# Data Analytics Using Python

# Day1:23/02/2024

# Task1:Review of core Python concepts: DataTypes ,Operators,Control Flow,Functions,Modules,Packages.

# DataTypes:

* It is a type of data which is used in the program.There is no need of data type to declare a variable in Python
* Numeric Types: int, float, complex
* Sequence Types: list, tuple, range
* Mapping Type: dict
* Set Types: set, frozenset
* Boolean Type: bool
* Binary Types: bytes, bytearray, memoryview
* None Type: NoneType
* Text Type: str

# Operators:

* It is a special symbol which is used to perform logical or mathematical operation on data or variable
* Arithmetic Operators:+,-,*,/,%,*\*\*
* Relational Operators:==, !=,<=,>=
* Logical Operators:and,or,not
* Assignment Operators:=,+=,-=,=,/=,%=,\*\*=
* Bitwise Operators:&,|,<<,>>,^
* Membership Operators:in,not in
* Identity Operators:is,is not

# Control Flow:

* Conditional statements: if, elif, else
* Looping statements: for, while
* Control flow statements: break, continue, pass

## 

## Functions:

* It is a collection of statements that performs a specific task.
* It executes when it is called by its name.
* Syntax of function:

def fun\_name(parameter list):

function\_body

* Parameter List:

It is the place where we can pass a number of parameters/variables.

* Body:

It is the place where the actual code is written to perform the specific task

## Modules:

* Modules in Python are simply Python files with a .py extension. They can contain functions, classes, and variables. You can import modules using the import statement.

# Packages:

* Packages are a way of organising modules into a hierarchical structure. A package is essentially a directory containing Python modules and a special \_init\_.py file. You can import packages using dot notation.

Task 2:NumPy For Efficient

# 1) IDE :- Integrated Development Environment(google colab)

# 2) Github Repository creation(folder name:-DAP\_Repository)

# 3) NUMPY:- Numpy is used for numerical or scientific computing in arrays,vectors and matrices

# 4) To import a numpy by import numpy as np

Numpy Functions:

## Array Creation:To create an array using NumPy

## EX:import numpy as np

## #creating arrays

## arr = np.array([1,2,3,4,5])

## print(arr)

## Output:

## [1,2,3,4,5]

## 2.ones() Method :- Print/Display the array with Ones(1’s) by using ones() method

## Syntax:a = np.ones((r,c),dtype= data type)

## > r = number of rows

## > c = number of columns

## >dtype means data type (int,float etc….)

## EX:

# import numpy as np

# ones\_arr=np.ones((3,4),dtype=int)

# print(ones\_arr)

# 

## Output:

# [[1 1 1 1]

[1 1 1 1]

[1 1 1 1]]

## 3. zeros() Method :- Same like we can print the array with Zeros(0’s) by using zeros() method

## Syntax: a = np.zeros((r,c),dtype= data type)

## > r = number of rows

## > c = number of columns

## > dtype means data type (int,float etc….)

## >If datatype doesn’t consider the dtype(data type) by default it can prints in float

# Ex:import numpy as np

# zeros\_arr = np.zeros((3,2),dtype=float)

# print(zeros\_arr)

# Output:

[[0. 0.]

[0. 0.]

[0. 0.]]

## 4.arange() Method:- It is used to to arrange the number from specific number to specific number

## Syntax: a=np.arange(x,y,z,dtype= data type)

## x = Starting number

## y = Ending number

## z = Step number -> dtype means data type (int,float etc….)

Ex:import numpy as np

ar\_arange = np.arange(10)

print(ar\_arange)

Output:

[0 1 2 3 4 5 6 7 8 9]

## 5. reshape() Method:- This method is used to change the shape of the array

## Syntax:b=a.reshape(r,c)

## a = created array

## r = number of rows

## c = number of columns

## EX:import numpy as np

## arr=np.array([1,2,3])

## reshaped\_arr = arr.reshape(3,1)

## print(reshaped\_arr)

## Output:

[[1]

[2]

[3]]

6. T (Transpose):- Returns Transpose of the matrix

Syntax: b=a.T

* a = created array

Ex:

import numpy as np

a=np.array([[1,2,3],[4,5,6]])

b=a.T

print(b)

Output:

[[1 4]

[2 5]

[3 6]]

## 7. split() Method:- Split method is used split the array in number of times

## Syntax:- b=np.split(a,n)

## a = created array

## n = number of divisions/split

## Ex:import numpy as np

## a=np.array([1,2,3,4,5,6,7,8])

## b=np.split(a,4)

## print(b)

## Output:

[array([1, 2]), array([3, 4]), array([5, 6]), array([7, 8])]

## 

## 12).dot() Method:- This method returns the matrix multiplication of given matrices

## Ex:- c=np.dot(a,b)

## -> a=first matrix stored variable

## -> b=second matrix stored variable

## 13)linalg.eig() Method (Linear algebra eigenvalues):- Compute the eigenvalues and right eigenvectors of a square array.

## Ex:- b=np.linalg.eig(a)

## -> a = created array

## 14) random.rand():-This method is used to print random number

## Ex:- import numpy as np a=np.random.rand()

## 15) random.randint() Method:- It Returns the random value between the values

## Ex:- import numpy as np a=np.random.randint(2,10)

## 16) type() Method:- Returns the which type of the variable

## Ex:- import numpy as np a=np.array([1,2,3,4]) print(type(a))

## 17) ndim(Number of dimensions):- To give the dimension of array.

## Ex:- import numpy as np a=np.array([1,2,3,4]) print(a.ndim)

## 18)shape :- Returns the shape of the array.

## import numpy as np a=np.array([1,2,3,4]) print(a.shape)

## 19) linspace()(Line space) Method:- Create array filled evenly spaced values.

## Ex:- import numpy as np

## a=np.linspace(0.8,2,5)

## print(a)

## 20) If we can use ‘\*’ to multiply the elements in matrices.

## Ex:- import numpy as np

## a=np.array([1,2,3,4])

## b=np.array([5,6,7,8]) c=a\*b

## 21) If we can use ‘@ , .dot()’ to perform the matrix multiplication.

## Ex:- import numpy as np

## a=np.array([1,2,3,4])

## b=np.array([5,6,7,8])

## c=a@b

## 22) sum():- To perform the addition between the 2 arrays.

## Ex:- import numpy as np

## a=np.ones((3,3),dtype=int)

## print(a.sum())

## 23) axis:- 1) If axis=1 refers to the row

## 2) If axis=0 refers to the column

## Ex:- import numpy as np

## a=np.ones((3,3),dtype=int)

## print(a.sum(axis=1)) //row

## print(a.sum(axis=0)) //column

## 24) floor() method :- Returns the value of the ignores the decimal values

## 25) ceil() method :- Returns the value of the round the decimal value and returns

## 26) max() Method:- Returns the maximum value of the array

## 27) min() Method:- Returns the minimum value of the array

## 28) cumsum() Method:- Returns the cumulative sum of array/matrix

## 29) resize() Method:- we can modify the matrix /array size.

## Ex:- import numpy as np

## a=np.array([1,2,3,4,5,6])

## p=a.resize(2,3)

## 30) vstack():-This method is used to join the two maurices in vertical order

## Ex:- import numpy as np

## 

## a=np.array([1,2,3,4,5,6])

## b=np.array([7,8,9,10,11,12]) a.resize(2,3) a.resize(2,3)

## p=np.vstack(a,b)

## 31) hstack():-This method is used to join the two maurices in horizontal order

## Ex:- import numpy as np

## a=np.array([1,2,3,4,5,6])

## b=np.array([7,8,9,10,11,12]) a.resize(2,3) a.resize(2,3) p=np.hstack(a,b)

## 32) dstack() Method:- number of rows become number of groups number of columns become rows group become column

## Ex:- import numpy as np

## a=np.array([1,2,3,4,5,6])

## b=np.array([7,8,9,10,11,12]) a.resize(2,3) a.resize(2,3) p=np.dstack(a,b)

## 

## DAY2:-(24-02-2024)

## 1) eye() Method:- This method is used to print the diagonal values with “1”

## Ex:- import numpy as np

## a=np.eye(4)

## print(a)

## 2) full() Method:- This method is used to fill the given matrix with given number

## Ex:- import numpy as np

## a=np.full([2,2],2)

## print(a)

## 3) asarray() Method:- This is a symmetric array of the array. It is one of the method to create the array

## Ex:- import numpy as np

## x=[1,2,3,4,5] a=np.asarray(x)

## print(a)

## 4) inner() method :- It is the sum of the element wise multiplication in an array.

## 5) outer() method :- It is the 1st array element is multiply with the 2nd array of all elements.

## 6) cross() Method :- cross multiplication of 1st array and 2nd array

## 7) rint() Method:- This method is returns the round the elements in array to nearest integer

## 8)true\_divide() method :- This method is used for the 1st element array with 2nd array elements wise.

## Ex:- import numpy as np

## a=np.array([1,2,3,4,5])

## b=np.array([2,3,4,5,6])

## c=np.true\_divide(a,b)

## print(c)

## 9) unique() :- this method removes all duplicate values and prints.

## Ex:- import numpy as np

## a=np.array([1,1,2,2,3,3,4,4,5,5])

## b=np.unique(a)

## print(b)

## 10) union1d() Method :- It returns the to join the all array without the duplicate values and prints them in one array.

## Ex:- import numpy as np

## a=np.array([[1,2,3,4,5],[8,4,9,8,1]]) b=np.array([[2,3,4,5,6],[4,6,8,9,3]])

## c=np.union1d(a,b)

## print(c)

## 11) intersect1d() Method :- It returns the to common elements in two arrays

## Ex:- import numpy as np

## a=np.array([[1,2,3,4,5],[8,4,9,8,1]]) b=np.array([[2,3,4,5,6],[4,6,8,9,3]])

## c=np.intersect1d(a,b)

## print(c)

## 12) setdiff1d() Method :- It returns the first array elements without which element is common in both arrays.

## Ex:- import numpy as np

## a=np.array([[1,2,3,4,5],[8,4,9,8,1]]) b=np.array([[2,3,4,5,6],[4,6,8,9,3]])

## c=np.setdiff1d(a,b)

## print(c)

## 13)hypot() Method:- This method is works on hypotenuse formula

## Ex:- import numpy as np

## a=8 b=6 c=np.hypot(a,b)

## print(c)

## 14) sin() Method :- It can return the sin of values.

## import numpy as np x=90 c=np.sin(x) print(c)

## 15) divmod() Method :- It can print the two arrays. The 1st array is division of both arrays in element wise, 2nd array is modular division of both arrays by element wise.

## Ex:- import numpy as np

## a=np.array([1,2,3,4,5])

## b=np.array([2,3,4,5,6])

## c=np.divmod(a,b)

## print(c) 16) mod() Method :- It can print the modular division (remainder) of both arrays in element wise.

## Ex:- import numpy as np a=np.array([1,2,3,4,5])

## b=np.array([2,3,4,5,6])

## c=np.mod(a,b)

## print(c)

## 17) div() Method :- It can print the exact division of both arrays in element wise

## Ex:- import numpy as np

## a=np.array([1,2,3,4,5])

## b=np.array([2,3,4,5,6])

## c=np.div(a,b)

## print(c)

## 18) multiply() Method:- This method is used to multiply the elements in arrays in element wise.

## Ex:- import numpy as np

## a=np.array([1,2,3,4,5])

## b=np.array([2,3,4,5,6]) c=np.mul(a,b)

## print(c)

## 19) random.normal() Method:- This method returns random values in given size of the matrix.

## Ex:- from numpy import random

## a=random.normal(size(2,3))

## print(a)

## a=random.normal(loc=1,scale=2,size(2,3))

## ->loc=mean

## ->scale=standard deviation

## 20) random.binomial() Method:-

## Ex:- from numpy import random a=random.binomial(n=12,p=0.5,size=10) print(a)

## 21) random.poisson() Method:-

## Ex:- from numpy import random a=random.binomial(lam=2,size=10) print(a) -> lam=lamida 22)random.choice() Method:- This method is used to print the random value in given range.

## Ex:- from numpy import random a=random.choice([1,2,3,4,5,6,7,8,9,10])

* print(a)

Pandas:

## Pandas is a data manipulation package in Python for tabular data.That is, data in the form of rows and columns also known

## as DataFrames

## Pandas-Library used for working with datasets.It has functions for analysing,cleaning,exploring,and manipulating data.

## Pandas has a reference to both “Panel Data” and“Python Data Analysis”.

# Applications of Pandas:

# Data cleaning and preparation

# Data analysis

# Data visualisation

# Machine learning

Series() #gives index values to series

Ex:-

import pandas as pd

a=["jwalitha",'ramya','durga','jahnavi','lahari','sunny','Dhanush']

r=pd.Series(a,index=[67,43,44,89,34,45,23])

print(r)

OUTPUT:

67 jwalitha

43 ramya

44 durga

89 jahnavi

34 lahari

45 sunny

23 Dhanush

dtype: object

read() #from csv file

Ex:-

df=pd.read\_csv("//content/diabetcsv.csv")

print(df)

OUTPUT:OUTPUT:

preg plas pres skin insu mass pedi age class

0 6 148 72 35 0 33.6 0.627 50 tested\_positive

1 1 85 66 29 0 26.6 0.351 31 tested\_negative

2 8 183 64 0 0 23.3 0.672 32 tested\_positive

3 1 89 66 23 94 28.1 0.167 21 tested\_negative

4 0 137 40 35 168 43.1 2.288 33 tested\_positive

.. ... ... ... ... ... ... ... ... ...

763 10 101 76 48 180 32.9 0.171 63 tested\_negative

764 2 122 70 27 0 36.8 0.340 27 tested\_negative

765 5 121 72 23 112 26.2 0.245 30 tested\_negative

766 1 126 60 0 0 30.1 0.349 47 tested\_positive

767 1 93 70 31 0 30.4 0.315 23 tested\_negative

[768 rows x 9 columns]

Ex:-

df=pd.read\_csv("/content/grades\_withnulls.csv")

print(df)

OUTPUT:

Names Initials SEM1 SEM2 SEM3 Grade Placed

0 Joe K 9.8 10.0 9.9 A+ 1

1 Rajesh M 8.9 9.1 9.3 A 1

2 Kissan V 9.9 9.8 10.0 A 0

3 Mary N 7.7 8.0 NaN B 0

4 Jeen K 9.8 9.1 9.9 A+ 1

5 Raj M 8.9 9.1 9.3 A 1

6 Hassan V 9.9 9.0 9.2 A 1

7 Mari N 7.7 8.0 7.1 B 1

8 Jess K NaN 9.1 9.9 A+ 1

9 Rajini M NaN 9.1 9.3 A 0

10 Kiran V NaN 9.3 9.2 A 0

11 Maya N 7.7 8.0 7.1 B 0

12 Jolin K 9.8 9.1 9.9 A+ 1

13 Rajesh M 8.9 9.1 9.3 A 1

14 Riya M 9.3 9.9 10.0 A 1

15 Sana V 9.9 9.3 9.2 A 0

16 Mark N 7.7 8.0 7.0 B 0

df=pd.read\_csv("/content/drive/MyDrive/Dataset/diabetcsv.csv",sep=" ")

print(df)

output:

preg,plas,pres,skin,insu,mass,pedi,age,class

0 6,148,72,35,0,33.6,0.627,50,tested\_positive

1 1,85,66,29,0,26.6,0.351,31,tested\_negative

2 8,183,64,0,0,23.3,0.672,32,tested\_positive

3 1,89,66,23,94,28.1,0.167,21,tested\_negative

4 0,137,40,35,168,43.1,2.288,33,tested\_positive

.. ...

763 10,101,76,48,180,32.9,0.171,63,tested\_negative

764 2,122,70,27,0,36.8,0.34,27,tested\_negative

765 5,121,72,23,112,26.2,0.245,30,tested\_negative

766 1,126,60,0,0,30.1,0.349,47,tested\_positive

767 1,93,70,31,0,30.4,0.315,23,tested\_negative

[768 rows x 1 columns]

read() #from excel file

Ex:-

df=pd.read\_excel("/content/diabetcsv.xls")

print(df)

OUTPUT:

preg plas pres skin insu mass pedi age class

0 6 148 72 35 0 33.6 0.627 50 tested\_positive

1 1 85 66 29 0 26.6 0.351 31 tested\_negative

2 8 183 64 0 0 23.3 0.672 32 tested\_positive

3 1 89 66 23 94 28.1 0.167 21 tested\_negative

4 0 137 40 35 168 43.1 2.288 33 tested\_positive

.. ... ... ... ... ... ... ... ... ...

763 10 101 76 48 180 32.9 0.171 63 tested\_negative

764 2 122 70 27 0 36.8 0.340 27 tested\_negative

765 5 121 72 23 112 26.2 0.245 30 tested\_negative

766 1 126 60 0 0 30.1 0.349 47 tested\_positive

767 1 93 70 31 0 30.4 0.315 23 tested\_negative

[768 rows x 9 columns]

Ex:-

df=pd.read\_excel("/content/diabetcsv.xls",sheet\_name=1)

print(df)

OUTPUT:

543 30 0.484 32 tested\_positive

0 543 45.8 0.551 31 tested\_positive

1 543 29.6 0.254 31 tested\_positive

2 543 43.3 0.183 33 tested\_negative

3 543 34.6 0.529 32 tested\_positive

4 543 39.3 0.704 27 tested\_negative

5 543 35.4 0.388 50 tested\_negative

6 543 39.8 0.451 41 tested\_positive

7 543 29.0 0.263 29 tested\_positive

8 543 36.6 0.254 51 tested\_positive

9 543 31.1 0.205 41 tested\_positive

10 543 39.4 0.257 43 tested\_positive

11 543 23.2 0.487 22 tested\_negative

12 543 22.2 0.245 57 tested\_negative

head() #prints top specified lines

df=pd.read\_csv("/content/drive/MyDrive/Dataset/diabetcsv.csv")

print(df.head())

tail() #prints last specified lines

print(df.tail(n=10))

#output:

**Names Initials SEM1 SEM2 SEM3 Grade Placed**

0 Joe K 9.8 10.0 9.9 A+ 1

1 Rajesh M 8.9 9.1 9.3 A 1

2 Kissan V 9.9 9.8 10.0 A 0

3 Mary N 7.7 8.0 NaN B 0

4 Jeen K 9.8 9.1 9.9 A+ 1

**Names Initials SEM1 SEM2 SEM3 Grade Placed**

7 Mari N 7.7 8.0 7.1 B 1

8 Jess K NaN 9.1 9.9 A+ 1

9 Rajini M NaN 9.1 9.3 A 0

10 Kiran V NaN 9.3 9.2 A 0

11 Maya N 7.7 8.0 7.1 B 0

12 Jolin K 9.8 9.1 9.9 A+ 1

13 Rajesh M 8.9 9.1 9.3 A 1

14 Riya M 9.3 9.9 10.0 A 1

15 Sana V 9.9 9.3 9.2 A 0

16 Mark N 7.7 8.0 7.0 B 0

Describe #gives count of values,mean,STD

df=pd.read\_csv("/content/grades\_withnulls.csv")

print(df.describe)

OUTPUT:

bound method NDFrame.describe of Names Initials SEM1 SEM2 SEM3 Grade Placed

0 Joe K 9.8 10.0 9.9 A+ 1

1 Rajesh M 8.9 9.1 9.3 A 1

2 Kissan V 9.9 9.8 10.0 A 0

3 Mary N 7.7 8.0 NaN B 0

4 Jeen K 9.8 9.1 9.9 A+ 1

5 Raj M 8.9 9.1 9.3 A 1

6 Hassan V 9.9 9.0 9.2 A 1

7 Mari N 7.7 8.0 7.1 B 1

8 Jess K NaN 9.1 9.9 A+ 1

9 Rajini M NaN 9.1 9.3 A 0

10 Kiran V NaN 9.3 9.2 A 0

11 Maya N 7.7 8.0 7.1 B 0

12 Jolin K 9.8 9.1 9.9 A+ 1

13 Rajesh M 8.9 9.1 9.3 A 1

14 Riya M 9.3 9.9 10.0 A 1

15 Sana V 9.9 9.3 9.2 A 0

16 Mark N 7.7 8.0 7.0 B 0>

describe().T

df=pd.read\_csv("/content/drive/MyDrive/Dataset/grades\_withnulls.csv")

print(df.describe().T)

#**output:**

**count mean std min 25% 50% 75% max**

SEM1 14.0 8.992857 0.929404 7.7 8.0 9.1 9.8 9.9

SEM2 17.0 9.000000 0.647109 8.0 9.0 9.1 9.3 10.0

SEM3 16.0 9.100000 1.057670 7.0 9.2 9.3 9.9 10.0

Placed 17.0 0.588235 0.507300 0.0 0.0 1.0 1.0 1.0

shape

df=pd.read\_csv("/content/drive/MyDrive/Dataset/grades\_withnulls.csv")

# get the number of rows and columns

print(df.shape) #output:(17,7)

# get the number of rows

print(df.shape[0]) #output:17

# get the number of column

print(df.shape[1]) #output:7

# creates list of column names

df=pd.read\_csv("/content/drive/MyDrive/Dataset/grades\_withnulls.csv")

print(df.columns)

#output:Index(['Names', 'Initials', 'SEM1', 'SEM2', 'SEM3', 'Grade', 'Placed'], dtype='object')

#making null

df=pd.read\_csv("/content/drive/MyDrive/Dataset/grades\_withnulls.csv")

df2=df.copy()

df2.loc[2:5,'SEM1'] = None

d2.head(7)

#output:

|  | **name** | **initial** | **SEM1** | **SEM2** | **SEM3** | **Grade** | **placed** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | Joe | K | 9.8 | 10.0 | 9.9 | A+ | 1 |
| 1 | Rajesh | M | 8.9 | 9.1 | 9.3 | A | 1 |
| 2 | Kissan | V | NaN | 9.8 | 10.0 | A | 0 |
| 3 | Mary | N | NaN | 8.0 | NaN | B | 0 |
| 4 | Jeen | K | NaN | 9.1 | 9.9 | A+ | 1 |
| 5 | Raj | M | NaN | 9.1 | 9.3 | A | 1 |
| 6 | Hassa | V | 9.9 | 9.0 | 9.2 | A | 1 |

df2.isnull().head(7) #gives true if there exists null

|  | **Name** | **initials** | **SEM1** | **SEM2** | **SEM3** | **Grade** | **Placed** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | False | False | False | False | False | False | False |
| **1** | False | False | False | False | False | False | False |
| **2** | False | False | True | False | False | False | False |
| **3** | False | False | True | False | True | False | False |
| **4** | False | False | True | False | False | False | False |
| **5** | False | False | True | False | False | False | False |
| **6** | False | False | False | False | False | False | False |

df2.isnull().tail(5)

#**output:**

|  | **Names** | **Initials** | **SEM1** | **SEM2** | **Sem3** | **Grade** | **Placed** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **12** | False | False | False | False | False | False | False |
| **13** | False | False | False | False | False | False | False |
| **14** | False | False | False | False | False | False | False |
| **15** | False | False | False | False | False | False | False |
| **16** | False | False | False | False | False | False | False |

isnull().sum() #gives the count of null values in each column

print(df2.isnull().sum())

**#output:**

Names 0

Initials 0

SEM1 7

SEM2 0

SEM3 1

Grade 0

Placed 0

dtype: int64

print(df2.isnull().sum().sum()) #output:8

print(df2[['SEM1','Names']]) #extracting column

**#output:**

SEM1 Names

0 9.8 Joe

1 8.9 Rajesh

2 NaN Kissan

3 NaN Mary

4 NaN Jeen

5 NaN Raj

6 9.9 Hassan

7 7.7 Mari

8 NaN Jess

9 NaN Rajini

10 NaN Kiran

11 7.7 Maya

12 9.8 Jolin

13 8.9 Rajesh

14 9.3 Riya

15 9.9 Sana

16 7.7 Mark

print(df[df.index==1]) #prints particular row

#output:

Names Initials SEM1 SEM2 SEM3 Grade Placed

1 Rajesh M 8.9 9.1 9.3 A 1

print(df[df.index.isin(range(2,4))]) #prints particular rows in given range

Names Initials SEM1 SEM2 SEM3 Grade Placed

2 Kissan V 9.9 9.8 10.0 A 0

3 Mary N 7.7 8.0 NaN B 0

print(df2.loc[2]) # prints one row as series

#output:

Names Kissan

Initials V

SEM1 NaN

SEM2 9.8

SEM3 10.0

Grade A

Placed 0

df2.loc[6:15,'SEM1'] #prints particular column in given range

6 NaN

7 NaN

8 NaN

9 NaN

10 NaN

11 NaN

12 9.8

13 8.9

14 9.3

15 9.9

df2.loc[[1,3]] #prints the rows which are specified

#output:

|  | **Names** | **Initials** | **SEM1** | **SEM2** | **SEM3** | **Grade** | **Placed** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | Rajesh | M | 8.9 | 9.1 | 9.3 | A | 1 |
| **3** | Mary | N | NaN | 8.0 | NaN | B | 0 |

df2.loc[10:15,['Names','Grade']] #prints the columns with in given range of rows

#output:

|  | **Names** | **Grade** |
| --- | --- | --- |
| **10** | Kiran | A |
| **11** | Maya | B |
| **12** | Jolin | A+ |
| **13** | Rajesh | A |
| **14** | Riya | A |
| **15** | Sana | A |

df2.iloc[10:14,:3] #row,column

#output:

|  | **Names** | **Initials** | **SEM1** |
| --- | --- | --- | --- |
| **10** | Kiran | V | NaN |
| **11** | Maya | N | NaN |
| **12** | Jolin | K | 9.8 |
| **13** | Rajesh | M | 8.9 |

df2.loc[:5,["Names"]]

#output:

|  | **Names** |
| --- | --- |
| **0** | Joe |
| **1** | Rajesh |
| **2** | Kissan |
| **3** | Mary |
| **4** | Jeen |
| **5** | Raj |

df2.loc[df['Names']=='Rajesh'] #particular details

#output:

|  | **Names** | **Initials** | **SEM1** | **SEM2** | **SEM3** | **Grade** | **Placed** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | Rajesh | M | 8.9 | 9.1 | 9.3 | A | 1 |
| **13** | Rajesh | M | 8.9 | 9.1 | 9.3 | A | 1 |

df[df.Grade=='A']

df[df.SEM2==10]

#output:

| **Names** | **Initials** | **SEM1** | **SEM2** | **SEM3** | **Grade** | **Placed** |
| --- | --- | --- | --- | --- | --- | --- |
| joe | k | 9.8 | 10.0 | 9.9 | A+ | 1 |
|  |  |  |  |  |  |  |

#PRINTS particular column which satisfies the condition

df.loc[df['SEM1'] > 9.8,['Names']]

#output:

|  | **Names** |
| --- | --- |
| **2** | Kissan |
| **6** | Hassan |
| **15** | Sana |

df3=df.copy()

df3

#drops columns which consists nan

df3.dropna(inplace=True,axis=1) # inplace says not to rechange in df

df3

#output:

|  | **Names** | **Initials** | **SEM2** | **Grade** | **placed** |
| --- | --- | --- | --- | --- | --- |
| **0** | Joe | K | 10.0 | A+ | 1 |
| **1** | Rajesh | M | 9.1 | A | 1 |
| **2** | Kissan | V | 9.8 | A | 0 |
| **3** | Mary | N | 8.0 | B | 0 |
| **4** | Jeen | K | 9.1 | A+ | 1 |
| **5** | Raj | M | 9.1 | A | 1 |
| **6** | Hassan | V | 9.0 | A | 1 |
| **7** | Mari | N | 8.0 | B | 1 |
| **8** | Jess | K | 9.1 | A+ | 1 |
| **9** | Rajini | M | 9.1 | A | 0 |
| **10** | Kiran | V | 9.3 | A | 0 |
| **11** | Maya | N | 8.0 | B | 0 |
| **12** | Jolin | K | 9.1 | A+ | 1 |
| **13** | Rajesh | M | 9.1 | A | 1 |
| **14** | Riya | M | 9.9 | A | 1 |
| **15** | Sana | V | 9.3 | A | 0 |
| **16** | Mark | N | 8.0 | B | 1 |

#drops both rows and columns which consists nan

df3.dropna(inplace=True,how='all') #inplace says not to rechange in df

df3

# Inplace of nan mean value is replaced

mv=df['SEM1'].mean()

df=df.fillna(mv)

print(df)

#output:

**Names Initials SEM1 SEM2 SEM3 Grade Placed**

0 Joe K 9.800000 10.0 9.900000 A+ 1

1 Rajesh M 8.900000 9.1 9.300000 A 1

2 Kissan V 9.900000 9.8 10.000000 A 0

3 Mary N 7.700000 8.0 8.992857 B 0

4 Jeen K 9.800000 9.1 9.900000 A+ 1

5 Raj M 8.900000 9.1 9.300000 A 1

6 Hassan V 9.900000 9.0 9.200000 A 1

7 Mari N 7.700000 8.0 7.100000 B 1

8 Jess K 8.992857 9.1 9.900000 A+ 1

9 Rajini M 8.992857 9.1 9.300000 A 0

10 Kiran V 8.992857 9.3 9.20000

preg plas pres skin insu mass pedi age class

0 6 148 72 35 0 33.6 0.627 50 tested\_positive

1 1 85 66 29 0 26.6 0.351 31 tested\_negative

2 8 183 64 0 0 23.3 0.672 32 tested\_positive

3 1 89 66 23 94 28.1 0.167 21 tested\_negative

4 0 137 40 35 168 43.1 2.288 33 tested\_positive

.. ... ... ... ... ... ... ... ... ...

763 10 101 76 48 180 32.9 0.171 63 tested\_negative

764 2 122 70 27 0 36.8 0.340 27 tested\_negative

765 5 121 72 23 112 26.2 0.245 30 tested\_negative

766 1 126 60 0 0 30.1 0.349 47 tested\_positive

767 1 93 70 31 0 30.4 0.315 23 tested\_negative

[768 rows x 9 columns]

Ex:-

df=pd.read\_csv("/content/grades\_withnulls.csv")

print(df)

OUTPUT:

Names Initials SEM1 SEM2 SEM3 Grade Placed

0 Joe K 9.8 10.0 9.9 A+ 1

1 Rajesh M 8.9 9.1 9.3 A 1

2 Kissan V 9.9 9.8 10.0 A 0

3 Mary N 7.7 8.0 NaN B 0

4 Jeen K 9.8 9.1 9.9 A+ 1

5 Raj M 8.9 9.1 9.3 A 1

6 Hassan V 9.9 9.0 9.2 A 1

7 Mari N 7.7 8.0 7.1 B 1

8 Jess K NaN 9.1 9.9 A+ 1

9 Rajini M NaN 9.1 9.3 A 0

10 Kiran V NaN 9.3 9.2 A 0

11 Maya N 7.7 8.0 7.1 B 0

12 Jolin K 9.8 9.1 9.9 A+ 1

13 Rajesh M 8.9 9.1 9.3 A 1

14 Riya M 9.3 9.9 10.0 A 1

15 Sana V 9.9 9.3 9.2 A 0

16 Mark N 7.7 8.0 7.0 B 0

df=pd.read\_csv("/content/drive/MyDrive/Dataset/diabetcsv.csv",sep=" ")

print(df)

output:

preg,plas,pres,skin,insu,mass,pedi,age,class

0 6,148,72,35,0,33.6,0.627,50,tested\_positive

1 1,85,66,29,0,26.6,0.351,31,tested\_negative

2 8,183,64,0,0,23.3,0.672,32,tested\_positive

3 1,89,66,23,94,28.1,0.167,21,tested\_negative

4 0,137,40,35,168,43.1,2.288,33,tested\_positive

.. ...

763 10,101,76,48,180,32.9,0.171,63,tested\_negative

764 2,122,70,27,0,36.8,0.34,27,tested\_negative

765 5,121,72,23,112,26.2,0.245,30,tested\_negative

766 1,126,60,0,0,30.1,0.349,47,tested\_positive

767 1,93,70,31,0,30.4,0.315,23,tested\_negative

[768 rows x 1 columns]

read() #from excel file

Ex:-

df=pd.read\_excel("/content/diabetcsv.xls")

print(df)

OUTPUT:

preg plas pres skin insu mass pedi age class

0 6 148 72 35 0 33.6 0.627 50 tested\_positive

1 1 85 66 29 0 26.6 0.351 31 tested\_negative

2 8 183 64 0 0 23.3 0.672 32 tested\_positive

3 1 89 66 23 94 28.1 0.167 21 tested\_negative

4 0 137 40 35 168 43.1 2.288 33 tested\_positive

.. ... ... ... ... ... ... ... ... ...

763 10 101 76 48 180 32.9 0.171 63 tested\_negative

764 2 122 70 27 0 36.8 0.340 27 tested\_negative

765 5 121 72 23 112 26.2 0.245 30 tested\_negative

766 1 126 60 0 0 30.1 0.349 47 tested\_positive

767 1 93 70 31 0 30.4 0.315 23 tested\_negative

[768 rows x 9 columns]

Ex:-

df=pd.read\_excel("/content/diabetcsv.xls",sheet\_name=1)

print(df)

OUTPUT:

543 30 0.484 32 tested\_positive

0 543 45.8 0.551 31 tested\_positive

1 543 29.6 0.254 31 tested\_positive

2 543 43.3 0.183 33 tested\_negative

3 543 34.6 0.529 32 tested\_positive

4 543 39.3 0.704 27 tested\_negative

5 543 35.4 0.388 50 tested\_negative

6 543 39.8 0.451 41 tested\_positive

7 543 29.0 0.263 29 tested\_positive

8 543 36.6 0.254 51 tested\_positive

9 543 31.1 0.205 41 tested\_positive

10 543 39.4 0.257 43 tested\_positive

11 543 23.2 0.487 22 tested\_negative

12 543 22.2 0.245 57 tested\_negative

head() #prints top specified lines

df=pd.read\_csv("/content/drive/MyDrive/Dataset/diabetcsv.csv")

print(df.head())

tail() #prints last specified lines

print(df.tail(n=10))

#output:

**Names Initials SEM1 SEM2 SEM3 Grade Placed**

0 Joe K 9.8 10.0 9.9 A+ 1

1 Rajesh M 8.9 9.1 9.3 A 1

2 Kissan V 9.9 9.8 10.0 A 0

3 Mary N 7.7 8.0 NaN B 0

4 Jeen K 9.8 9.1 9.9 A+ 1

**Names Initials SEM1 SEM2 SEM3 Grade Placed**

7 Mari N 7.7 8.0 7.1 B 1

8 Jess K NaN 9.1 9.9 A+ 1

9 Rajini M NaN 9.1 9.3 A 0

10 Kiran V NaN 9.3 9.2 A 0

11 Maya N 7.7 8.0 7.1 B 0

12 Jolin K 9.8 9.1 9.9 A+ 1

13 Rajesh M 8.9 9.1 9.3 A 1

14 Riya M 9.3 9.9 10.0 A 1

15 Sana V 9.9 9.3 9.2 A 0

16 Mark N 7.7 8.0 7.0 B 0

Describe #gives count of values,mean,STD

df=pd.read\_csv("/content/grades\_withnulls.csv")

print(df.describe)

OUTPUT:

bound method NDFrame.describe of Names Initials SEM1 SEM2 SEM3 Grade Placed

0 Joe K 9.8 10.0 9.9 A+ 1

1 Rajesh M 8.9 9.1 9.3 A 1

2 Kissan V 9.9 9.8 10.0 A 0

3 Mary N 7.7 8.0 NaN B 0

4 Jeen K 9.8 9.1 9.9 A+ 1

5 Raj M 8.9 9.1 9.3 A 1

6 Hassan V 9.9 9.0 9.2 A 1

7 Mari N 7.7 8.0 7.1 B 1

8 Jess K NaN 9.1 9.9 A+ 1

9 Rajini M NaN 9.1 9.3 A 0

10 Kiran V NaN 9.3 9.2 A 0

11 Maya N 7.7 8.0 7.1 B 0

12 Jolin K 9.8 9.1 9.9 A+ 1

13 Rajesh M 8.9 9.1 9.3 A 1

14 Riya M 9.3 9.9 10.0 A 1

15 Sana V 9.9 9.3 9.2 A 0

16 Mark N 7.7 8.0 7.0 B 0>

describe().T

df=pd.read\_csv("/content/drive/MyDrive/Dataset/grades\_withnulls.csv")

print(df.describe().T)

#**output:**

**count mean std min 25% 50% 75% max**

SEM1 14.0 8.992857 0.929404 7.7 8.0 9.1 9.8 9.9

SEM2 17.0 9.000000 0.647109 8.0 9.0 9.1 9.3 10.0

SEM3 16.0 9.100000 1.057670 7.0 9.2 9.3 9.9 10.0

Placed 17.0 0.588235 0.507300 0.0 0.0 1.0 1.0 1.0

shape

df=pd.read\_csv("/content/drive/MyDrive/Dataset/grades\_withnulls.csv")

# get the number of rows and columns

print(df.shape) #output:(17,7)

# get the number of rows

print(df.shape[0]) #output:17

# get the number of column

print(df.shape[1]) #output:7

# creates list of column names

df=pd.read\_csv("/content/drive/MyDrive/Dataset/grades\_withnulls.csv")

print(df.columns)

#output:Index(['Names', 'Initials', 'SEM1', 'SEM2', 'SEM3', 'Grade', 'Placed'], dtype='object')

#making null

df=pd.read\_csv("/content/drive/MyDrive/Dataset/grades\_withnulls.csv")

df2=df.copy()

df2.loc[2:5,'SEM1'] = None

d2.head(7)

#output:

|  | **name** | **initial** | **SEM1** | **SEM2** | **SEM3** | **Grade** | **placed** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | Joe | K | 9.8 | 10.0 | 9.9 | A+ | 1 |
| 1 | Rajesh | M | 8.9 | 9.1 | 9.3 | A | 1 |
| 2 | Kissan | V | NaN | 9.8 | 10.0 | A | 0 |
| 3 | Mary | N | NaN | 8.0 | NaN | B | 0 |
| 4 | Jeen | K | NaN | 9.1 | 9.9 | A+ | 1 |
| 5 | Raj | M | NaN | 9.1 | 9.3 | A | 1 |
| 6 | Hassa | V | 9.9 | 9.0 | 9.2 | A | 1 |

df2.isnull().head(7) #gives true if there exists null

|  | **Name** | **initials** | **SEM1** | **SEM2** | **SEM3** | **Grade** | **Placed** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | False | False | False | False | False | False | False |
| **1** | False | False | False | False | False | False | False |
| **2** | False | False | True | False | False | False | False |
| **3** | False | False | True | False | True | False | False |
| **4** | False | False | True | False | False | False | False |
| **5** | False | False | True | False | False | False | False |
| **6** | False | False | False | False | False | False | False |

df2.isnull().tail(5)

#**output:**

|  | **Names** | **Initials** | **SEM1** | **SEM2** | **Sem3** | **Grade** | **Placed** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **12** | False | False | False | False | False | False | False |
| **13** | False | False | False | False | False | False | False |
| **14** | False | False | False | False | False | False | False |
| **15** | False | False | False | False | False | False | False |
| **16** | False | False | False | False | False | False | False |

isnull().sum() #gives the count of null values in each column

print(df2.isnull().sum())

**#output:**

Names 0

Initials 0

SEM1 7

SEM2 0

SEM3 1

Grade 0

Placed 0

dtype: int64

print(df2.isnull().sum().sum()) #output:8

print(df2[['SEM1','Names']]) #extracting column

**#output:**

SEM1 Names

0 9.8 Joe

1 8.9 Rajesh

2 NaN Kissan

3 NaN Mary

4 NaN Jeen

5 NaN Raj

6 9.9 Hassan

7 7.7 Mari

8 NaN Jess

9 NaN Rajini

10 NaN Kiran

11 7.7 Maya

12 9.8 Jolin

13 8.9 Rajesh

14 9.3 Riya

15 9.9 Sana

16 7.7 Mark

print(df[df.index==1]) #prints particular row

#output:

Names Initials SEM1 SEM2 SEM3 Grade Placed

1 Rajesh M 8.9 9.1 9.3 A 1

print(df[df.index.isin(range(2,4))]) #prints particular rows in given range

Names Initials SEM1 SEM2 SEM3 Grade Placed

2 Kissan V 9.9 9.8 10.0 A 0

3 Mary N 7.7 8.0 NaN B 0

print(df2.loc[2]) # prints one row as series

#output:

Names Kissan

Initials V

SEM1 NaN

SEM2 9.8

SEM3 10.0

Grade A

Placed 0

df2.loc[6:15,'SEM1'] #prints particular column in given range

6 NaN

7 NaN

8 NaN

9 NaN

10 NaN

11 NaN

12 9.8

13 8.9

14 9.3

15 9.9

df2.loc[[1,3]] #prints the rows which are specified

#output:

|  | **Names** | **Initials** | **SEM1** | **SEM2** | **SEM3** | **Grade** | **Placed** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | Rajesh | M | 8.9 | 9.1 | 9.3 | A | 1 |
| **3** | Mary | N | NaN | 8.0 | NaN | B | 0 |

df2.loc[10:15,['Names','Grade']] #prints the columns with in given range of rows

#output:

|  | **Names** | **Grade** |
| --- | --- | --- |
| **10** | Kiran | A |
| **11** | Maya | B |
| **12** | Jolin | A+ |
| **13** | Rajesh | A |
| **14** | Riya | A |
| **15** | Sana | A |

df2.iloc[10:14,:3] #row,column

#output:

|  | **Names** | **Initials** | **SEM1** |
| --- | --- | --- | --- |
| **10** | Kiran | V | NaN |
| **11** | Maya | N | NaN |
| **12** | Jolin | K | 9.8 |
| **13** | Rajesh | M | 8.9 |

df2.loc[:5,["Names"]]

#output:

|  | **Names** |
| --- | --- |
| **0** | Joe |
| **1** | Rajesh |
| **2** | Kissan |
| **3** | Mary |
| **4** | Jeen |
| **5** | Raj |

df2.loc[df['Names']=='Rajesh'] #particular details

#output:

|  | **Names** | **Initials** | **SEM1** | **SEM2** | **SEM3** | **Grade** | **Placed** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | Rajesh | M | 8.9 | 9.1 | 9.3 | A | 1 |
| **13** | Rajesh | M | 8.9 | 9.1 | 9.3 | A | 1 |

df[df.Grade=='A']

df[df.SEM2==10]

#output:

| **Names** | **Initials** | **SEM1** | **SEM2** | **SEM3** | **Grade** | **Placed** |
| --- | --- | --- | --- | --- | --- | --- |
| joe | k | 9.8 | 10.0 | 9.9 | A+ | 1 |
|  |  |  |  |  |  |  |

#PRINTS particular column which satisfies the condition

df.loc[df['SEM1'] > 9.8,['Names']]

#output:

|  | **Names** |
| --- | --- |
| **2** | Kissan |
| **6** | Hassan |
| **15** | Sana |

df3=df.copy()

df3

#drops columns which consists nan

df3.dropna(inplace=True,axis=1) # inplace says not to rechange in df

df3

#output:

|  | **Names** | **Initials** | **SEM2** | **Grade** | **placed** |
| --- | --- | --- | --- | --- | --- |
| **0** | Joe | K | 10.0 | A+ | 1 |
| **1** | Rajesh | M | 9.1 | A | 1 |
| **2** | Kissan | V | 9.8 | A | 0 |
| **3** | Mary | N | 8.0 | B | 0 |
| **4** | Jeen | K | 9.1 | A+ | 1 |
| **5** | Raj | M | 9.1 | A | 1 |
| **6** | Hassan | V | 9.0 | A | 1 |
| **7** | Mari | N | 8.0 | B | 1 |
| **8** | Jess | K | 9.1 | A+ | 1 |
| **9** | Rajini | M | 9.1 | A | 0 |
| **10** | Kiran | V | 9.3 | A | 0 |
| **11** | Maya | N | 8.0 | B | 0 |
| **12** | Jolin | K | 9.1 | A+ | 1 |
| **13** | Rajesh | M | 9.1 | A | 1 |
| **14** | Riya | M | 9.9 | A | 1 |
| **15** | Sana | V | 9.3 | A | 0 |
| **16** | Mark | N | 8.0 | B | 1 |

#drops both rows and columns which consists nan

df3.dropna(inplace=True,how='all') #inplace says not to rechange in df

df3

# Inplace of nan mean value is replaced

mv=df['SEM1'].mean()

df=df.fillna(mv)

print(df)

#output:

**Names Initials SEM1 SEM2 SEM3 Grade Placed**

0 Joe K 9.800000 10.0 9.900000 A+ 1

1 Rajesh M 8.900000 9.1 9.300000 A 1

2 Kissan V 9.900000 9.8 10.000000 A 0

3 Mary N 7.700000 8.0 8.992857 B 0

4 Jeen K 9.800000 9.1 9.900000 A+ 1

5 Raj M 8.900000 9.1 9.300000 A 1

6 Hassan V 9.900000 9.0 9.200000 A 1

7 Mari N 7.700000 8.0 7.100000 B 1

8 Jess K 8.992857 9.1 9.900000 A+ 1

9 Rajini M 8.992857 9.1 9.300000 A 0

10 Kiran V 8.992857 9.3 9.20000 9.3 9.200000 A 0

11 Maya N 7.700000 8.0 7.100000 B 0

12 Jolin K 9.800000 9.1 9.900000 A+ 1

13 Rajesh M 8.900000 9.1 9.300000 A 1

14 Riya M 9.300000 9.9 10.000000 A 1

15 Sana V 9.900000 9.3 9.200000 A 0

16 Mark N 7.700000 8.0 7.000000 B 0

DAY 3:26-02-2024

**MATPLOTLIB LIBRARY:**Python library that enables you to create a wide variety of plots, charts, and visualizations.

Major Applications of Matplotlib**:**

1) Data Analysis

2) Scientific computing

3) Machine learning

>”matplotlib.pyplot” is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure:

e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels,etc.

>>To import matplotlib : “import matplotlib.pyplot as plt”

Dataset : The data in the form of a file is called dataset.

Operations with data set:

1) loading the dataset into the program.

→”data\_name=np.loadtxt("path of .txt file",dtype=data\_type)”

2) Saving the dataset from the program.

→”data\_name=np.savetxt("/content/new\_file\_name.txt",data\_file\_name)”

3) Plotting using the dataset.

→”plt.plot(array\_name)”

4) Scatter plot the data.

→”plt.scatter(x,y,color=’color\_name’)

→To give title : “plt.title(‘title\_name’)”

→To display graph: “plt.show()”

→To display label : “plt.legend(loc=’best’)”

5) Plot the data.

→”plt.plot(x,y,color=’color\_name’)

→To give title : “plt.title(‘title\_name’)”

→To display graph : “plt.show()”

→To display label : “plt.legend(loc=’best’)”

6) Display data in the form of pie chart.

→”plt.pie(array\_1,array\_2,colors=array\_of\_colors,startangle=angle)”

#cricket

import numpy as np

#importing Matplotlib

import matplotlib.pyplot as plt

runs=np.array([100,50,91,78,89,25,34,19,9,10])

w=np.array([1,0,2,0,3,7,8,9,7,5])

#scater only point

# plot connects points

plt.scatter(runs,w,color='orange',label='India')

#plt.scatter(w,runs,color='red')

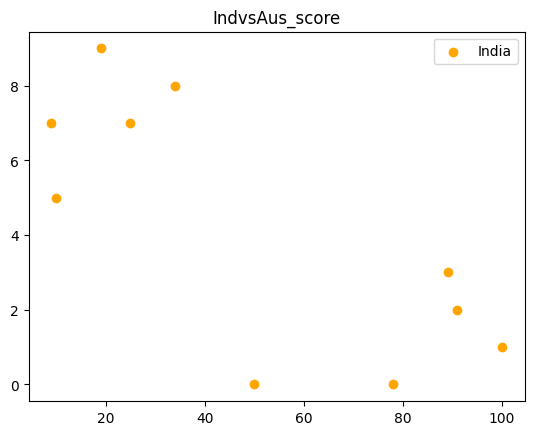
plt.title('IndvsAus\_score')

#legend is related to label locates the label at best place

plt.legend(loc='best')

plt.show()

OUTPUT:



import numpy as np

import matplotlib.pyplot as plt

# Generate array of 200 values between -pi & pi

tigar = np.linspace(-2\*np.pi, 2\*np.pi, 100)

print(tigar)

plt.plot(tigar, np.sin(tigar) ,color='black',linestyle='dotted') # SIN

plt.title("sin(x)")

# Display plot

plt.show()

OUTPUT:

[-6.28318531 -6.15625227 -6.02931923 -5.9023862 -5.77545316 -5.64852012

-5.52158709 -5.39465405 -5.26772102 -5.14078798 -5.01385494 -4.88692191

-4.75998887 -4.63305583 -4.5061228 -4.37918976 -4.25225672 -4.12532369

-3.99839065 -3.87145761 -3.74452458 -3.61759154 -3.4906585 -3.36372547

-3.23679243 -3.10985939 -2.98292636 -2.85599332 -2.72906028 -2.60212725

-2.47519421 -2.34826118 -2.22132814 -2.0943951 -1.96746207 -1.84052903

-1.71359599 -1.58666296 -1.45972992 -1.33279688 -1.20586385 -1.07893081

-0.95199777 -0.82506474 -0.6981317 -0.57119866 -0.44426563 -0.31733259

-0.19039955 -0.06346652 0.06346652 0.19039955 0.31733259 0.44426563

0.57119866 0.6981317 0.82506474 0.95199777 1.07893081 1.20586385

1.33279688 1.45972992 1.58666296 1.71359599 1.84052903 1.96746207

2.0943951 2.22132814 2.34826118 2.47519421 2.60212725 2.72906028

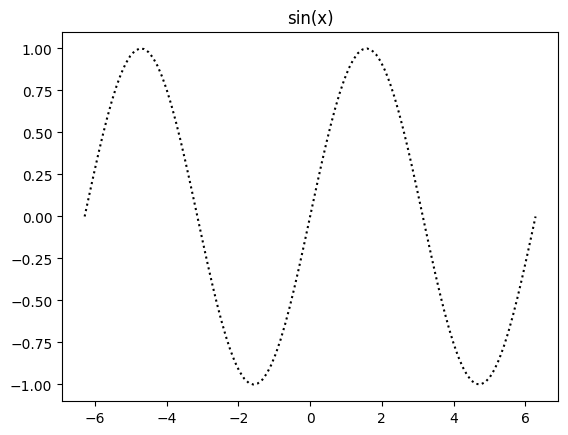
2.85599332 2.98292636 3.10985939 3.23679243 3.36372547 3.4906585

3.61759154 3.74452458 3.87145761 3.99839065 4.12532369 4.25225672

4.37918976 4.5061228 4.63305583 4.75998887 4.88692191 5.01385494

5.14078798 5.26772102 5.39465405 5.52158709 5.64852012 5.77545316

5.9023862 6.02931923 6.15625227 6.28318531]



#plotting all in one graph

# use : plt.legend(loc='best')

#plt.plot(x,y,color,label)

import numpy as np

import matplotlib.pyplot as plt

#creating x

overs = np.arange(5,50,5)

overs\_a = np.arange(5,30,5)

#creating y

runs\_i = np.array([25,51,84,131,160,189,220,250,267])

runs\_a = np.array([15,41,94,110,151])

wickets = np.array([12,32,96])

#plotting

plt.plot(overs,runs\_i,color='blue',label='India')

plt.plot(overs\_a,runs\_a,color='yellow',label = 'Aus')

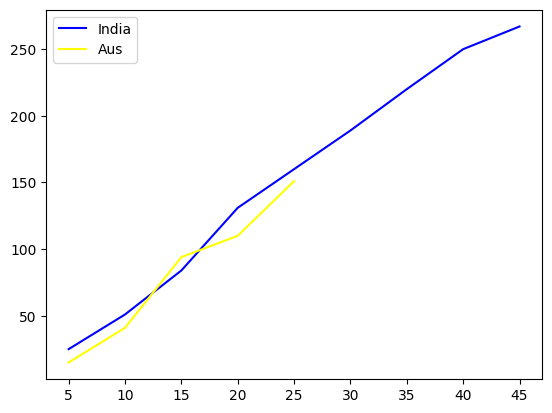
#combining two graphs

plt.legend(loc='best')#locates the label at best place

#displaying the final graph

plt.show()

OUTPUT:



import matplotlib.pyplot as plt

a = [230,560,780,127,128]

b = [200,160,270,127,400]

years = [1,2,3,4]

profit\_a = [(a[i]-a[i-1]) for i in range(1,len(a))]

profit\_b = [(b[i]-b[i-1]) for i in range(1,len(b))]

plt.subplot(2,1,2)

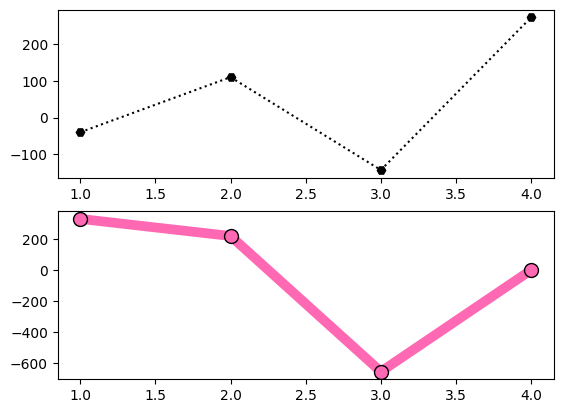
plt.plot(years,profit\_a,color='hotpink',linewidth = '7',label ='CompanyA',marker='.',ms='20',mec='k')

plt.subplot(2,1,1)

plt.plot(years,profit\_b,color='black',linestyle='dotted',label ='CompanyB',marker = 'H')

plt.show()

OUTPUT:



a = np.array([25,60,5,10])

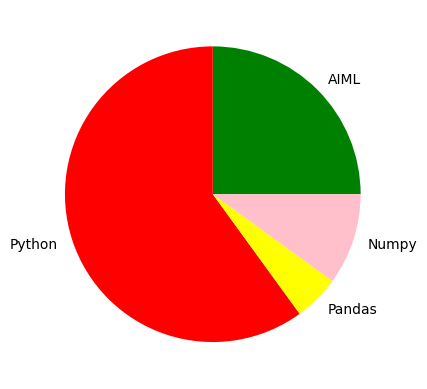
labe = ["AIML","Python","Pandas","Numpy"]

colors=['Green','red','yellow','pink']

plt.pie(a,labels = labe,colors=colors)

plt.show()

OUTPUT:



a = np.array([25,60,5,10])

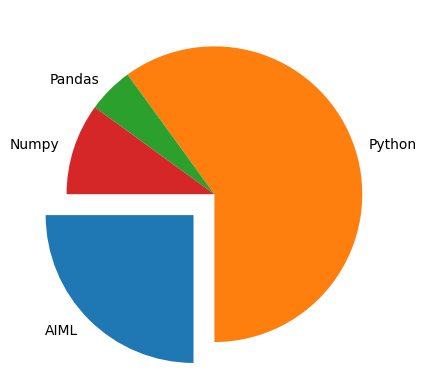
labe = ["AIML","Python","Pandas","Numpy"]

explo = [0.2,0,0,0]

plt.pie(a,labels = labe,explode = explo,startangle = 180)

plt.show()

OUTPUT:



a = np.array([25,60,5,10])

labe = ["AIML","Python","Pandas","Numpy"]

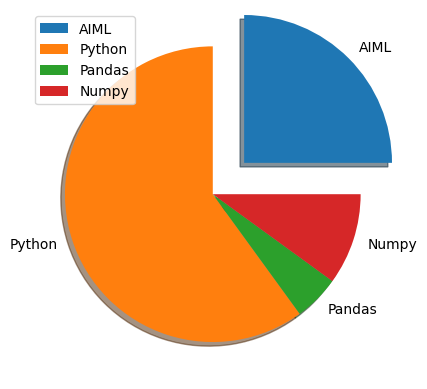
explo = [0.3,0,0,0]

plt.pie(a,labels = labe,explode = explo,shadow=True)

plt.legend()

plt.show()

OUTPUT:



a = np.array([25,60,5,10])

labe = ["AIML","Python","Pandas","Numpy"]

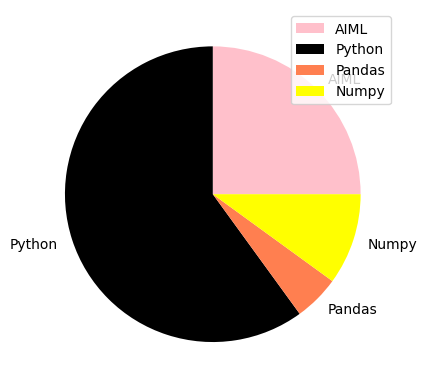
color = ['pink','black','coral','yellow']

plt.pie(a,labels = labe,colors= color)

plt.legend()

plt.show()

OUTPUT:



**Seaborn :** Seaborn is a library for making statistical graphics in Python.It builds on top of matplotlib and integrates closely with pandas data structures.

>>It contains the builtin datasets.

>>To import seaborn : “import seaborn as sns”

Operations:

1) Load the dataset into the program.

→”var\_name=sns.load\_dataset(“dataset\_name”)”

2) Scatterplot the data in a dataset.

→”sns.scatterplot(x=”label1\_name”.y=”label2\_name”,data=var\_name)”

3) Violinplot the data in a dataset.

→”sns.violinplot(x=”label1\_name”.y=”label2\_name”,data=var\_name)”

4) Heatmap from the data in a dataset.

>>Heatmap : a technique of data visualisation that makes use of colour in order to exhibit how a value of interest varies on the basis of the values of the two other variables. To sum it up, using different colours to represent data gives you a general view of the numerical data.

>>Correlation matrix : The correlation matrix is a matrix that shows

the correlation between variables.

→correlation\_matrix=”var\_name.corr()”

→sns.heatmap(correlation\_matrix,annot=True,cmap=”coolwarm”)”

pip install seaborn

import seaborn as sns

import matplotlib.pyplot as plt

#load example dataset

iris=sns.load\_dataset("iris")

print(iris)

plt.legend(loc='best')

#create a scatter plot

sns.scatterplot(x="sepal\_length",y="sepal\_width",data=iris)

plt.title("scatter plot of sepal\_length vs sepal\_width")

plt.xlabel("sepal\_length($)")

plt.ylabel("sepal\_width ($)")

#plt.legenda(loc='best')

plt.show()

OUTPUT:

sepal\_length sepal\_width petal\_length petal\_width species

0 5.1 3.5 1.4 0.2 setosa

1 4.9 3.0 1.4 0.2 setosa

2 4.7 3.2 1.3 0.2 setosa

3 4.6 3.1 1.5 0.2 setosa

4 5.0 3.6 1.4 0.2 setosa

.. ... ... ... ... ...

145 6.7 3.0 5.2 2.3 virginica

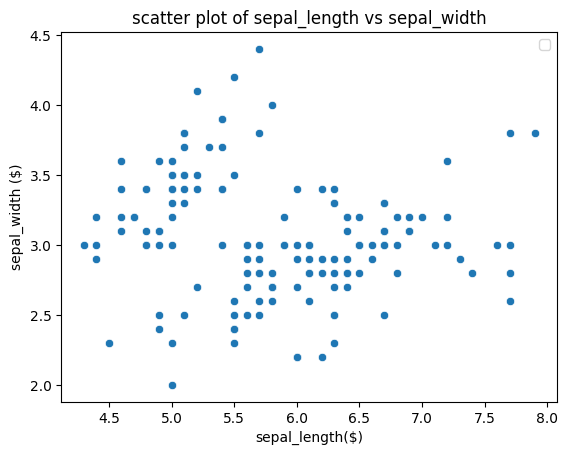
146 6.3 2.5 5.0 1.9 virginica

147 6.5 3.0 5.2 2.0 virginica

148 6.2 3.4 5.4 2.3 virginica

149 5.9 3.0 5.1 1.8 virginica

[150 rows x 5 columns]



import seaborn as sns

import matplotlib.pyplot as plt

#load example dataset

tips=sns.load\_dataset("tips")

print(tips)

plt.legend(loc='best')

#create a scatter plot

#sns.scatterplot(x="day",y="total\_bill",data=tips)

sns.barplot(x="day",y="total\_bill",data=tips)

plt.title("scatter plot of sepal\_length vs sepal\_width")

plt.xlabel("day($)")

plt.ylabel("total\_bill ($)")

#plt.legenda(loc='best')

plt.show()

OUTPUT:

total\_bill tip sex smoker day time size

0 16.99 1.01 Female No Sun Dinner 2

1 10.34 1.66 Male No Sun Dinner 3

2 21.01 3.50 Male No Sun Dinner 3

3 23.68 3.31 Male No Sun Dinner 2

4 24.59 3.61 Female No Sun Dinner 4

.. ... ... ... ... ... ... ...

239 29.03 5.92 Male No Sat Dinner 3

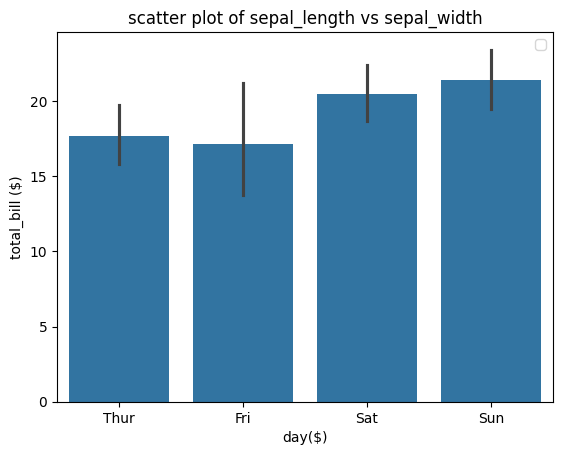
240 27.18 2.00 Female Yes Sat Dinner 2

241 22.67 2.00 Male Yes Sat Dinner 2

242 17.82 1.75 Male No Sat Dinner 2

243 18.78 3.00 Female No Thur Dinner 2

[244 rows x 7 columns]



import seaborn as sns

import matplotlib.pyplot as plt

#load example dataset

tips=sns.load\_dataset("tips")

print(tips)

plt.legend(loc='best')

#create a scatter plot

#sns.scatterplot(x="day",y="total\_bill",data=tips)

sns.boxplot(x="day",y="total\_bill",data=tips)

plt.title("day Vs total bill")

plt.xlabel("day($)")

plt.ylabel("total\_bill ($)")

#plt.legenda(loc='best')

plt.show()

OUTPUT:

total\_bill tip sex smoker day time size

0 16.99 1.01 Female No Sun Dinner 2

1 10.34 1.66 Male No Sun Dinner 3

2 21.01 3.50 Male No Sun Dinner 3

3 23.68 3.31 Male No Sun Dinner 2

4 24.59 3.61 Female No Sun Dinner 4

.. ... ... ... ... ... ... ...

239 29.03 5.92 Male No Sat Dinner 3

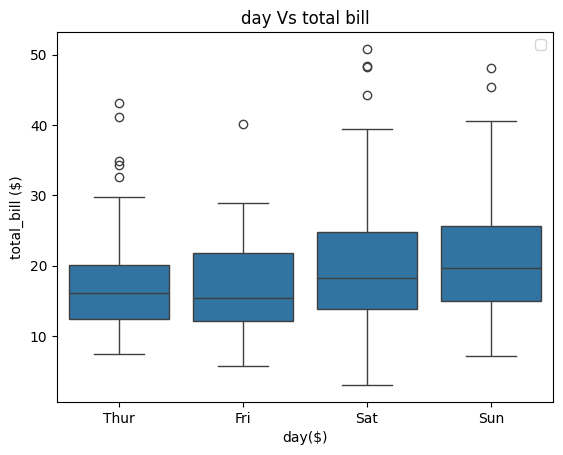
240 27.18 2.00 Female Yes Sat Dinner 2

241 22.67 2.00 Male Yes Sat Dinner 2

242 17.82 1.75 Male No Sat Dinner 2

243 18.78 3.00 Female No Thur Dinner 2

[244 rows x 7 columns]



import seaborn as sns

import matplotlib.pyplot as plt

#load example dataset

tips=sns.load\_dataset("tips")

print(tips)

plt.legend(loc='best')

#create a scatter plot

#sns.scatterplot(x="day",y="total\_bill",data=tips)

sns.violinplot(x="day",y="total\_bill",data=tips)

plt.title("scatter plot of sepal\_length vs sepal\_width")

plt.xlabel("day($)")

plt.ylabel("total\_bill ($)")

#plt.legenda(loc='best')

plt.show()

OUTPUT:

total\_bill tip sex smoker day time size

0 16.99 1.01 Female No Sun Dinner 2

1 10.34 1.66 Male No Sun Dinner 3

2 21.01 3.50 Male No Sun Dinner 3

3 23.68 3.31 Male No Sun Dinner 2

4 24.59 3.61 Female No Sun Dinner 4

.. ... ... ... ... ... ... ...

239 29.03 5.92 Male No Sat Dinner 3

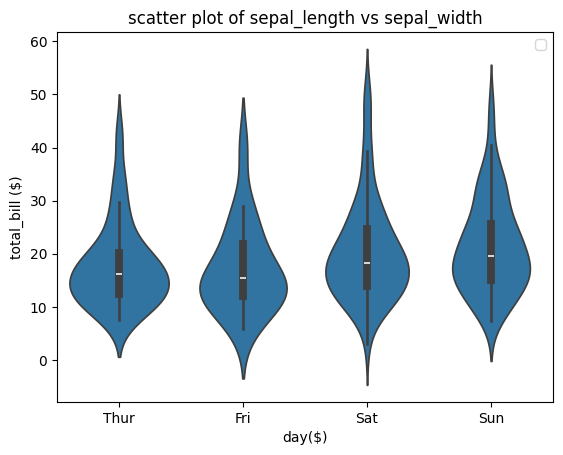
240 27.18 2.00 Female Yes Sat Dinner 2

241 22.67 2.00 Male Yes Sat Dinner 2

242 17.82 1.75 Male No Sat Dinner 2

243 18.78 3.00 Female No Thur Dinner 2

[244 rows x 7 columns]



import pandas as pd

import seaborn as sns

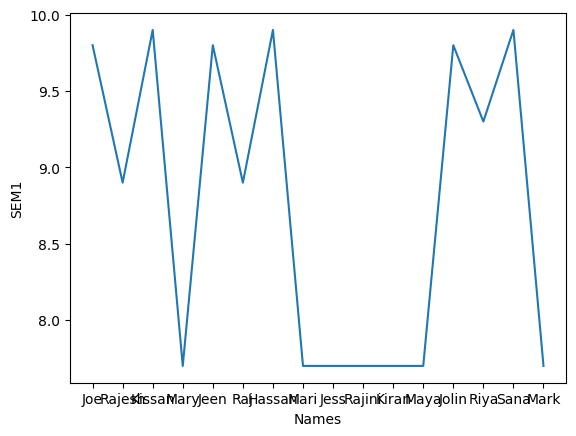
import matplotlib.pyplot as plt

a=pd.read\_csv("/content/grades\_withnulls.csv")

b=sns.lineplot(x='Names',y='SEM1',data=a)

plt.show()

OUTPUT:



import seaborn as sns

import matplotlib.pyplot as plt

#load example dataset

iris=sns.load\_dataset("iris")

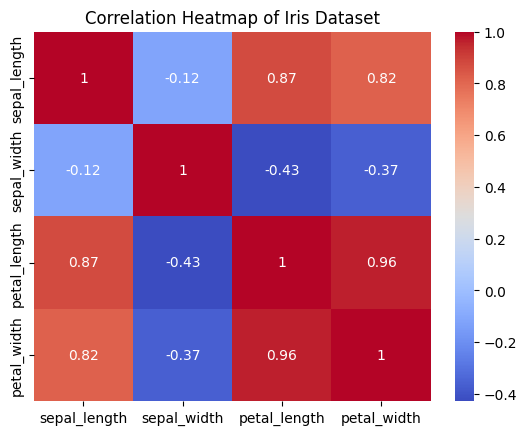
correlation\_matrix=iris.corr()

sns.heatmap(correlation\_matrix,annot=True,cmap="coolwarm")

plt.title("Correlation Heatmap of Iris Dataset")

plt.show()

OUTPUT:



import seaborn as sns

import matplotlib.pyplot as plt

#load example dataset

tips=sns.load\_dataset("tips")

print(tips)

plt.legend(loc='best')

#create a scatter plot

#sns.scatterplot(x="day",y="total\_bill",data=tips)

sns.jointplot(x="day",y="total\_bill",data=tips)

plt.title("day Vs total bill")

plt.xlabel("day($)")

plt.ylabel("total\_bill ($)")

#plt.legenda(loc='best')

plt.show()

OUTPUT:

total\_bill tip sex smoker day time size

0 16.99 1.01 Female No Sun Dinner 2

1 10.34 1.66 Male No Sun Dinner 3

2 21.01 3.50 Male No Sun Dinner 3

3 23.68 3.31 Male No Sun Dinner 2

4 24.59 3.61 Female No Sun Dinner 4

.. ... ... ... ... ... ... ...

239 29.03 5.92 Male No Sat Dinner 3

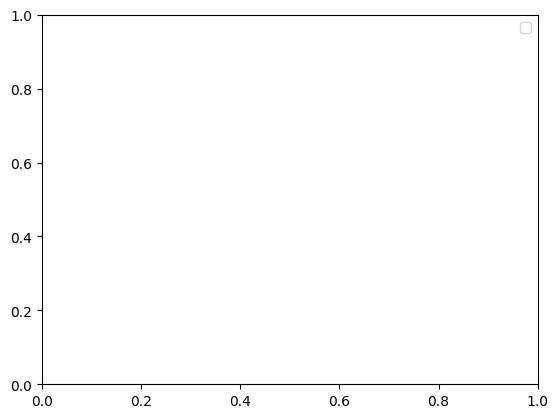
240 27.18 2.00 Female Yes Sat Dinner 2

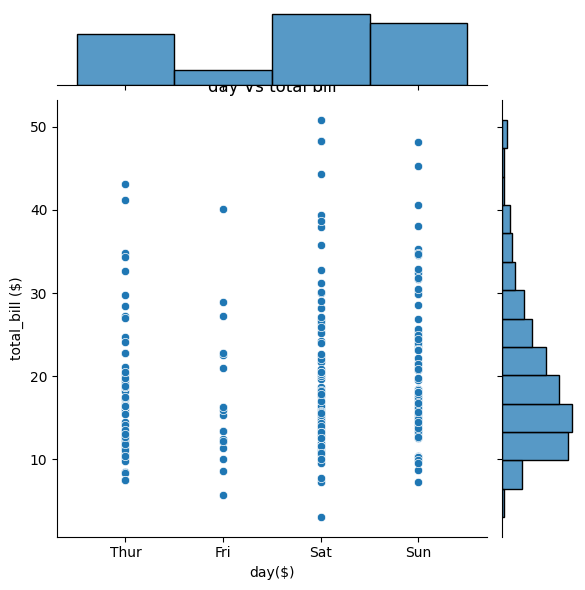
241 22.67 2.00 Male Yes Sat Dinner 2

242 17.82 1.75 Male No Sat Dinner 2

243 18.78 3.00 Female No Thur Dinner 2

[244 rows x 7 columns]





import seaborn as sns

import matplotlib.pyplot as plt

#load example dataset

titanic=sns.load\_dataset("titanic")

print(titanic)

plt.legend(loc='best')

#create a scatter plot

#sns.scatterplot(x="day",y="total\_bill",data=tips)

sns.countplot(x="class",data=titanic)

plt.title("day Vs total bill")

plt.xlabel("class")

plt.ylabel("count")

#plt.legenda(loc='best')

plt.show()

OUTPUT:

survived pclass sex age sibsp parch fare embarked class \

0 0 3 male 22.0 1 0 7.2500 S Third

1 1 1 female 38.0 1 0 71.2833 C First

2 1 3 female 26.0 0 0 7.9250 S Third

3 1 1 female 35.0 1 0 53.1000 S First

4 0 3 male 35.0 0 0 8.0500 S Third

.. ... ... ... ... ... ... ... ... ...

886 0 2 male 27.0 0 0 13.0000 S Second

887 1 1 female 19.0 0 0 30.0000 S First

888 0 3 female NaN 1 2 23.4500 S Third

889 1 1 male 26.0 0 0 30.0000 C First

890 0 3 male 32.0 0 0 7.7500 Q Third

who adult\_male deck embark\_town alive alone

0 man True NaN Southampton no False

1 woman False C Cherbourg yes False

2 woman False NaN Southampton yes True

3 woman False C Southampton yes False

4 man True NaN Southampton no True

.. ... ... ... ... ... ...

886 man True NaN Southampton no True

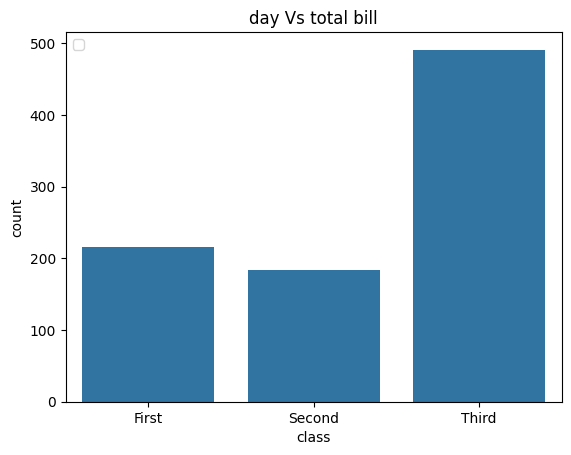
887 woman False B Southampton yes True

888 woman False NaN Southampton no False

889 man True C Cherbourg yes True

890 man True NaN Queenstown no True

[891 rows x 15 columns]



import seaborn as sns

import matplotlib.pyplot as plt

#load example dataset

tips=sns.load\_dataset("tips")

print(tips)

plt.legend(loc='best')

#create a scatter plot

#sns.scatterplot(x="day",y="total\_bill",data=tips)

sns.lmplot(x="total\_bill",y="tip",data=tips)

plt.title("day Vs total bill")

plt.xlabel("total\_bill")

plt.ylabel("tip")

#plt.legenda(loc='best')

plt.show()

OUTPUT:

total\_bill tip sex smoker day time size

0 16.99 1.01 Female No Sun Dinner 2

1 10.34 1.66 Male No Sun Dinner 3

2 21.01 3.50 Male No Sun Dinner 3

3 23.68 3.31 Male No Sun Dinner 2

4 24.59 3.61 Female No Sun Dinner 4

.. ... ... ... ... ... ... ...

239 29.03 5.92 Male No Sat Dinner 3

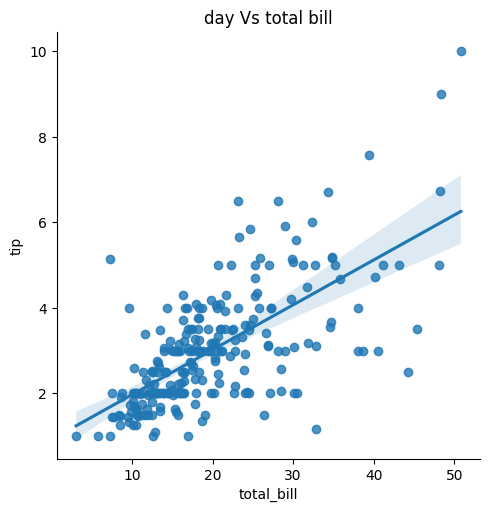
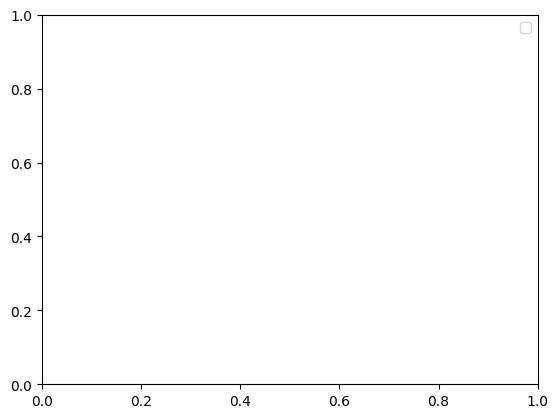
240 27.18 2.00 Female Yes Sat Dinner 2

241 22.67 2.00 Male Yes Sat Dinner 2

242 17.82 1.75 Male No Sat Dinner 2

243 18.78 3.00 Female No Thur Dinner 2

[244 rows x 7 columns]



import seaborn as sns

import matplotlib.pyplot as plt

#load example dataset

titanic=sns.load\_dataset("titanic")

print(titanic)

#create a scatter plot

#sns.scatterplot(x="day",y="total\_bill",data=titanic)

g=sns.FacetGrid(titanic,col="class")

g.map(sns.histplot, "age")

plt.subplots\_adjust(top=0.8)

g.fig.suptitle("Age Distribution by Passenger Class")

#plt.legenda(loc='best')

plt.show()

OUTPUT:

survived pclass sex age sibsp parch fare embarked class \

0 0 3 male 22.0 1 0 7.2500 S Third

1 1 1 female 38.0 1 0 71.2833 C First

2 1 3 female 26.0 0 0 7.9250 S Third

3 1 1 female 35.0 1 0 53.1000 S First

4 0 3 male 35.0 0 0 8.0500 S Third

.. ... ... ... ... ... ... ... ... ...

886 0 2 male 27.0 0 0 13.0000 S Second

887 1 1 female 19.0 0 0 30.0000 S First

888 0 3 female NaN 1 2 23.4500 S Third

889 1 1 male 26.0 0 0 30.0000 C First

890 0 3 male 32.0 0 0 7.7500 Q Third

who adult\_male deck embark\_town alive alone

0 man True NaN Southampton no False

1 woman False C Cherbourg yes False

2 woman False NaN Southampton yes True

3 woman False C Southampton yes False

4 man True NaN Southampton no True

.. ... ... ... ... ... ...

886 man True NaN Southampton no True

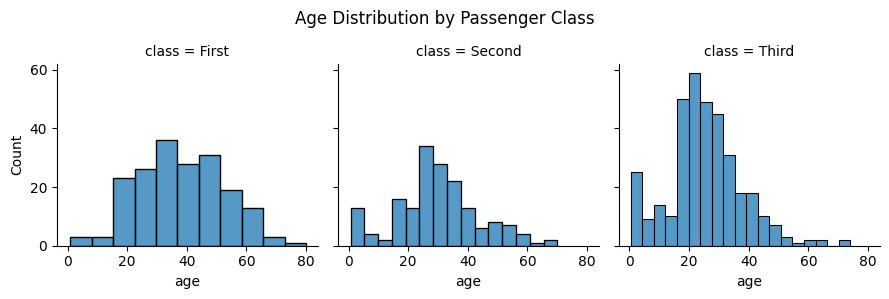
887 woman False B Southampton yes True

888 woman False NaN Southampton no False

889 man True C Cherbourg yes True

890 man True NaN Queenstown no True

[891 rows x 15 columns]



CNN : A Convolutional Neural Network (CNN) is a type of Deep Learning neural network architecture . Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.

CNN Architecture:

Input layer->Hidden layer(Convolution layer->Activation function->Max Pooling->Avg Pooling->Dense layer)->Output layer

Input Layer • It's the layer in which we give input to our model. In CNN, Generally, the input will be an image or a sequence of images. This layer holds the raw input of the image with width 32, height 32, and depth 3.

Convolutional Layer:

This is the layer, used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices usually 2x2, 3x3, or 5x5 shape. it slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch.

• The output of this layer is referred ad feature maps. Suppose we use a total of 12 filters for this layer we'll get an output volume of dimension 32 x 32 x 12.

Activation Layer:

By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network, it will apply an element-wise activation function to the output of the convolution layer. Some common activation functions are RELU: max(0, x), Tanh, Leaky RELU, etc.

• The volume remains unchanged hence output volume will have dimensions 32 x 32 x 12

Pooling Layer:

• This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are max pooling and average pooling. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.

Output Layer:

• The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.

Activation Function:

• The activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it. The purpose of the activation function is to introduce non-linearity into the output of a neuron. We know, the neural network has neurons that work in correspondence with weight, bias, and their respective activation function. In a neural network, we would update the weights and biases of the neurons on the basis of the error at the output. This process is known as back-propagation.

• Tanh

•Sigmoid

•Relu

•Softmax

Sigmoid Function

• It is a function which is plotted as 'S' shaped graph.

•Equation: A = 1/(1 + e-x)

•Nature: Non-linear.

• Value Range: 0 to 1

• Uses: Usually used in output layer of a binary classification, where result is either 0 or 1, as value for sigmoid function lies between 0 and 1 only so, result can be predicted easily to be 1 if value is greater than 0.5 and 0 otherwise.

Tanh Function

• Value Range :- -1 to +1

•Nature: non-linear

•Uses: Usually used in hidden layers of a neural network as it's values lies between -1 to 1

Relu

•It stands for Rectified linear unit.It is the most widely used activation layers •Equation:- A(x) = max(0,x). It gives an output x if x is positive and 0 otherwise.

•Value Range: [0, inf)

•Nature: non-linear

•Uses:- ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation. RELU learns much faster than sigmoid and Tanh function.

Softmax

•The softmax function is also a type of sigmoid function but is handy when we are trying to handle multi-class classification problems

•Nature:-non-linear

•Uses: Usually used when trying to handle multiple classes, the softmax function was commonly found in the output layer of image classification problems. The softmax function would squeeze the outputs for each class between 0 and 1 and would also divide by the sum of the outputs.

•Output:- .The basic rule of thumb is if really don't know what activation function to then simply use RELU as it is a general activation function in hidden layers and is used in most cases these days.

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

from tensorflow.keras.preprocessing.image import ImageDataGenerator

#Define image size and batch size

IMG\_SIZE=224

BATCH\_SIZE=32

#Define data generators for train,validation and test sets

train\_datagen=ImageDataGenerator(

rescale=1./255,

validation\_split=0.2

)

train\_generator=train\_datagen.flow\_from\_directory('/content/drive/MyDrive/Database1/Brain\_Tumor\_Detection/train',

target\_size=(IMG\_SIZE,IMG\_SIZE),

batch\_size=BATCH\_SIZE,

class\_mode='binary',

subset='training'

)

val\_generator=train\_datagen.flow\_from\_directory('/content/drive/MyDrive/Database1/Brain\_Tumor\_Detection/train',

target\_size=(IMG\_SIZE,IMG\_SIZE),

batch\_size=BATCH\_SIZE,

class\_mode='binary',

subset='validation'

)

test\_datagen=ImageDataGenerator(rescale=1./255)

test\_generator=test\_datagen.flow\_from\_directory('/content/drive/MyDrive/Database1/Brain\_Tumor\_Detection/train',

target\_size=(IMG\_SIZE,IMG\_SIZE),

batch\_size=BATCH\_SIZE,

class\_mode='binary',

)

#Define the model

model=keras.Sequential([

layers.Conv2D(32,(3,3),activation='relu',input\_shape=(IMG\_SIZE,IMG\_SIZE,3)),

layers.MaxPooling2D((2,2)),

layers.Conv2D(64,(3,3),activation='relu'),

layers.MaxPooling2D((2,2)),

layers.Conv2D(128,(3,3),activation='relu'),

layers.MaxPooling2D((2,2)),

layers.Flatten(),

layers.Dense(128,activation='relu'),

layers.Dense(1,activation='sigmoid'),

])

#compile the model

model.compile(optimizer='adam',loss='binary\_crossentropy',metrics=['accuracy'])

model.save("Model.h5","label.txt")

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing import image

import numpy as np

#load the saved model

model=load\_model('/content/Model.h5')

#load and preprocess the test image

test\_image\_path='/content/drive/MyDrive/Database1/Brain\_Tumor\_Detection/train/no/No13.jpg'

img=image.load\_img(test\_image\_path,target\_size=(224,224))

img\_array=image.img\_to\_array(img)

img\_array=np.expand\_dims(img\_array,axis=0)#Add batch dimension

img\_array /=255. #Normalize the pixel values

#Make predictions

prediction=model.predict(img\_array)

#Print the prediction

if prediction<0.5:

print("Prediction:No tumor (Probability:",prediction[0][0],")")

else:

print("Prediction:Tumor present (Probability:",prediction[0][0],")")

output:

1/1 [==============================] - 0s 251ms/step

Prediction:No tumor (Probability: 2.0546553e-07 )

**1) eye() Method:- Th**

## 

## 