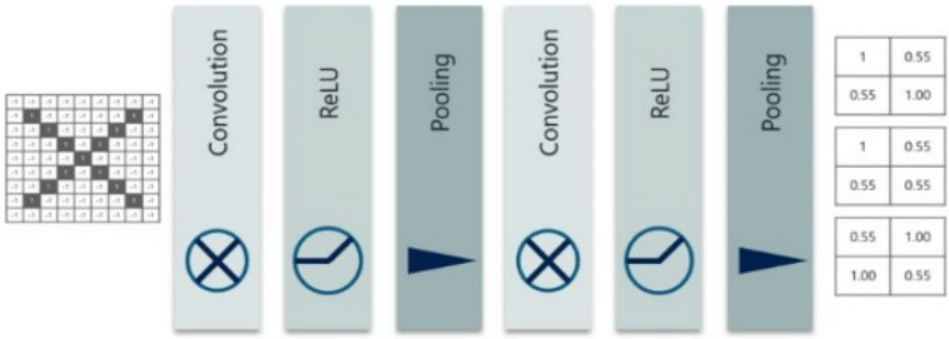
- **Pooling layer**
  - It is shrinking of an image in smaller size.
  - Selecting a block and picking a maximum value



  - It is shrinking of an image in smaller size.
  - Selecting a block and picking a maximum value



- Further shrinking the lock to make it 2x2 matrix by implementing pooling concept.



- **Fully connected layer**
  - The first phase of this method contains converting the multi dimensional array into single dimensional, and collects them in the form of single dimensional array.

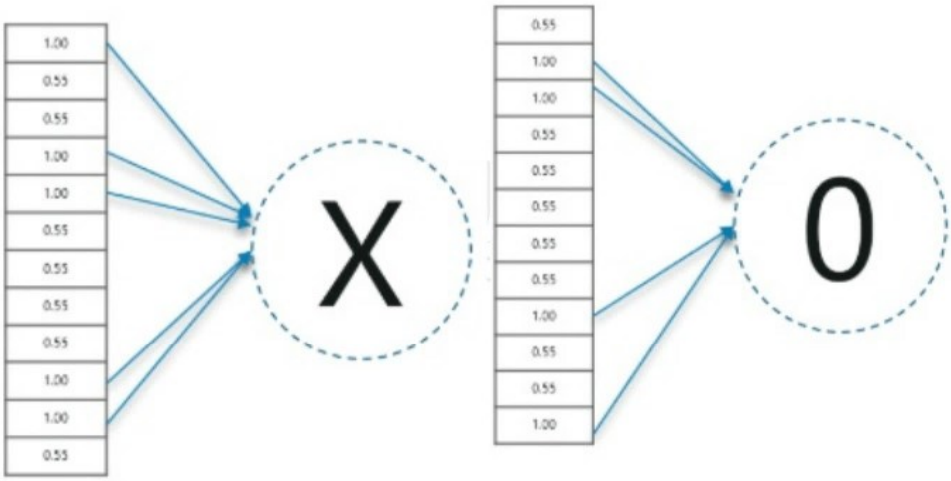
1	0.55
0.55	1.00

1	0.55
0.55	0.55

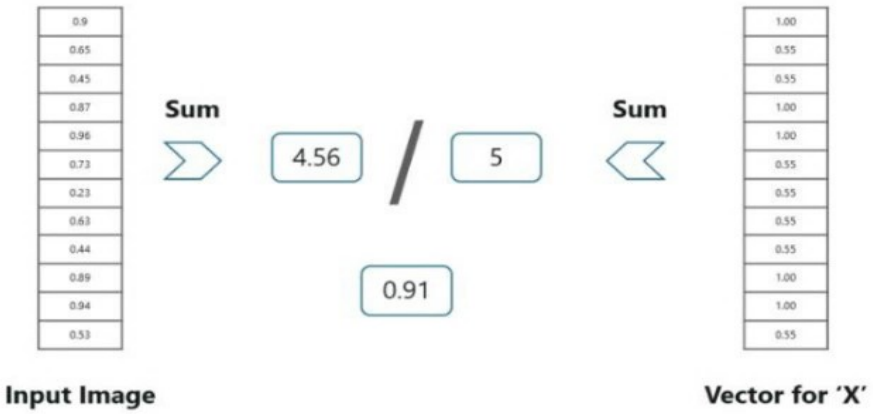
0.55	1.00
1.00	0.55

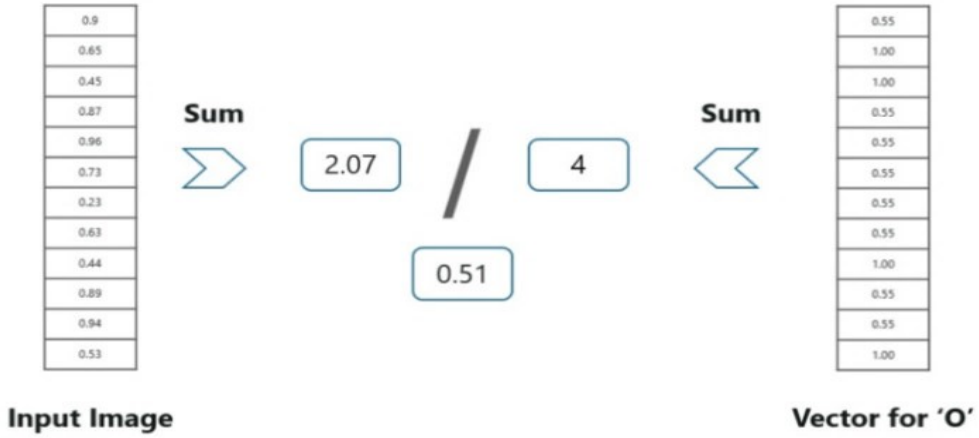
1.00
0.55
0.55
1.00
1.00
0.55
0.55
0.55
0.55
1.00
1.00
0.55

- Consider the image below for X and O and find out the distinguishable one it contains. So according it will be easily find out their key difference.



- Get the results by comparing the obtained value with list of X or O.





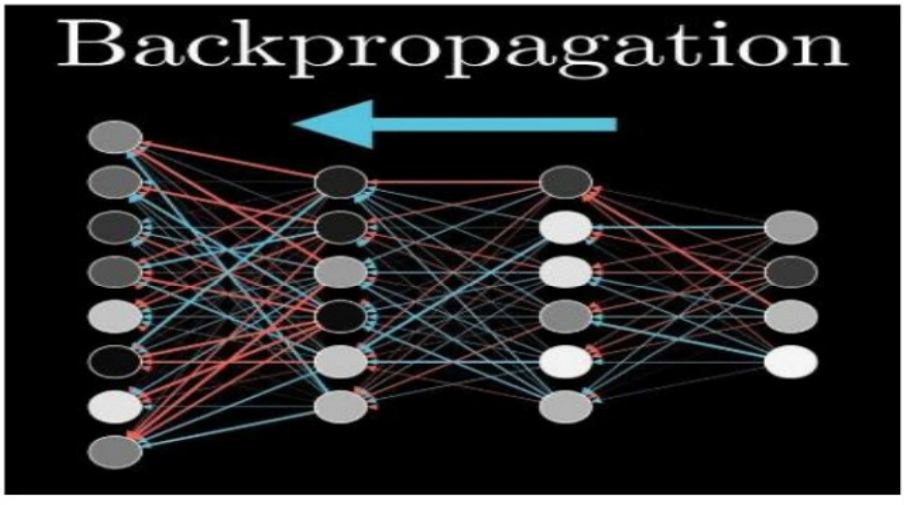
Comparing both we can easily find out the ratio in case of X is more compared to O. So we can say that the input image contain X in place of O.

Back propagation in neural network

Introduction

- It is a method use for training the neural network
- Its main goal is to optimize the weight so that the neural network can learn how to correctly map for increasing the accuracy of model.
- On every iteration output of neural network changes as every time keras assign a random weight to the neurons.
- It is done for multi perceptron network.

Working flow



Let we find out the practical reason of working out with back propagation with an example of a dataset giving the information about the exit state of a customer on multi scenario. Then will use ANN to make it solve and then will find out the accuracy of overall model.

Now the purpose of Back propagation comes. If we get some accuracy from ANN algorithm back propagation provides the way of increasing the accuracy.

Let see an example in which an algorithm is implemented using ANN and then will find out how back propagation increases the accuracy of overall model.

Data set to use:

RowNumb	CustomerId	Surname	CreditScor	Geography	Gender	Age	Tenure	Balance	NumOfPro	HasCrCard	IsActiveMe	EstimatedS	Exited
1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348.9	1
2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.6	0
3	15619304	Onio	502	France	Female	42	8	159660.8	3	1	0	113931.6	1
4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
5	15737888	Mitchell	850	Spain	Female	43	2	125510.8	1	1	1	79084.1	0
6	15574012	Chu	645	Spain	Male	44	8	113755.8	2	1	0	149756.7	1
7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
8	15656148	Obinna	376	Germany	Female	29	4	115046.7	4	1	0	119346.9	1
9	15792365	He	501	France	Male	44	4	142051.1	2	0	1	74940.5	0
10	15592389	H?	684	France	Male	27	2	134603.9	1	1	1	71725.73	0
11	15767821	Bearce	528	France	Male	31	6	102016.7	2	0	0	80181.12	0
12	15737173	Andrews	497	Spain	Male	24	3	0	2	1	0	76390.01	0
13	15632264	Kay	476	France	Female	34	10	0	2	1	0	26260.98	0
14	15691483	Chin	549	France	Female	25	5	0	2	0	0	190857.8	0
15	15600882	Scott	635	Spain	Female	35	7	0	2	1	1	65951.65	0
16	15643966	Goforth	616	Germany	Male	45	3	143129.4	2	0	1	64327.26	0
17	15737452	Romeo	653	Germany	Male	58	1	132602.9	1	1	0	5097.67	1
18	15788218	Henderson	549	Spain	Female	24	9	0	2	1	1	14406.41	0
19	15661507	Muldrow	587	Spain	Male	45	6	0	1	0	0	158684.8	0
20	15568982	Hao	726	France	Female	24	6	0	2	1	1	54724.03	0
21	15577657	McDonald	732	France	Male	41	8	0	2	1	1	170886.2	0
22	15597945	Dellucci	636	Spain	Female	32	8	0	2	1	0	138555.5	0
23	15699309	Gerasimov	510	Spain	Female	38	4	0	1	1	0	118913.5	1

Implementation of algorithm without back propagation

Importing the libraries

```
import pandas as pd

import numpy as np
```

Importing the dataset using pandas and extract independent and dependent variables

```
dataset = pd.read_csv('Churn_Modelling.csv')

X = dataset.iloc[:, 3:13].values

y = dataset.iloc[:, 13].values
```

Use to convert variables (string) into values using scikitlearn (labelencoder)

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X_1 = LabelEncoder()
X[:, 1] = labelencoder_X_1.fit_transform(X[:, 1])
labelencoder_X_2 = LabelEncoder()
X[:, 2] = labelencoder_X_2.fit_transform(X[:, 2])
```

Some of the variables are coded as 0,1 and 2 due to making their labels, labels are made through alphabets sequence so it is obvious that our resultant have no order or rank. So their dummies variables are created. OneHotEncoder for dummy variable creation using scikitlearn

```
onehotencoder = OneHotEncoder(categorical_features = [1])
X = onehotencoder.fit_transform(X).toarray()
X = X[:, 1:]
```

The data set is requiring splitting into two parts one is the training and other is testing. Training set is use to train our system by teaching them the relationship between input and output variables.

Use train_test_split to divide data. Training set split ratio taken as 80:20.

Standards are 80:20, 75:25, 60:40

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
print(y_train)
print(X_train)
```

Scaling of data is required as some of the data is very large and some of the data is very small. StandardScalar is use to scale the data

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
print(y_train)
```

Till now the data is ready to implement into algorithm. Next we have to initialize neural network using sequential module.

Importing the Keras libraries and packages

```
from keras.models import Sequential
from keras.layers import Dense
```

Initializing Neural Network

```
classifier = Sequential()
```

Using dense function will add hidden layer one by one.

Output_dim:-no of nodes we want to add to this layer

Init:- initialization of stochastic gradient decent

In neural network want to assign weight to each each mode which is nothing but the importance of that node.

Initialially we are randomly initializing weights using uniform function.

Input_dim:- only use for first layer as model doesn't know the no of input variables.

In our case total input are 11. in 2nd layer system automatically knows about the input from the first hidden layer.

Activation function:-neuron applies activation function to weighted sum(summation of Wi*Xi where w is weight and x is input and i is suffix of w and x).

If any neuron have value nearly 1 then it will be more activated and more passes signal.

CTRL_BREAK_EVENT which activation function use is crucial task.

Using rectifier(relu) function in hidden layer and sigmoid function in output layer as we require binary result from output layer.

If no of category in output layer is more than 2 then use softmax function.

Adding the input layer and the first hidden layer

```
classifier.add(Dense(units= 6, kernel_initializer = 'uniform', activation = 'relu', input_dim = 11))
```

Adding the second hidden layer

```
classifier.add(Dense(units= 6, kernel_initializer = 'uniform', activation = 'relu'))
classifier.add(Dense(units= 6, kernel_initializer = 'uniform', activation = 'relu'))
classifier.add(Dense(units= 6, kernel_initializer = 'uniform', activation = 'relu'))
classifier.add(Dense(units= 6, kernel_initializer = 'uniform', activation = 'relu'))
```

Adding the output layer

```
classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))
```

Optimizer:-use to find optimal set of weights. algo is stochastic gradient descent(SGD).

We will use adam algo in SGD.

SGD depends on loss thus our second parameter is loss.

Our dependent variable is binary so we have to use binary_crossentropy.

For more than two categories use categorical_crossentropy.

To improve performance of neural network add metrics as accuracy

Compiling Neural Network

```
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

Deep learning neural network is formed

Batch size is used to specify the no of observations after which you want to update weight.

Epoch is total no of iterations.

Choosing value of batch size and epoch is trial and error there is no specific rule for that.

Fitting our model

```
classifier.fit(X_train, y_train, batch_size = 10, epochs = 100)
```

Predicting the test results.

Prediction function gives us the probability of the customer leaving the company.

Predicting the Test set results

```
y_pred = classifier.predict(X_test)
y_pred = (y_pred> 0.5)
```

Evaluating our model performance.

To check the accuracy build confusion matrix.

Creating the Confusion Matrix

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

Results: As shown in the image acc is the variable for accuracy and the accuracy we are getting is 79.92 percent

```
Using TensorFlow backend.
Epoch 1/100
8000/8000 [==============================] - 5s 583us/step - loss: 0.5608 - acc: 0.7992
Epoch 2/100
8000/8000 [==============================] - 1s 106us/step - loss: 0.5014 - acc: 0.7994
Epoch 3/100
8000/8000 [==============================] - 1s 151us/step - loss: 0.5014 - acc: 0.7994
Epoch 4/100
8000/8000 [==============================] - 1s 117us/step - loss: 0.5014 - acc: 0.7994
Epoch 5/100
8000/8000 [==============================] - 1s 157us/step - loss: 0.5015 - acc: 0.7994
Epoch 6/100
8000/8000 [==============================] - 1s 136us/step - loss: 0.5015 - acc: 0.7994
Epoch 7/100
8000/8000 [==============================] - 4s 456us/step - loss: 0.5013 - acc: 0.7994
Epoch 8/100
8000/8000 [==============================] - 1s 172us/step - loss: 0.5013 - acc: 0.7994
Epoch 9/100
8000/8000 [==============================] - 1s 145us/step - loss: 0.5016 - acc: 0.7994
Epoch 10/100
8000/8000 [==============================] - 1s 136us/step - loss: 0.5015 - acc: 0.7994
Epoch 11/100
8000/8000 [==============================] - 3s 435us/step - loss: 0.5015 - acc: 0.7994
```

Implementation of algorithm with back propagation

Importing the libraries

```
import pandas as pd
import numpy as np
import tensorflow as tf
import random
np.random.seed(500)
tf.set_random_seed(500)
```

Importing the dataset using pandas and extract independent and dependent variables

```
dataset = pd.read_csv('Churn_Modelling.csv')
X = dataset.iloc[:, 3:13].values
y = dataset.iloc[:, 13].values
```

Use to convert variables (string) into values using scikitlearn (labelencoder)

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X_1 = LabelEncoder()
X[:, 1] = labelencoder_X_1.fit_transform(X[:, 1])
labelencoder_X_2 = LabelEncoder()
X[:, 2] = labelencoder_X_2.fit_transform(X[:, 2])
```

Some of the variables are coded as 0,1 and 2 due to making their labels, labels are made through alphabets sequence so it is obvious that our resultant have no order or rank. So their dummies variables are created. OneHotEncoder for dummy variable creation using scikitlearn

```
onehotencoder = OneHotEncoder(categorical_features = [1])
X = onehotencoder.fit_transform(X).toarray()
X = X[:, 1:]
```

The data set is requiring splitting into two parts one is the training and other is testing. Training set is use to train our system by teaching them the relationship between input and output variables.

Use train_test_split to divide data. Training set split ratio taken as 80:20.

Standards are 80:20, 75:25, 60:40

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
print(y_train)
print(X_train)
```

Scaling of data is required as some of the data is very large and some of the data is very small. StandardScalar is use to scale the data

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
print(y_train)
```

Till now the data is ready to implement into algorithm. Next we have to initialize neural network using sequential module.

Importing the Keras libraries and packages

```
from keras.models import Sequential
from keras.layers import Dense
```

Initializing Neural Network

```
classifier = Sequential()
```

Using dense function will add hidden layer one by one.

Output_dim:-no of nodes we want to add to this layer

Init:- initialization of stochastic gradient decent

In neural network want to assign weight to each each mode which is nothing but the importance of that node.

Initialially we are randomly initializing weights using uniform function.

Input_dim:- only use for first layer as model doesn't know the no of input variables.

In our case total input are 11. in 2nd layer system automatically knows about the input from the first hidden layer.

Activation function:-neuron applies activation function to weighted sum(summation of Wi*Xi where w is weight and x is input and i is suffix of w and x).

If any neuron have value nearly 1 then it will be more activated and more passes signal.

CTRL_BREAK_EVENT which activation function use is crucial task.

Using rectifier(relu) function in hidden layer and sigmoid function in output layer as we require binary result from output layer.

If no of category in output layer is more than 2 then use softmax function.

Adding the input layer and the first hidden layer

```
classifier.add(Dense(units= 6, kernel_initializer = 'uniform', activation = 'relu', input_dim = 11))
```

Adding the second hidden layer

```
classifier.add(Dense(units= 6, kernel_initializer = 'uniform', activation = 'relu'))
classifier.add(Dense(units= 6, kernel_initializer = 'uniform', activation = 'relu'))
classifier.add(Dense(units= 6, kernel_initializer = 'uniform', activation = 'relu'))
classifier.add(Dense(units= 6, kernel_initializer = 'uniform', activation = 'relu'))
```

Adding the output layer

```
classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))
```

Optimizer:-use to find optimal set of weights. algo is stochastic gradient descent(SGD).

We will use adam algo in SGD.

SGD depends on loss thus our second parameter is loss.

Our dependent variable is binary so we have to use binary_crossentropy.

For more than two categories use categorical_crossentropy.

To improve performance of neural network add metrics as accuracy

Compiling Neural Network

```
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

Deep learning neural network is formed

Batch size is used to specify the no of observations after which you want to update weight.

Epoch is total no of iterations.

Choosing value of batch size and epoch is trial and error there is no specific rule for that.

Fitting our model

```
classifier.fit(X_train, y_train, batch_size = 10, epochs = 100)
```

Predicting the test results.

Prediction function gives us the probability of the customer leaving the company.

Predicting the Test set results

```
y_pred = classifier.predict(X_test)

y_pred = (y_pred> 0.5)
```

Evaluating our model performance.

To check the accuracy build confusion matrix.

Creating the Confusion Matrix

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
```

Results: As shown in the image acc is the variable for accuracy and the accuracy we are getting is 84 percent. This is showing that how back propagation makes increase in accuracy

```
Epoch 1/100
8000/8000 [==============================] - 1s 185us/step - loss: 0.4890 - acc: 0.8006
Epoch 2/100
8000/8000 [==============================] - 1s 116us/step - loss: 0.4242 - acc: 0.8006
Epoch 3/100
8000/8000 [==============================] - 1s 115us/step - loss: 0.4172 - acc: 0.8054
Epoch 4/100
8000/8000 [==============================] - 1s 106us/step - loss: 0.4121 - acc: 0.8290
Epoch 5/100
8000/8000 [==============================] - 1s 114us/step - loss: 0.4094 - acc: 0.8322
Epoch 6/100
8000/8000 [==============================] - 3s 343us/step - loss: 0.4074 - acc: 0.8357
Epoch 7/100
8000/8000 [==============================] - 1s 135us/step - loss: 0.4062 - acc: 0.8361
Epoch 8/100
8000/8000 [==============================] - 1s 131us/step - loss: 0.4039 - acc: 0.8371
Epoch 9/100
8000/8000 [==============================] - 1s 127us/step - loss: 0.4032 - acc: 0.8354
Epoch 10/100
8000/8000 [==============================] - 1s 122us/step - loss: 0.4021 - acc: 0.8400
Epoch 11/100
8000/8000 [==============================] - 3s 373us/step - loss: 0.4015 - acc: 0.8384
Epoch 12/100
8000/8000 [==============================] - 1s 123us/step - loss: 0.4000 - acc: 0.8395
```

APPLICATIONS

Object detection

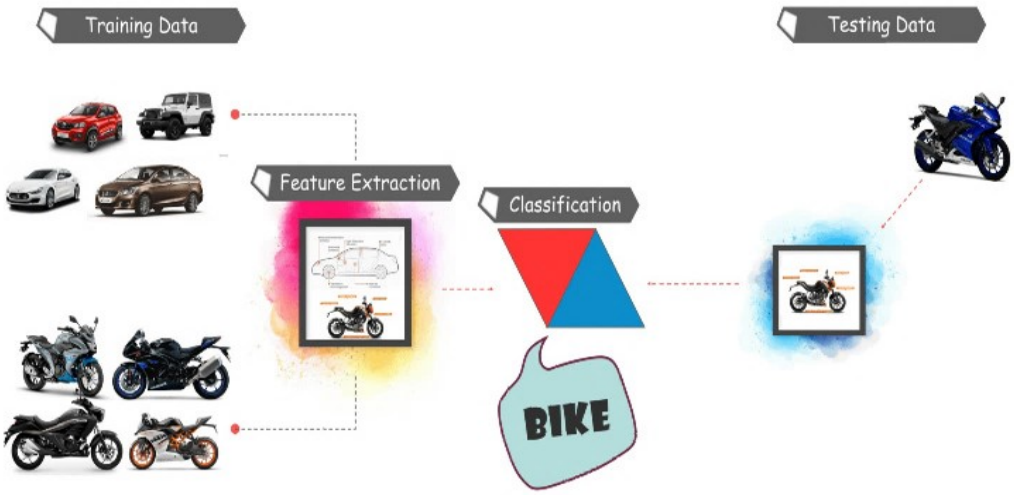
Object detection using TF

- Making accurate models to predict an object from an image was a challenging task but with the advancement of deep learning makes this task easy.
- Object detection API is a framework works on top of TF.
- It is the process of finding real time objects into an image.
- Ways of object detection
  - Feature based
  - Jones Viola
  - SVM based
  - Deep learning
- In the above methods Deep learning provides the best results with higher accuracy and higher speed.

Applications

- Face detection
- People counting
- Driverless car
- Security system
- Industrial process management

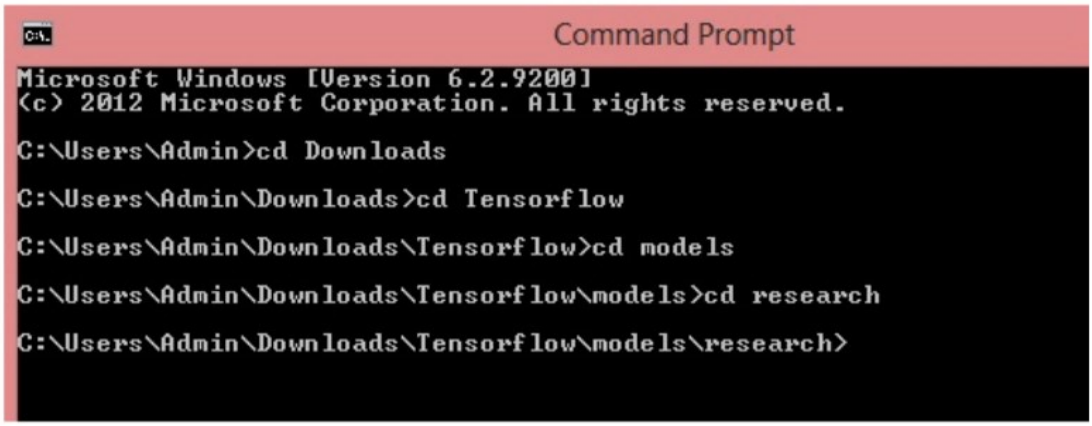
Work Flow



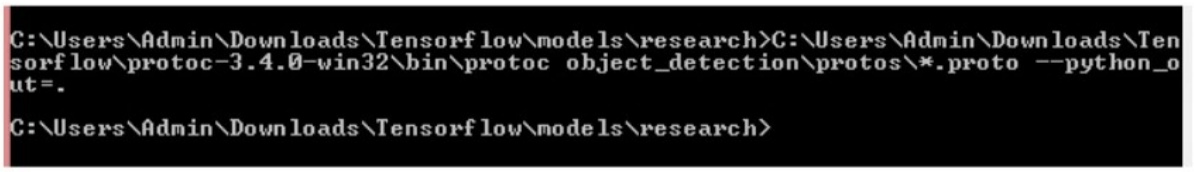
Object detection: Case study

Settings up the environment for object detection

- Download libraries using conda and pip package.
  - Pip install tensorflow
  - Pip install tensorflow-gpu
  - Pip install pandas
  - Pip install matplotlib
  - Pip install xlrd
- Download protobuf above 3.4.0 version 32bit to compile the protoc files.
- Clone and download tensorflow models from github.
- Open command prompt and reach to “path of tensorflow model/model-masters/research/” using cd as shown.



- Now write the path where we have protobuf upto bin folder and then write \proto object_detection\protos*.proto –python_out=. If this command execute without showing any error as shown in the image it means the equivalent file of proto is created in python environment.



- Common objects in context data set if use to train our system. To download the data set go to "http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v1_coco_11_06_2017.tar.gz"
- Now it is time to write the code for object detection. Open python tool and save a file in object_detection folder in tensorflow models. Write the below code in it and execute. Make sure that webcam is working fine.

Import libraries for object detection

```
import numpy as np
import os
import sys
import tarfile
import tensorflow as tf
import cv2
from utils import label_map_util
from utils import visualization_utils as vis_util
```

Make an object of webcam using VideoCapture in OPEN-CV.

```
cap = cv2.VideoCapture(0)
```

Using sys path set file path to the permanent system path

```
sys.path.append("..")
```

Extract the training data which is downloaded in ".tar.gz" format and access the file members

```
tar_fil = tarfile.open('C:/Users/Admin/Downloads/ssd_mobilenet_v1_coco_11_06_2017.tar.gz')
for file in tar_fil.getmembers():
  file_name = os.path.basename(file.name)
  if 'frozen_inference_graph.pb' in file_name:
    tar_fil.extract(file, os.getcwd())
```

Create a tensorflow graph object and set the model set.

```
detection_graph = tf.Graph()
with detection_graph.as_default():
  od_graph = tf.GraphDef()
  with tf.gfile.GFile('ssd_mobilenet_v1_coco_11_06_2017/frozen_inference_graph.pb', 'rb') as fid:
    serialized_graph = fid.read()
    od_graph.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph, name='')
```

A category index is created containing multiple objects for training purpose

```
label_map = label_map_util.load_labelmap(os.path.join('data', 'mscoco_label_map.pbtxt'))
```

```
categories =
label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=90, use_display_name=True)
category_index =
label_map_util.create_category_index(categories)
```

The detection graph object created is set as default and and taken for tensorflow session

```
with detection_graph.as_default():
  with tf.Session(graph=detection_graph) as sess:
```

Implement the next part of program in infinite loop for continuously capturing the frames from the video for infinite number of times.

```
while True:
        ret, image_np = cap.read()
```

Dimension of an image is expended as the image expected will have the shape [1, None, None, 3]

```
      image_np_expanded = np.expand_dims(image_np, axis=0)
       image_tensor =
detection_graph.get_tensor_by_name('image_tensor:0')
```

A kind of image segmentation is happening in this line as each part of an image is taken as box and it represent a particular object.

```
boxes =
detection_graph.get_tensor_by_name('detection_boxes:0')
```

On every image a value got display that value if the confidence of machine for each object. If with confirmation machine can detect that object then the confidence value remains high

```
scores =
detection_graph.get_tensor_by_name('detection_scores:0')
       classes =
detection_graph.get_tensor_by_name('detection_classes:0')
```

```
    num_detections =
detection_graph.get_tensor_by_name('num_detections:0')
```

Actual detection of object

```
(boxes, scores, classes, num_detections) = sess.run(
        [boxes, scores, classes, num_detections],
        feed_dict={image_tensor: image_np_expanded})
```

Visualization of the result of detection

```
vis_util.visualize_boxes_and_labels_on_image_array(
          image_np,
          np.squeeze(boxes),
          np.squeeze(classes).astype(np.int32),
          np.squeeze(scores),
          category_index,
          use_normalized_coordinates=True,
          line_thickness=8)
```


Show the image with detected objects using OPEN-CV. Using wait key function the closing of the process makes dependent on a key ‘q’.

```
cv2.imshow('object detection', cv2.resize(image_np, (640,480)))
if cv2.waitKey(1) & 0xFF == ord('q'):
        cap.release()
        cv2.destroyAllWindows()
        break
```

Chat-bot

Introduction to chat-bot

- It is the application of NLP in which the concept of chatting with the robot is discussed.
- In chat-bot the main challenge is to understand a robot language.

- It contains conversion of a voice in the format of a text.
- It uses regular expression matching to match for the results.

Chat-bot: case study

Settings up the environment for Chat-bot

- Download libraries using conda and pip package.
  - from nltk.chat.util import Chat, reflections
- Now it is time to write the code for chat-bot.

Import library for chat-bot

```
from nltk.chat.util import Chat, reflections
```

Let involve some of the questions with their solution we want from our chat bot. r in every question is representing raw string.

```
chats = [
    [
        r"my name is (.*)",
        ["Hello %1, How are you today ?",]
    ],
     [
        r"what is your name ?",
        ["My name is chatter and I'm a chatbot ?",]
    ],
    [
```

```
    r"how are you ?",
    ["I'm doing good\nHow about You ?",]
  ],
  [
    r"sorry (.*)",
    ["Its alright","Its OK, never mind",]
  ],
  [
    r"i'm (.*) doing good",
    ["Nice to hear that","Alright :)",]
  ],
  [
    r"hi|hey|hello",
    ["Hello", "Hey there",]
  ],
  [
    r"(.*) age?",
    ["I'm a computer program dude\n Seriously you are asking me this?",]

  ],
  [
    r"what (.*) want ?",
    ["Make me an offer I can't refuse",]
      ],
  [
    r"(.*) created ?",
    ["LiveWire created me using Python's NLTK library ","dnt share it with anyone ;)",]
```

```
    ],
    [
        r"(.*) (location|city) ?",
        ['Jaipur, Rajasthan',]
    ],
    [
        r"how is weather in (.*)?",
        ["Weather in %1 is awesome like always","Too hot man here in %1","Too cold man here in %1","Never even heard about %1"]
    ],
    [
        r"i work in (.*)?",
        ["%1 is an Amazing company, I have heard about it. But they are in huge loss these days.",]
    ],
[
        r"(.*)raining in (.*)",
        ["No rain from past two week %2","It’s too much raining %2"]
    ],
    [
        r"how (.*) health(.*)",
        ["I am a computer program so I am healthy always",]
    ],
    [
        r"(.*) (sports|game) ?",
        ["I'm a very big fan of cricket",]
    ],
    [
        r"who (.*) sportsperson ?",
```

```
        ["Dhoni","Kohli","Ma long"]
],
    [
        r"who (.*) (moviestar|actor)?",
        ["Salman khan"]
],
    [
        r"exit",
        ["BBye take care. See you soon :) ","It was nice talking to you.:)"]
],
]
```

This is the function created to initialize chat-bot and start the conversation till exit return by the user.

```
def chatter():
    print("Hi, I'm chatter and I chat alot ;)\nPlease type lowercase English language to start a conversation. Type quit to leave ") #default message at the start
    chat = Chat(chats, reflections)
    chat.converse()
```

It is a special method defines in python when interpreter executes the main module/program. This special method is utilized to call chatter function.

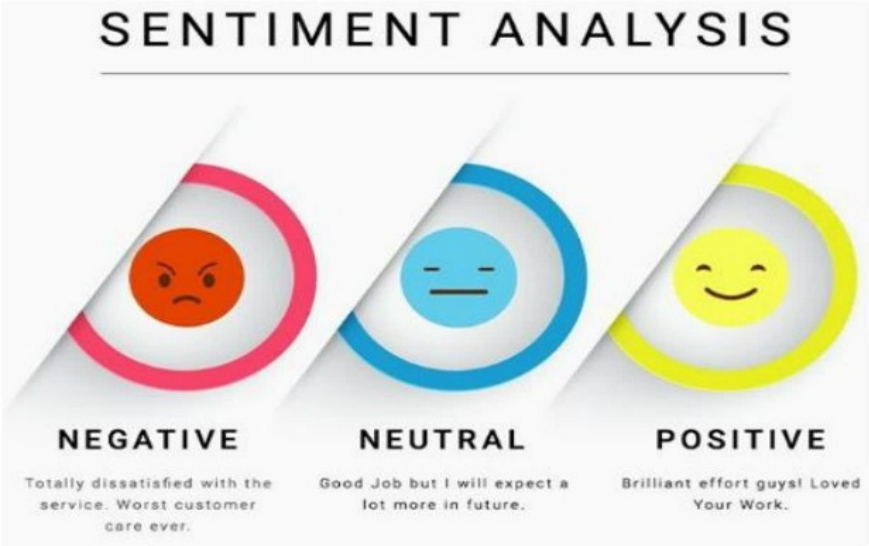
```
if __name__ == "__main__":
    chatter()
```

Sentimental analysis

Introduction to sentimental analysis

- Sentimental analysis is a common task of NLP that data scientist use to perform a specific task.

- This will going to provides a response on the basis of the text we are giving.
- The data set we are using is of movie reviews containing 50,000 review of positive type and negative type. We are going to divide the data in training (25k) and testing set(25k). Now this 25k will contain 12.5k positive reviews and 12.5k negative reviews.



Sentimental analysis: case study

Settings up the environment for sentimental analysis

- Download libraries using conda and pip package.
  - Pip install nltk
    - After downloading it on python console write “import nltk” and then write “nltk.download()”. This will open a window where multi package of nltk were there so they are require to download. These packages contain the dataset and supporting library for nltk.
  - Pip install pandas
  - Pip install matplotlib
  - Pip install xlrd
  - Pip install numpy
  - Pip install sklearn
- Now it is time to write the code for sentimental analysis.

Import the libraries

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

```
from sklearn.feature_extraction.text import CountVectorizer
import os
import nltk
print(os.listdir(nltk.data.find("corpora")))
from nltk.corpus import movie_reviews
```

Check the number of categories in movie dataset and check the file ids for positive and negative reviews

```
print(movie_reviews.categories())
print(movie_reviews.fileids('pos'))
print(movie_reviews.fileids('neg'))
```

Collect the positive reviews words into a variable named as filename

```
filename=nltk.corpus.movie_reviews.words('pos/cv000_29590.txt')
```

Collect the words for negative and positive reviews and separate unused spaces from it. Finally store these words into an empty list.


```
dataSet_reviews=[]
neg_re=nltk.corpus.movie_reviews.fileids('neg')
for l in neg_re:
    rev_text=l=nltk.corpus.movie_reviews.words(l)
    rev_one=" ".join(rev_text)
    rev_one=rev_one.replace(' ,',',')
    rev_one=rev_one.replace(' .','.')
    rev_one=rev_one.replace('\"',"'")
    rev_one=rev_one.replace(' \"',"'")
    dataSet_reviews.append(rev_one)
posi=nltk.corpus.movie_reviews.fileids('pos')
for l in posi:
    rev_text=l=nltk.corpus.movie_reviews.words(l)
    rev_one=" ".join(rev_text)
```

```
rev_one=rev_one.replace(' ,',',')
rev_one=rev_one.replace(' .','.')
rev_one=rev_one.replace('\"',"'")
rev_one=rev_one.replace(' \"',"'")
dataSet_reviews.append(rev_one)
```

Now it's time to define the target variable. For negative reviews zero will be pass and for positive reviews one will be pass.

```
NegTar=np.zeros((1000,),dtype=np.int)
PosTar=np.ones((1000,),dtype=np.int)
Target_list=[]
for i in NegTar:
    Target_list.append(i)
for i in PosTar:
    Target_list.append(i)
```

Convert the content into pandas series and check the top most entries.

```
Y=pd.Series(Target_list)
Y.head()
```

Now it's time to represent the text data into numerical format. Using it we will create a vector with dimensionality equal to the size of our vocabulary. Result of count vectorizer is very large vector. This will not going to provide any information about relationship information. The functionality is somewhere related to one hot encoder in dummies creation.

```
from sklearn.feature_extraction.text import CountVectorizer
Count_vect=CountVectorizer(lowercase=True,stop_words='english',min_df=2)
X_count_vect=Count_vect.fit_transform(dataSet_reviews)
X_count_vect.shape  #this gives the dimension of the data
X_names=Count_vect.get_feature_names()
```

```
X_count_vect=pd.DataFrame(X_count_vect.toarray(),columns=X_
names) #Still dimension remain same
X_count_vect.head()
```

Using train_test_split library the dataset is splitting into training set and testing set. Imported confusion matrix to find the relationship between actual and predicted results.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn import metrics
X_train,X_test,y_train,y_test=train_test_split(X_count_vect,Y,test_
size=0.2)
```

Using GaussianNB as the problem is classification type. Gnb is the object created for GaussianNB.

```
from sklearn.naive_bayes import GaussianNB
Gnb=GaussianNB()
```

Fit the input and output training set using fit method. Predict for the test data. Finally a confusion matrix is drawn

```
Gnb.fit(X_train,y_train)
Y_pred=Gnb.predict(X_test)
confG=confusion_matrix(y_test,Y_pred]
```

Using MultinomialNB as the problem is classification type. Clf is the object created for MultinomialNB.

```
from sklearn.naive_bayes import MultinomialNB
Clf=MultinomialNB()
```

Fit the input and output training set using fit method. Predict for the test data. Finally a confusion matrix is drawn

```
Clf.fit(X_train,y_train)
Y_pred_clf=Clf.predict(X_test)
#print(metrics.accuracy_score(y_test,Y_pred))
confM=confusion_matrix(y_test,Y_pred_clf)
```

Skilling India Since 1988

Now it's official!

Certificate No.: NSDC/2018-19/4326

N·S·D·C National Skill Development Corporation

Transforming the skill landscape

Certificate of Partnership

This is to certify that

CADD Centre Training Services Pvt. Ltd.

is an **Approved Training Partner** of
National Skill Development Corporation (NSDC)
for the financial year 2018-19

Date: 1 April, 2018

Skill India
कौशल भारत-कुशल भारत

MANISH KUMAR
MD and CEO

301, 3rd Floor, West Wing, World Mark 1, Aerocity, New Delhi - 110037
T: +91-11-47451600 • F: +91-11-46580417 • E-mail: mdtr@nsdcindia.org | CIN- U85300DL2008NPL181612

To verify NSDC training partner details, visit: www.nsdcindia.org or write to monitoring@nsdcindia.org

CADD Centre
is now
NSDC
Approved
Training
Partner

CADD Centre
has won the prestigious
"Best International Organization in Skill Development 2017" by ASSOCHAM

ASSOCHAM
INDIA

Summit-cum-Awards on Skilling India

From Skills to Prosperity

2016-2017

On advise of Members of the Jury

Awarded Gold Trophy to

CADD Centre Training Services Private Limited

in recognition of outstanding contribution and certificate of merit in

Best International Organization in Skill Development

09th August, 2017
New Delhi

D.S. Rawat
Secretary General
ASSOCHAM

LIVEWIRE is a division of CADD Centre, a NSDC approved Training Partner

No. 91, Dr. Radhakrishanan Salai, 8th Floor.
GEE GEE Crystal, Office No. 8C & 8D, Mylapore, Chennai 600 004.

1800 425 0220 | Info@livewireindia.com | www.livewireindia.com

LIVEWIRE™
FOR LIVE CAREERS