**#** **Data analytics** Data analytics is the science of analyzing raw data to find trends and answer question

**# What is python?** python is both compiled and interpreted, object-oriented, high-level programming language with dynamic semantics.

**# Single line comments** to add single line comment, # hash is used. python completely ignores anything written after #.

**# Variables** 1, Variables are placeholders, which can store a value. In simple words, variable is a container that hold data inside it as a value. Input- a= “hello world” Print(A) Output- hello world 2, Make sure to not use spaces while creating a variable. One can use (\_) underscore to separate the names while writing a variable. 3, a variable name should never start with a number or special symbols.

**# Datatypes:** 1, Text- type: string (str) 2, Numeric types: integer(int), floating point(float), complex3, Sequence types: list, tuple and range 4, Mapping type: dictionaries (dict)5, Set type: set, frozenset 6, Boolean type: bool7, Binary types: bytes, bytearray, memoryview

**# User-inputs** to ask for the input from the user. Default datatype is sting.Input:Name = input (“enter your name here”)Print (name)Age = int (input (“enter your name here”))Print (age)Output:Enter your name here mayank bhattEnter your age here 15

**Type casting** Conversion of one datatype to another is called type-casting. There are two types of types of type-casting:1, implicit type conversion: where python itself converts one datatype to another.2, explicit type conversion: where the user converts one datatype to another.

**Operators and operands:** Operators indicates what operation is to be performed while operands indicate on what the action or the operation should be performed.X + y = 0In the given expression, x, y, and 0 are operands and operators.

**Types of operators:** Operator can be further divided into 6 categories:1 arithmetic operator 2 comparison operators 3 logical operators4 assignment operators 5 identity operators 6 membership operators7 bitwise operators

**Arithmetic operators** Addition (+), subtraction (-), multiplication (\*), division (/), floor division (//) ka matlad hota hai quotient before zero, exponentiation (\*\*) is ka matlad hota hai power, modulus (%) ka matlad hai reminder before zero.

**comparison operators** ‘<’ less then, ‘<=’ less than or equal to, ‘!=’ not equal to, ‘==’ equal to, ‘>=’ greater than or equal to, ‘>’ greater then

**logical operators**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Meaning** | **Example** |
| And | True if both the operands are true | X and y |
| Or | True if either of the operands is true | X or y |
| Not | True if operand is false (complement the operand) | Not x |

|  |  |  |
| --- | --- | --- |
| **Operator** | **Example** | **Equivalent to** |
| = | X = 6 | X = 6 |
| += | X += 6 | X = x + 6 |
| -= | X -= 6 | X = x – 6 |
| \*= | X \*= 6 | X = x \* 6 |

**assignment operators**

assignment operators are used in python to assign values to variable.A = 6 is a simple operator that assigns the value 6 on the variable a on the left.

**identity operators** identity operator is used to compare items to see if they are the same object with the same memory address. Types: is, is not

**membership operators** membership operators are used to check the presence of sequence in an object.Types: in, not in

**bitwise operators** these operators are used to compare the binary numberstypes: AND (&) operator, OR (|) operator, XOR (^) operator, << zero fill left shift, >> zero fill right shift

|  |  |
| --- | --- |
| Operator | Result |
| 0 & 0 | 0 |
| 1 & 0 | 0 |
| 0 & 1 | 0 |
| 1 & 1 | 1 |

AND operator:Implementation of and operation on binary digits

To find a binary = Print (bin(number))

|  |  |
| --- | --- |
| Operation | Result |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |

OR operator Bitwise Or operations

XOR operator Bitwise XOR Operations

|  |  |
| --- | --- |
| Operation | Result |
| 0 ^ 0 | 0 |
| 1 ^ 0 | 1 |
| 0 ^ 1 | 1 |
| 1 ^ 1 | 0 |

**Conditional statements** Conditional statement allows computer to execute a certain condition only if it is true. Types of conditional statements: 1 If the statement 2 if-else statement 3 if-elif-else statement 4 nested statements 5 short hand if statement 6 short hand if-else statement

If statement the if statement is the most fundamental decision-making statement. the if statement in python has the subsequent syntax: if expression statement -: a = 10 if (condition): body of if print (“thankyou”)

if-else statement if-else statement is used when you want to give two conditions to the computer. Here if one condition is false, program executes another condition. If condition: #will executes this block if the condition is true Else: #will executes this block if the condition is false statement -: a = 10 if (condition): body of if else: body of else print (“thankyou”) if-elif-else statement in this case, the if condition is evaluated first. If it is false, the elif statement will be executed, if it also comes false then else statement will be executed.

For multiple condition: body of if elif condition: body of elif else: body of else print (“thankyou”)

nested statement a nested if statement is one in which an if statement is nestled inside another if statement. This is used when a variable must be processed more than once. The nested if statement in python has the following syntax: if(condition1): #executes if condition 1 is true If (condition2): # executes if condition 2 is true # condition 2 ends here # condition 1 end here

short hand if statements short hand if statement is used when only one statement needs to be executed inside the same line which holds the if statement The short hand if statement in python has the following syntax: If condition: statement

short hand if-else statement it is used to mention if-else statements in one line in which there is only one statement to execute in both if and else blocks. In simple words, if you have only one put it all on the same line.

**Loops** A loop means to repeat something in the exact same way.Types of loops are:1 for loop 2 while loop 3 while true4 nested loops

For loop For loop is a loo that repeats something in a given range. The range has a staring point, ending point and a step/gap in it +1 is added to the ending point while defining a range

While loop While loop executes till the given condition is true. In while loop, the increment is done inside the loop.

While true It is an infinite loop. To break a while true loop, break statement is used.

Nested loop A loop inside a loop is called as nested loop. Nested loops are also used to solve pattern problems.

**For loop with conditional statements.** The use of if-else statement increases the ability of for loop to completes the task effectively. By using if-else we can provide with special conditions inside for loop.

**Break and continue statement** Continue statement: - continue statement is used when you want to skip a particular condition.Break statement: - break statement is used when you want to destroy a loop at a certain condition and come out of the loop.

**String.** String are the combination of number, symbols and letters, enclosed inside quotations.Creation of a string: strings can be created by enclosing numbers, letters or special symbols inside double quotations.It means assigning a string value to a variable.A = (“hello world”)Print(A)

**String function’s: -** 1, length = (len) 2, count = (.count) 3, upper = (.upper)4, lower = (.lower) 5, index = (.index) 6, capitalize = (.capitalize)7, casefold = (.casefold) 8, find = (.find) 9, format = (.format)10, center = (.center)

Format for format: - Name = “mayank” A = “my name is { }” Print a.format (name) 11, isalnum - returns true if all characters in the string are alphanumeric. 12, isalpha - returns true if all characters in the string are in the alphabet. 13, isdecimal - returns true if all characters in the string are decimal. 14, isdight - returns true if all characters in the string are digits. 15, isnumeric - returns true if all characters in the string are numeric 16, islower - converts a string into lower case. 17, isupper - returns true if all characters in the string are upper case. 18, isspace - returns true if all characters in the string are whitespaces. 19, istittle - returns true if the string follows the rules of a title. 20, endwith() – returns true if the string ends with the specified value 21, startwith() - returns true if the string starts with the specified value 22, swapcase() - swaps cases, lower case became upper case and vice versa 23, strip() – returns a trimmed version of the string 24, split() - splits the string at the specified separator, and returns a list 25, ljust() - returns a lift justified version of the string 26, rjust() – returns a right justified version of the string 27, replace() – returns a string where a specified value is replaced with a specified value 28, rindex() – searches the string for a specified value and returns the last position of where it was found 29, rfind() – searches the string for a specified value and returns the last position of where it was found

**Slicing in string** Is used cut in small pieces (::)

**Introduction to Lists** Lists is the collection of ordered and mutable data.1, Lists are written inside the squared brackets.2, the value inside a list is separated by coma (,).3, mutable means once created, they can be changed.4, multiple datatypes can be written inside a list.

**Slicing lists.** Is used to make cut in small pieces.a = ["Ironman","Thor","Captain america","Hulk"]ironman ka index number 0 hai

**List Iteration.** Is used to put in loop in particular line. Iteration used for loop. Iteration using for loop with range and length function.Iteration using while loop. Using short-hand for loop.

**List function part (1)** 1, to find the length of a list (len) 2, to count an occurrence of a particular element (.count)3, to add to the list (.append) 4, to add to a specific location (.insert)5, to remove from a list (.remove) 6, to remove from a certain location (.pop) 7, to create a copy of a list (.copy) 8, to access an element (.index)9, to entend the list (.extend) 10, to reverse the list (.reverse)11, to sort the list (.sort) 11, to clear all the data from the list (.clear)

**List comprehension.** It is used to copy list and write again.

**Tuples.** Tuples are the collection of ordered and un-mutable data. For tuples no brackets are mandatory. By choice one can use parentheses. The value inside a tuple is separated by coma (,). Once created, tuples cannot be changed. Multiple datatypes can be written inside a tuple. Identify by (,)

**Slicing and iteration in tuples.** Slicing is used to cut tuples in small part Identify by (,) Iteration is used to copy data in vertical line by the help of loop

**Conversion of tuples and tuple functions.** Conversion of tuples is used to change the type of any word and function of tuple are count method and index method.

**Introduction to dictionary** Dictionary allows user to write the data in the form of keys and values. Dictionaries are enclosed inside curly brackets {} Keys and values are separated by colon Every key values are separated by a coma(,).

**Iteration in dictionary** { for example a = (“name” : “mayank” )} 1, by for loop to get (for x in a: print(x)) to get “name” from example 2, by for loop with [] (for x in a: print(a[x])) to get “mayank” from example

3, by .values (for x in a.values(): print(x)) to get “mayank” from example 4, by .items (for x,y in a.items(): print(x,”:”,”y”)) to get “name : mayank” from example

**Dictionary functions** 1, get (.get) 2, get (.get) 3, keys (.keys) 4, values (.values) 5, copy (.copy) 6,setdefault(.setdefault) 7,update (.update) 8, pop (.pop) 9, popitem() 10,clear(.clear)

**Nested dictionaries** nested dictionaries are a part of dictionaries in which you can store more then one dict in one line

**Sets** Sets are unordered collection of data. Every element inside the set is unique and mutable. 1, sets are written inside the curly brackets. 2, the values inside a set are separated by coma (,). 3, mutable means once created, they can be changed.

**Set functions**  1, add(.add) 2, pop(.pop)3,remove(.remove) 4, discard(.discard)5, copy(.copy) 6,isdisjoint(.isdisjoint) 7, issubset(.issubset) 8,issuperset(.issuperset) 7,issuperset(.issuperset) 10, update(.update) 11, clear(.clear) 12,union(.union) 13, difference(.difference) 14, difference update(.difference update) . 15, intersection(.intersection) 16, intersection update(.intersection update) 17, symmetric difference(.symmetric difference) 18, symmetric difference(.symmetric difference)

**Functions** Functions are a set of cord, which once created they can be used throughout the program. Functions help break our program into smaller parts and helps it look more organized and manageable.

**Parameter and Arguments:** Parameters: Parameters are the variable written inside the parentheses with the name of function. Arguments: Arguments are the values passed to the parameters while calling the function.

**Return statement and recursion in Python** Return statementReturn keyword in python is used to exit a function and return the value of the function. Recursion in Python Recursion in most commonly used mathematical and programming concept. In simple words, recursion means a function can call itself, giving us a benefit of looping though data in order to get a result.

**Advantages and Disadvantages of Recursion** Advantages: 1, they make the code look clean and organized. 2, By the use of recursive functions, a complex task can be broken down into small sub- parts. 3, Sequence generation becomes easier. Disadvantages: 1, Recursive functions take up a lot of memory. 2, sometime the logic becomes hard to follow. 3, Sometime debugging makes code difficult.

**Lambda function in python**  1, It is used when an anonymous function is required for a short period of time. 2, it can take numerous arguments. 3, it can only have one expression.

**Local and global variables** local Variable: local variables are restricted to only one block of code and cannot be changed throughout the program. Global variables: Global variables are not restricted to one block of code they can be changed throughout the program.

**Introduction to modules** Modules is the (.py) files, that contains set of functions you want to include in your program.

**In-built Modules in python** 1, datetime = {staring mai datetime and import datetime} # (.striftime(“%a”)) = to get day of given date # (.striftime(“%m”)) = to get month of given date # (.striftime(“%y”)) = to get year of given date # (.striftime(“%p”)) = to get pm/am of given date # (.striftime(“%M”)) = to get minutes of given date # (.striftime(“%S”)) = to get second of given date # (.striftime(“%f”)) = to get micro second of given date 2, random = {staring mai random and import random} # (.randint) = for to get random number # (.choice) = to get choice of anything 3, math = {staring mai import math} # max = to get a maximum number # min = to get a minimum number # pow = to get power of (2,4) = 24 # .sqrt = to get square root of given number(starting mai (math)) # abs = to get positive number # .ceil = by round of + one number (2.3=3) (starting mai (math)) # .floor = by round of – one number (2.3=2) (starting mai (math))

**Creation of Modules** To create a module, all you need to do is create a .py file in a similar path to your python file inside that file, you can add required functions you need program to perform. To call the module inside your program, all you need to do is use import keyword followed by the name of your.py file

**Introduction NumPy** NumPy: NumPy is the short form of Numerical Python. In 2005, Travis Oliphant created NumPy package. NumPy is a package that defines a multi-dimension array object and associates fast math functions that operate on it. It also has functions for working in domain of linear algebra, Fourier transformation and matrices. In simple words, it is the fundamental package for scientific computing in python.

**Arrays:** 1, An array is defined as a collection of items that are stored at contiguous memory locations. 2, it is a container which can hold a fixed number of items, and these items should be of the same type. 3, A combination of arrays saves a lot of time. The arrays can reduce the overall size of the code.

**Advantages of using Arrays:** 1, NumPy uses much less memory to store data. 2, NumPy makes it extremely easy to perform mathematical operations on it. 3, Used for the creation of n-dimensional arrays. 4, Finding elements in NumPy array is easy. 5, A list cannot directly handle mathematical operations, while Array can. 6, an array consumes less memory than a list.

7, Using an array is faster than list. 8, a list can store different datatypes, while you can’t do that in an array.

**Arrays v/s Lists:** 1, A list cannot directly handle mathematical operations, while Array can. 2, An Array consumes less memory than a list. 3, Using an array is faster than list. 4, a list can store different datatypes, while you can’t do that in an Array. 6, A list is easier to modify since a list store each element individually, it is easier to add and delete an element individually, it is easier to add and delete an element than an array does. 7, In lists one can have nested data with different size, while you cannot do the same in array.

**NumPy-Creating Arrays, Slicing and Attributes:** Creating Arrays: Firstly import numpy then a=numpy.array([10,20,30,40]) Slicing: simply like print([0:3]) Attributes: 1, print(numpy.shape (a)) = to count row and columns 2, print(numpy.size(a)) = to count total number 3, print(numpy.ndim(a)) = to count dimension 4, print(a.dtype) = to know data type 5, print(len(a)) = to know the length 6, print(a.astype(int)) = convert to different type

**Mathematical Operations and Functions on Arrays** Function: 1, np.subtract(#,#) = to subtract 2, p.add(#,#) = to add 3, np.divide(#,#) = to divide 4 , np.multiply(#,#) = to multiply 5, np.exp(#,#) = to 6, np.sqrt(#) to make square root 7, np.pow(#) = to make power

**Combining and splitting Arrays** Concatenate**:** firstly import numpy then a = numpy.array([[a,b],[c,d]]) then print(np.concatenate([a]) Horizontal concatenation: print(numpy.hstack([a])) Vertical concatenation:print(numpy.vstack([a])) Second Type: print(np.concatenate([a]),axis = 0)) for Vertical concatenationprint(np.concatenate([a]),axis = 1)) for Horizontal concatenation array\_split: a = numpy.arry([[20,40,30],[40,20,10]]) b = numpy.array\_split(a,5) print(b) [array([[20,40,30]]), array([[40,20,10]]) to split the array

**Adding and removing elements in the arrays** 1, np.append(h,g) = append items to an array 2, np.insert(a,1,5) (array, index, value) = inserts items in an array 3, np.delete(a,[1]) = delete items from an array

**NumPy- Sort, Filter and Search** Sort: sort is used to sort array ar = np.array([3,4,1,7,8]) print(np.short(ar)) filter: we use where ar = np.array([3,4,1,7,8]) s = np.where(ar % 2 == 0) print(s) Search: it is used to search ar = np.array([3,4,1,7,8]) ss = np.searchsorted(ar,5) print(ss)

**Aggregating functions in numpy** 1, (.sum) = print(np.sum(a)) to get sum 2, (.min) = print(np.min(a)) to get minimum . 3, (.max) = print(np.max(a)) to get maximum 4, (.size) = print(np.size(a)) to get size of array . 5, (.mean) = print(np.mean(a)) to get mean 6, (.cumsum) = print(np.cumsum(a)) to get sum like a,a+b… 7, (.median) = print(np.median(a)) to get median 8, (.cumprod) = print(np.cumprod(a)) to get a,a\*b,a\*b\*c

**Statistical Functions Numpy**  1, (.mean) = sum of all the values/number of values 2, (.median) =central value after sorting 3, (.mode) = most repeated value import statistics as stats print(stats.mode(####)) 4, (.std) = a standard deviation is a statistical measure indicating the spread of a distribution of data . 5, (.var) = square root of standard deviation 6, (.corrcoef) = show the relation between both -1 represent inversely proportional relationship 1 represent proportional relationship 0 means no relationship

**Introduction to pandas** Pandas is a Python package providing fast, flexible and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. Here are just a few of the things that pandas do well: 1, it has functions for analyzing, cleaning, exploring, and manipulating data. 2, The name “Pandas” has a reference to both “Panel Data”, and “Python data Analysis” and was created by McKinney in 2008. 3, Easy handling of missing data. 4, Size mutability: columns can be inserted and data frame and higher dimensional objects. 5, Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let series, Data Frame etc. 6, Automatically align the data for you in computations. 7, Powerful, flexible group by functionality. 8, Intelligent label-based slicing, fancy indexing, and sub setting of large data sets. 9, Flexible reshaping and pivoting of data sets.

**Data structures in Pandas** the best way to think about the pandas data structures is as flexible containers for lower dimensional data. For example, Data frame is a container for series, and series is a container for scalars. We would like to be able to insert and remove objects from these containers in a dictionary-like fashion.

**Series in pandas:** Pandas series is a one-dimensional ladled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called index. Pandas series is nothing but a column in an excel sheet. The object supports both integer and label-based indexing and provides a host of methods for performing operations involving the index.

**Data frames:** Pandas Data frames is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas Data frame consists of three principal components, the data, rows, and columns.

**Exploring Data in Pandas Creation of Data Frames in Pandas** 1, DataFram: to create your own data Import pandas data = {“###”: [“###”, “###”, “###”],} df = pandas.DataFrame(data) print(df) 2, To take it from csv firstly take the csv file in jupyter notebook then if not in jupyter notebook folder copy the path of the file. data = pd.read\_csv(“file name. csv”) print(data) 3, to take it from excel go on command prompt (cmd) type (pip install openpyxl) same step data = pd.read\_excel(“file name.xlsx”) print(data) 4, to print big fille to see upper data print(data.head(##no you want to see##)) to see lower data print(data.tail(##no you want to see##)) 5, to get info of data print(data.info()) 6, to describe data data.describe() 7, to see is data null print(data,isnull().sum())

**Handling Duplicate Values in Pandas** to see any duplicate values print(data[“##any column## ”].duplicated().sum()) to drop the same name print(data.drop-duplicates(“##any column##”))

**Working with Missing Data in Pandas** 1,to fill all nan value firstly import pandas print(data.replace(np.nan, “##value you want to fill##”)) 2, to fill selected value data[“##selected value##”] = data[“##selected value##”].replace(np.nan, “##what you want it to be”) print(data) 3, to fill nan value by the value blow them print(data.fillna(method = “bfill”)) 4, to fill nan value by the value above them print(data.fillna(method = “ffill”)) 5, to fill same value to all nan value print(data.fillna(“## word you want to fill ##”)) 6, to fill numeric value take the mean of all the data then fill it in all nan value. 7, to fill string value use (“bfill” or “ffill”).

**Columns transformation in pandas** 1, to make a new column from given data like firstly call data from your file

data.loc[(data[“## from where you want to find it from column name##”] == 0\*\* Write your condition\*\*), “##new column##”] = “##what you want to write in it (yes)##” data.loc[(data[“## from where you want to find it from column name##”] > 0\*\* Write your condition\*\*), “##new column##”] = “##what you want to write in it (no)##” 2, to join two columns firstly call data from your file. Data[“## New column you want to create ##”] = data[“column you want to add”] + “ ” + data[“second column you want to add ”] print(data) 3, to write data in certain way like .str.upper() = MAYANK .str.capitalize() = Mayank ya data[“column you want to add”] is ka baad aata hai 4, to get the short form of any data like print a data then def extract(value): return value[0:3] data[“## what you want to name the short data which will be created ##”] = data[“## name of a column you want to make short ##”].map(extract) print(data)

**Group by in pandas** to make in summary of given data firstly import data then a = data.groupby(“## on which dasis ##”).agg({“ of what you want to ”: “ count ”}) print(a) a = data.groupby([“## on which dasis ##”, “## on which dasis ##”]).agg({“ of what you want to ”: “##anything you want to find##”}) print(a)

**Merge, Join and Concatenate in Pandas** To Merge, Join And Concatenate in Pandas firstly import data then 1, Merge: to add two data print(pd.merge( first data, second data, on = “ similar column between them ”)) 2, Join: To join two data but some data is different with nan value print(pd.merge(left = first data,right = second data,on = “ similar column between them ”, how = “## left , right ##”)) 3, Concatenate: it also help to join two data firstly import data then make their name short like df1 = pandas.DataFram(data1) same for second data df2 = pandas.DataFram(data2) print(pandas.concat([df1,df2]))

**Pandas | Compare DataFrames**  Compare DataFrames is used to compare the data of same kind like same data but of different month to compare the data follow the step: 1, print your both data in pandas.DataFrame {Pandas location} if you have same data to update every month in that case we use (##data we printed pandas.DataFrame ##.loc[##row no of place you want to change##, “## column name you want to change ##”]=##from what you want to change ##) 2, print(data1.compare(data2)) to see the comparison of data in separate column. 3, print(df1.compare(df2,aling\_axis=0 )) to see in row ways in same column. 3. print(df1.compare(df2,keep\_equal=##True,False##)) to see the comparison of data in separate column. In False we see it in point form. In True we see it in normal value. 4, print(df1.compare(df2,keep\_shape=##True, False##)) to see the comparison of data in separate column. In False we see it in point form. In True we see all values but unchanged value became nan.

**Pandas – Pivoting and Melting DataFrames** PivotingIt help to arrange the values in order like this data then use data then make it in pd.DataFrame then to change the data print (df.pivot("keys","Names","House")) print (df.pivot(index="keys",columns="Names",values=["House", “Grades”])) to get values in two different boxes Melting It help to arrange the values in order like this data then use data then make it in pd.DataFrame then to change the data print(pd.melt(df,id\_vars=["Names"],value\_vars=["Houses"])) ya same Pivoting ki jasa hai bas vertical mai hai

Name Mayank Satyam Anju Sonali keys k1 Nan green red Nan k2 blue Nan Nan Red

keys Names House 0 k1 Mayank red 1 k2 Satyam blue 2 k1 Anju green 3 k2 Sonali red

**Introduction to matplotlib** Data visualization: Data visualization is the graphical representation of information and data. In the world of Big Data, data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions. Matplotlib is a low-level graph plotting library in python that serves as a visualization utility. Matplotlib was created by John D. Hunter. Matplotlib is open-source and we can use it freely. **1, Bar Plot Matplotlib**= firstly import matplotlib.pyplot then put data colors = (## Name of colors ##) plt.bar(\_\_ ya wala, || ya wala, color = colors, edgecolor = “##name of color ##” ) to name the x axis = plt.xlabel(“## name of axis ##” , fontsize = ## your preferred size ##) to name the y axis = plt.ylabel(“## name of axis ##” , fontsize = ## your preferred size ##) to name the title plt.title(## Name of title ##, fontsize = ## your preferred size ##) if you want to take data from any file firstly impot pandas and matplotlib.pyplot then take data df = pandas.DataFrame(data) grouped\_by = df.grouped(“by which column you want to make data from it ”)[“## place where value is given ##”].sum() print(grouped\_by) plt.bar(grouped\_by.index,grouped\_by.values) plt.show()

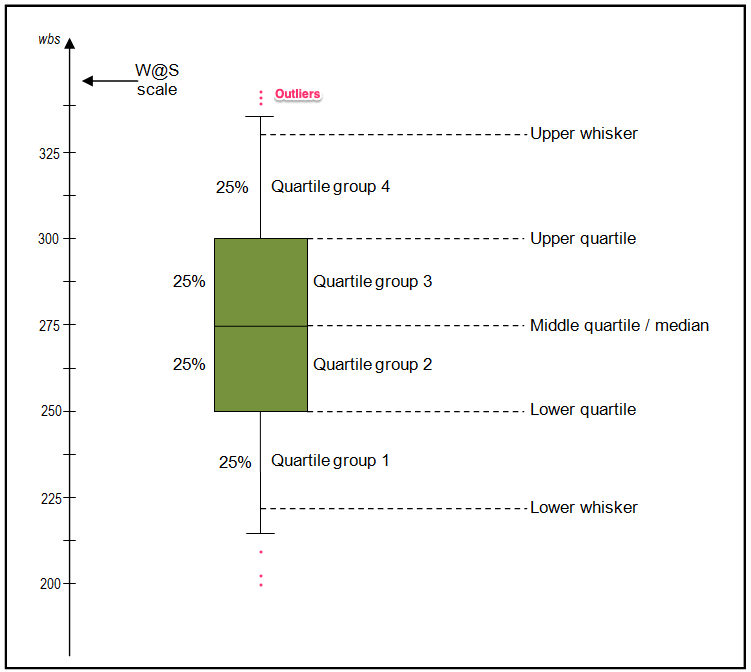
**2, Line plot-Matplotlib**= firstly import matplotlib.pyplot then put data and numpy plt.plot(\_\_ ya wala,|| ya wala, marker = “## jo point per lagana hai ##”, ls = “## line ki shape ##”, colors = (## Name of colors ##),label= “## Name of line ##” ) plt.plot(\_\_ ya wala,|| ya wala, marker = “## jo point per lagana hai ##”, ls = “## line ki shape ##”, colors = (## Name of colors ##),label= “## Name of line ##” ) plt.legend() plt.show() if you want to take data from any file firstly impot pandas and matplotlib.pyplot then take data df = pandas.DataFrame(data) grouped\_by = df.grouped(“by which column you want to make data from it ”)[“## place where value is given ##”].sum() print(grouped\_by) plt.plot(grouped\_by.index,group\_by.values) print(df) plt.show()

**3, Scatter Plot Matplotlib=** firstly import matplotlib.pyplot then put data and numpy plt.scatter(\_\_ ya wala,|| ya wala, marker = “## jo point per lagana hai ##”,color = “## Name of colors ##”) ya wala after x and y axis ka baad color = ## Name of colors ## plt.scatter(\_\_ ya wala,|| ya wala, marker = “## jo point per lagana hai ##”,camp = “viridis” ,c = color) . plt.colorbar() plt.show() if you want to take data from any file firstly impot pandas and matplotlib.pyplot then take data df = pandas.DataFrame(data) grouped\_by = df.grouped(“by which column you want to make data from it ”)[“## place where value is given ##”].sum() print(grouped\_by) plt.scatter(df[ “## collum name ## ”],df[“## collum name ##”]) plt.show() print(df)

**4, Pie Chart Matplotlib=** firstly import matplotlib.pyplot then put data and numpy then your values c = [## colors you want to put ##] to point some part ex = [## 0 for not changing 0.1 to show it special ##] plt.pie(## part where value Is in number ##, labels =## name of value ##,colors = c , explode = ex, shadow = True, autopct = “%.f” ## %.f ka matlab value of number , %.2f ka matlab value of number in point ##) plt.show() if you want to take data from any file firstly impot pandas and matplotlib.pyplot then take data df = pandas.DataFrame(data) grouped\_by = df.grouped(“by which column you want to make data from it ”)[“## place where value is given ##”].sum() print(grouped\_by) plt.pie(grouped\_by.values, labels = grouped\_by.index) plt.show() print(df)

**5, Box Plot-Matplotlib=** firstly import matplotlib.pyplot then put the values plt.boxpllot(## value name ##) plt.show() if you want to take data from any file firstly impot pandas and matplotlib.pyplot then take data df = pandas.DataFrame(data) plt.boxplot(df [“## thing you want to print on the chart ##”]) plt.show() Min/LF = Q1 – 1.5 ( IQR ) Max/UF = Q3 + 15 ( IQR ) IQR = Q3 – Q1 Q1 = 25/100 \* ##no of element + 1##





**6, Histogram plot Matplotlib=** firstly import matplotlib.pyplot then put the values plt.hist(## value name ##, bins = ## no of collum you want to make ## , edgecolor = “## name of color you want in your outline ##”, color = “ Name of color you want in your collum ” ) plt.show() if you want to take data from any file firstly impot pandas and matplotlib.pyplot then take data df = pandas.DataFrame(data) print(df) plt.hist(df[“by which column you want to make data from it”], bins = ## no of collum you want to make ##) plt.show()

**7, Violin Plot Matplotlib:** firstly import matplotlib.pyplot then put the values plt.violinplot(## value name ##, showmedians = True) plt.show() if you want to take data from any file firstly impot pandas and matplotlib.pyplot then take data df = pandas.DataFrame(data) print(df) plt.violinplot(df[“by which column you want to make data from it”], showmedians = True ) plt.show()

8, Stem plot Matplotlib: firstly import matplotlib.pyplot then put the values plt.stem(## value name ##, linefmt = “## according to your preference like -- \*\*##”, markerfmt = “D”, orientation = “horizontal / vertical”) plt.show()

if you want to take data from any file firstly impot pandas and matplotlib.pyplot then take data df = pandas.DataFrame(data) if data is to big df2 = (df.head(## no of you want to see ##)) plt.stem(df2[“## what you want to compared ##”]) plt.plot(df2[“## what you want to compared ##”]) plt.show()

**9, stack plot Matplotlib:** firstly import matplotlib.pyplot then put the values and the value you want to compare it to plt.stackplot(## the values ## , ## the values you want to compare it to ## , labels = [“ ## name them ”]) plt.legend() plt.show() if you want to take data from any file firstly impot pandas and matplotlib.pyplot then take data df = pandas.DataFrame(data) print(df) grouped = df.groupby([“## from where you want it to compared ##”])[“## what you want to compared ##”].mean() plt.stackplot(df[“## from where you want it to compared ##”].unique(), grouped[“## what you want to compared separately ##”]) plt.show() **10, Step plot Matplotlib:** firstly import matplotlib.pyplot then put the values and the value you want to compare it to plt.step(## the values ## , ## the values you want to compare it to ## , where = “post”, marker = “o”) plt.show if you want to take data from any file firstly impot pandas and matplotlib.pyplot then take data print(df) grouped = df.groupby(“## from where you want it to compared ##”).agg({“## what you want to compared ##”: “sum”}) print(group) plt.step(group.index, group[“## what you want to compared ##”], where = “mid”, marker = “o”) plt.show()

**11, Legend Matplotlib:** is a function of python to make your graph more presentable for it you firstly import matplotlib.pyplot then put the values then if you want to make your graph in shape plt.figure(figsize = [5,3 you can use whatever you want]) plt.plot(## the values ## , ## the values you want to compare it to ## , labels = [“ ## name them ”]) to see feature of plt.legend(loc = 0) to set location of legend plt.legend([“anything”, “you want”, “name them”]) plt.legend(bbox\_to\_anchor = (0.8,1,2)) it make size of box plt.legend(ncols = 2) it help to make collum in legend plt.show()

**12, Subplot- Matplotlib:** it is used to make graph of your preferred size so for it firstly import matplotlib.pyplot then put the values x and y then plt.subplot(1,2,1) plt.title(“ name of your graph ”) plt.plot(x,y) plt.show()

**13, Save a chart Using Matplotlib:** to save the graph in png form if you want to take data from any file firstly impot pandas and matplotlib.pyplot then take data df = pandas.DataFrame(data) grouped\_by = df.grouped(“by which column you want to make data from it ”)[“## place where value is given ##”].sum() print(grouped\_by) plt.bar(grouped\_by.index,grouped\_by.values) plt.savefig(“##by which name you want to same##.png ”, facecolor = “## name of color ##”) plt.show()

**Introduction to Seaborn:** Data visualization is the graphical representation of information and data. In the world of big data, data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions.

**Seaborn:** is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

**1, Line Plot Seaborn=** import seaborn, pandas and matplotlib.pyplot then put your data = your values df = pd.DataFrame(data) print(df) sns.lineplot(data= data, x = “your value”, y = “your other value”) plt.show() if you want to take data from any file firstly impot seaborn, pandas and matplotlib.pyplot then take data print(data) color = sns.color\_palette(“viridis”) sns.lineplot(data = data, x= “your value”, y = “your other value”) plt.show()

**2, Bar Plot Seaborn=** import seaborn, pandas and matplotlib.pyplot then data = sns.load\_dataset(“ you can see them in seaborn github datasheet”) sns.barplot(data = data, x = “####”, y = “####”, estimator = “ whatever you want to fill ” hue = “## on which basic you want to see them ##”, palette = “## you can search them in google ##”, order = [“## you can arrange them by yourself ##”], errorbar = (“ci”,0)## to remove errorbar from bar ## ) plt.show()

**3, Hist plot seaborn=** import seaborn, pandas and matplotlib.pyplot then data = sns.load\_dataset(“ you can see them in seaborn github datasheet”) sns.histplot(data, x = “####”, y = “####”,hue = “## on which basic you want to see them ##”, kde = True ## for line ##) plt.show()

**4, scatter Plot Seaborn=** import seaborn, pandas and matplotlib.pyplot then data = sns.load\_dataset(“ you can see them in seaborn github datasheet”) sns.scatterplot(data = data, x = “####”, y = “####”,hue = “## on which basic you want to see them ##”,size = “##it tell no of it ##” )

**5, Heatmap Seaborn=** import seaborn, pandas and matplotlib.pyplot then data = sns.load\_dataset(“ you can see them in seaborn github datasheet”) gp = data.groupby(“## x axis ##”).agg({“ y axis ”: “ what you want to see ”}) plt.show()

**6, Count Plot Seaborn=** import seaborn, pandas and matplotlib.pyplot then data = sns.load\_dataset(“ you can see them in seaborn github datasheet”) sns.countplot(data = data, x = “on which basis you want to make poles ” ) plt.show()

**7, Violin Plot Seaborn=** import seaborn, pandas and matplotlib.pyplot then data = sns.load\_dataset(“ you can see them in seaborn github datasheet”) sns.violinplot(data = data, x = “## what you want to see ##”) plt.show()

**8, Pair Plot Seaborn=** import seaborn, pandas and matplotlib.pyplot then data = sns.load\_dataset(“ you can see them in seaborn github datasheet”) sns.pairplot(data, hue = “## on which basic you want to see them ##”) plt.show()

**9, Strip Plot Seaborn=** import seaborn, pandas and matplotlib.pyplot then data = sns.load\_dataset(“ you can see them in seaborn github datasheet”) sns.stripplot(data = data, x = “####”, y = “####”,hue = “## on which basic you want to see them ##” , dodge = True show them separately) plt.show()

**10, Box Plot Seaborn=** import seaborn, pandas and matplotlib.pyplot then data = sns.load\_dataset(“ you can see them in seaborn github datasheet”) sns.boxplot(data = data, x = “####”, y = “####” , orient = “vertical” to make it vertically) plt.show()

**11, Cat Plot Seaborn=** import seaborn, pandas and matplotlib.pyplot then data = sns.load\_dataset(“ you can see them in seaborn github datasheet”) sns.catplot(data = data , x=”####”,y = “####”, kind = “## whatever type of plot you want ##”) plt.show()

**12, Style And Color in Plots Seaborn=** firstly, for style import seaborn, pandas and matplotlib.pyplot then data = sns.load\_dataset(“ you can see them in seaborn github datasheet”) sns.set\_style(style = “ticks”) for small line in front of x and y axis. sns.set\_style(style = “whitegrid”) for line in background. sns.##any diagram you want to make ##(data = data , x = “####”, y = “####”) plt.show() For color import seaborn, pandas and matplotlib.pyplot then sns.palplot(sns.color\_palette(“## goggle palette color ##”)) plt.show()

**13, Multiple plots in seaborn=** import seaborn, pandas and matplotlib.pyplot then . data = sns.load\_dataset(“ you can see them in seaborn github datasheet”) a = sns.FaceGrid(data, col = “## what you want to make collum of ##”) a.map(sns.##any diagram you want to make ##,”## x = axis ##”, “## y = axis ##”) plt.show()

**14, Relational Plot Seaborn=** import seaborn, pandas and matplotlib.pyplot then . data = sns.load\_dataset(“ you can see them in seaborn github datasheet”) sns.relplot(data = data, x = “####”,y = “####”, kind = “## whatever type of plot you want ##”) plt.show()

**15, Swarm Plot Seaborn=** import seaborn, pandas and matplotlib.pyplot then . data = sns.load\_dataset(“ you can see them in seaborn github datasheet”) sns.swarmplot(data = data, x = “####”, y = “####”) plt.show()

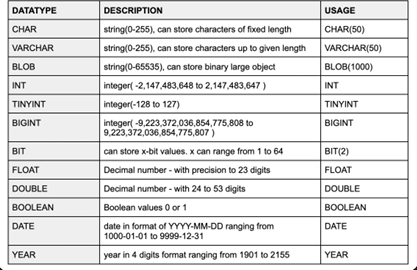
**16, Kde Plot Seaborn=** import seaborn, pandas and matplotlib.pyplot then . data = sns.load\_dataset(“ you can see them in seaborn github datasheet”) sns.kdeplot(data = data, x = “####”) plt.show()

**SQL** MySQL is a relational database management system (RDBMS) developed by Oracle that is based on structured query Language (SQL). The software used to store, manage, query, and retrieve data stored in a relational database is called a relational database management system (RDBMS). The RDBMS provides an interface between users and applications and the database, as well as administrative functions for managing data storage, access and performance

**What is SQL?**  Structured Query Language SQL is a programming language used to interact with relational databases. It is used to perform CRUD operations: Create, Read, Update and Delete

**Types of SQL Commands** DDL (Data Definition Language): create, alter, rename, truncate & drop DQL (Data Query Language): select DML (Data Manipulation Language): select, insert, update & delete DCL (Data Control Language): grant & revoke permission to users TCL (Transaction Control Language): start transaction, commit, rollback etc.

**Sql Datatypes**

****

**Keys** Primary Key It is a column (or set of columns) in a table that uniquely identifies each row. (a unique id) There is only 1 PK & it should be NOT null. Foreign Key A foreign key is a column (or set of columns) in a table that refers to the primary key in another table. There can be multiple FKs. FKs can have duplicate & null values

**Constraints**  SQL constraints are used to specify rules for data in a table. NOT NULL UNIQUE PRIMARY KEY columns cannot have a null value UNIQUE all values in column are different PRIMARY KEY makes a column unique & not null but used only for one

**Where Clause** Using Operators in WHERE Arithmetic Operators: +(addition), -(subtraction), \*(multiplication), /(division), %(modulus) Comparison Operators: = (equal to), != (not equal to), > , >=, <= *Logical Operators*: AND, OR , NOT, IN, BETWEEN, ALL, LIKE, ANY Bitwise Operators : & (Bitwise AND), | (Bitwise OR

**Operators** AND (to check for both conditions to be true) OR (to check for one of the conditions to be true) Between (selects for a given range) In (matches any value in the list) NOT (to negate the given condition)

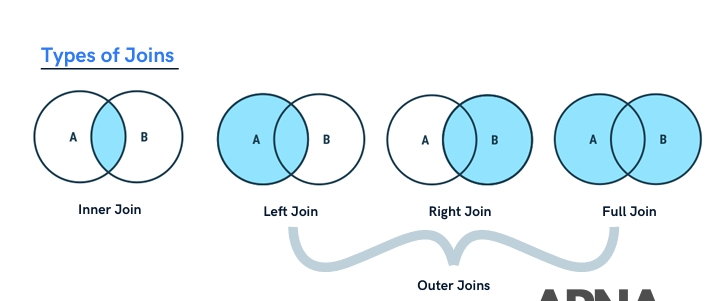
**Aggregate Functions** Aggregate functions perform a calculation on a set of values, and return a single value. COUNT() MAX( ) MIN( ) SUM( ) AVG( )

**Group By Clause** Groups rows that have the same values into summary rows. It collects data from multiple records and groups the result by one or more column. \*Generally, we use group by with some aggregation function.

**Having Clause** Similar to Where i.e. applies some condition on rows. Used when we want to apply any condition after grouping.

**Cascading for FK** On Delete Cascade When we create a foreign key using this option, it deletes the referencing rows in the child table when the referenced row is deleted in the parent table which has a primary key. On Update Cascade When we create a foreign key using UPDATE CASCADE the referencing rows are updated in the child table when the referenced row is updated in the parent table which has a primary key.

**Joins in SQL** Join is used to combine rows from two or more tables, based on a related column between them.



Inner Join: Returns records that have matching values in both tables. Left Join: Returns all records from the left table, and the matched records from the right table. Right Join: Returns all records from the right table, and the matched records from the left table. Full Join: Returns all records when there is a match in either left or right table.

**Union** It is used to combine the result-set of two or more SELECT statements. Gives UNIQUE records. To use it: every SELECT should have same no. of columns columns must have similar data types columns in every SELECT should be in same order

**SQL Sub Queries** A Subquery or Inner query or a Nested query is a query within another SQL query. It involves 2 select statements. Query Sub Queries

**TO SEE MORE SEE IN PDF OF APNA COLLEGE**