केन्द्रीय विद्यालय संगठन, जयपुर संभाग

KENDRIYA VIDYALAYA SANGATHAN
JAIPUR REGION

STUDY MATERIAL

CLASS– 12

SUBJECT—INFORMATICS PRACTICES

SESSION : 2021-22 (Term-I)

CHIEF PATRON

Sh. B L Morodia
Deputy Commissioner
KVS (RO), Jaipur

PATRON

Shri D R Meena
Assistant Commissioner
KVS(RO), Jaipur

CONTENT COORDINATOR/COURSE DIRECTOR

Sh. Narsi Lal
Principal
KV Suratgarh Cantt

Coordinators:

Sh. Mahendra Sethi, PGT Computer Science, KV Suratgarh Cantt
Sh. Pankaj Singh, PGT Computer Science, KV Bharatpur
Sh. Ashish Kumar Joshi, PGT Computer Science, KV Baran

GROUP	NAME OF PGT COMPUTER SCIENCE	NAME OF KV	WORK ASSIGNED FOR 3 DAYS WORKSHOP
GROUP-1	MANISH SONI	NO. 1 AFS SURATGARH	Introduction to Python libraries- Pandas, Matplotlib. ● Data structures in Pandas - Series and data frames.Series: Creation of series from dictionary, scalar value; mathematical operations; series attributes, head and tail functions; selection, indexing and slicing.
	MS. KULDEEP KAUR	NO.2 AFS JODHPUR	
	V D MEENA	AVIKANAGAR	
	DR AJAY KUMAR GARG	K V NO 3 NAL BIKANER	
	PRADEEP SWAMI	JHUNJHUNU	
GROUP-2	VISHAL GOSWAMI	NO1 BIKANER	Data Frames: creation of data frames from dictionary of series, list of dictionaries, text/CSV files, display, iteration. Operations on rows and columns: add (insert /append) , select, delete (drop column and row), rename, Head and Tail functions, indexing using labels, Boolean indexing.
	SATISH CHANDRA JANGIR	KV NO. 3 JAIPUR	
	PRITHVI RAJ CHOUHAN	KV NO.1 AFS JODHPUR	
	MRS. NIPUN KALRA WALIA	K V NO 6 JAIPUR	
	RAJESH SUYAL	ITARANA	
GROUP-3	NEHA TYAGI	KV NO 3 JAIPUR	Data Visualization ● Data Visualization : Purpose of plotting, drawing and saving of plots using Matplotlib (line plot, bar graph, histogram). Customizing plots:; adding label, title, and legend in plots.
	MRS. MAMTA JAIN	BHILWARA	
	ADARSH BHATNAGAR	KV NO.2, BIKANER	
	MR. AAKIB JAVED	BSF JODHPUR	
	PREM PRAKASH MEENA	ALWAR	
GROUP-4	BIRBAL JAT	DABLA	Digital footprint, net and communication etiquettes, ● Data protection, intellectual property rights (IPR), plagiarism, licensing and copyright, ● Free and open source software (FOSS)
	PANKAJ MEHRA	KV JHALAWAR	
	P KACHHAWA	KV NO 2 AJMER	
	SH VIJAY KUMAR GARG	KV GANGAPUR CITY	
	PINKY KUMARI MEENA	KV NO2 ARMY JODHPUR	
GROUP-5	AMIT KUMAR JAIN	NO.4 JAIPUR	Cybercrime and cyber laws, hacking, phishing, cyber bullying, overview of Indian IT Act. ● E-waste: hazards and management. Awareness about health concerns related to the usage of technology.
	GHANSHYAM CHITARA	AFS UTTARLAI	
	VIKRAM SINGH PAREVA	KV CHITTORGARH	
	GAJRAJ MEENA	KV KARAULI	
	KAVITA ACHARYA	KV BANSWARA	
GROUP-6	SANDEEP ARORA	KENDRIYA VIDYALAYA NO.1 UDAIPUR	3 Sample Question Paper for Term-I as per CBSE pattern
	MR. ARVIND KUMAR	KV NO. 1, JAIPUR	
	SH. P. R. GOLIA	KV NASIRABAD	
	MRS. PREETI MEHARISHI	KV AFS JAISALMER	
	VIJETA DARA	NO 5 (I SHIFT) JAIPUR	
GROUP-7	DILIP SINGH	BANAR JODHPUR	3 Sample Question Paper for Term-I as per CBSE pattern
	SH. PRAVEEN KUMAR YADAV	SAWAI MADHOPUR	
	USHA BENIWAL	K V NO 2, JAIPUR	
	KRISHAN KUMAR KUMAWAT	KV 1 AJMER	
	NAVNEET	KENDRIYA VIDYALAYA CHURU	

Informatics Practices

CLASS XII

Term - 1 Distribution of Theory Marks

Unit No	Unit Name	Marks
1	Data Handling using Pandas and Data Visualization	25
4	Societal Impacts	10
	Total	35

Unit 1: Data Handling using Pandas and Data Visualization

Data Handling using Pandas -I

● **Introduction to Python libraries**- Pandas, Matplotlib.

● **Data structures in Pandas** - Series and data frames.Series: Creation of series from dictionary, scalar value; mathematical operations; series attributes, head and tail functions; selection, indexing and slicing.

● **Data Frames**: creation of data frames from dictionary of series, list of dictionaries, text/CSV files, display, iteration. Operations on rows and columns: add (insert /append) , select, delete (drop column and row), rename, Head and Tail functions, indexing using labels, Boolean indexing.

Data Visualization

● **Data Visualization :** Purpose of plotting, drawing and saving of plots using Matplotlib (line plot, bar graph, histogram). Customizing plots:; adding label, title, and legend in plots.

Unit 4: Societal Impacts

● Digital footprint, net and communication etiquettes,

● Data protection, intellectual property rights (IPR), plagiarism, licensing and copyright,

● Free and open source software (FOSS),

● Cybercrime and cyber laws, hacking, phishing, cyber bullying, overview of Indian IT Act.

● E-waste: hazards and management. Awareness about health concerns related to the usage of technology.

INDEX:

Python Pandas Series & DataFrame

Python Pandas:

Pandas is most popular library. It provides various functions related to Scientific Data analysis, like

Pandas is Python's library for data analysis. Pandas has derived its name from PANel DAta System. Pandas developed by Wes McKinney

- It can read and write different data formats like int, float, double
- It can calculate data that is organized in row and columns.
- It can select sub set of data values and merge two data sets.
- It can support reshape of data values.
- It can support visualization library matplotlib.

Data Structure:

Pandas Data Structure is a way to store & organize data values in a specific manner so that various specific functions can be applied on them. Examples- array, stack, queue, linked list, series, DataFrame etc.

"Series" Vs "DataFrame"

Property	Series	DataFrame
Dimensions	One-Dimensional	Two-Dimensional
Types of data	Homogenous (In Series, all data values should be of same type)	Heterogeneous (In DataFrame, data values may be of different type)
Value Mutable	Yes, Mutable	Yes, Mutable
Size Mutable	Size is Immutable. Once the size of series created, it cannot be changed. (If add/delete element, then new series object will be created.)	Size is Mutable. Once the size of DataFrame created, it can be changed.

"Series" Data Structure:

A Series is a Pandas Data Structure that represent 1–Dimensional array of indexed data. The series structure contains two parts. It requires to import pandas and numpy package.

1. An array of actual data values
2. An associated array of indexes (Used to access data values)

Creation of Series:

A series of object can be created by using many ways. Like

1. Creation of empty series by using Series()
2. Creation of non- empty series with Series()

1. Creation of empty series:

Syntax:

Series_object = pandas.Series()
# S is capital in Series()

Example:

```
import pandas
Ser_obj1 = pandas.Series()
# It will create an empty series of float type.
```

2. Creation of Non empty series

Syntax:

Series_object = pandas.Series(data, index=idx)

Where data is array of actual data value of series. Index is any valid numpy datatype. Index can be any type of following.

- A Python sequence
- An nd array
- A Python dictionary
- A scalar value

Example:

```
Ser_obj2 = pandas.Series([1,3,5])
```

Output:

```
0    1
1    3
2    5
```

```
Ser_obj3 = pandas.Series([1.5,3.5,5.5])
```

Output:

```
0    1.5
1    3.5
2    5.5
```

Creation of Series for various Objects:

Series of List (Integer values)

```
import pandas as pd
S1=pd.Series([2,4,6])
print(" Series of Object-1")
print(S1)
```

```
Series of Object-1
0    2
1    4
2    6
```

Series of Tuple (Integer values)

```
import pandas as pd
S2=pd.Series((20,40,60))
print(" Series of Object-2")
print(S2)
```

```
Series of Object-2
0    20
1    40
2    60
```

Series of List (Character values)

```
import pandas as pd
S3=pd.Series(['K','V','S'])
print(" Series of Object-3")
print(S3)
```

```
Series of Object-3
0    K
1    V
2    S
```

Series of List (string value)

```
import pandas as pd
S4=pd.Series(["KVS JJN"])
print(" Series of Object-4")
print(S4)
```

```
Series of Object-4
0    KVS JJN
```

Series of List (String values)

```
import pandas as pd
S5=pd.Series(["KVS","JJN"])
print(" Series of Object-5")
print(S5)
```

```
Series of Object-5
0    KVS
1    JJN
```

Series of array using arange()

```
import pandas as pd
import numpy as np
nd1=np.arange(3, 13, 3.5)
S6=pd.Series(nd1)
print(" Series of Object-6")
print(S6)
```

```
Series of Object-6
0    3.0
1    6.5
2    10.0
```

Series of array using linspace()

```
import pandas as pd
import numpy as np
nd2=np.linspace(24, 64, 5)
S7=pd.Series(nd2)
print(" Series of Object-7")
print(S7)
```

```
Series of Object-7
0    24.0
1    34.0
2    44.0
3    54.0
4    64.0
```

Series of dictionary

```
import pandas as pd
import numpy as np
S8=pd.Series({'Jan':31,
'Feb':28,'Mar":31})
print(" Series of Object-8")
print(S8)
```

```
Series of Object-8
Feb    28
Jan    31
Mar    31
```

Series using range()

```
import pandas as pd
S9=pd.Series(10,
index=range(0,3))
print(" Series of Object-9")
print(S9)
```

```
Series of Object-9
0    10
1    10
2    10
```

Series of scalar values using user defined index

```
import pandas as pd
S11=pd.Series(20,
index=['Raj','PB','HR'])
print(" Series of Object-11)
print(S11)
```

```
Series of Object-11
Raj    20
PB     20
HR     20
```

Series of NaN (Not a Number) values

```
import pandas as pd
import numpy as np
S12=pd.Series([9.5,np.NaN,5.5])
print(" Series of Object-12)
print(S12)
```

```
Series of Object-12
0    9.5
1    NaN
2    5.5
```

Series of None values

```
import pandas as pd
import numpy as np
S13=pd.Series([9.5,np.None,
5.5])
print(" Series of Object-13)
print(S13)
```

```
Series of Object-13
0    9.5
1    None
2    5.5
```

Series by using for loop

```
import pandas as pd
import numpy as np
ind=x for x in 'ABCDE'
S15=pd.Series(range(1,15,3),
index=ind)
print(" Series of Object-14)
print(S14)
```

```
Series of Object-14
A    1
B    4
C    7
D   10
E   13
```

Series() Special examples

```
import pandas as pd
import numpy as np
arr=np.array([31,28,31,30])
day=['Jan','Feb','Mar','Apr']
S15=pd.Series(data=arr,inde
x=day, dtype=np.float64)
print("Series of Object-15")
print(S15)
```

```
Series of Object-15
Jan    31.0
Feb    28.0
Mar    31.0
Apr    30.0
```

Series() Special examples

```
import pandas as pd
import numpy as np
a=np.arange(9,13)
S16=pd.Series(index=a,
data=a**2)
print("Series of Object-16")
print(S16)
```

```
Series of Object-16
9     81
10    100
11    121
12    144
```

Series() Special examples

```
import pandas as pd
import numpy as np
lst=[9,10,11]
S17=pd.Series(data=lst*2)
print("Series of Object-17")
print(S17)
```

```
Series of Object-17
0    9
1    10
2    11
3    9
4    10
5    11
```

Attributes of Series Object

Attribute	Description
Series_object.index	It show the indexes of series object
Series_object.values	It show the nd-array values of series object

Series_object.dtype	It show the data types of data values of series object
Series_object.shape	It show tuple of shape underlying data of series object
Series_object.nbytes	It show the number of bytes of underlying data of series object
Series_object.ndim	It show the number of dimensions of underlying data of series object
Series_object.size	It show the number elements in series object
Series_object.itemsize	It show the size of data type of underlying data of series object
Series_object.hasnans	It show True if there is NaN / None value in Series, otherwise returns False.
Series_object.empty	It returns True if series is empty, otherwise returns False.

Example with Attribute	**Output**
# Example of Series import numpy as np import pandas as pd Ind=['Jan','Feb','Mar','Apr'] Val=[31,28,31,30] Sr_Obj=pd.Series(data=Val, index=Ind)	
print(Sr_Obj.index)	Index(['Jan', 'Feb', 'Mar', 'Apr'], dtype='object')
print(Sr_Obj.values)	[31 28 31 30]
print(Sr_Obj.dtype)	int64
print(Sr_Obj.itemsize)	8
print(Sr_Obj.size)	4
print(Sr_Obj.ndim)	1
print(Sr_Obj.empty)	False
print(Sr_Obj.hasnans)	False
print(Sr_Obj.nbytes)	32
print(Sr_Obj.shape)	(4,)

Accessing individual element of Series Syntax: Series_Object[Valid index]	
import numpy as np import pandas as pd Ind=['Jan','Feb','Mar','Apr'] Val=[31,28,31,30] Sr_Obj=pd.Series(data=Val, index=Ind)	
# print Whole series print(Sr_Obj)	Jan 31 Feb 28 Mar 31 Apr 30 dtype: int64
print(Sr_Obj['Feb'])	28
print(Sr_Obj['Apr'])	30

Accessing Slice of Series

Slicing takes place position wise (built in Index) and not the index wise in a series object.

Syntax: Series_Object[Start: End: Step]

Where,

Start is Lower Limit (default is 0)

End is Upper Limit

Step is updation (default is 1)

Note: slicing may be –ve also

print(Sr_Obj[1:3:1])	Feb 28 Mar 31
print(Sr_Obj[-1:-3:-1])	Apr 30 Mar 31
print(Sr_Obj[1::])	Feb 28 Mar 31 Apr 30
print(Sr_Obj[::1])	Jan 31 Feb 28 Mar 31 Apr 30
print(Sr_Obj[::-1])	Apr 30 Mar 31 Feb 28 Jan 31

Modifying Elements of of Series

Syntax: Series_Object[index / slice]= new value

Sr_Obj[1]=29 print(Sr_Obj)	Jan 31 Feb 29 Mar 31 Apr 30
Sr_Obj[:-3:-1]=31 print(Sr_Obj)	Change 31 in Last 2 place Jan 31 Feb 29 Mar 31 Apr 31
print("Add New element 100") Sr_Obj['May']=100 print(Sr_Obj)	Add New element 100 Jan 31 Feb 29 Mar 31 Apr 31 May 100
print("Delete Last index") del Sr_Obj['May'] print(Sr_Obj)	Delete Last index Jan 31 Feb 29

	Mar 31 Apr 31
print("Rename Index") Sr_Obj.index=['J','F','M','A'] print(Sr_Obj)	Rename Index J 31 F 29 M 31 A 31

head( ) and tail( )

head( ) returns first n rows and tail( ) returns last n rows from series.
If n is not given then by default it will return 5 rows.

Sytax:
Series_Object.head([n])
Series_Object.tail([n])

```
import numpy as np
import pandas as pd
Ind=['Jan','Feb','Mar','Apr','May','Jun','Jul']
Val=[31,28,31,30,31,30,31]
Sr_Obj=pd.Series(data=Val, index=Ind)
```

print("Display First 2 Rows") print(Sr_Obj.head(2))	Display First 2 Rows Jan 31 Feb 28
print("Display First 5 Rows") print(Sr_Obj.head())	Display First 5 Rows Jan 31 Feb 28 Mar 31 Apr 30 May 31
print("Display First 6 Rows") print(Sr_Obj.head(6))	Display First 6 Rows Jan 31 Feb 28 Mar 31 Apr 30 May 31 Jun 30
print("Display Last 2 Rows") print(Sr_Obj.tail(2))	Display Last 2 Rows Jun 30 Jul 31
print("Display Last 5 Rows") print(Sr_Obj.tail())	Display Last 5 Rows Mar 31 Apr 30 May 31 Jun 30 Jul 31
print("Display Last 6 Rows") print(Sr_Obj.tail(6))	Display Last 6 Rows Feb 28 Mar 31 Apr 30 May 31 Jun 30 Jul 31

Vector operations on Series Object

Similar to nd-array, the vector operations can be applied on series object also. Scalar operation mean, one operation can be applied to each element of series object at a time.

```
import pandas as pd
import numpy as np
Sr_Obj=pd.Series(index=['A' , 'B' , 'C' , 'D'],
data=[10,20,30,40])
```

print("Add 5 in each element of Sr_Obj") print(Sr_Obj+5)	Add 5 in each element of Sr_Obj A 15 B 25 C 35 D 45
print("Multiply by 5 in each element of Sr_Obj") print(Sr_Obj*5)	Add 5 in each element of Sr_Obj A 50 B 100 C 150 D 200
print("Divide 5 in each element of Sr_Obj") print(Sr_Obj/5)	Add 5 in each element of Sr_Obj A 2.0 B 4.0 C 6.0 D 8.0
print(Sr_Obj>20)	A False B False C True D True
print("Sr_Obj**2") print(Sr_Obj**2)	A 100 B 400 C 900 D 1600

#Adding two Series of similar indexes

```
import numpy as np
import pandas as pd
class11=pd.Series(data=[30,40,50],index=['scien
ce','arts','commerce'])
class12=pd.Series(data=[60,80,100],index=['scie
nce','arts','commerce'])
```

```
print("Total number of students")
print(class11+class12)

Output:
Total number of students
science    90
arts       120
commerce   150
```

#Adding two Series of dissimilar indexes

```
class11=pd.Series(data=[30,40,50],index=['science','arts','commerce'])
class12=pd.Series(data=[60,80,100],index=['sci','arts','commerce'])
print("Total number of students")
print(class11+class12)

Output:
Total number of students
arts       120.0
commerce   150.0
sci        NaN
science    NaN
```

Filtering Entries of Series

```
import pandas as pd
Info=pd.Series(data=[31,41,51])

print("info>40\n", info>40)

print("info[info>40]\n", info[info>40])
```

```
info>40
0    False
1    True
2    True
info[info>40]
1    41
2    51
```

Sorting Series Values based on Values

```
import pandas as pd
import numpy as np
Sr_Obj=pd.Series(index=['A' , 'B' , 'C' , 'D'],
data=[200,400,300,100])
Sr_Obj.sort_values()  OR
Sr_Obj.sort_values(ascending= True)
```

```
D    100
A    200
C    300
B    400
(By default order is Ascending)
```

```
Sr_Obj.sort_values(ascending= False)
```

```
B    400
C    300
A    200
D    100
```

Sorting Series Values based on Indexes

```
Sr_Obj.sort_index()  OR
Sr_Obj.sort_index(ascending= True)
```

```
A    200
B    400
C    300
D    100
```

```
Sr_Obj.sort_index(ascending= False)
```

```
D    100
C    300
B    400
A    200
```

Arithmetic on Series

```
import pandas as pd
import numpy as np
s1=pd.Series(data=[20,40,60], index=['A','B','C'])
s2=pd.Series(data=[2,4,6], index=['A','B','C'])
print("Addition of Series: s1+s2")
print(s1+s2)
```

```
Addition of Series-s1+s2:
A  22
B  44
C  66
```

```
print("Division of Series: s1/s2")
print(s1/s2)
```

```
Division of Series: s1/s2
A  10.0
B  10.0
C  10.0
```

```
print("Addition of Series: S3=s1+s2")
s3=s1+s2
print(s3)
```

```
Addition of Series: S3=s1+s2
A  22
B  44
C  66
```

NumPy Arrays Vs Series Object

1. In ndarray, vector operations can only be performed if shape of both array match, otherwise it will generate error.
2. In Series, vector operations can have performed with different shapes series also. For different shape series operation gives NaN values.
3. In ndarray, the indexes always numeric and start with 0 onwards. But in series, indexes can have any type of indexes.

DATAFRAME –

A Data frame is a two- dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

Features are:

- Two-dimensional
- size-mutable &
- data mutable
- Contains heterogeneous data
- Contains rows and columns index
- The DataFrame contains labelled axes (rows or axis = 0 and columns or axis = 1).
- All elements within a single column have the same data type, but different columns can have different data types.

Have a look to know the 2- D form representation of a DataFrame -

Row ↓	Column ↓ Roll /0	Column ↓ Name/1	Column ↓ Mark /2
FIRST/0	D[0][0]	D[0][1]	D[0][2]
SECOND/1	D[1][0]	D[1][1]	D[1][2]
THIRD/2	D[2][0]	D[2][1]	D[2][2]

For using the DataFrame object we must import the pandas library as below:

import pandas OR import pandas as ALIAS-NAME

Creating a DataFrame

Mainly DataFrame() function of pandas library is used. There are different ways of creating a DataFrame using -

A - Empty DataFrame

Let us learn with help of an example to create and print a DataFrame.

```
import pandas
DF = pandas.DataFrame()
print(DF)
```

OUTPUT –

```
Empty DataFrame
Columns: []
Index: []
```

B - Dictionary of Series

Example 1 –

```
s = pandas.Series(100 , index =['a','b','c','d'])
print(s)
df = pandas.DataFrame(s)
print(df)
```

OUTPUT –

```
a    100
b    100
c    100
d    100
dtype: int64
```

```
     0
a  100
b  100
c  100
d  100
```

Since DataFrames are two-dimensional, to create DataFrame from Series, we can also take two or more Series objects to create a DataFrame.

Example 2 –

```
import pandas
roll = pandas.Series( [10, 12, 13, 16])
name = pandas.Series(['Aruna', 'Kavita',
'Gaurav','Sumit'])
DF = pandas.DataFrame( { 'Roll_No' : roll ,
'SName' : name } )
print(DF)
```

OUTPUT –

```
   Roll_No   SName
0       10   Aruna
1       12  Kavita
2       13  Gaurav
3       16   Sumit
```

Example 3 –

```
import pandas
s1 = pandas.Series( { 101 : 'Amit', 102 : 'Anita',
103:'Geetu', 105:'Jatin'})
s2 = pandas.Series( { 101 : 93 , 102 : 87 , 103 :
82 , 104 : 93 , 105 : 90 } )
dfs = pandas.DataFrame( {'Name' : s1 , 'Marks' :
s2 } )
print("Series 1")
print(s1)
print("Series 2")
print(s2)
print("data frame from the above series ")
print(dfs)
```

```
Series 1
101      Amit
102     Anita
103     Geetu
105     Jatin
dtype: object
Series 2
101     93
102     87
103     82
104     93
105     90
dtype: int64
data frame from the above series
       Name  Marks
101    Amit     93
102   Anita     87
103   Geetu     82
104     NaN     93
105   Jatin     90
```

C - List of Dictionary

Recall that dictionary is of the form { key1 : value1 , key2 : value2 , - - - }

The **keys** of the dictionary become the **column names** in the DataFrame object and the **values** of the dictionary become the **column-values** of the DataFrame object

Example 1–

```
import pandas
d1 = { 'roll' : 101 , 'name' : 'Astha' , 'tot_mark' : 456 }
d2 = {'roll' : 104 , 'name' : 'Gautam', 'tot_mark' : 478 }
d3 = {'roll' : 105 , 'name' : 'Deepika', 'tot_mark' : 453 , 'grade' : 'A2' }
L = [ d1 , d2, d3 ]
df_list = pandas.DataFrame(L)
print("Data Frame from list of dictionaries ")
print(df_list)
```

OUTPUT –

```
Data Frame from list of dictionaries
   roll     name  tot_mark grade
0   101    Astha       456   NaN
1   104   Gautam       478   NaN
2   105  Deepika       453    A2
```

As shown in the output, NaN (Not a Number) is automatically added for missing places.

Example 2 –

Instead of the default row labels: 0, 1, 2, 3, … we can specify our own row labels by using the index=[list_of _row_labels] parameters in the DataFrame() function.

```
import pandas
L = [ {'roll' : 101 , 'name' : 'Astha' } ,\
      {'roll' : 104 , 'name' : 'Gautam', 'mark' : 478} ]
DF = pandas.DataFrame( L , index= ['s1' , 's2'] )
print("DataFrame from List of Dictionaries with Row-Index")
print(DF)
```

OUTPUT –

```
DataFrame from List of Dictionaries with Row-Index
    roll    name   mark
s1   101   Astha    NaN
s2   104  Gautam  478.0
```

Example 3 –

We can also use the **index**=[list_of_row_labels] and **columns**=[list_of_column_labels] to specify the **row index as well as the column index**

Example 3, dataframe from a list of dictionaries with row index & column index

```
import pandas
L = [ {'roll' : 101 , 'name' : 'Astha' } ,\
      {'roll' : 104 , 'name' : 'Gautam', 'mark' : 478} ]
DF = pandas.DataFrame( L , index= ['s1' , 's2'] , columns =['roll' , 'name'] )
#note , here column 'mark' is skipped
print("First DataFrame" )
print(DF)
DF2 = pandas.DataFrame(L , index= ['s1','s2'] , columns=['roll' , 'name' , 'age'] )
#Here, column 'age' is additonal column, which does not exist in List of Dictionary
print("Second DataFrame is")
print(DF2)
```

OUTPUT –

```
First DataFrame
    roll    name
s1   101   Astha
s2   104  Gautam
Second DataFrame is
    roll    name  age
s1   101   Astha  NaN
s2   104  Gautam  NaN
```

D - Text/CSV Files -

A CSV (Comma Separated Values) file can be imported directly to a DataFrame object using the read_csv() method.

Simple form of Syntax is –

<data-frame-name> = read_csv(<file-name-path>)

Let us take a csv file named “stu_result.csv” as below -

Adm_No	Name	Class	Marks
1201	Aniket Sharma	XII	83
1203	Anita Gupta	XII	91
1206	Gautam Kumar	XI	89
1207	Mahesh Singh	XII	94
1209	Pratik Mehra	XI	90
1214	Nikita Verma	XII	92

Example 1 –

#read the csv file in a DataFrame –

```
import pandas as pp
# pp = alias-name of pandas library
sdf = pp.read_csv("D:/CPP/python_practice/stu_result.csv")
#OR, read_csv("stu_result.csv"), if file is in same folder as our program
print(sdf)
```

OUTPUT –

```
    Adm_No            Name Class  Marks
0  1201.0  Aniket Sharma   XII   83.0
1  1203.0    Anita Gupta   XII   91.0
2     NaN             NaN   NaN    NaN
3  1206.0   Gautam Kumar    XI   89.0
4  1207.0   Mahesh Singh   XII   94.0
5  1209.0   Pratik Mehra    XI   90.0
6  1214.0   Nikita Verma   XII   92.0
```

The read_csv() method has many parameters to control the kind of data imported to create the DataFrame.

Example 2 –

To show the shape (number of rows and columns) of CSV file imported in a DataFrame

```
r ,c = sdf.shape
print("\nTotal rows", r, "Total columns", c)
```

OUTPUT –

```
Total rows 7 Total columns 4
```

Similary, we can use <data-frame>.size to find number of values of DataFrame

Example 3 –

To read CSV file with specific / selected columns

```
#usecols = to display selected columns only
DF3 = pp.read_csv("stu_result.csv", usecols = ['Adm_No' , 'Name', 'Class'] )
print("\nDataFrame is\n", DF3)
```

OUTPUT –

```
DataFrame is
    Adm_No            Name Class
0  1201.0  Aniket Sharma   XII
1  1203.0    Anita Gupta   XII
2     NaN             NaN   NaN
3  1206.0   Gautam Kumar    XI
4  1207.0   Mahesh Singh   XII
5  1209.0   Pratik Mehra    XI
6  1214.0   Nikita Verma   XII
```

Example 4 –

To read CSV file with specific / selected rows

```
#nrows = we will use to display only first four records
DF = pp.read_csv("stu_result.csv", nrows = 4)
print("\nFirst four records of DataFrame \n ", DF)
```

OUTPUT –

```
First four records of DataFrame

    Adm_No            Name Class  Marks
0  1201.0  Aniket Sharma   XII   83.0
1  1203.0    Anita Gupta   XII   91.0
2     NaN             NaN   NaN    NaN
3  1206.0   Gautam Kumar    XI   89.0
```

Example 5 –

To read CSV file without header

```
# header = to omit(None) the display of headings of columns
DH = pp.read_csv("stu_result.csv", header = None )
print("The DataFrame is\n", DH)
```

OUTPUT –

```
The DataFrame is

        0               1       2       3
0  Adm_No            Name   Class   Marks
1    1201   Aniket Sharma     XII      83
2    1203     Anita Gupta     XII      91
3     NaN             NaN     NaN     NaN
4    1206    Gautam Kumar      XI      89
5    1207    Mahesh Singh     XII      94
6    1209    Pratik Mehra      XI      90
7    1214    Nikita Verma     XII      92
```

Example 6 –

To read CSV file without index

```
#when we do not want to display the  row indices
df2 = pp.read_csv("stu_result.csv", index_col = 0 )
print(df2)
```

OUTPUT –

Adm_No	Name	Class	Marks
1201.0	Aniket Sharma	XII	83.0
1203.0	Anita Gupta	XII	91.0
NaN	NaN	NaN	NaN
1206.0	Gautam Kumar	XI	89.0
1207.0	Mahesh Singh	XII	94.0
1209.0	Pratik Mehra	XI	90.0
1214.0	Nikita Verma	XII	92.0

Here, Adm_No will be the first column instead of indices.

Example 7 –

To read CSV file with new column names
#to use different names of column from default data, use **skiprows along-with names**

```
DF = pp.read_csv("stu_result.csv", skiprows =1 , names = ['StuNo' , 'SName', 'SClass','T_Marks'] )
print('DataFrame\n', DF)
```

OUTPUT –

	StuNo	SName	SClass	T_Marks
0	1201.0	Aniket Sharma	XII	83.0
1	1203.0	Anita Gupta	XII	91.0
2	NaN	NaN	NaN	NaN
3	1206.0	Gautam Kumar	XI	89.0
4	1207.0	Mahesh Singh	XII	94.0
5	1209.0	Pratik Mehra	XI	90.0
6	1214.0	Nikita Verma	XII	92.0

Display/Iteration of DataFrame:-

```
import pandas as pd
L1=[1,2,3,4,5]
L2=[10,20,30,40,50]
df=pd.DataFrame ([L1,L2],columns=[ 'a','b','c','d','e'])
print(df) # display entire DataFrame
```

Output:

	a	b	c	d	e
0	**1**	**2**	**3**	**4**	**5**
1	**10**	**20**	**30**	**40**	**50**

Display columns

```
print(df['a']) # display data of particular column (column a)
```

Output:

```
0   1
1   10
Name: a, dtype: int64
```

```
print(df[['a','c','e']]) # display data of multiple columns (columns a,c and e)
```

Output:

```
   a  c  e
0  1  3  5
1 10 30 50
```

Display rows using loc method:-

Syntax-

<DataFrame object>.loc[<startrow>:<endrow>,<startcolumn>:<endcolumn>]

Examples:

```
print(df.loc[1]) # display data of particular single row (row 1)
```

Output:

```
a   10
b   20
c   30
d   40
e   50
Name: 1, dtype: int64
```

```
print(df.loc[0:1])          #display data of multiple rows by using slicing(rows 0 and 1)
```

Output:

```
    a  b  c  d  e
0   1  2  3  4  5
1  10 20 30 40 50
```

```
print(df.loc([0:1,'a']      # display data of multiple rows with single column by using slicing
```

Output: (rows 0,1 and column a)

```
0   1
1   10
Name: a, dtype: int64
```

```
print(df.loc[0:1,'a':'c'])          # display data of multiple rows with multiple columns using slicing method(rows 0,1 and columns a,b,c)
```

Output:

```
    a  b  c
0   1  2  3
1  10 20 30
```

Display rows using iloc method:-

This method is used when DataFrame object does not have row and column labels or even we may not remember them. It works on numeric index.

Syntax:-

<DataFrame object>.iloc[<startrowindex>:<endrowindex>,<startcolumnindex>:<endcolumnindex>]

Examples:

print(df.iloc[0:2,1:3]) *# display rows exist on index 0,1 and columns exist on index 1,2*

Output:

```
   b  c
0  2  3
1 20 30
```

print(df.iloc[0:2,:]) *# display rows exist on index 0,1 with all columns*

Output:

```
    a  b  c  d  e
0   1  2  3  4  5
2  10 20 30 40 50
```

Difference between loc and iloc method:-

In loc method both start label and end label are included but in iloc method end index is excluded when given as strat:end.

Operations on rows and columns in DataFrames:-We can perform some basic operations on rows and columns of a DataFrame like selection, deletion, addition, and renaming

```
import pandas as pd

dict={ 'Arnab': pd.Series([90, 91, 97],
index=['Maths','Science','Hindi']),

'Ramit': pd.Series([92, 81, 96],
index=['Maths','Science','Hindi']),

'Samridhi': pd.Series([89, 91, 88],
index=['Maths','Science','Hindi']),

'Riya': pd.Series([81, 71, 67],
index=['Maths','Science','Hindi']),

'Mallika': pd.Series([94, 95, 99],
index=['Maths','Science','Hindi']) }

ResultDF = pd.DataFrame(dict)

print(ResultDF)
```

Output:

	Arnab	RamitSamridhi	Riya	Mallika
Maths	90	92	89	81	94
Science	91	81	91	71	95
Hindi	97	96	88	67	99

>>>

Adding a New Column to a DataFrame: To add a new column to a DataFrameResultDFwe can write the following statement:

```
>>>ResultDF['Radha']=[89,78,76]
Or
ResultDF.loc[:,'Radha']=[89,78,76]
Or
ResultDF.at[:,'Radha']=[89,78,76]
>>>print(ResultDF)
```

or

Output:-

	Arnab	RamitSamridhi	Riya	Mallika	Radha
Maths	90	92	89	81	94	89
Science	91	81	91	71	95	78
Hindi	97	96	88	67	99	76

Note: Assigning values to a new column label that does not exist will create a new column at the end If already exists then the assignment statement will update the values of the already existing column

Example :

```
ResultDF['Ramit']=[99, 98, 78]
>>>print(ResultDF)
```

Output:

	Arnab	RamitSamridhi	Riya	Mallika	Radha
Maths	90	99	89	81	94	89
Science	91	98	91	71	95	78
Hindi	97	78	88	67	99	76

Adding a New Row to a DataFrame: To add a new row to a DataFrame we can use the **DataFrame.loc[ ]** method.

Suppose we want to add English marks in above DataFrame, we can write the following statement:

```
ResultDF.loc['English'] = [85, 86, 83, 80, 90, 89]
>>>print(ResultDF)
Or
ResultDF.at['English'] = [85, 86, 83, 80, 90, 89]
>>>print(ResultDF)
```

Output:

	Arnab	RamitSamridhi	Riya	Mallika	Radha
Maths	90	99	89	81	94	89
Science	91	98	91	71	95	78

Hindi	97	78	88	67	99	76
English	85	86	83	80	90	89

DataFrame.loc[] method can also be used to change the data values of a row to a particular value.

Example: to set marks in 'Maths' for all columns to 0:
>>>ResultDF.loc['Maths']=0
>>>print(ResultDF)
Output:

	Arnab	Ramit	Samridhi	Riya	Mallika	Radha
Maths	0	0	0	0	0	0
Science	91	98	91	71	95	78
Hindi	97	78	88	67	99	76
English	85	86	83	80	90	89

>>>ResultDF[:] = **0 # Set all values in ResultDF to 0**
>>>ResultDF

	Arnab	Ramit	Samridhi	Riya	Mallika	Radha
Maths	0	0	0	0	0	0
Science	0	0	0	0	0	0
Hindi	0	0	0	0	0	0
English	0	0	0	0	0	0

Selecting / Accessing Data from DataFrame :

DataFrame : DF5

	Population	**Hospital**	**Schools**
Delhi	**10927986**	**189**	**7916**
Mumbai	**12691836**	**208**	**8508**
Kolkata	**4631392**	**149**	**7226**

Selecting / Accessing a column: Just use the following syntax

<DF_object>[column_name] or <DF_object>.<column_name>

Example : >>>DF5[‘Population’] or >>>DF5.Population

Output:-

Delhi	10927986
Mumbai	12691836
Kolkata	4631392

Selecting / Accessing multiple columns: Just use the following syntax

<DF_object>[[<column_name1>,<column_name2>,<column_name3>.....]]

Example : >>>DF5[[‘Population’, ‘Hospital’]]

Output:-	**Population**	**Hospital**
Delhi	**10927986**	**189**
Mumbai	**12691836**	**208**
Kolkata	**4631392**	**149**

Selecting /Accessing a subset from a DataFrame using Row / Column Names: Use the following syntax :-
<DF_object>.loc[<start_row>:<end_row>,<start_column>:<end_column>]

or

<DF_object>.iloc[<start_row_index>:<end_row_index>,<start_column_index>:<end_column_index>]

Example 1.>>>DF5.loc[‘Mumbai’:’Kolkata’ , :]

Output:

	Population	Hospital	Schools
Mumbai	12691836	208	8508

Example 2. >>>DF5.iloc[0:2,0:2]

Output: -

	Population	Hospital
Delhi	10927986	189
Mumbai	12691836	208

Deleting Rows or Columns from a DataFrame: DataFrame.drop() method is used to delete rows and columns from a DataFrame. To delete a row set the parameter axis=0 and

for deleting a column set axis=1. Consider the following DataFrame:

	Arnab	Ramit	Samridhi	Riya	Mallika	Radha
Maths	**90**	**99**	**89**	**81**	**94**	**89**
Science	**91**	**98**	**91**	**71**	**95**	**78**
Hindi	**97**	**78**	**88**	**67**	**99**	**76**
English	**85**	**86**	**83**	**80**	**90**	**89**

To delete the row with label 'Science' we can write the following statement:

```
>>>ResultDF = ResultDF.drop('Science', axis=0)
>>>ResultDF
```

Output :

	Arnab	Ramit	Samridhi	Riya	Mallika	Radha
Maths	90	99	89	81	94	89
Hindi	97	78	88	67	99	76
English	85	86	83	80	90	89

To delete the columns having labels 'Samridhi', 'Ramit' and 'Riya': we can write the following statement:-

```
>>>ResultDF = ResultDF.drop(['Samridhi','Ramit','Riya'], axis=1)
>>>ResultDF
```

Output:

	Arnab	Mallika	Radha
Maths	90	94	89
Hindi	97	99	76
English	85	90	89

Renaming Row Labels of a DataFrame: DataFrame.rename() method is used to rename the row and column label. To rename the row indices Maths to sub1, Hindi to sub2 in above DataFrame we can write the following statement:-

```
ResultDF=ResultDF.rename({'Maths':'Sub1', 'Hindi':'Sub2'}, axis='index')
Print(ResultDF)
```

Output:

	Arnab	Mallika	Radha
Sub1	90	94	89
Sub2	97	99	76
English	85	90	89

Note: The parameter axis='index' is used to specify that the row label is to be changed and axis='columns' to specify that the column label is to be changed

Renaming Column Labels of a DataFrame:

```
ResultDF=ResultDF.rename({'Arnab':'Student1','Mallika':'Student2','Radha':'Student3'}, axis='columns' )
>>>print(ResultDF)
```

Output:

	Student1	Student2	Student3
Sub1	90	94	89
Sub2	97	99	76
English	85	90	89

```
>>>
```

Operations on rows and columns in DataFrames:-We can perform some basic operations on rows and columns of a DataFrame like selection, deletion, addition, and renaming

```
import pandas as pd
dict={ 'Arnab': pd.Series([90, 91, 97], index=['Maths','Science','Hindi']),
'Ramit': pd.Series([92, 81, 96], index=['Maths','Science','Hindi']),
'Samridhi': pd.Series([89, 91, 88], index=['Maths','Science','Hindi']),
'Riya': pd.Series([81, 71, 67], index=['Maths','Science','Hindi']),
'Mallika': pd.Series([94, 95, 99], index=['Maths','Science','Hindi']) }
ResultDF = pd.DataFrame(dict)
print(ResultDF)
```

Output:

	Arnab	Ramit	Samridhi	Riya	Mallika
Maths	90	92	89	81	94
Science	91	81	91	71	95
Hindi	97	96	88	67	99

>>>

Adding a New Column to a DataFrame: To add a new column to a DataFrameResultDFwe can write the following statement:

>>>ResultDF['Radha']=[89,78,76]

Or

ResultDF.loc[:,'Radha']=[89,78,76]

Or

ResultDF.at[:,'Radha']=[89,78,76]

>>>print(ResultDF)

or

Output:-

	Arnab	Ramit	Samridhi	Riya	Mallika	Radha
Maths	90	92	89	81	94	89
Science	91	81	91	71	95	78
Hindi	97	96	88	67	99	76

Note: Assigning values to a new column label that does not exist will create a new column at the end If already exists then the assignment statement will update the values of the already existing column

Example :

ResultDF['Ramit']=[99, 98, 78]

>>>print(ResultDF)

Output:

	Arnab	Ramit	Samridhi	Riya	Mallika	Radha
Maths	90	99	89	81	94	89
Science	91	98	91	71	95	78
Hindi	97	78	88	67	99	76

Adding a New Row to a DataFrame: To add a new row to a DataFramewe can use the **DataFrame.loc[ ]** method.

Suppose we want to add English marks in above DataFrame, we can write the following statement:

ResultDF.loc['English'] = [85, 86, 83, 80, 90, 89]

>>>print(ResultDF)

Or

ResultDF.at['English'] = [85, 86, 83, 80, 90, 89]

>>>print(ResultDF)

Output:

	Arnab	Ramit	Samridhi	Riya	Mallika	Radha
Maths	90	99	89	81	94	89
Science	91	98	91	71	95	78
Hindi	97	78	88	67	99	76
English	85	86	83	80	90	89

DataFrame.loc[] method can also be used to change the data values of a row to a particular value.

Example: to set marks in 'Maths' for all columns to 0:

>>>ResultDF.loc['Maths']=0

>>>print(ResultDF)

Output:

	Arnab	Ramit	Samridhi	Riya	Mallika	Radha
Maths	0	0	0	0	0	0
Science	91	98	91	71	95	78
Hindi	97	78	88	67	99	76
English	85	86	83	80	90	89

>>>ResultDF[:] = **0 # Set all values in ResultDF to 0**

>>>ResultDF

	Arnab	Ramit	Samridhi	Riya	Mallika	Radha

Maths	0	0	0	0	0	0	0
Science	0	0	0	0	0	0	0
Hindi	0	0	0	0	0	0	0
English	0	0	0	0	0	0	0

Selecting / Accessing Data from DataFrame :

DataFrame : DF5

	Population	Hospital	Schools
Delhi	**10927986**	**189**	**7916**
Mumbai	**12691836**	**208**	**8508**
Kolkata	**4631392**	**149**	**7226**

Selecting / Accessing a column: Just use the following syntax

<DF_object>[column_name] or <DF_object>.<column_name>

Example : >>>DF5['Population'] or >>>DF5.Population

Output:-

Delhi 10927986

Mumbai 12691836

Kolkata 4631392

Selecting / Accessing multiple columns: Just use the following syntax

<DF_object>[[<column_name1>,<column_name2>,<column_name3>.....]]

Example : >>>DF5[['Population', 'Hospital']]

Output:-

	Population	Hospital
Delhi	**10927986**	**189**
Mumbai	**12691836**	**208**
Kolkata	**4631392**	**149**

Selecting /Accessing a subset from a DataFrame using Row / Column Names: Use the following syntax :-

<DF_object>.loc[<start_row>:<end_row>,<start_column>:<end_column>]

or

<DF_object>.iloc[<start_row_index>:<end_row_index>,<start_column_index>:<end_column_index>]

Example 1.>>>DF5.loc['Mumbai':'Kolkata' , :]

Output:

	Population	Hospital	Schools
Mumbai	12691836	208	8508

Example 2. >>>DF5.iloc[0:2,0:2]

Output: -

	Population	Hospital
Delhi	10927986	189
Mumbai	12691836	208

Deleting Rows or Columns from a DataFrame: DataFrame.drop() method is used to delete rows and columns from a DataFrame. To delete a row set the parameter axis=0 and for deleting a column set axis=1. Consider the following DataFrame:

	Arnab	Ramit	Samridhi	Riya	Mallika	Radha
Maths	**90**	**99**	**89**	**81**	**94**	**89**
Science	**91**	**98**	**91**	**71**	**95**	**78**
Hindi	**97**	**78**	**88**	**67**	**99**	**76**
English	**85**	**86**	**83**	**80**	**90**	**89**

To delete the row with label 'Science' we can write the following statement:

>>>ResultDF = ResultDF.drop('Science', axis=0)

>>>ResultDF

Output : Arnab Ramit Samridhi Riya Mallika Radha

Maths 90 99 89 81 94 89

Hindi 97 78 88 67 99
76

English 85 86 83 80 90
89

To delete the columns having labels 'Samridhi', 'Ramit' and 'Riya': we can write the following statement:-

>>>ResultDF =
ResultDF.drop(['Samridhi','Ramit','Riya'], axis=1)
>>>ResultDF

Output: Arnab Mallika Radha
Maths 90 94 89
Hindi 97 99 76
English 85 90 89

Renaming Row Labels of a DataFrame:

DataFrame.rename() method is used to rename the row and column label. To rename the row indices Maths to sub1, Hindi to sub2 in above DataFrame we can write the following statement:-

ResultDF=ResultDF.rename({'Maths':'Sub1', 'Hindi':'Sub2'}, axis='index')

Print(ResultDF)

Output: Arnab Mallika Radha

Sub1 90 94 89

Sub2 97 99 76

English 85 90 89

Note: The parameter axis='index' is used to specify that the row label is to be changed and axis='columns' to specify that the column label is to be changed

Renaming Column Labels of a DataFrame:

ResultDF=ResultDF.rename({'Arnab':'Student1','Mallika':'Student2','Radha':'Student3'}, axis='columns')

>>>print(ResultDF)

Output: Student1 Student2 Student3

Sub1 90 94 89

Sub2 97 99 76

English 85 90 89

>>>

Indexing and Boolean indexing:-

In Boolean indexing, we select data based on the actual values of the data and not on their row/column labels or integer locations. If we provide list of Boolean values as index then only those rows will be selected where True is stored. Consider following code for the **df1**

	Hindi	**English**	**IP**
Aditya	34	23	67
Aman	34	85	56
Rajesh	60	80	91
Mohit	45	21	32

print(df1[[True,False,False,True]])

OUTPUT

	Hindi	**English**	**IP**
Aditya	34	23	67
Mohit	45	21	32

Consider the following command df1['English']>50 is will result a Series of **False, True, True, False,** so this Boolean expression can be used as index, hence df1[df1['English']>50] will select the rows where English marks are more than 50.

OUTPUT

	Hindi	**English**	**IP**
Aman	34	**85**	56
Rajesh	60	**80**	91

Find the details of student who secured 34 marks in Hindi

df1['Hindi']==34 will result Series of [True,True,False,False]

so df1[df1['Hindi']==34] will select the rows where 34 marks is stored in Hindi

OUTPUT

	Hindi	English	IP
Aditya	**34**	23	67
Aman	**34**	85	56

Find the details of student who secured marks is IP subject which is more than average marks of IP subject

df1['IP'].mean() will return average marks for IP which is 61.5

so df1['IP']>df1['IP'].**mean()** will return Series of [True,False,True,False]

So this code can be used as index to get desired result

Hence df1[df1['IP']>df1['IP'].mean()]

output

	Hindi	**English**	**IP**
Aditya	34	23	**67**
Rajesh	60	80	**91**

We can include specific column(s) in our output in two ways

To display only IP column in place of all columns we can modify above code as given below

df1**['IP']**[df1['IP']>df1['IP'].mean()]

OR

df1[df1['IP']>df1['IP'].mean()]**['IP']**

Output

Aditya 67

Rajesh 91

Name:IP, dtype: int64

If Hindi and IP marks to be displayed for the same problem stated above the code will be

df1**[['Hindi','IP']]**[df1['IP']>df1['IP'].mean()]

OR

df1[df1['IP']>df1['IP'].mean()]**[['Hindi','IP']]**

output

	Hindi	**IP**
Aditya	34	**67**
Rajesh	60	**91**

Data Visualization: -

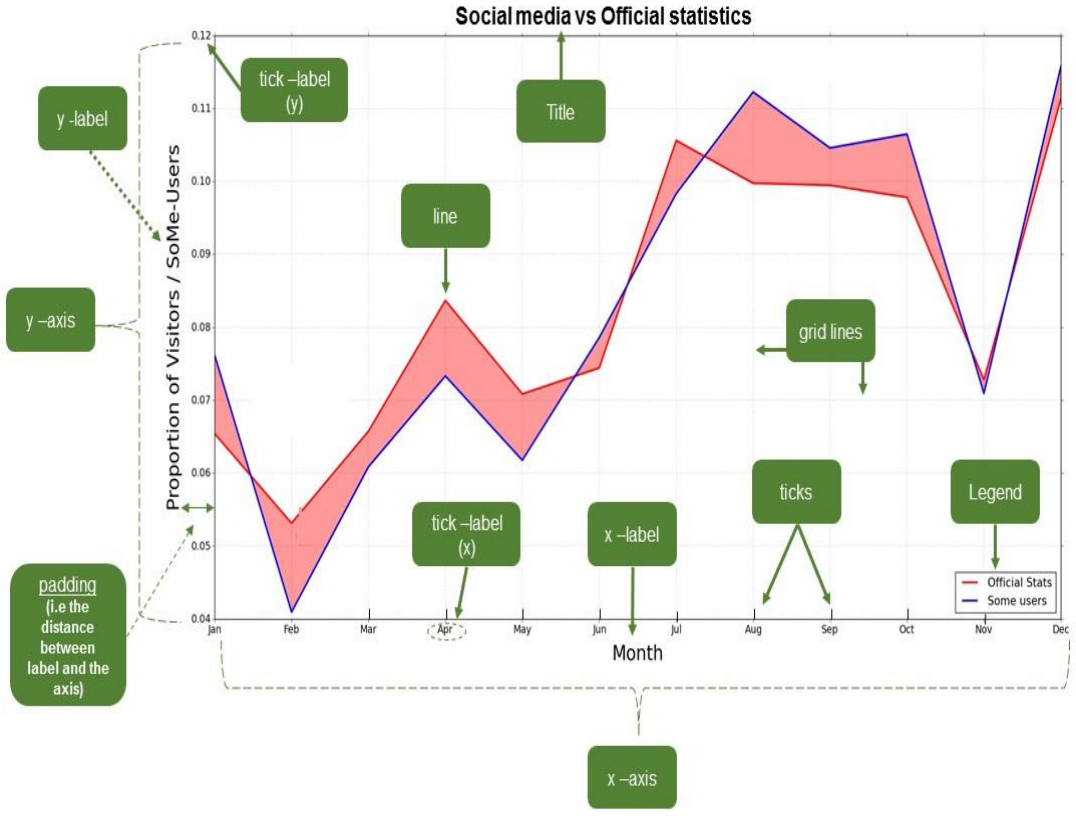
Data Visualization

- ***Data Visualization refers to the graphical or visual representation of data and information using visual elements like charts, graphs, maps etc.***
- Data visualization is the discipline of trying to expose the data to understand it by placing it in a visual context.
- Its main goal is to distill large datasets into visual graphics to allow for easy understanding of complex relationships within the data.

Purpose of Data visualization

- Better analysis
- Quick action
- Identifying patterns
- Finding errors
- Understanding the story
- Exploring business insights
- Grasping the latest trends plotting library

Anatomy of a Chart



Introduction to matplotlib

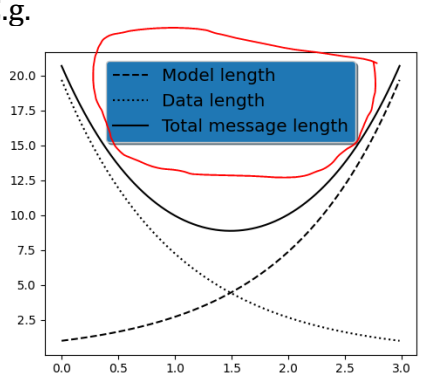
- matplotlib.pyplot is a collection of functions for 2D plotting.
- Some of the types of plots: Line, Bar, Histogram, Pie and Boxplot.

Matplotlib – pyplot features

- ***PyPlot is a collection of methods within matplotlib library of python which allows users to construct 2D plots easily and interactively.***
- **Drawing** – plots can be drawn based on passed data through specific functions.
- **Customization** – plots can be customized as per requirement after specifying it in the arguments of the functions. Like color, style (dashed, dotted), width; adding label, title, and legend in plots can be customized.

- **Saving** – After drawing and customization plots can be saved for future use.
- **Figure**: Pyplot by default plots every chart into an area called Figure. A figure contains other elements of the plot in it.
- **Axes**: The axes define the area (mostly rectangular in shape for simple plots) on which actual plot (line or bar or graph etc.) will appear. Axes have properties like label, limits and tick marks on them.

There are two axes in a plot:
(i) X-axis the horizontal axis,
ii) Y – axis the vertical axis
a) **Axis Label**: It defines the name for an axis. It is individually defined for X– axis and Y–axis each.
b) **Limits:** These define the range of values and number of values marked on X–axis and Y – axis.
c) **Tick_Marks**: The tick marks are individual point marked on the X – axis or Y – axis.
Title: This is the text that appears on the top of the plot. It defines what the chart is about.
Legends: These are different colors that identify different sets of data plotted on the plot. The legends are shown in a corner of the plot. We use legend as following types:
plt.legend (loc="upper left") or plt.legend (loc=2)
E.g.



To import the library for plotting

import matplotlib.pyplot as pl
Basic steps to follow while plotting:
(a) Choose appropriate plot type and then the **function**

- Line plot: plot ()
- Bar plot: bar () and barh()
- Histogram: hist ()

(b) Understand the data and assign the legend values

- assign the axis labels
- assign plot title

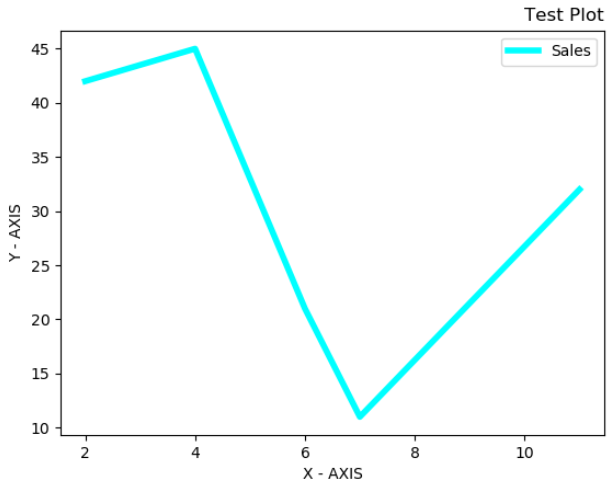
Different color codes:

Character	Color	Character	Color	Character	Color
'b'	Blue	'm'	Magenta	'c'	Cyan
'g'	Green	'y'	Yellow	'w'	White
'r'	Red	'k'	Black		

Line Plot:

Definition: A line plot/chart is a graph that shows the frequency of data occurring along a number line.
Eg.

```
import matplotlib.pyplot as pl
x = [2,4,6,3,8]
y = [42, 45, 21, 11, 32]
pl.plot(x, y,'r', label = "Sales", linewidth = 4,color='cyan')
pl.title ("Test Plot", loc="right")
pl.xlabel ("X - AXIS")
pl.ylabel ("Y - AXIS")
pl.legend ()
pl.show ()
```

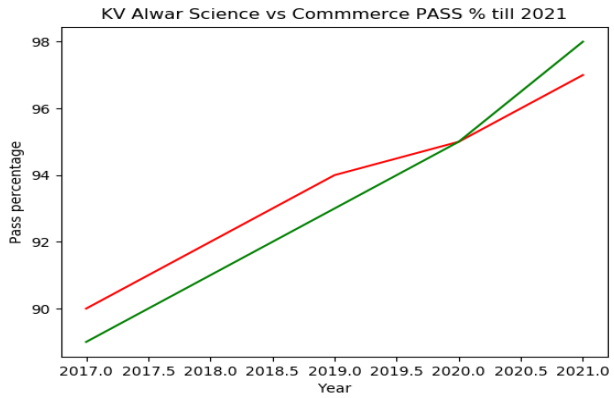


Multiple line plots

In this we will take help of two plot functions to make comparison between two line plots.
E.g.

```
import numpy as np
import matplotlib.pyplot as plt
year = [2017, 2018, 2019, 2020, 2021]
Sciencepasspercentage = [90, 92, 94, 95, 97]
commercepasspercentage = [89, 91, 93, 95, 98]
plt.plot (year, Sciencepasspercentage, color='red')
plt.plot (year, commercepasspercentage, color='green')
```

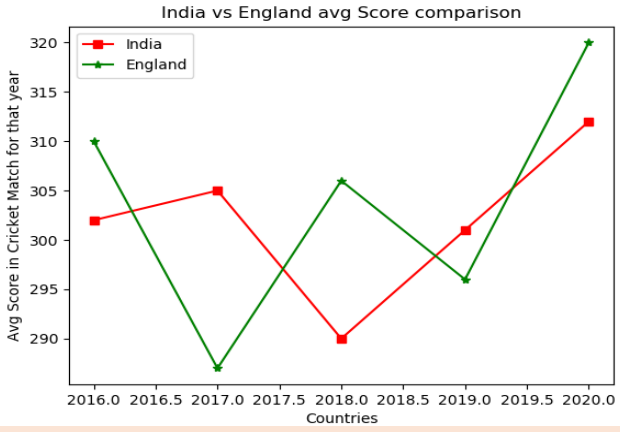
plt.xlabel ('Year')
plt.ylabel ('Pass percentage')
plt.title ('KV Alwar Science vs Commerce PASS % till 2021')
plt.show ()



Or

Eg.

```
import matplotlib.pyplot as plt
year = [2016, 2017, 2018, 2019, 2020]
Indianavgscore = [302,305,290,301,312]
Englandavgscore = [310,287,306,296,320]
plt.plot (year, Indianavgscore, color='red', marker='s', label='India')
plt.plot (year, Englandavgscore, color='green', marker='*', label='England')
plt.xlabel ('Countries')
plt.ylabel('Avg Score in Cricket Match for that year')
plt.title ('India vs England avg. Score comparison')
plt.legend ()
plt.show ()
```



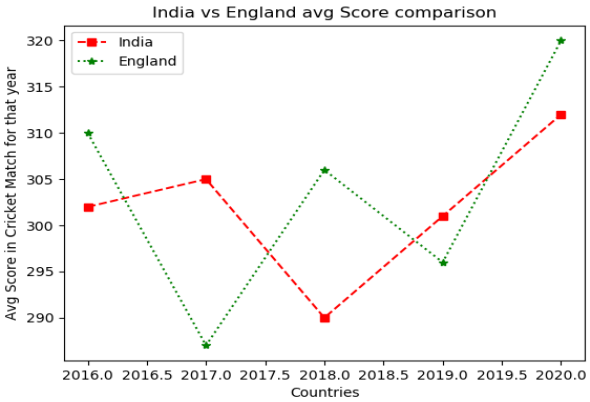
To change the line style

We can add following additional optional argument in plot (): **linestyle or ls = ['solid' | 'dashed' | 'dashdot' | 'dotted']**

If we apply the customization for line style in above example then, we will apply the linestyle type in both the plot statements.

E.g.

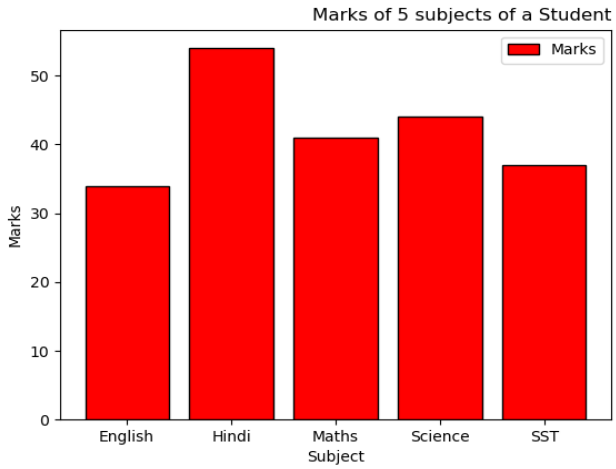
- plt.plot (year, Indianavgscore, color='red', marker='s', label='India',linestyle='dashed')
- plt.plot (year, Englandavgscore, color='green', marker='*', label='England', linestyle='dotted')



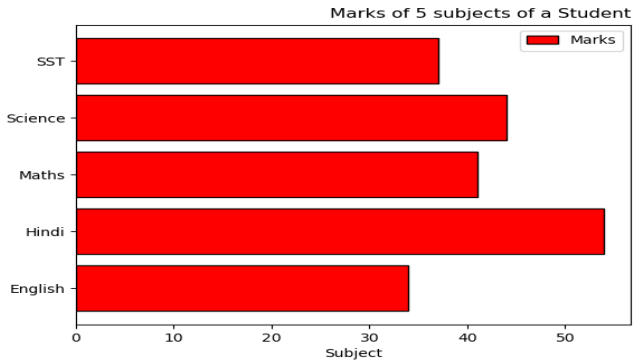
Bar Plot:

Definition: A graph drawn using rectangular bars to show how large each value is. The bars can be horizontal or vertical. E.g.

```
import matplotlib.pyplot as pl
x = ['English','Hindi','Maths','Science','SST']
y = [34, 54, 41, 44, 37]
pl.bar (x, y, width =0.8, label= "Marks", color='red', edgecolor="black")
pl.title ("Marks of 5 subjects of a Student", loc="right")
pl.xlabel ("Subject")
pl.ylabel ("Marks")
pl.legend ()
pl.show ()
```



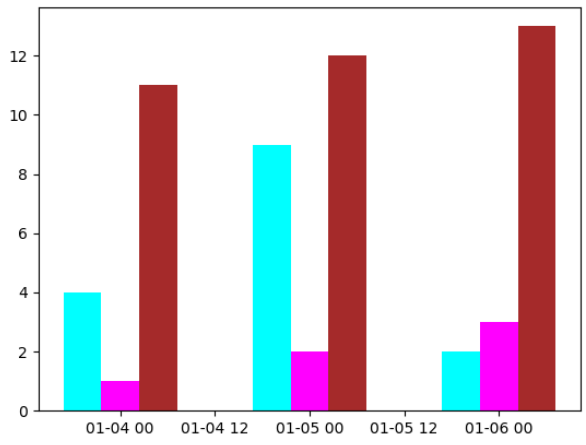
Note: – use barh () for creating horizontal bar graphs. If we apply the barh() in above example, then following figure will be appeared:



Multiple Bar Plots

Eg.

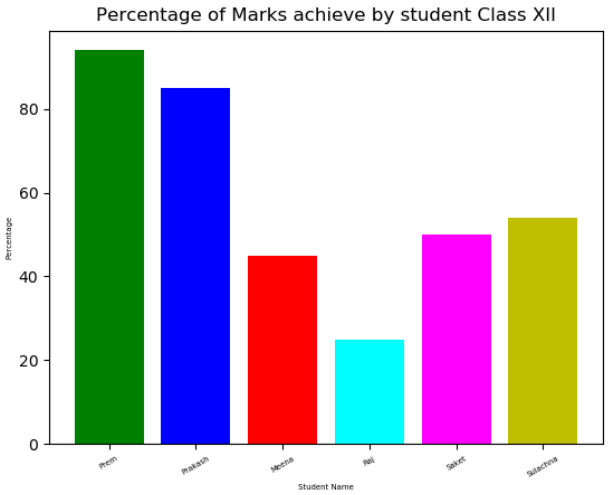
```
import matplotlib.pyplot as plt
from matplotlib.dates import date2num
import datetime
x = [datetime.datetime (2011, 1, 4, 0, 0),
   datetime.datetime (2011, 1, 5, 0, 0),
   datetime.datetime (2011, 1, 6, 0, 0)]
x = date2num(x)
y = [4, 9, 2]
z = [1, 2, 3]
k = [11, 12, 13]
ax = plt.subplot (111)
ax.bar(x-0.2, y, width=0.2, color='cyan',
align='center')
ax.bar(x, z, width=0.2, color='magenta',
align='center')
ax.bar(x+0.2, k, width=0.2, color='brown',
align='center')
ax.xaxis_date()
plt.show ()
```



or

E.g.-We can also use arange() function to generate the data.

```
import matplotlib.pyplot as plt
import numpy as np
label = ['Prem', 'Prakash', 'Meena', 'Raj', 'Saket',
'Sulachna']
per = [94, 85, 45, 25, 50, 54]
index = np.arange (len(label))
clr=['g', 'b','r', 'cyan', 'magenta','y']
plt.bar (index, per, color=clr)
plt.xlabel ('Student Name', fontsize=5)
plt.ylabel ('Percentage', fontsize=5)
plt.xticks (index, label, fontsize=5, rotation=30)
plt.title ('Percentage of Marks achieve by
student Class XII')
plt.show ()
```



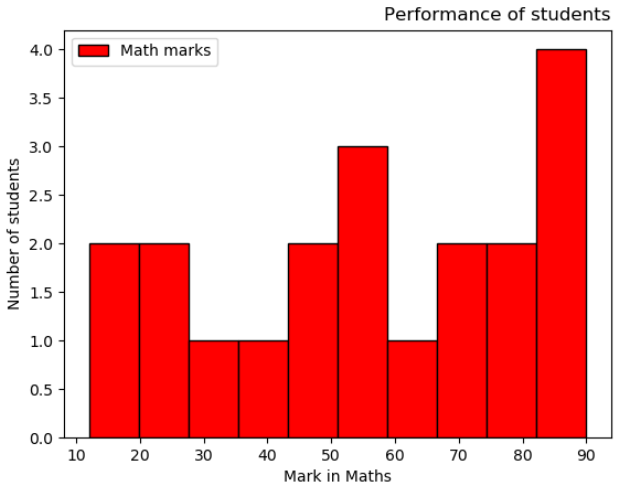
Histogram:

A histogram is a graphical representation which organizes a group of data points into user-specified ranges.

histtype: [‘bar’, ‘barstacked’, ‘step’, ‘stepfilled’], It is optional by default is ‘bar’ orientation: [‘vertical’, ’horizontal’]

Eg.

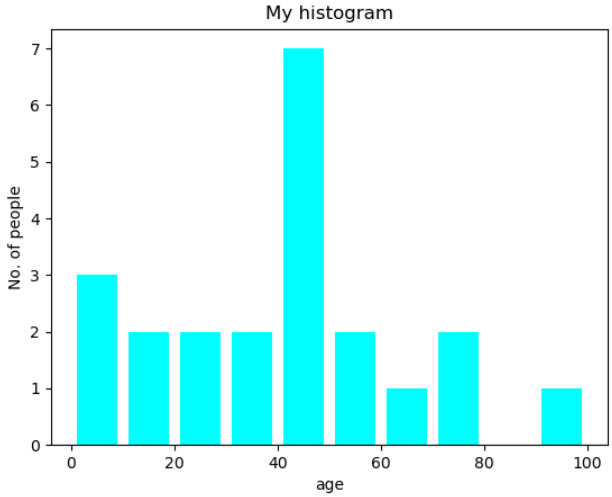
```
import matplotlib.pyplot as pl
import numpy as np
math=
[12,23,45,56,57,67,72,83,65,22,87,53,12,90,78,
83, 45, 75, 37,28]
x = np.arange (len(math))
freq, bin, patches = pl.hist (math,
bins=10,color='red',edgecolor = "black", label =
"Math marks")
pl.title ("Performance of students", loc="right")
pl.xlabel ("Mark in Maths")
pl.ylabel ("Number of students")
pl.legend ()
pl.show ()
```



Or

Eg.

```
import matplotlib.pyplot as plt
ages=[2,5,70,40,30,45,50,45,43,40,44,60,7,13,57
,18,90,77,32,21,20,40]
range = (0, 100)
bins = 10
plt.hist(ages, bins, range, color = 'cyan', histtype
= 'bar', rwidth = 0.8)
plt.xlabel ('age')
plt.ylabel ('No. of people')
plt.title ('My histogram')
plt.show ()
```



Save Plot:

To save any plot we have to use savefig() function E.g. plt.savefig ("plot.png")

Here plot.png is the name of the file where plot is saved. Eg.

```
plt.savefig ('Student_Data.pdf')
plt.savefig ('Student_Data.svg')
plt.savefig ('Student_Data.png')

plt.savefig ('line_plot.jpg', dpi=400, quality=60,
optimize=True, progressive=True)
```

Example of Save Plot with full path

```
pl.savefig('F:\line_plot.png')
```

here the figure will be saved in F drive with name lineplot of Computer System.

Societal Impact (Marks-10)

Digital Footprint

A digital footprint, sometimes called digital dossier is a body of data that you create while using the Internet. It includes the websites you visit, emails you send, and information you submit to online services and can be traced back by an individual.

It is of two types:

1. Passive digital footprints
2. Active digital footprints

- **A passive digital footprint** is created when data is collected without the owner knowing. A more personal aspect of your passive digital footprint is your search history, which is saved by some search engines while you are logged in.

- **Active digital footprints** are created when a user, for the purpose of sharing information about oneself by means of websites or social media, deliberately. An "active digital footprint" includes data that you intentionally submit online. Sending an email contributes to your active digital footprint, since you expect the data be seen and/or saved by another person. The more email you send, the more your digital footprint grows.

Publishing a blog and posting social media updates are another popular ways to expand your digital footprint. Every tweet you post on Twitter, every status update you publish on Facebook, and every photo you share on Instagram contributes to your digital footprint.

How to reduce the footprint?

1.Double-check privacy settings
2.Logout after you're done surfing a website
3.Think before putting anything online/public platform
4. Don's post personal information online

Net and Communication Etiquettes

Netiquette is short for "Internet etiquette." Just like etiquette is a code of polite behaviour in society, netiquette is a code of good behaviour on the Internet. This includes several aspects of the Internet, such as email, social media, online chat, web forums, website comments, multiplayer gaming, and other types of online communication. While there is no official list of netiquette rules or guidelines, the general idea is to respect others online.

Below are some examples of rules to follow for good netiquette:

- *Avoid posting inflammatory or offensive comments online.*
- *Respect others' privacy by not sharing personal information, photos, or videos that another person may not want published online.*
- *Never spam others by sending large amounts of unsolicited email.*
- *Show good sportsmanship when playing online games, whether you win or lose.*
- *Don't troll people in web forums or website comments by repeatedly nagging or annoying them.*
- Stick to the topic when posting in online forums or when commenting on photos or videos, such as YouTube or Facebook comments.
- Don't swear or use offensive language.
- Avoid replying to negative comments with more negative comments. Instead, break the cycle with a positive post.
- If someone asks a question and you know the answer, offer to help.
- Thank others who help you online.

Data Protection

Data protection refers to the practices, safeguards, and binding rules put in place to protect your personal information and ensure that you remain in control of it.

In short, you should be able to decide whether you want to share some information or not, who has access to it, for how long, for what reason, and who be able to modify some of this information Personal data is any information relating to you, whether it relates to your private, professional, or public life.

In the online environment, where vast amounts of personal data are shared and transferred around the globe instantaneously,

It is increasingly difficult for people to maintain control of their personal information. This is where data protection comes in.

Intellectuals Property Rights (IPR)

Intellectual property refers to intangible property that has been created by individuals and corporations for their benefit or usage such as copyright, trademark, patent and digital data.

It is therefore unethical to copy or steal the creativity and efforts of someone else.

Intellectual property is divided into categories which are-

- Industrial property which majorly speaks about protecting inventions on the other hand.
- Copyright majorly protects literary and artistic works.
-

licensing of intellectual property:

Copyright , Patent and Trademark,

- Code of the software will be protected by a **copyright**.
- Functional expression of the idea will be protected by a **patent**
- The name and logo of the software will come under a registered **trademark**

PLAGIARISM

Plagiarism pronounced as **plei** ·juh·ri·zm

Plagiarism means not giving authors credit after copying that author's work.

It involves lying, cheating, theft and dishonesty.

For example, copying papers written by other people and professional and claims it as written by you can be an example of plagiarism.

It can be classified as:

- Accidental/unintentional
- Deliberate/intentional

Accidental/unintentional Plagiarism: . Involves careless paraphrasing (changing the words or sentence construction of a copied document), quoting text excessively along with poor documentation. Accidental Plagiarism cases are less serious whereas

Deliberate/intentional Plagiarism : Includes copying someone else's work, cutting and passing blocks of text or any kind of information from electronic sources without the permission of the original author. Deliberate plagiarism that may result in serious implications.

HOW TO AVOID PLAGIARISM?

Plagiarism should be avoided by the following simple measures:

- Use your own ideas and words.
- Always provide a reference or give credit to the source from where you have received information.
- Cite the name of the website, a URL or the name of authors, and acknowledge them if you have used their work after rearranging the order of a sentence and changing some of the work.
- Take the information in the form of bulleted notes in your words.
- Use online tools to check for plagiarism.
- Develop your writing skills.

Licensing and copyright

A Software license is a legal permission or right to use or redistribution of that software. The software can run on a certain number of computers as per license agreement.

PROPRIETARY LICENSES:-

Exclusive rights in the software are retained with the owner /developer /publisher. They reserve all the freedom and rights to use and distribute this proprietary software.

PERMISSIVE LICENSES :-

Permissive licenses provide a royalty-free license to do virtually anything with the source code.

They permit using, copying, modifying, merging, publishing, distributing, sublicense and/or selling ,but distribution can only be made without the source code as source code modifications can lead to permissive license violation.

Example:

General Public License (GPL),

Creative Commons License (CC),

Lesser General Public License (LGPL),

Mozilla public License (MPL) etc.

COPYLEFT LICENSE

In the case of copyleft licenses, source code has to be provided.

Distribution and modification of source code is permitted.

COPYRIGHT

It is a form of protection given to the authors of "original works of authorship". This is given in the field of literature, dramatics, music, software, art etc. This protection applies to published as well as unpublished work.

Software copyright is used by software developers and proprietary software companies to prevent the unauthorized copying of their software. Free and open source licenses also rely on copyright law to enforce their terms. Copyright protects your software from someone else copying it and using it without your permission. When you hold the copyright to software, you can-

- Make copies of it.
- Distribute it.
- Modify it

Cyber Crime:

Cybercrime is any criminal offence involves the use of electronic communication, computer or internet. The term "Cybercrime" that covers phishing, Identity theft, credit card frauds, illegal downloading, child pornography, cyber bullying, cyber trolls, cyber stalking, cyber terrorism, distribution of viruses, spam, and industrial intelligence and so on.

1. Identity theft:

When we buy or sell goods using social media or we give out private data to business for the right usage. Personal data or login details cannot be used for harmful reasons like posting comments on someone else with stolen identity is called identity theft.

2. Cyber Trolls:

Posting insulted messages online targeting people is called cyber trolls. It is closely related to cyber bullying.

3. Cyber Bullying:

Harassing people or acting like someone or posting negative comments to someone or acting like someone using modern technologies such as internet, email, cell phone, instant massagers', social networks etc is called as Cyber Bullying.

4. Cyber Stalking:

Cyber stalking is a crime in which the attacker harasses a specific victim using electronic communication such as email or online message. Stalkers know their victims and they attack online instead resolving issues off line.

5. Phishing:

Phishing is a cyber attack that uses email or website as a weapon trick the email recipient into believing that the message is something they want or need — a request from their bank, to click a link or download an attachment. They try to gather personal information or debit/credit card information.

6. Child pornography:

Chile Pornography is defined as any visual or written representation including images or video that depicts sexual activity of anyone under the age of 18. Child pornography is sometimes called "child sexual abuse images".

Online Fraud:

Fraud committed using the internet is called online fraud and may occur in many ways

- Non-delivery goods
- Non-existent companies
- Stealing information
- Fraudulent payments etc.

Digital Forensics:

It refers to methods used for interpretation of computer media for digital evidence.

Cyber Law and IT Act:

Cyber law refers to all the legal and regulatory aspects of internet and WWW. Cyber law touches all the transactions and activities of internet, WWW.

In India cyber law was enforced through IT Act, 2000 based on UNCITRAL (United Nations Commission for International Trade Related Laws). It purpose is to provide legal recognition to electronic commerce.

The Act was later amended in December 2008 to provide additional focus on information security i.e. It Act, 2008. Major amended are

Digital Signatures i.e authentication of electronic records.

Electronic Governance i.e. E-documents get legal recognition.

The maximum penalty for any damage to computers is fine up to 1 crore.

Other amended acts such as IPC 1860, 1872, 1891 and 1934.

Technology & Society:

Technologies whose value and impact arise primarily from their use in economic and social sectors. The impacts of ICT have had on the development of economies, societies and culture include

Economic impacts include the globalization of production in goods and services, changes in international trade and distribution network, changes in pattern of consumption, virtualization of some products and behaviors and growing the importance of ICT sector within the world. The economic benefit include

Secure transactions, Ease of availability, Net banking, Global market

Social impact include mass market access to an increased information resources, enhanced, new pattern of work and human settlement and changes in the relationships between government, citizen and the state.

E-Waste Management:

Electronic waste describes discarded electrical or electronic devices. “Electronic waste” may also be defined as discarded computers, office electronic equipment, entertainment device electronics, mobile phones, television sets and refrigerators. This includes used electronics which are destined for reuse, resale, salvage, recycling or disposal.

Electrical and Electronic eq**uipm**ent contains metallic and non-metallic elements such as Copper, Aluminium, Gold, Silver, Palladium, Platinum, Nickel, Tin, Lead, Iron, Sulphur, Phosphorous, Arsenic etc.

The recycle and recovery includes the following unit operations

- Dismantling involves removal of parts containing dangerous substances, parts containing valuable substances.

- Separation of ferrous metal, non-ferrous metal and plastic.
- Repair and reuse.
- Recovery of valuable materials.
- Disposal of dangerous materials.

The e-waste disposal and recycling are very much necessary and important for the benefit of people, environment and the nation. The key benefits are

- Allows for recovery of valuable precious metals
- Protects public health and water quality.
- Creates jobs
- Toxic waste
- Saves landfill space.

Awareness about health concerns related to the usage of Technology

Awareness about health concern related to the usage of technology:-

1. Digital eye strain

- Symptoms of digital eye strain may include:
- Blurred vision
- Dry eyes
- Headaches
- Neck and shoulder pain

2. Emotional problems

- Makes you feel anxious or depressed.

3. Sleep problems

4. Musculoskeletal problems

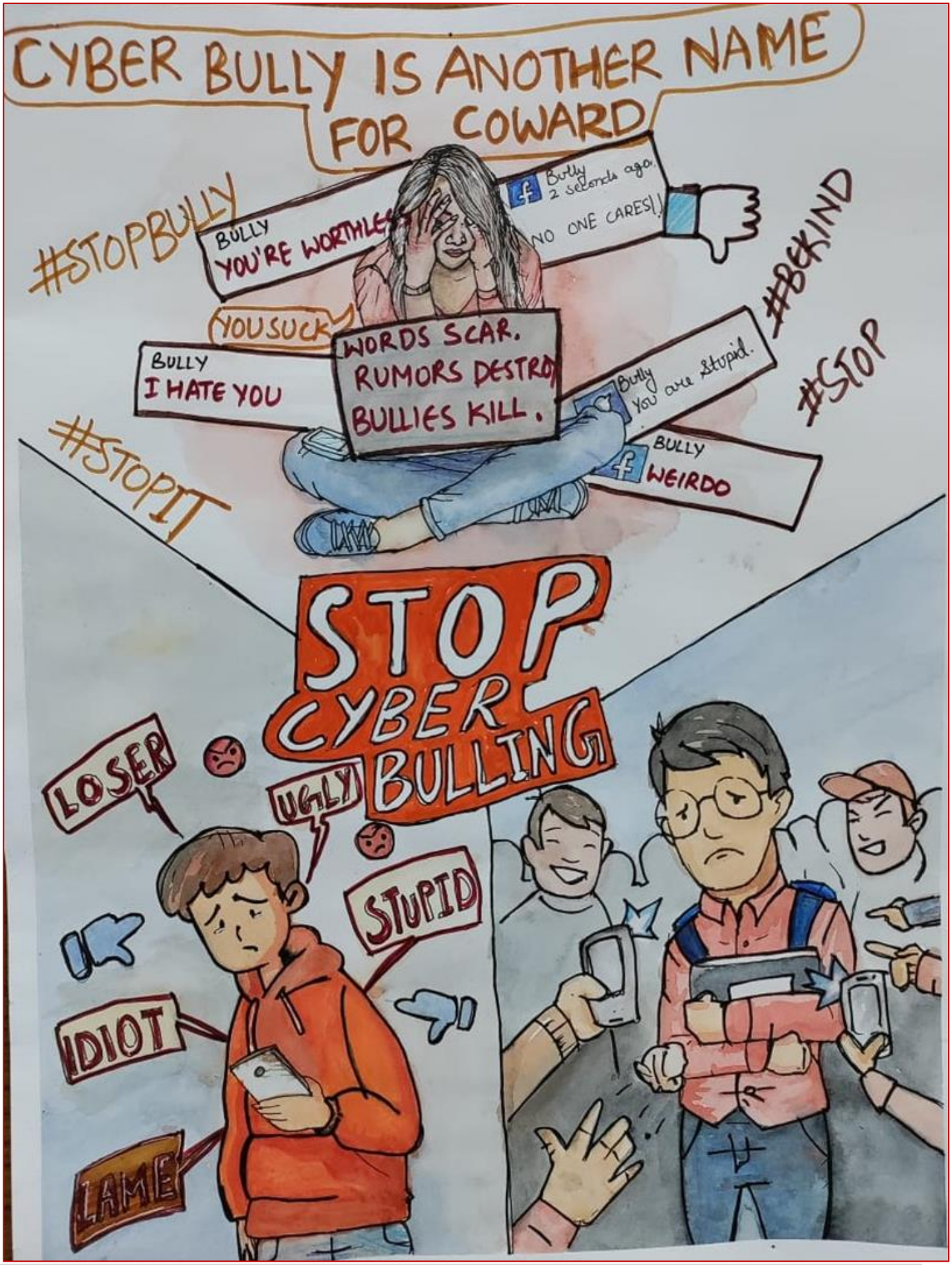
- When you use a Smartphone, the chances are that you're holding your head in an unnatural forward-leaning position.
- This position puts a lot of stress on your neck, shoulders, spine and repetitive strain injuries
- of the fingers, thumbs, and wrists.

5. Negative effects of technology on kids:

- Too much screen time or low-quality screen time may lead to
- Behavioral problems
- less time for play and loss of social skills
- obesity
- sleep problems
- violence

MEASURES TO SAFEGUARD FROM NEGATIVE TECHNOLOGICAL EFFECTS

- Clear your phone of unessential apps to keep you from constantly checking it for updates.
- Take frequent breaks to stretch, create an ergonomic workspace and maintain proper posture while using devices
- Carve out a specific, limited amount of time to use your devices.
- Turn some television time into physical activity time.
- Keep electronic devices out of the bedroom. Charge them in another room. Turn clocks and other glowing devices toward the wall at bedtime.
- Make mealtime gadget-free time.
- Prioritize real-world relationships over online relationships.
- CHECK OUT else you will be WIPED OUT:-
- Technology is a part of our lives. It can have some negative effects, but it can also offer many positive benefits and play an important role in education, health, and general welfare.
- Knowing the possible negative effects can help you take steps to identify and minimize them so that you can still enjoy the positive aspects of technology.


CYBER BULLY IS ANOTHER NAME FOR COWARD
#STOPBULLY
BULLY
YOU'RE WORTHLESS
Bully
2 seconds ago
NO ONE CARES!!
#BEKIND
#STOP
YOU SUCK
WORDS SCAR.
RUMORS DESTRO
BULLIES KILL.
BULLY
I HATE YOU
Bully
You are stupid.
#STOPIT
BULLY
WEIRDO
STOP
CYBER
BULLING
LOSER
UGLY
STUPID
IDIOT
LAME