**Data Analysis with Python**

**1. Concept Overview -Python**

**1.1. Variable:**

Container to store values.

Code:

a=5

**1.2. Print:**

It is a function which used to display

Code:

a=5

print("i am ",a,"years old")

Output:

i am 5 years old

**1.3. Operators:**

Operators are a special symbols combination of symbols or key words that designates some type of computation.

**Arithmetic Operators:**

An Arithmetic operator is a special type of mathematical function that performs a calculation on two operands.The different types of arithmetic operators include addition,subtraction,multiplication,division,modulus,exponentiation and floor division.

Addition,Subtraction,Multiplication,Division

Code:

a=5

b=25

print(a+b)

print(a-b)

print(a\*b)

print(a/b)

Output:

30

-20

125

0.2

Power or Exponent

Code:

print(2\*\*4)

Output:

16

Floor Division

Code:

print(25//2)

Output:

12

**Relational Operators:**

A relational operator in python is a symbol or keyword used to compare two values and determine the relation between them .The different types of relational operators include are <(less than),>(greater than),<=(less than or equal to),>=(greater than or equal to),!=(not equal to).

Code:

a=6

b=6

print(a==b)

print(a>b)

print(a<b)

print(a!=b)

print(a>=b)

print(a<=b)

Output:

True

False

False

False

True

True

**Logical Operators:**

Logical operators are used to combine multiple conditions together less than as a single boolean expression.There are three types of logical operators in python ‘and’,’or’,’not’.

Code:

a=9

b=25

print((a>b) and (a<b)) #false and true

print((a>b) or (a<b)) #false or true

Output:

False

True

**Membership Operators:**

Checks whether the given value is a member of sequence such as strings,lists and tuples. Two primary types:’in’(return True if specified value is found within the sequence)and ‘not in’(returns True if specified value is not in the given sequence).

Code:

a="sasidharani"

print("a" in a)

print("U" in a)

Output:

True

False

**1.4. Control flow - Conditional Statements :**

**Conditional - if :**

An if statement is a conditional statement used to check the condition,and executes if the condition is true.It is also a control flow statement which utilises decision-making to control the flow of execution.

#write a program to get a no from the user and check whether it is positive or negative

Code:

a=int(input("enter a number:"))

if a>0:

print(a,"is positive")

Output:

enter a number:6

6 is positive

**Conditional -if-else:**

The if-else statement used to execute both the true and the false part of a given condition.

#write a program to get a no from the user and check whether it is positive or negative

Code:

a=int(input("enter a number:"))

if(a>0):

print("a is positive")

else:

print("a is negative")

Output:

enter a number:23

a is positive

**Conditional if-else-ladder:**

Common programming construct that is based upon nested if is the if-else-if ladder.

#write a program to get a no from the user and check whether it is positive , negative , zero

Code:

a=int(input("enter a number:"))

if(a>0):

print(a, "is positive")

elif(a==0):

print("print zero")

else:

print(a,"is negative")

Output:

enter a number:0

print zero

**1.5. Control flow - Looping Statements :**

**Table by using - For loop:**

A control flow statement that is used to repeatedly execute the group of statements.

Code:

a=int(input("enter a number:"))

for i in range(1,11,1):

n=a\*i

print(n)

Output:

enter a number:4

4

8

12

16

20

24

28

32

36

40

**Table by using - While loop:**

A control flow statement which repeatedly executes a block of code until the condition is satisfied.

Code:

a=int(input("enter a number:"))

i=1

while(i<=10):

n=a\*i

print(n)

i=i+1

Output:

enter a number:5

5

10

15

20

25

30

35

40

45

50

**1.6. Data Slicing :**

The slice() method extracts a section of data and returns it as new data, without modifying it. This means users can take a specific range of elements without changing it.

Syntax: slice(start,stop,step)

Code:

a="Python is easy"

print(a[0:6]) #start:stop:step

Output:

Python

Reverse

Code:

a="Python is easy"

print(a[::-1]) #start:stop:step

Output:

ysae si nohtyP

Slicing by step 2

Code:

a="Python is easy"

print(a[::2])#start:stop:2

Output:

Pto ses

**1.7. Type Casting :**

It is a process that converts a variable’s data type into another data type.

Implicit type casting: By interpreter.

Implicit Type Casting

Code:

a=12.6 #float

b=4 #int

print(a\*b) #interpreter automatically taken the output as float

Output:

50.4

Explicit type casting: By user.

Explicit Type Casting

Code:

a=12.6 #float

b=4 #int

print(int(a\*b)) #user

Output:

50

Code:

a="45"

a=int(a)

print(type(a))

Output:

<class 'int'>

**1.8. Collection List:**

**LIST :**

* List is a collection of elements.
* List is Heterogenous.
* Mutable(modifiable).

List

Code:

list=[1,"akhi",3,4,"sasi",6]

print(list)

Output:

[1, 'akhi', 3, 4, 'sasi', 6]

Lists collection

Code:

s1=[34,"sasi",123,149,"sri"]

for i in l1:

print(i)

Output:

1

2

6

[1, 4, 9]

List Append

Code:

l1=[1,2,6]

l2=[1,4,9]

l1.append(l2)

print(l1)

Output:

[1, 2, 6, [1, 4, 9]]

Insert at specific index

Code:

l1.insert(2,"sai") #list\_name.insert(index,element)

print(l1)

Output:

[1, 2, 'sai', 6, [1, 4, 9]]

List Adding

Code:

print(l1+l2)

Output:

[1, 2, 'sai', 6, [1, 4, 9], 1, 4, 9]

List Extend

Code:

a=[1,4,3]

b=[0,8,2]

a.extend(b)

print(a)

Output:

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

List pop

Code:

s=[1,"sasi",3,"lavs",4]

s.pop(3) #list\_name.pop(index\_no)

print(s)

List remove

s.remove(1) #list\_name.remove(member element)

print(s)

Output:

[1, 'sasi', 3, 4]

['sasi', 3, 4]

List max

#min and max works for only numeric data type

Code:

a=[1,2,3,4,50,6,7,8,9]

print(min(a))

print(max(a))

Output:

1

50

**1.9. List Comprehensions:**

* Iterates.
* Applies some function on every element.
* Conditions.
* Output:list.

Code:

l1=[45,67,89,90]

l2=[i\*\*2 for i in l1] #output iterate condition

print(l2)

Output:

[2025, 4489, 7921, 8100]

Code:

l1=[45,67,89,90]

l2=[i\*\*2 for i in l1 if i>50] #output iterate condition

print(l2)

l2=[i\*\*2 for i in l1 if i<60] #output iterate condition

print(l2)

Output:

[4489, 7921, 8100]

[2025]

#the salaries of 5 employees in a company taken as a list .the tax is 10% if the salary is less than or = 50000 or it is %15

#create a new list with tax amounts

#[67000,45000,89000,34000,50000]

#list\_name=[(boby of if) (condition) else (boby of else) iterate]

Code:

sal=[67000,45000,89000,34000,50000]

tax=[]

for i in sal:

if i<=50000:

t=i\*0.1

tax.append(t)

else:

t=i\*0.15

tax.append(t)

print(tax)

Output:

[10050.0, 4500.0, 13350.0, 3400.0, 5000.0]

Another Method:

Code:

sal=[67000,45000,89000,34000,50000]

tax=[i\*0.1 if i<=50000 else i\*0.15 for i in sal]

print(tax)

Output:

[10050.0, 4500.0, 13350.0, 3400.0, 5000.0]

**2.0. Library Function :**

**2.1.Numpy:**

Numpy is a python library used for working with arrays.

* Numpy stands for Numerical Python.
* It also has functions for the working domain of linear algebra,fourier transform and matrices.

#Importing

import numpy as np

**2.2.Arrays :**

Stores the similar type of data.

Types:1D,2D,3D-arrays

Creating 1D array #rows and columns

Code:

import numpy as np

A = np.array([2,3,4,5])

print(type(A))

Output:

<class 'numpy.ndarray'>

Creating 2D array #rows and columns

Code:

import numpy as np

B = np.array([[2,3,4],[7,8,9]])

print(B)

Output:

[[2 3 4]

[7 8 9]]

#Creating 3D array #rows,columns and groups

Code:

C = np.array([[[2,3,4],[6,7,8]],[[6,9,0],[2,3,1]]])

print(C)

Output:

[[[2 3 4]

[6 7 8]]

[[6 9 0]

[2 3 1]]]

Checking dimensions

Code:

print(A.ndim)

print(B.ndim)

print(C.ndim)

Output:

1

2

3

**Ones:**

Ones

Code:

D=np.ones((2,3,3)) #groups,rows,columns,3D

print(D)

Output:

[[[1. 1. 1.]

[1. 1. 1.]

[1. 1. 1.]]

[[1. 1. 1.]

[1. 1. 1.]

[1. 1. 1.]]]

Ones

Code:

E=np.ones((3,2)) #rows,columns,2D

print(E)

Output:

[[1. 1.]

[1. 1.]

[1. 1.]]

**Zeros:**

Zeros

Code:

F=np.zeros((3,2)) #2D

print(F)

Output:

[[0. 0.]

[0. 0.]

[0. 0.]]

**Eye:**

Code:

J=np.eye(4)

print(J)

Output:

[[1. 0. 0. 0.]

[0. 1. 0. 0.]

[0. 0. 1. 0.]

[0. 0. 0. 1.]]

Code:

R=np.eye(4,3)

print(R)

Output:

[[1. 0. 0.]

[0. 1. 0.]

[0. 0. 1.]

[0. 0. 0.]]

**2.3. Arange:**

**Arange** :The arange function in python is used to create a sequence of Numbers with a specified start,stop and step value

Arange

Code:

H=np.arange(3,31,3) #arange(start,stop,step)

print(H)

Output:

[ 3 6 9 12 15 18 21 24 27 30]

**Reshape:**

Used to modify the array.ie,setting the no.of rows and columns.

Arange

Code:

H=np.arange(3,31,3).reshape(5,2) #arange(start,stop,step)

print(H)

Output:

[[ 3 6]

[ 9 12]

[15 18]

[21 24]

[27 30]]

Arange

Code:

H=np.arange(5,1001,5)

print(H)

Output:

[ 5 10 15 20 25 30 35 40 45 50 55 60 65 70

75 80 85 90 95 100 105 110 115 120 125 130 135 140

145 150 155 160 165 170 175 180 185 190 195 200 205 210

215 220 225 230 235 240 245 250 255 260 265 270 275 280

285 290 295 300 305 310 315 320 325 330 335 340 345 350

355 360 365 370 375 380 385 390 395 400 405 410 415 420

425 430 435 440 445 450 455 460 465 470 475 480 485 490

495 500 505 510 515 520 525 530 535 540 545 550 555 560

565 570 575 580 585 590 595 600 605 610 615 620 625 630

635 640 645 650 655 660 665 670 675 680 685 690 695 700

705 710 715 720 725 730 735 740 745 750 755 760 765 770

775 780 785 790 795 800 805 810 815 820 825 830 835 840

845 850 855 860 865 870 875 880 885 890 895 900 905 910

915 920 925 930 935 940 945 950 955 960 965 970 975 980

985 990 995 1000]

Arange

Code:

H=np.arange(5,1001,5).reshape(40,5)

print(H)

Output:

[[ 5 10 15 20 25]

[ 30 35 40 45 50]

[ 55 60 65 70 75]

[ 80 85 90 95 100]

[ 105 110 115 120 125]

[ 130 135 140 145 150]

[ 155 160 165 170 175]

[ 180 185 190 195 200]

[ 205 210 215 220 225]

[ 230 235 240 245 250]

[ 255 260 265 270 275]

[ 280 285 290 295 300]

[ 305 310 315 320 325]

[ 330 335 340 345 350]

[ 355 360 365 370 375]

[ 380 385 390 395 400]

[ 405 410 415 420 425]

[ 430 435 440 445 450]

[ 455 460 465 470 475]

[ 480 485 490 495 500]

[ 505 510 515 520 525]

[ 530 535 540 545 550]

[ 555 560 565 570 575]

[ 580 585 590 595 600]

[ 605 610 615 620 625]

[ 630 635 640 645 650]

[ 655 660 665 670 675]

[ 680 685 690 695 700]

[ 705 710 715 720 725]

[ 730 735 740 745 750]

[ 755 760 765 770 775]

[ 780 785 790 795 800]

[ 805 810 815 820 825]

[ 830 835 840 845 850]

[ 855 860 865 870 875]

[ 880 885 890 895 900]

[ 905 910 915 920 925]

[ 930 935 940 945 950]

[ 955 960 965 970 975]

[ 980 985 990 995 1000]]

**Linspace:**

Code:

V=np.linspace(12,24,10)

print(V)

Output:

[12. 13.33333333 14.66666667 16. 17.33333333 18.66666667

20. 21.33333333 22.66666667 24. ]

Code:

L=np.arange(1,7)

print(L)

Output:

[1 2 3 4 5 6]

**Reshape**:This function is used to reshape an array into a given shape without changing data.

Code:

L=np.arange(1,7).reshape(2,3)

print(L)

Output:

[[1 2 3]

[4 5 6]]

Code:

L=np.arange(1,7).reshape(2,3)

print(L)

R=np.arange(9,15).reshape(2,3)

print(R)

Output:

[[1 2 3]

[4 5 6]]

[[ 9 10 11]

[12 13 14]]

Code:

print(L+R)

Output:

[[10 12 14]

[16 18 20]]

Code:

G=np.sum((L,R))

print(G)

Output:

90

Code:

G=np.sum((L,R),axis=0)

print(G)

Output:

[[10 12 14]

[16 18 20]]

Code:

G=np.sum((L,R),axis=1)

print(G)

Output:

[[ 5 7 9]

[21 23 25]]

Code:

G=np.sum((L,R),axis=2)

print(G)

Output:

[[ 6 15]

[30 39]]

Code:

h=np.ones((4,2))

g=np.ones((4,2))

print(np.sum((h,g),axis=0))

Output:

[[2. 2.]

[2. 2.]

[2. 2.]

[2. 2.]]

Code:

h=np.ones((4,2))

g=np.ones((4,2))

print(np.sum((h,g),axis=1))

Output:

[[4. 4.]

[4. 4.]]

Code:

h=np.ones((4,2))

g=np.ones((4,2))

print(np.sum((h,g),axis=2))

Output:

[[2. 2. 2. 2.]

[2. 2. 2. 2.]]

Code:

A=np.array([[1,1],[0,1]])

B=np.array([[2,0],[3,4]])

print(A\*B)

Output:

[[2 0]

[0 4]]

Code:

print(A@B)

Output:

[[5 4]

[3 4]]

b=np.array[25,289,361,81] #find square roots and iterate through result value output : 5 square is 25

Code:

b=np.array([25,289,361,81])

for i in b:

print(np.sqrt(i),"square of",i)

Output:

b=np.array([25,289,361,81])

for i in b:

print(np.sqrt(i),"square of",i)

**2.4. Array joining :**

#Array\_joins

Code:

a=np.array([34,35,36,37,38,39])

a.resize(2,3)

b=np.array([4,5,6,7,8,9])

b.resize(2,3)

print(np.vstack((a,b))) #columns

print("\n")

print(np.hstack((a,b))) #rows

Output:

[[34 35 36]

[37 38 39]

[ 4 5 6]

[ 7 8 9]]

[[34 35 36 4 5 6]

[37 38 39 7 8 9]]

#array\_joins

Code:

a=np.array([34,35,36,37,38,39])

a.resize(2,3)

b=np.array([4,5,6,7,8,9])

b.resize(2,3)

print(a)

print(b)

Output:

[[34 35 36]

[37 38 39]]

[[4 5 6]

[7 8 9]]

**dstack :** it is used to stack arrays in sequence depth wise.

Code:

a=np.arange(30).reshape(2,3,5)

print(a)

print("\n")

print(np.dstack(a))

#number of rows becomes number of groups

#columns becomes rows

#group becomes columns

Output:

[[[ 0 1 2 3 4]

[ 5 6 7 8 9]

[10 11 12 13 14]]

[[15 16 17 18 19]

[20 21 22 23 24]

[25 26 27 28 29]]]

[[[ 0 15]

[ 1 16]

[ 2 17]

[ 3 18]

[ 4 19]]

[[ 5 20]

[ 6 21]

[ 7 22]

[ 8 23]

[ 9 24]]

[[10 25]

[11 26]

[12 27]

[13 28]

[14 29]]]

importing

import numpy as np

Code:

import numpy as np

a=np.ones((8,4))

print(a)

Output:

[[1. 1. 1. 1.]

[1. 1. 1. 1.]

[1. 1. 1. 1.]

[1. 1. 1. 1.]

[1. 1. 1. 1.]

[1. 1. 1. 1.]

[1. 1. 1. 1.]

[1. 1. 1. 1.]]

#random module is sub-package of numpy

Code:

A=np.random.rand(1)

print(A)

Output:

[0.48129352]

Code:

A=np.random.rand(8,4) #rand-range is between 0 and 1

print(A)

Output:

[[0.53901421 0.50592912 0.73483082 0.59349801]

[0.56743961 0.52816629 0.85009268 0.18914805]

[0.480714 0.72974503 0.32252532 0.15773461]

[0.42252389 0.60638159 0.50548979 0.62641512]

[0.47580072 0.62903925 0.60093666 0.33920848]

[0.0866525 0.05458333 0.5763455 0.56808612]

[0.49506289 0.23786728 0.08967322 0.93061835]

[0.35640194 0.56047029 0.80635368 0.284242 ]]

Code:

A=np.floor(10\*np.random.rand(8,4)) #rand-range is between 0 and 1

print(A)

Output:

[[8. 2. 5. 8.]

[0. 5. 8. 0.]

[0. 9. 9. 9.]

[4. 2. 2. 4.]

[8. 3. 0. 6.]

[4. 2. 3. 3.]

[8. 5. 6. 1.]

[3. 2. 8. 5.]]

Code:

A=np.arange(1,33).reshape(8,4)

print(A)

Output:

[[ 1 2 3 4]

[ 5 6 7 8]

[ 9 10 11 12]

[13 14 15 16]

[17 18 19 20]

[21 22 23 24]

[25 26 27 28]

[29 30 31 32]]

**2.5. Vsplit:**

Vsplit function is used to split the array into two subarrays along the vertical axis(rows).

#Splitting of an array

Code:

print(np.vsplit(A,4))

Output:

[array([[1, 2, 3, 4],

[5, 6, 7, 8]]), array([[ 9, 10, 11, 12],

[13, 14, 15, 16]]), array([[17, 18, 19, 20],

[21, 22, 23, 24]]), array([[25, 26, 27, 28],

[29, 30, 31, 32]])]

Code:

print(np.vsplit(A,4))

Output:

[array([[1, 2, 3, 4],

[5, 6, 7, 8]]), array([[ 9, 10, 11, 12],

[13, 14, 15, 16]]), array([[17, 18, 19, 20],

[21, 22, 23, 24]]), array([[25, 26, 27, 28],

[29, 30, 31, 32]])]

Code:

print(np.vsplit(A,(3,5))) #split it after 3rd row and 5th row

Output:

[array([[ 1, 2, 3, 4],

[ 5, 6, 7, 8],

[ 9, 10, 11, 12]]), array([[13, 14, 15, 16],

[17, 18, 19, 20]]), array([[21, 22, 23, 24],

[25, 26, 27, 28],

[29, 30, 31, 32]])]

**=>In case of wrong division,it gives an error.**

**np.vsplit(A,3)**

**ERROR:**

**---------------------------------------------------------------------------**

**ValueError Traceback (most recent call last)**

[**<ipython-input-10-bf600901fad4>**](https://localhost:8080/#) **in <cell line: 1>()**

**----> 1 np.vsplit(A,3)**

[**/usr/local/lib/python3.10/dist-packages/numpy/lib/shape\_base.py**](https://localhost:8080/#) **in split(ary, indices\_or\_sections, axis)**

**870 N = ary.shape[axis]**

**871 if N % sections:**

**--> 872 raise ValueError(**

**873 'array split does not result in an equal division') from None**

**874 return array\_split(ary, indices\_or\_sections, axis)**

**ValueError: array split does not result in an equal division**

Code:

A=np.arange(1,33).reshape(4,8)

print(A)

Output:

[[ 1 2 3 4 5 6 7 8]

[ 9 10 11 12 13 14 15 16]

[17 18 19 20 21 22 23 24]

[25 26 27 28 29 30 31 32]]

**2.6. hsplit:**

split function is used to split the array into two subarrays along the horizontal axis(columns).

Code:

print(np.hsplit(A,4))

Output:

[array([[ 1, 2],

[ 9, 10],

[17, 18],

[25, 26]]), array([[ 3, 4],

[11, 12],

[19, 20],

[27, 28]]), array([[ 5, 6],

[13, 14],

[21, 22],

[29, 30]]), array([[ 7, 8],

[15, 16],

[23, 24],

[31, 32]])]

Code:

print(np.hsplit(A,(3,4)))

Output:

[array([[ 1, 2, 3],

[ 9, 10, 11],

[17, 18, 19],

[25, 26, 27]]), array([[ 4],

[12],

[20],

[28]]), array([[ 5, 6, 7, 8],

[13, 14, 15, 16],

[21, 22, 23, 24],

[29, 30, 31, 32]])]

**2.7. Trigonometry :**

Code:

np.pi

Output:

3.141592653589793

Code:

A=[np.pi/4,np.pi/3,np.pi/2,np.pi]

print(A)

Output:

[0.7853981633974483, 1.0471975511965976, 1.5707963267948966, 3.141592653589793]

Code:

np.rad2deg(A) #convert radians to degrees

Output:

array([ 45., 60., 90., 180.])

Code:

A=[45,60,90,180]

np.deg2rad(A)

Output:

array([0.78539816, 1.04719755, 1.57079633, 3.14159265])

Code:

A=np.sin(1)

print(A)

B=np.cos(1)

print(B)

C=np.tan(1)

print(C)

Output:

**0.8414709848078965**

**0.5403023058681398**

**1.5574077246549023**

**Statistics:**

Code:

st=np.array([23,45,67,89,21,34])

Output:

46.5

Code:

np.median(st)

Output:

39.5

Code:

np.std(st)

Output:

24.452334585202017

Code:

np.var(st)

Output:

597.9166666666666

Code:

C=np.arange(1,5).reshape(2,2)

print(C)

Output:

[[1 2]

[3 4]]

Code:

np.linalg.inv(C) #Sub-package of numpy inverse

Output:

array([[-2. , 1. ],

[ 1.5, -0.5]])

**Max & Min function()**:It is used to find the maximum element in the matrix.

Code:

C=np.arange(1,25).reshape(6,4)

print(C)

print(np.argmax(C))

Output:

[[ 1 2 3 4]

[ 5 6 7 8]

[ 9 10 11 12]

[13 14 15 16]

[17 18 19 20]

[21 22 23 24]]

23

Code:

C=np.floor(10\*np.random.rand(24).reshape(6,4))

print(C)

print(np.argmax(C))

Output:

[[8. 7. 1. 3.]

[7. 2. 3. 1.]

[4. 9. 2. 8.]

[5. 9. 2. 2.]

[6. 8. 6. 2.]

[1. 0. 2. 4.]]

9

Code:

C=10\*np.random.rand(24).reshape(6,4)

print(C)

print(np.argmax(C))

Output:

[[6.79298645 4.45859517 4.01093878 6.9013812 ]

[6.26746355 7.64832734 3.62342629 1.31328202]

[1.90090825 4.01430464 3.04605398 9.33828239]

[1.24175377 6.08825543 7.45839796 4.03983079]

[0.18368667 1.13323821 7.37823101 5.79780023]

[2.84785532 9.31065779 0.09285886 1.91563997]]

11

Code:

print(np.argmax(C,axis=1)) #rows

Output:

[3 1 3 2 2 1]

Code:

print(np.argmax(C,axis=0)) #columns

Output:

[0 5 3 2]

Code:

print(np.argmin(C))

Output:

22

Code:

print(np.argmin(C,axis=0))

Output:

[4 4 5 1]

Code:

print(np.argmin(C,axis=1))

Output:

[0 1 3 2 1 2]

**Searching :**

#find the indexes where the values are even:

Code:

a=np.array([34,56,7,17,88,91])

print(np.where(a%2==0))

Output:

(array([0, 1, 4]),)

#used only for already sorted list:

Code:

a=np.array([6,7,8,9,10])

x=np.searchsorted(a,10)

print(x)

Output:

4

#used only for already unsorted list:

Code:

a=np.array([6,2,7,9,8,9,10])

x=np.searchsorted(a,3) #it is not applicable for unsorted list

print(x)

Output:

2

#find the indexes which are divisible by 6.

Code:

a=np.array([24,16,7,17,54,60])

print(np.where(a%6==0))

print(a)

Output:

(array([0, 4, 5]),)

[24 16 7 17 54 60]

**Sortings:**

**#Sorting array**

Code:

arr=np.array(['banana','cherry','apple'])

print(np.sort(arr))

Output:

['apple' 'banana' 'cherry']

Code:

arr=np.array([True,False,True]) #FALSE -0 ,TRUE-0

print(np.sort(arr))

Output:

[False True True]

#2d sort -sorting happens within row

Code:

arr=np.array([[3,2,4],[5,0,1]])

print(np.sort(arr))

Output:

[[2 3 4]

[0 1 5]]

**2.8. Filtering:**

filter() function is to process an iterable and extract those items that satisfy a given condition.

**Array Filters:**

**#array filter**

Code:

arr=np.array([40,42,50,44,67,78])

x=[True,False,True,False,True,True] #filter list

newarr=arr[x]

print(newarr)

Output:

[40 50 67 78]

Code:

arr=np.array([40,42,50,44,67,78]) #filter may be list or array

filt=np.where(arr%2==0)

print(filt)

Output:

(array([0, 1, 2, 3, 5]),)

Code:

names=np.array(["Sasi","Prasanna","Akhila","Lavanya","Jaya","Dharani"]) #Zip function is used to combine two or more arrays and lists

initials=np.array(["P","N","T","P","T","K"])

for i ,j in zip(initials,names):

print(i,".",j)

Output:

P . Sasi

N . Prasanna

T . Akhila

P . Lavanya

T . Jaya

K . Dharani

Code:

names=np.array(["Sasi","Prasanna","Akhila","Lavanya","Jaya","Dharani"]) #Zip function is used to combine two or more arrays and lists

initials=np.array(["P","N","T","P","T","K"])

rollno=np.array(["5b0","5a6","5c0","5b1","560","5b2"])

for i,j,k in zip(initials,names,rollno):

print(i,".",j,".",k)

Output:

P . Sasi . 5b0

N . Prasanna . 5a6

T . Akhila . 5c0

P . Lavanya . 5b1

T . Jaya . 560

K . Dharani . 5b2

Code:

a1=np.array([10,20,30,40,50,60])

a2=np.array([20,21,22,23,24,25])

a=np.multiply(a1,a2)

print(a)

b=np.divide(a1,a2)

print(b)

c=np.mod(a1,a2)

print(c)

d=np.divmod(a1,a2)

print(d)

Output:

[ 200 420 660 920 1200 1500]

[0.5 0.95238095 1.36363636 1.73913043 2.08333333 2.4 ]

[10 20 8 17 2 10]

(array([0, 0, 1, 1, 2, 2]), array([10, 20, 8, 17, 2, 10]))

Code:

a1=np.array([10,20,30,40,50,60])

a2=np.array([20,21,22,23,24,25])

a=np.multiply(a1,a2)

print(a)

Output:

[ 200 420 660 920 1200 1500]

Code:

a1=np.array([10,20,30,40,50,60])

a2=np.array([20,21,22,23,24,25])

b=np.divide(a1,a2)

print(b)

Output:

[0.5 0.95238095 1.36363636 1.73913043 2.08333333 2.4 ]

Code:

a1=np.array([10,20,30,40,50,60])

a2=np.array([20,21,22,23,24,25])

c=np.mod(a1,a2)

print(c)

Output:

[10 20 8 17 2 10]

Code:

a1=np.array([10,20,30,40,50,60])

a2=np.array([20,21,22,23,24,25])

d=np.divmod(a1,a2)

print(d)

Output:

(array([0, 0, 1, 1, 2, 2]), array([10, 20, 8, 17, 2, 10]))

**2.9. Logarithms**:

Code:

a=1.2

print(np.log(a)) #natural log e

Output:

0.1823215567939546

Code:

a=1.2

print(np.log2(a)) #log base 2

Output:

0.2630344058337938

Code:

a=1.2

print(np.log10(a)) #log base 10

Output:

0.07918124604762482

Code:

a=np.array([1,1.2,3,4])

print(np.log(a)) #natural log e

Output:

[0. 0.18232156 1.09861229 1.38629436]

Code:

a1=np.arange(1,10)

print(a1)

print(np.log10(a1)) #base 10

Output:

[1 2 3 4 5 6 7 8 9]

[0. 0.30103 0.47712125 0.60205999 0.69897 0.77815125

0.84509804 0.90308999 0.95424251]

**Other Mathematical Operations:**

Code:

a2=np.array([5,6,7,10])

x=np.cumprod(a2)

print(x)

Output:

[ 5 30 210 2100]

Code:

a2=np.array([5,6,7,10])

x=np.cumsum(a2)

print(x)

Output:

[ 5 11 18 28]

Code:

a1=np.array([10,15,25,15]) #e2-e1

newarr=np.diff(a2)

print(newarr)

Output:

[ 5 10 -10]

Code:

num1=455

num2=665

x=np.lcm(num1,num2)

print(x)

Output:

8645

Code:

num1=455

num2=665

x=np.gcd(num1,num2)

print(x)

Output:

35

Code:

a=np.array([12,15,60])

gc=np.gcd.reduce(a) #takes multiple inputs gives single output

print(gc)

Output:

3

Code:

a=np.array([12,15,60])

lc=np.lcm.reduce(a) #takes multiple inputs gives single output

print(lc)

Output:

60

**#If we give an array of elements to find gcd it gives an error**

**=>a=np.array([12,15,60])**

**gc=np.gcd(a) #takes multiple inputs gives single output**

**print(gc)**

**ERROR:**

**TypeError Traceback (most recent calllast)**

[**<ipython-input-86-3894ce52874b>**](https://localhost:8080/#) **in <cell line: 2>()**

**1 a=np.array([12,15,60])**

**----> 2 gc=np.gcd(a) #takes multiple inputs gives single output**

**3 print(gc)**

**TypeError: gcd() takes from 2 to 3 positional arguments but 1 were given**

**2.10. Matplotlib :**

Matplotlib is a cross\_platform,data visualisation and graphical plotting library for python and its numerical extension numpy.

**Plotting:**

import matplotlib.pyplot as plt

Code:

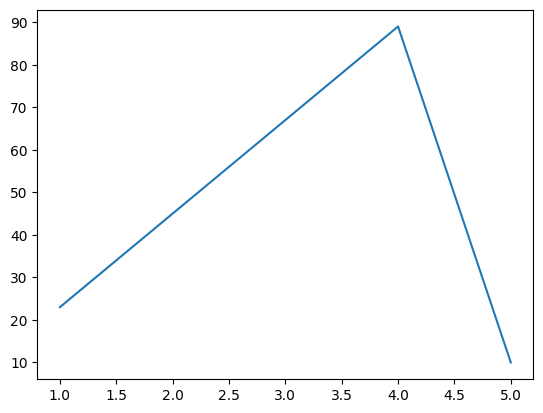
A=[23,45,67,89,10] #corona cases in first five days

B=[1,2,3,4,5]

plt.plot(B,A) #plot(x,y)

plt.show()

Output:



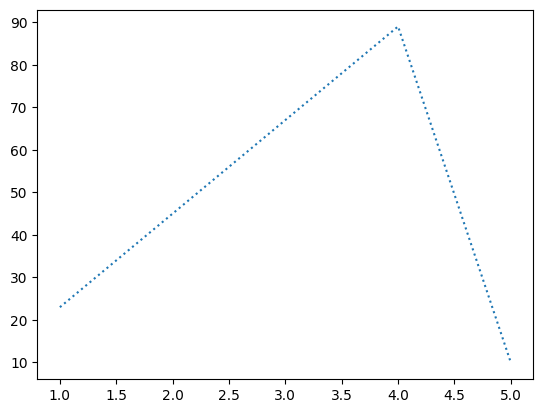
Code:

A=[23,45,67,89,10] #corona cases in first five days

B=[1,2,3,4,5]

plt.plot(B,A,linestyle=":") #:,.

plt.show()

Output:

Problem 1.

Runs scored by 10 new players[100,50,91,78,89,25,34,19,9,10] wickets taken by same 10 new players[1,0,2,0,3,7,8,9,7,5].Form clusters for batsmen and bowlers.

Code:

import numpy as np

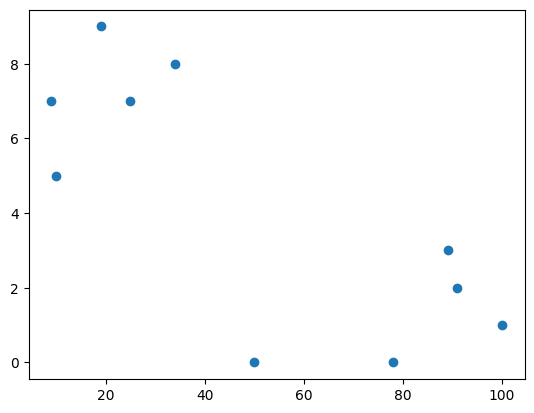
runs=np.array([100,50,91,78,89,25,34,19,9,10])

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

plt.scatter(runs,wic)

plt.show()

Output:



Code:

import numpy as np

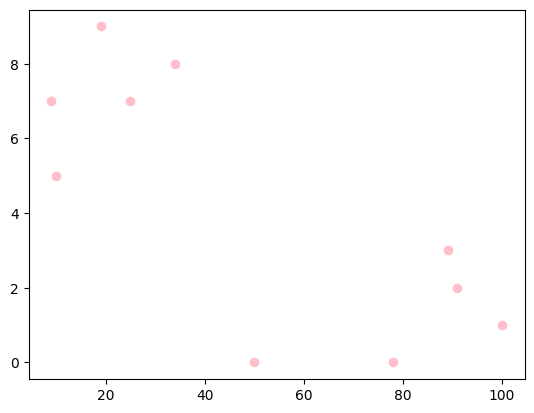
runs=np.array([100,50,91,78,89,25,34,19,9,10])

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

plt.scatter(runs,wic,color="pink") #r,b,k,g,m,w,c,y,pink,orange...-colors

plt.show()

Output:



Code:

# Marker-D,d,p,H,s,+,X,x,,,0,\*,h,v,^,<,>

import numpy as np

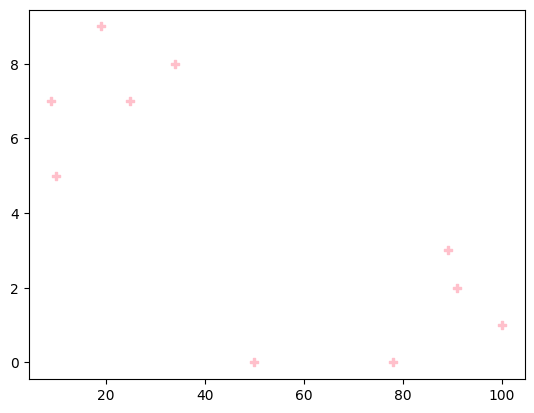
runs=np.array([100,50,91,78,89,25,34,19,9,10])

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

plt.scatter(runs,wic,color="pink", marker="P")

plt.show()

Output:



Program2:

Plot the Scores of 2 students in 5 different subjects . subjects as x axis and marks as y axis

Code:

Stu1=[56,78,92,95,78]

Stu2=[77,89,100,73,45]

sub=["maths","science","social","english","telugu"]

plt.plot(sub,Stu1,label="Student1")

plt.plot(sub,Stu2,label="Student2")

plt.legend()

Output:

<matplotlib.legend.Legend at 0x78054e6c8250>



Code:

#sub -plot

Stu1=[56,78,92,95,78]

Stu2=[77,89,100,73,45]

sub=["maths","science","social","english","telugu"]

plt.subplot(2,1,1)

plt.plot(sub,Stu1,label="Student1",color="m")

plt.legend()

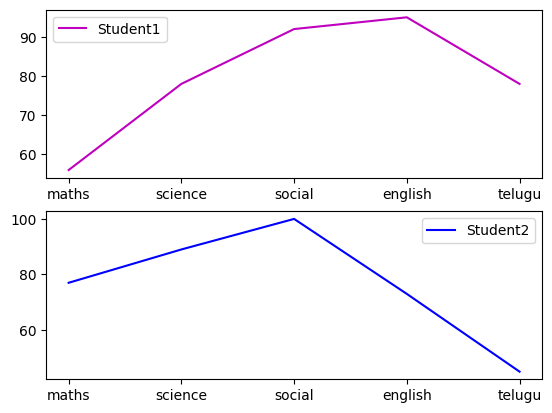
plt.subplot(2,1,2)

plt.plot(sub,Stu2,label="Student2",color="b")

plt.legend()

Output:

<matplotlib.legend.Legend at 0x78054e0c8310>



Problem3:

Create Subplots(b←) showing the profits of two companies. year. A B

1. 230 200
2. 560 160
3. 780 270
4. 127 127
5. 128 400

Code:

Com1=np.array([230,560,780,127,128])

Com2=np.array([200,160,270,127,400])

print("revenue of company ",Com1)

print("revenue of company ",Com2)

years=["19-20","20-21","21-22","22-23"]

profitA=np.diff(Com1)

print(profitA)

profitB=np.diff(Com2)

print(profitB)

plt.subplot(1,2,1)

plt.bar(years,profitA,label="Company1",color="k")

plt.legend(loc="best")

plt.subplot(1,2,2)

plt.bar(years,profitB,label="Company2",color="g")

plt.legend(loc="best")

plt.show()

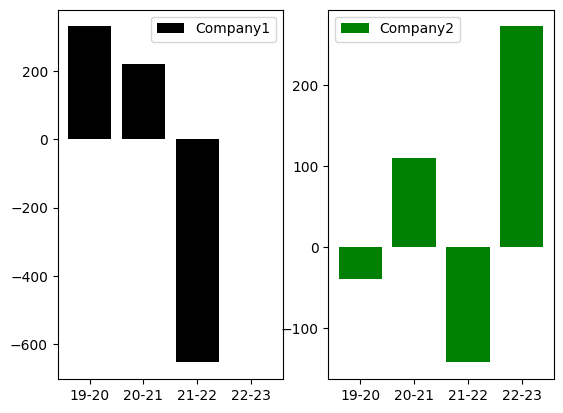
Output:

revenue of company [230 560 780 127 128]

revenue of company [200 160 270 127 400]

[ 330 220 -653 1]

[ -40 110 -143 273]



Code:

#pie chart

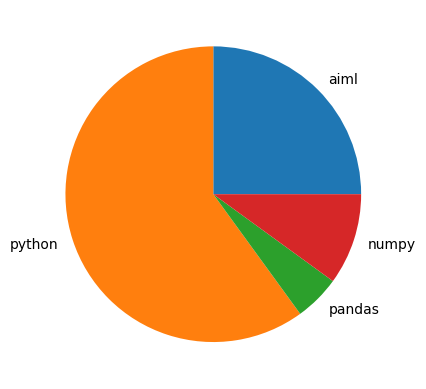
a=np.array([25,60,5,10])

labe=["aiml","python","pandas","numpy"]

plt.pie(a,labels = labe)

plt.show()

Output:



Code:

#shadow

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

labe=["aiml","python","pandas","numpy"]

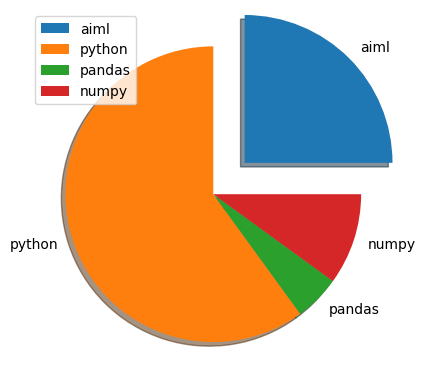
explo=[0.3,0,0,0]

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

plt.legend()

plt.show()

Output:



**3. Pandas:**

* Used for data manipulation.
* Create dataframes from excel,csv,txt,DBs.
* Dataframes(rows and columns readable by python).
* Data cleaning by dropping or replacing with mean.
* Visualise the data.

**3.1.Importing:**

Code:

#import pandas

import pandas as pd

**SERIES:**

Code:

a=["sasi","lavs","prasanna","akhila"]

index=[34,40,35,23]

ser1=pd.Series(a,index)

print(ser1)

Output:

34 sasi

40 lavs

35 prasanna

23 akhila

dtype: object

**Importing Files:**

* For csv and txt :read\_csv
* For excel : read\_excel

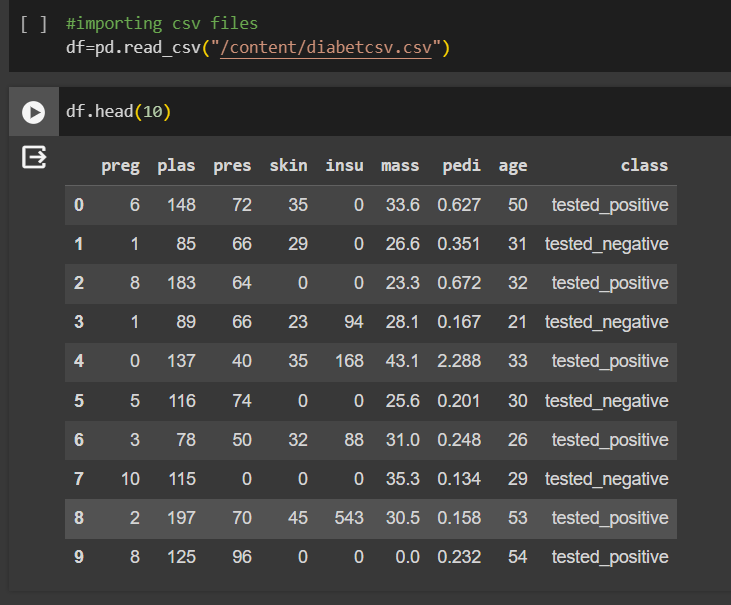
Code:

#importing csv files

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

df.head(10)

Output:



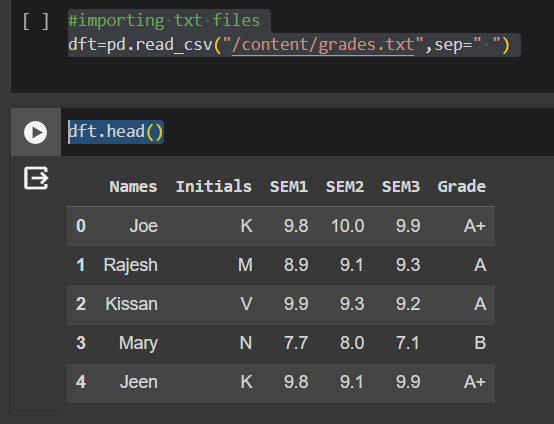
Code:

#importing text files

dft=pd.read\_csv("/content/grades.txt",sep=" ")

dft.head()

Output:



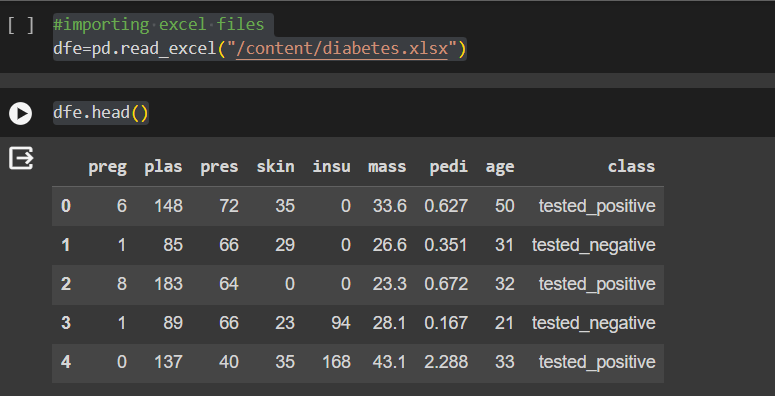
Code:

#importing excel files

dfe=pd.read\_excel("/content/diabetes.xlsx")

dfe.head()

Output:



**3.2. Describe**:

Code:

#describing the file (txt)

print(dft.describe)

Output:

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

0 Joe K 9.8 10.0 9.9 A+

1 Rajesh M 8.9 9.1 9.3 A

2 Kissan V 9.9 9.3 9.2 A

3 Mary N 7.7 8.0 7.1 B

4 Jeen K 9.8 9.1 9.9 A+

5 Raj M 8.9 9.1 9.3 A

6 Hassan V 9.9 9.0 9.2 A

7 Mari N 7.7 8.0 7.1 B

**8 Jess K 9.8 9.1 9.9 A+**

**9 Rajini M 7.0 9.1 9.3 A**

**10 Kiran V 9.9 9.3 9.2 A**

**11 Maya N 7.7 8.0 7.1 B**

**12 Jolin K 9.8 9.1 9.9 A+**

**13 Riya M 8.0 9.1 9.3 A**

**14 Sana V 9.9 9.3 9.2 A**

**15 Mark N 7.7 8.0 7.0 B>**

Code:

#describing the file (excel)

print(dfe.describe)

Output:

<bound method NDFrame.describe of 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]>

Code:

#shape of the file

dfe=pd.read\_excel("/content/diabetes.xlsx")

print(dfe.shape) #get the number of rows and columns

print(dfe.shape[0]) #get the number of rows only

print(dfe.shape[1]) #get the number of columns only

Output:

(768, 9)

768

9

**3.3. Accessing data:**

* **loc-accepts column names and index**
* **iloc-accepts only index**

Code:

print(dft.columns) # access columns

Output:

Index(['Names', 'Initials', 'SEM1', 'SEM2', 'SEM3', 'Grade'], dtype=’object')

Code:

print(dft[2:5]) #to access row

Output:

Names Initials SEM1 SEM2 SEM3 Grade

2 Kissan V 9.9 9.3 9.2 A

3 Mary N 7.7 8.0 7.1 B

4 Jeen K 9.8 9.1 9.9 A+

Code:

print(dft.loc[2:5,"Names"]) #row of specified column

Output:

2 Kissan

3 Mary

4 Jeen

5 Raj

Name: Names, dtype: object

Code:

print(dft.iloc[2:5]) #row of specified column

Output:

Names Initials SEM1 SEM2 SEM3 G**rade**

2 Kissan V 9.9 9.3 9.2 A

3 Mary N 7.7 8.0 7.1 B

4 Jeen K 9.8 9.1 9.9 A+

Code:

print(dft.iloc[2:5, :3]) #iloc[row range,column range] =>index

Output:

Names Initials SEM1

2 Kissan V 9.9

3 Mary N 7.7

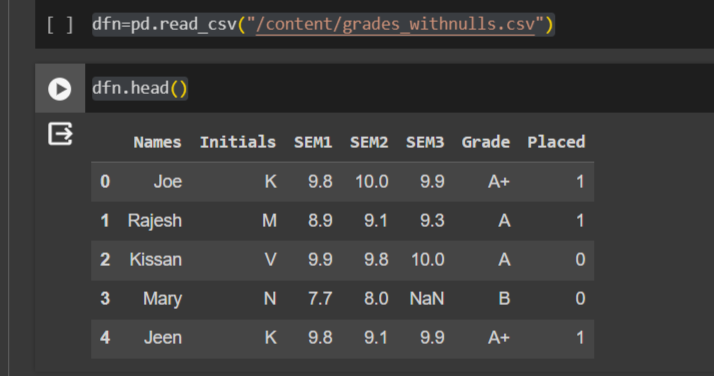
4 Jeen K 9.8

Code:

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

dfn.head()

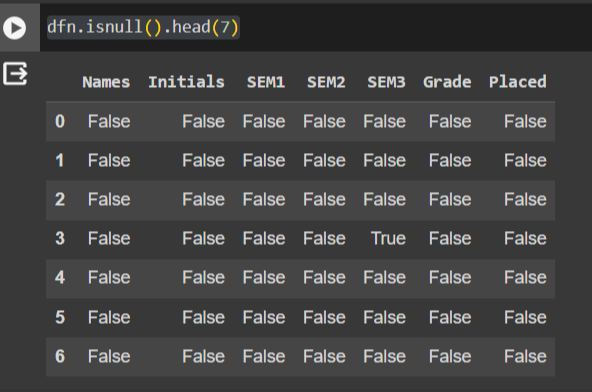
Output:



Code:

dfn.isnull().head(7)

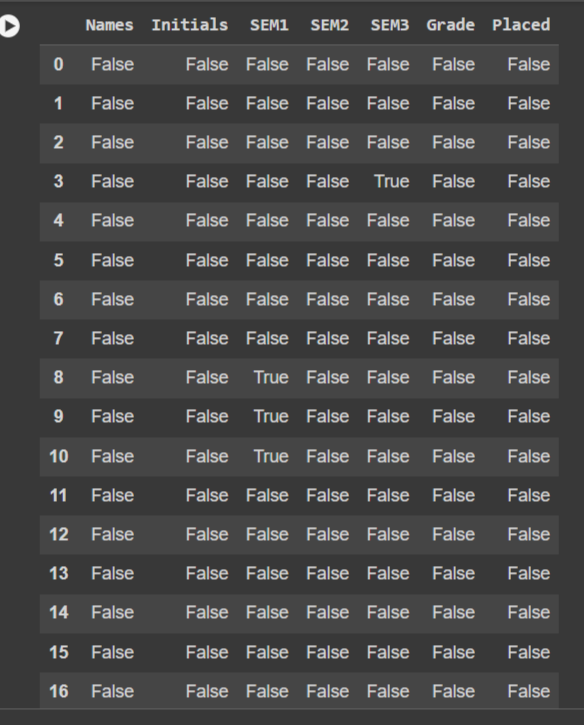
Output:



Code:

dfn.isnull()

Output:



Code:

dfn.isnull().sum() #to view how many nulls we have per column

Output:

Names 0

Initials 0

SEM1 3

SEM2 0

SEM3 1

Grade 0

Placed 0

dtype: int64

Code:

dfn.isnull().sum().sum() #to view total nulls

Output:

4

**3.4.dropna() :**  It is a function used to remove rows or columns with missing values ( NaN ) from a DataFrame

Code:

#dropping all the rows with nulls

dfn.dropna()

Output:



Code:

dfc=dfn.dropna()

print(dfc)

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

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

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

Code:

print(dfn)

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

**3.5 fullna():**

You can use the fillna() function to fill the null values in the dataset.

Code:

dfc1=dfn.fillna(5)

print(dfc1)

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 5.0 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 5.0 9.1 9.9 A+ 1

9 Rajini M 5.0 9.1 9.3 A 0

10 Kiran V 5.0 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

**3.6. Cleaning with mean:**

Code:

m=dfn['SEM3'].mean()

print(m)

Output:

9.100000000000001

Code:

dfc2=dfn.fillna(m)

print(dfc2)

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 9.1 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 9.1 9.1 9.9 A+ 1

9 Rajini M 9.1 9.1 9.3 A 0

10 Kiran V 9.1 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

**3.7. Dropping Duplicates:**

The drop\_duplicates() method removes duplicate rows**.**

Code:

dfc3=dfc2.drop\_duplicates()

print(dfc3)

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 9.1 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 9.1 9.1 9.9 A+ 1

9 Rajini M 9.1 9.1 9.3 A 0

10 Kiran V 9.1 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

14 Riya M 9.3 9.9 10.0 A 1

15 Sana V 9.9 9.3 9.2 A 0

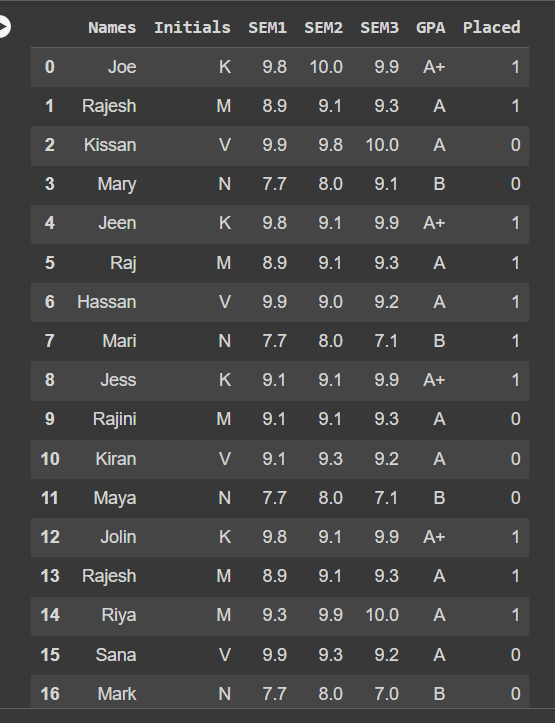
16 Mark N 7.7 8.0 7.0 B 0

**Columns:**

Code:

dfc2.rename(columns={"Grade":"GPA"}) #df.rename(column={old:new})

Output:



Code:

dfc2.head()

Output:



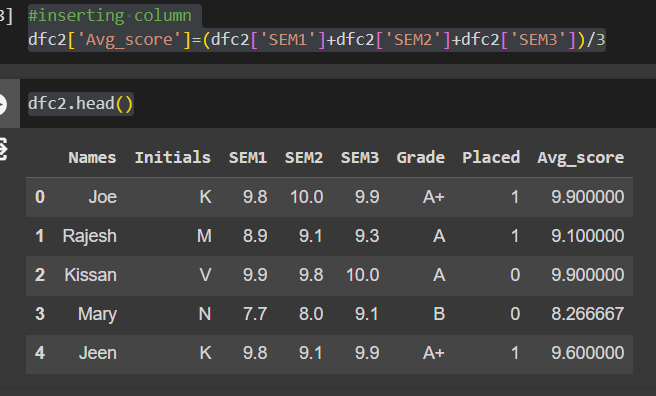
Code:

#inserting column

dfc2['Avg\_score']=(dfc2['SEM1']+dfc2['SEM2']+dfc2['SEM3'])/3

dfc2.head()

Output:



**3.8. Plotting :**

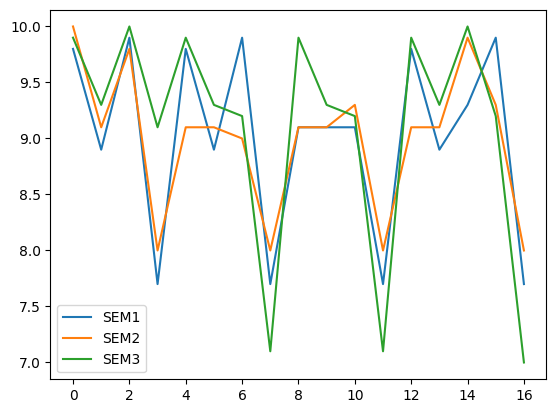
Code:

#plotting

dfc2[['SEM1','SEM2','SEM3']].plot.line()

Output:

<Axes: >



**3.9. Subplots:**

Code:

dfc2.plot.line(subplots=True)

Output:

array([<Axes: >, <Axes: >, <Axes: >, <Axes: >, <Axes: >], dtype=object)

**4.Seaborn:**

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

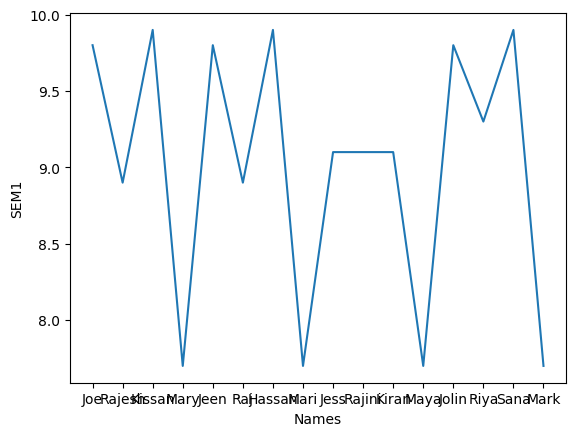
**4.1. Importing :**

Code:

import seaborn as sns

pl=sns.lineplot(x='Names',y='SEM1',data=dfc2)

Output:



**4.2. Relplot:**

The Seaborn Relational Plot (relplot) allows us to visualise how variables within a dataset relate to each other.

Code:

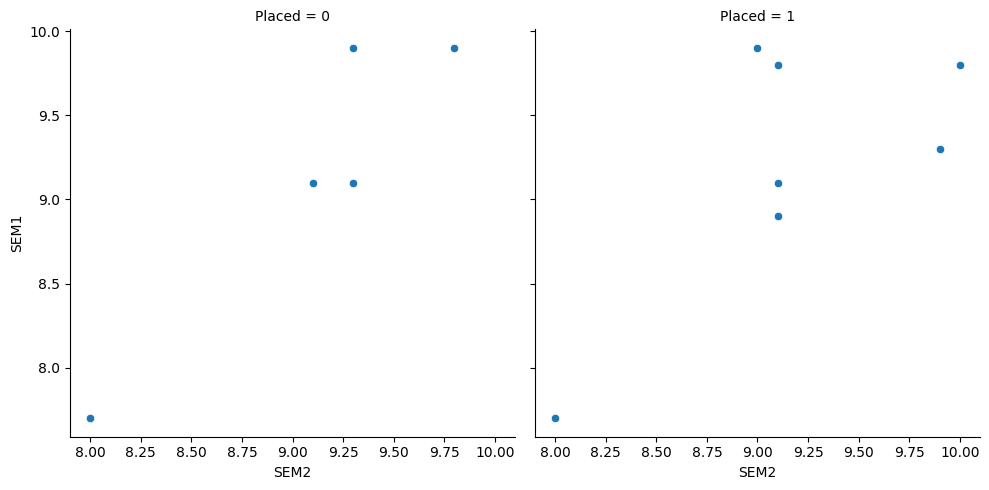
sns.relplot(

data=dfc2,

x="SEM2",y="SEM1",col="Placed")

Output:

<seaborn.axisgrid.FacetGrid at 0x788f2ccaaf80>

****

1. Load diabets.csv.
2. Create a relplot with age in the x axis and class as columns.

Code:

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

df['Index']=(range(0,768))

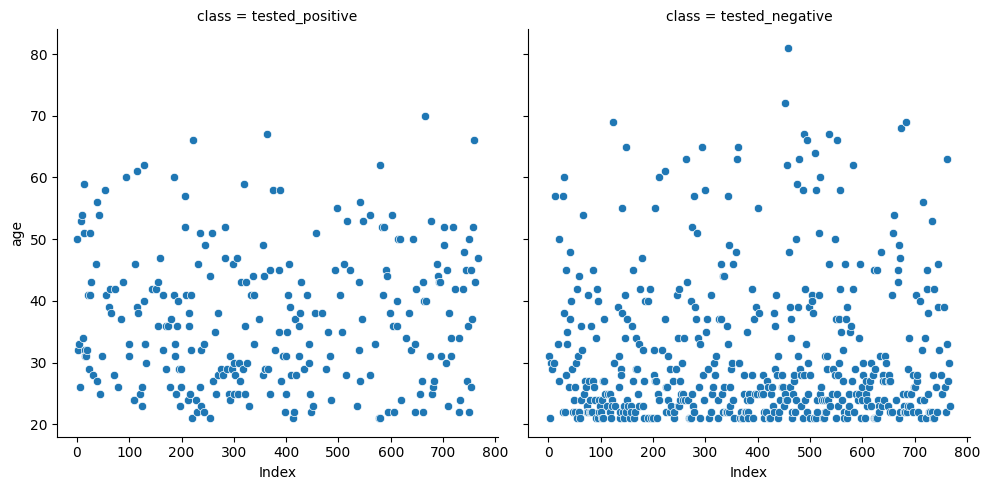
sns.relplot(

data=df,

x="Index",y="age",col="class")

Output:

<seaborn.axisgrid.FacetGrid at 0x788f2cb47a30>



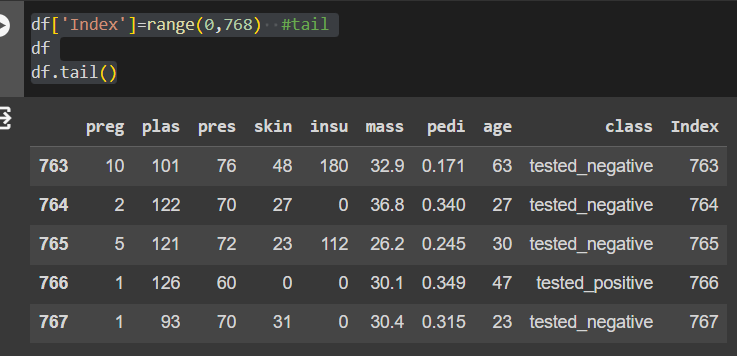
Code:

df['Index']=range(0,768) #tail

df

df.tail()

Output:



In-built datasets in seaborn

* tips
* dowjones
* fmri
* dots
* health exp
* To load datasets use:=>load\_datasets("dataset\_name")

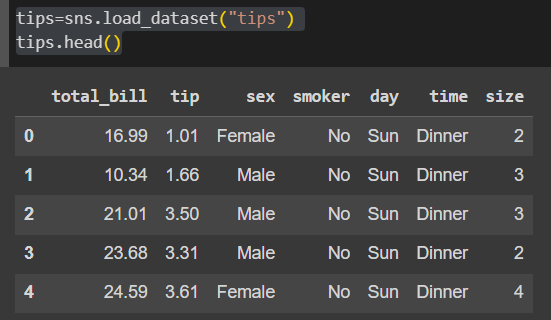
hue = different colour for diff category style=different colour for diff marker

Code:

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

tips.head()

Output:

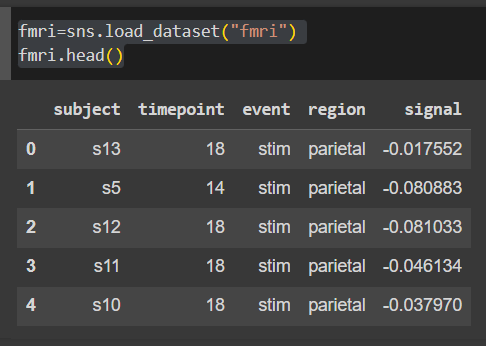


Code:

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

fmri.head()

Output:

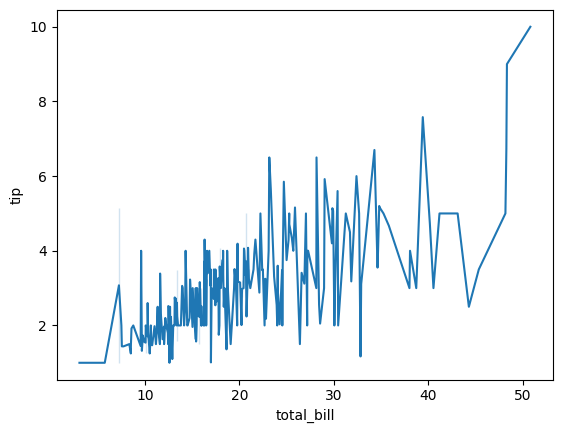


Code:

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

tips=sns.lineplot(x='total\_bill',y='tip',data=tips)

Output:



Code:

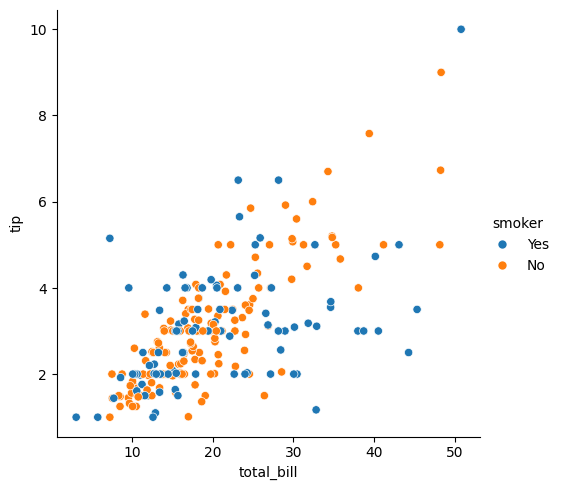
tips=sns.load\_dataset("tips")

sns.relplot(data=tips,x='total\_bill',y='tip',hue="smoker")

#hue- creating difference based on the column via colors

Output:

<seaborn.axisgrid.FacetGrid at 0x788f2c814970>



Code:

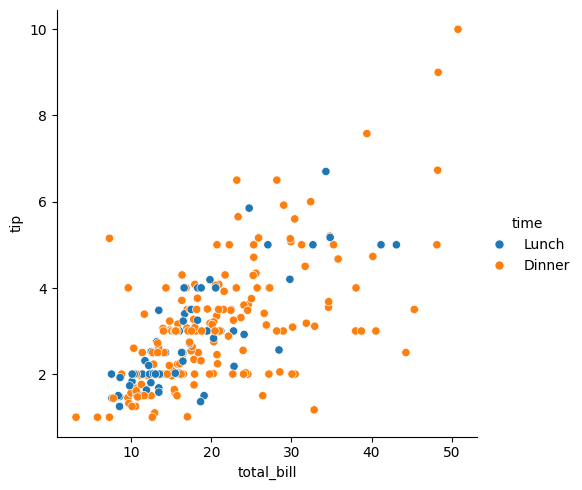
tips=sns.load\_dataset("tips")

sns.relplot(data=tips,x='total\_bill',y='tip',hue="time")

#hue- creating difference based on the column via time

Output:

<seaborn.axisgrid.FacetGrid at 0x788f2c5cab00

****

* subject means patient number
* region=brain location

Code:

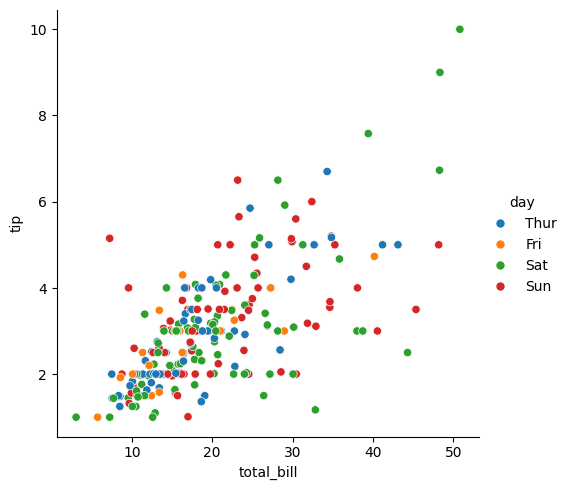
tips=sns.load\_dataset("tips")

sns.relplot(data=tips,x='total\_bill',y='tip',hue="day")

#hue- creating difference based on the column via day

Output:

<seaborn.axisgrid.FacetGrid at 0x788f38a09600>

****

* hue-> different color for diff cat
* style-> different color for diff marker

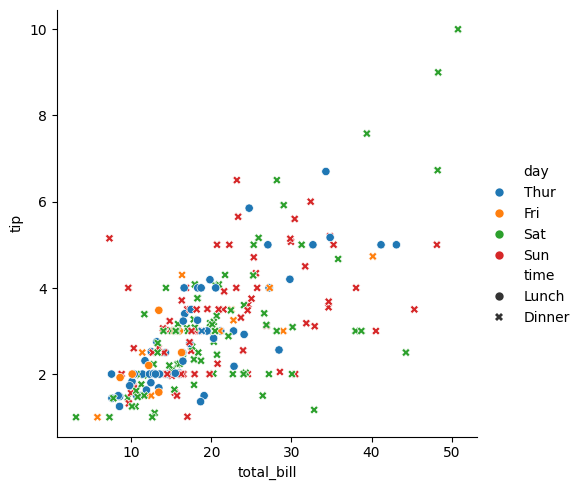
Code:

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

sns.relplot(data=tips,x='total\_bill',y='tip',hue="day",style="time")

Output:

<seaborn.axisgrid.FacetGrid at 0x788f2c4b1cc0>

****

**Color Palette:**

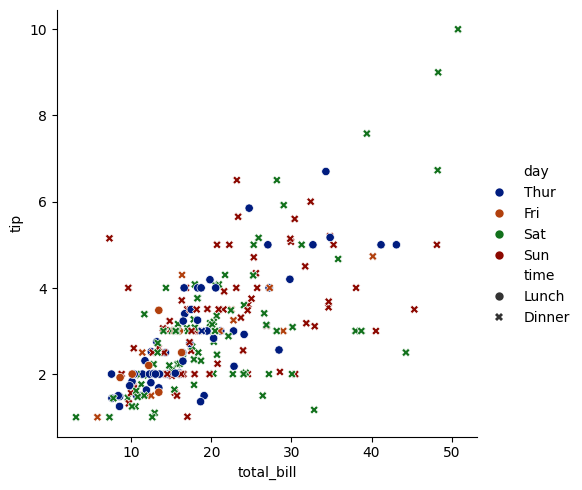
* pastel
* bright
* dark
* muted
* color blind
* Deep

Code:

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

sns.relplot(data=tips,x='total\_bill',y='tip',hue="day",style="time",palette="dark")

Output:

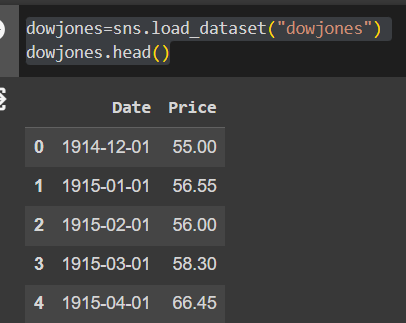
<seaborn.axisgrid.FacetGrid at 0x788f2c273dc0>

Code:

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

dowjones.head()

Output:



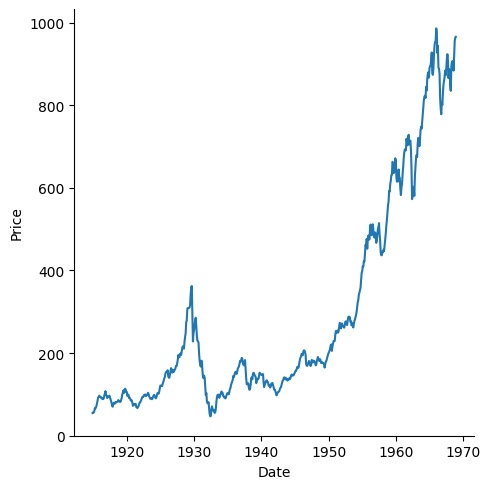
Code:

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

sns.relplot(data=dowjones,x="Date",y="Price",kind="line")

Output:

<seaborn.axisgrid.FacetGrid at 0x788f2c0afb20>

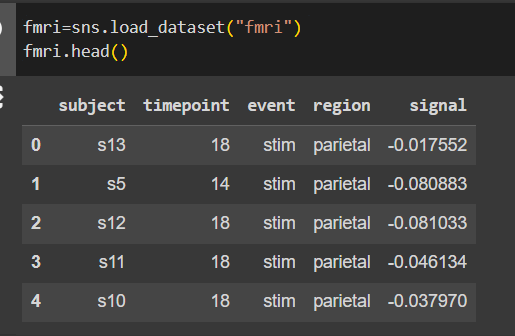


Code:

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

fmri.head()

Output:

****

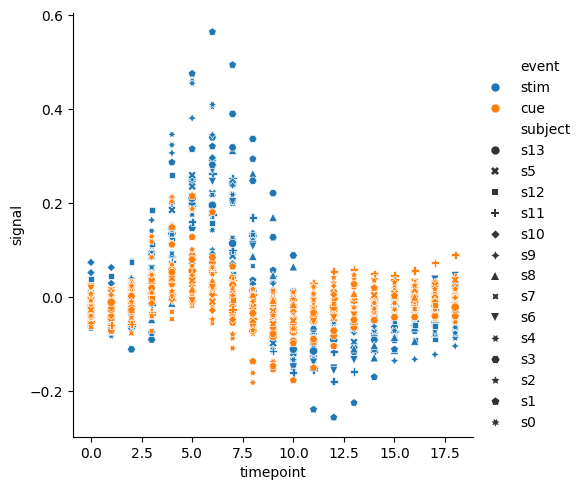
Code:

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

sns.relplot(data=fmri,x="timepoint",y="signal",hue="event",style="subject”)

Output:

<seaborn.axisgrid.FacetGrid at 0x788f2bcb03a0>



Subject means patients number

* region=brain location

Code:

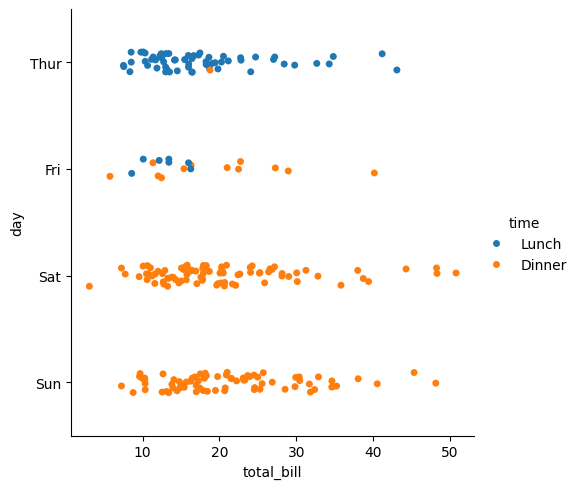
#category plot

sns.catplot(data=tips,x="total\_bill",y="day",hue="time")

Output:

<seaborn.axisgrid.FacetGrid at 0x788f2b76dcf0>

**Category plot :** The Seaborn catplot() function is used to create figure-level relational plots onto a Seaborn FacetGrid.

****

Code:

#category plot

sns.catplot(data=fmri,x="region",y="timepoint",kind="violin")

Output:

<seaborn.axisgrid.FacetGrid at 0x788f2b627df0>

****

Code:

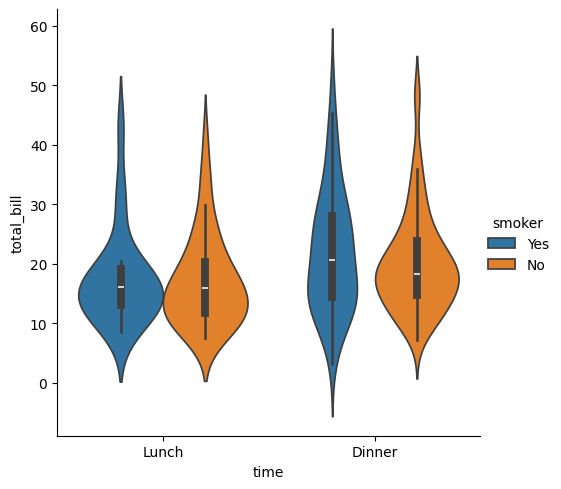
#category plot

sns.catplot(data=tips,x="time",y="total\_bill",kind="box",hue="smoker")

sns.catplot(data=tips,x="time",y="total\_bill",kind="violin",hue="smoker")

Output:

<seaborn.axisgrid.FacetGrid at 0x788f2ac337c0>

****

**Linear fit**

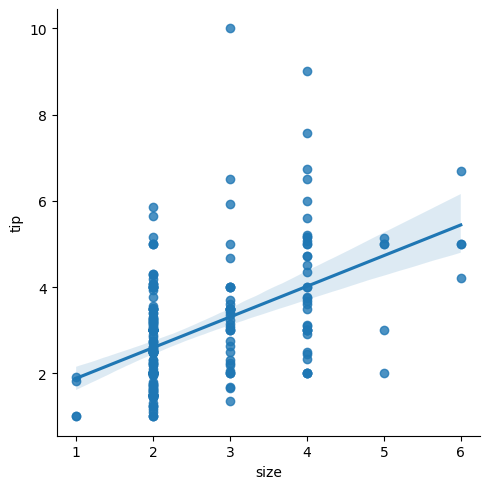
* lmplot()
* STEPS FOR LINEAR FIT
* plotting the dataset
* fitting the line
* predicting

lmplot() method is used to draw a scatter plot onto a FacetGrid.

Code:

sns.lmplot(x="size",y="tip",data=tips);

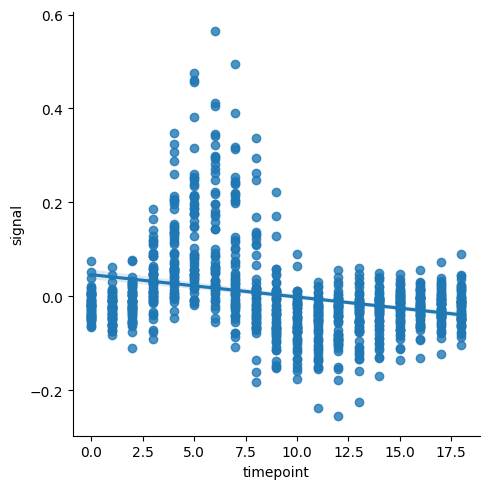
Output:

****

Code:

sns.lmplot(x="timepoint",y="signal",data=fmri);

Output:

****

**Web Scraping:**

* Loading data from a website.
* Unstructured in html.
* Convertible into spreadsheets/DB.
* Major websites have their APIs for web scraping.

**Scrapper:**

* Extract all the data on particular sites.
* Specific data that a user wants.

**Process:**

* URL->
* HTML code->
* Elements (CSS/JS)->
* Scrapes the required data->
* Saves it in required format(cvs,xlsx,Json)

**Applications:**

* Email Marketing.
* Sentiment Analysis.
* News Monitoring.
* Market Research.
* Price Monitoring.

**Libraries:**

* BeautifulSoup - scoop outs the html code -bs4 library
* Requests - to access the webpage - get(url)
* Selenium -
* Pandas - create data frames from excel,texts,cvs,DBs
* webdriver
* webdriver\_manager

To install any library:

**!pip install library**

**!pip install –upgrade library** #to upgrade to recent version

Import subpage:

from parent import child

Ex:- from bs4 import beautifulsoup

**Beautifulsoup:**

-Used to scrape data from static website

- Package bs4: subpackage BeautifulSoup

| **Function** | **Purpose** | **Attributes** |
| --- | --- | --- |
| BeautifulSoup() | To extract html code from a web page | .text  html |
| find() | To find first element of a kind | (‘element\_name) |
| find\_all() | To find all elements of a kind | (‘element\_name) |

**Requests:**

-Used to send the request to a web page

-get(‘url’)

**Project 1:**

**Extracting HTML code of any website**

Program flow

1. Import libraries
2. Save the url
3. Using request.get(),access the the web page
4. Using BeautifulSoup(),access HTML code

Step-1

#importing libraries

from bs4 import BeautifulSoup

import requests

Step-2

#saving the url

url="https://www.instagram.com/"

Step-3

#using requests.get(),access the web page

req=requests.get(url)

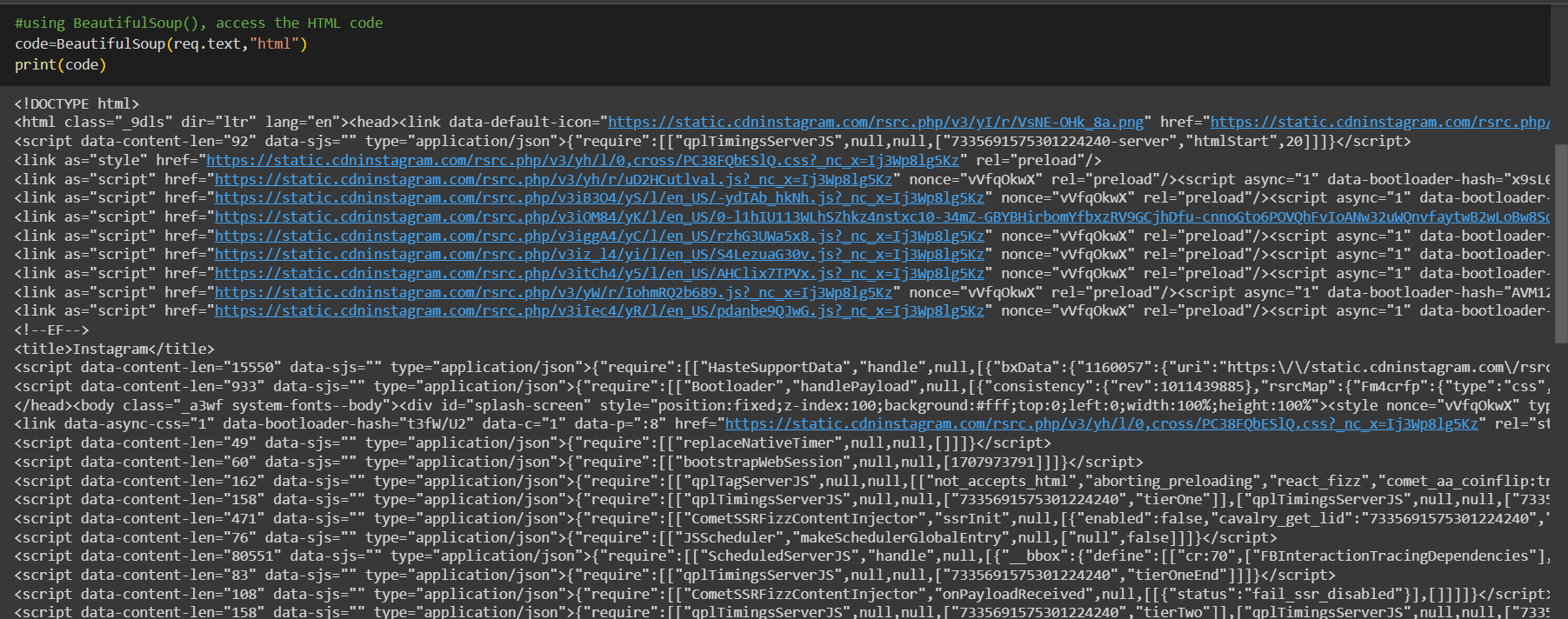
Step-4

#using BeautifulSoup(), access the HTML code

code=BeautifulSoup(req.text,"html")

print(code)

Output:



**``Accessing Elements:**

* Go to webpage
* Right click →Inspect
* Click on 
* Now hover on the element you want to access
* The html tags/code for that specific element is highlighted
* Extract the element name and import in your program using XPATH or By ID.

**Project 2 :**

**Extracting a table from HTML code of any website and saving it as.csv file**

Program flow

1. Import libraries
2. Save the url
3. Using requests.get(),access the web page
4. Using BeautifulSoup(),access the html code
5. Using find,access the table
6. Using find\_all,access the rows of the table
7. Using.text,extract only the text(removing html page)
8. Create a dataframe using pandas
9. Push all the extracted data into dataframes
10. Using to\_csv, save the dataframe in csv format.

Step-1

#importing libraries

from bs4 import BeautifulSoup

import requests

import pandas as pd

Step-2

url="https://www.forbesindia.com/article/explainers/top-10-richest-people-india/85909/1"

Step-3

#using requests.get(),access the web page

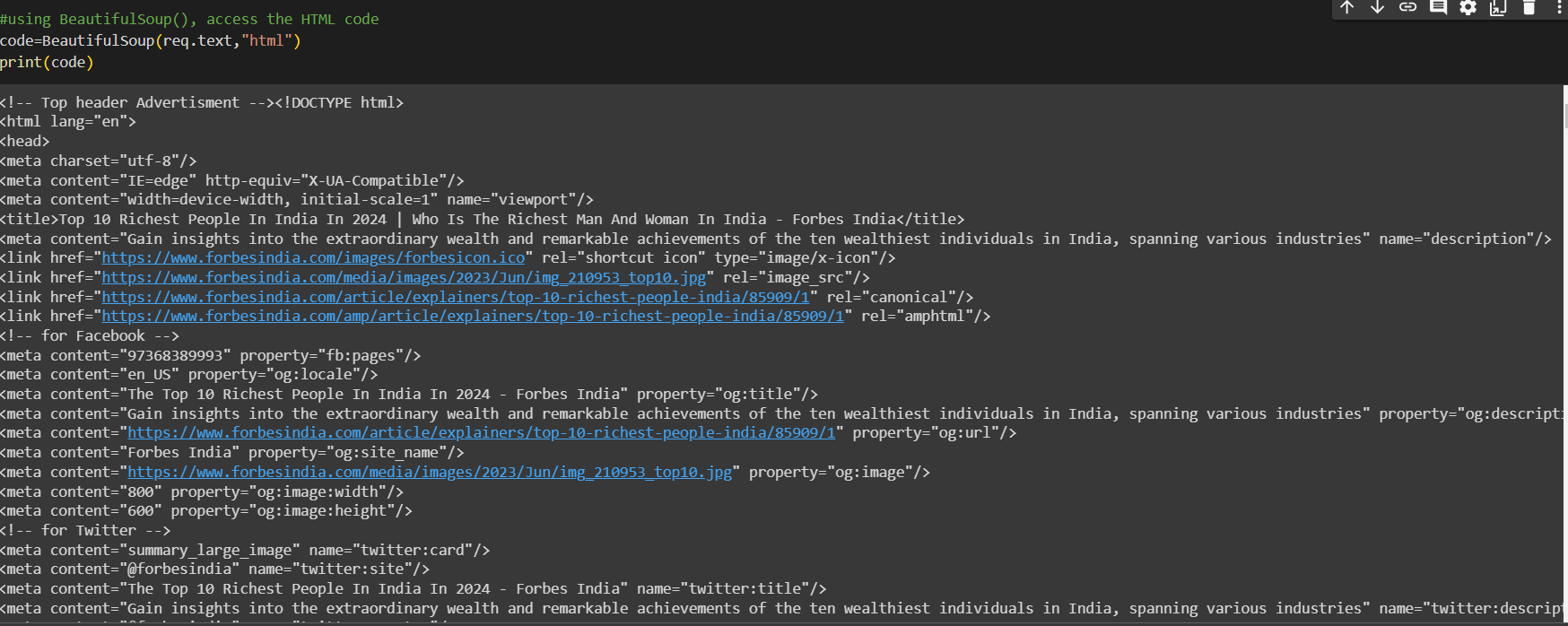
req=requests.get(url)

Step-4

#using BeautifulSoup(), access the HTML code

code=BeautifulSoup(req.text,"html")

print(code)

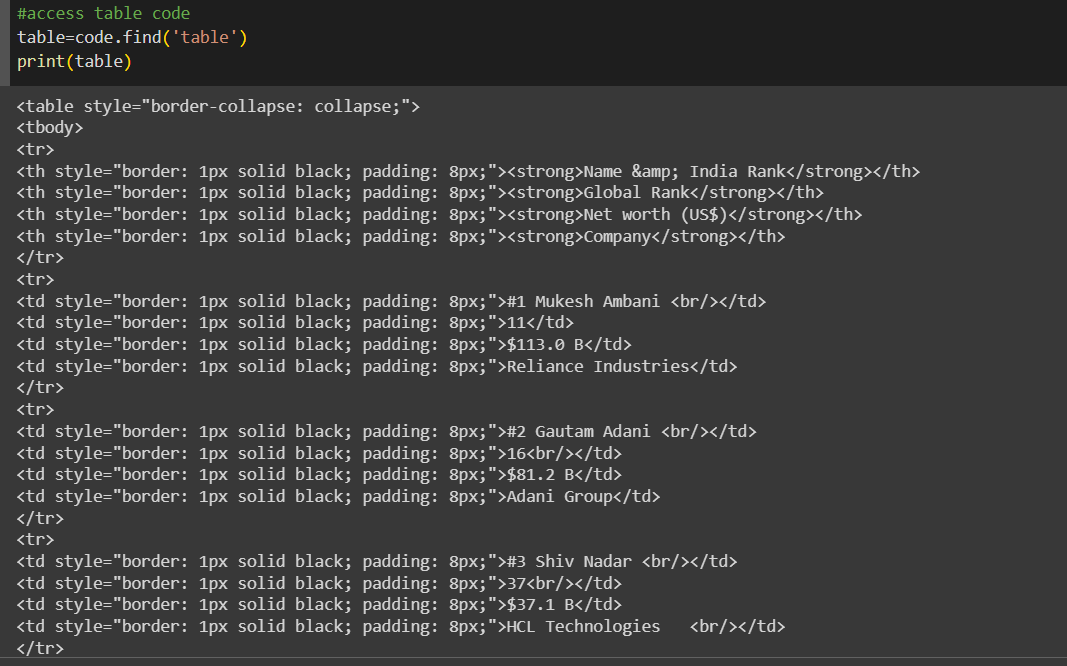


Step-5

#access table code

table=code.find('table')

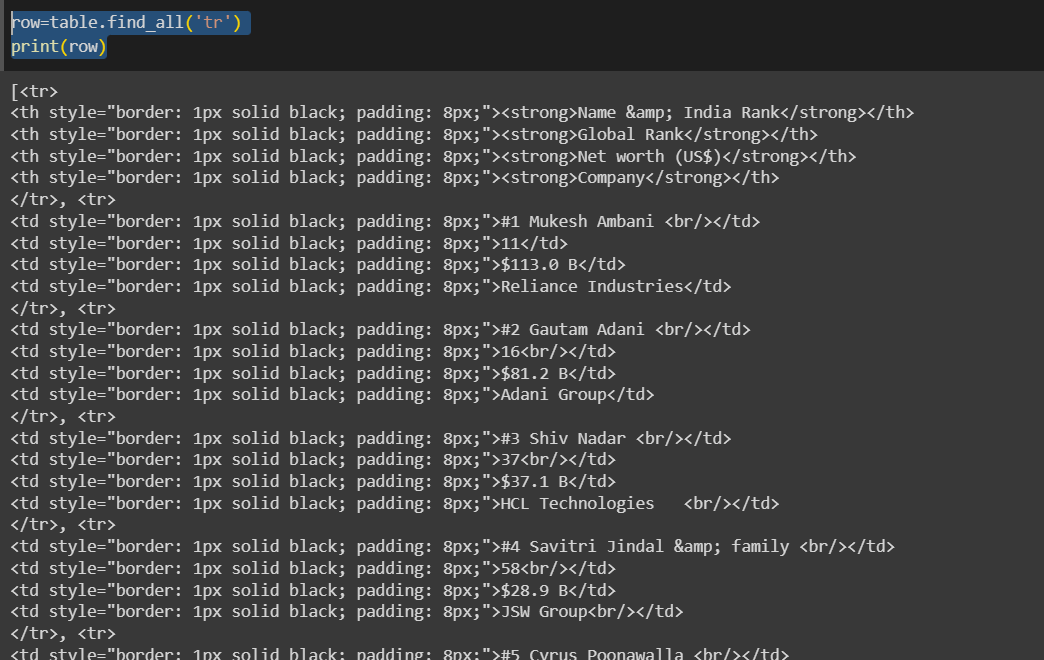
print(table)

****

Step-6

row=table.find\_all('tr')

print(row)



Step-7

#access each row from the table

for i in row[1:9]:

er=i.find\_all("td")

eachrow=[i.text for i in er]

print(eachrow)

Output:

['#1 Mukesh Ambani ', '11', '$113.0 B', 'Reliance Industries']

['#2 Gautam Adani ', '16', '$81.2 B', 'Adani Group']

['#3 Shiv Nadar ', '37', '$37.1 B', 'HCL Technologies\xa0\xa0 ']

['#4 Savitri Jindal & family ', '58', '$28.9 B', 'JSW Group']

['#5 Cyrus Poonawalla ', '68', '$25.6 B', 'Serum Institute of India']

['#6 Dilip Shanghvi ', '69', '$25.5 B', 'Sun Pharmaceutical Industries Ltd']

['#7 Kumar Birla ', '97', '$18.9 B', 'Aditya Birla Group']

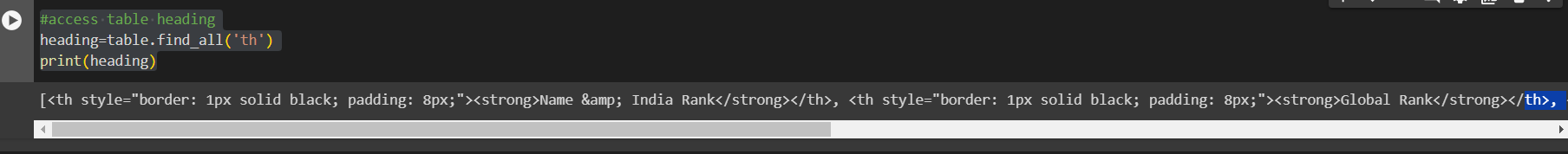
['#8 Kushal Pal Singh', '98', '$18.9 B', 'DLF Limited']

Step-8

#access table heading

heading=table.find\_all('th')

print(heading)



head=[i.text for i in heading]

print(head)

Output:

['Name & India Rank', 'Global Rank', 'Net worth (US$)', 'Company']

Step-9

a=0

for i in row[1:9]:

er=i.find\_all("td")

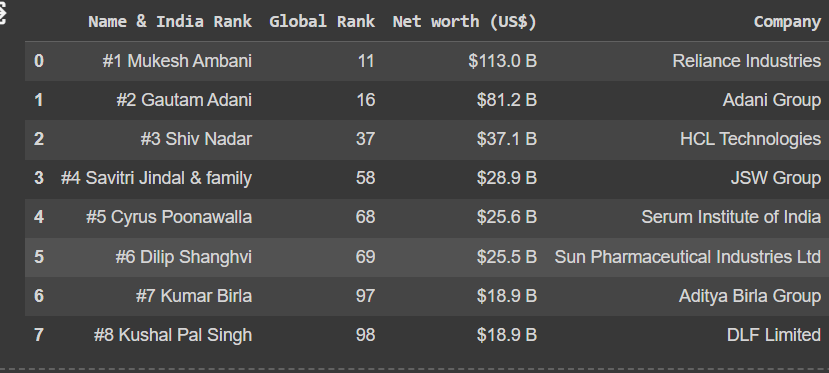
eachrow=[i.text for i in er]

df.loc[a]=eachrow

a=a+1

df

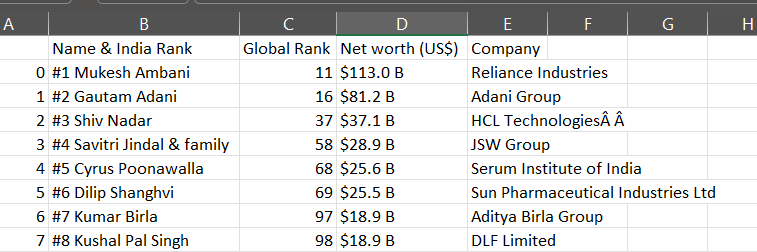
Output:



Step-10

df.to\_csv("Richmen.csv")

Output:



**Selenium:**

* Used to scrape data from static website
* Subpackage:Webdriver

| Function | Purpose | Attributes |
| --- | --- | --- |
| ChromeOptions() | Creates an instance of Chrome |  |
| .get | Access a webpage | ‘url’ |
| find\_element | To find first element of a kind | By.ID  By.XPATH |
| .click | To click a button in webpage |  |

**Problem 3:**

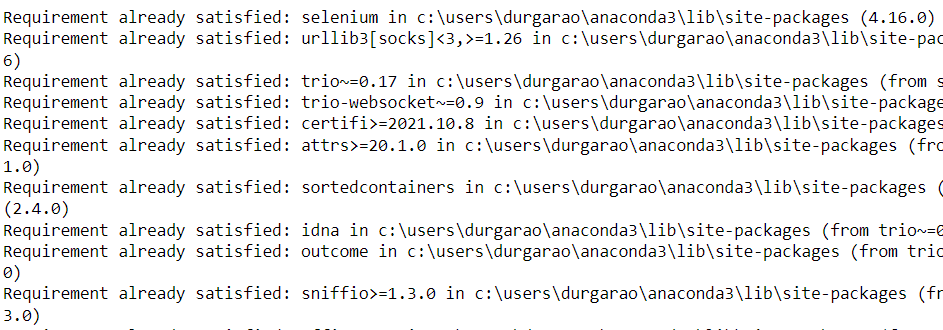
**Extracting Dell laptops from amazon in website and saving it as .csv**

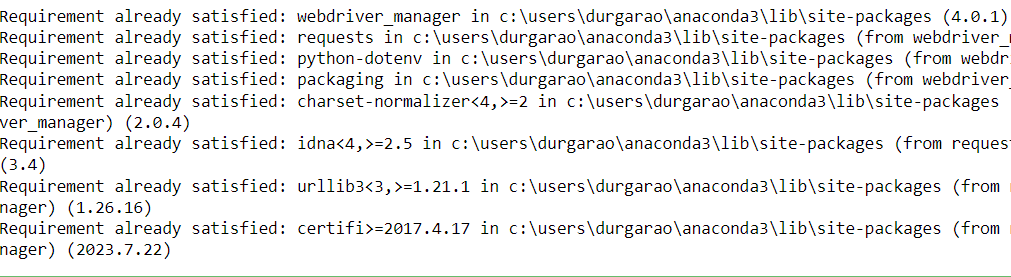
**Name of the Laptop , Price , No of Reviews.**

* To access the search box in amazon →id twotosearchtextbox
* To access the search button →id nav-search-submit-button

Step-1:

!pip install selenium



!pip install webdriver\_manager 

Step-2

from selenium import webdriver

from selenium.webdriver.chrome.options import Options

from webdriver\_manager.chrome import ChromeDriverManager

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

Step-3

#define options and set browser capabilities

options=webdriver.ChromeOptions()

options.add\_argument('--some-option')

#Create WebDriver instance with options

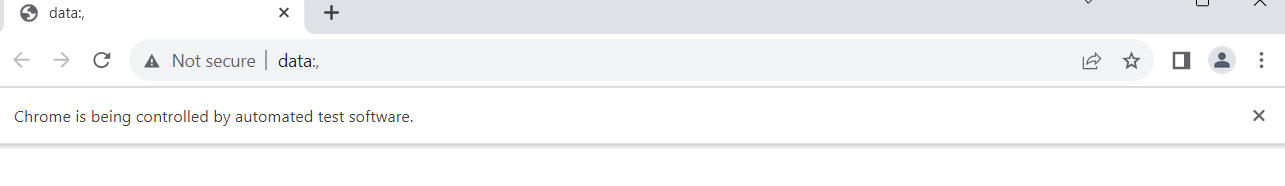
driver=webdriver.Chrome(options=options)

#Access browser capabilities

browser\_name=options.to\_capabilities()['browserName']

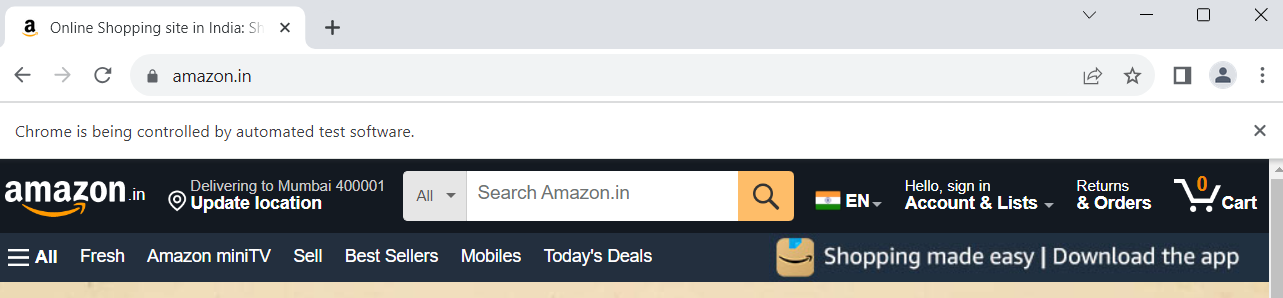
print(browser\_name)

chrome



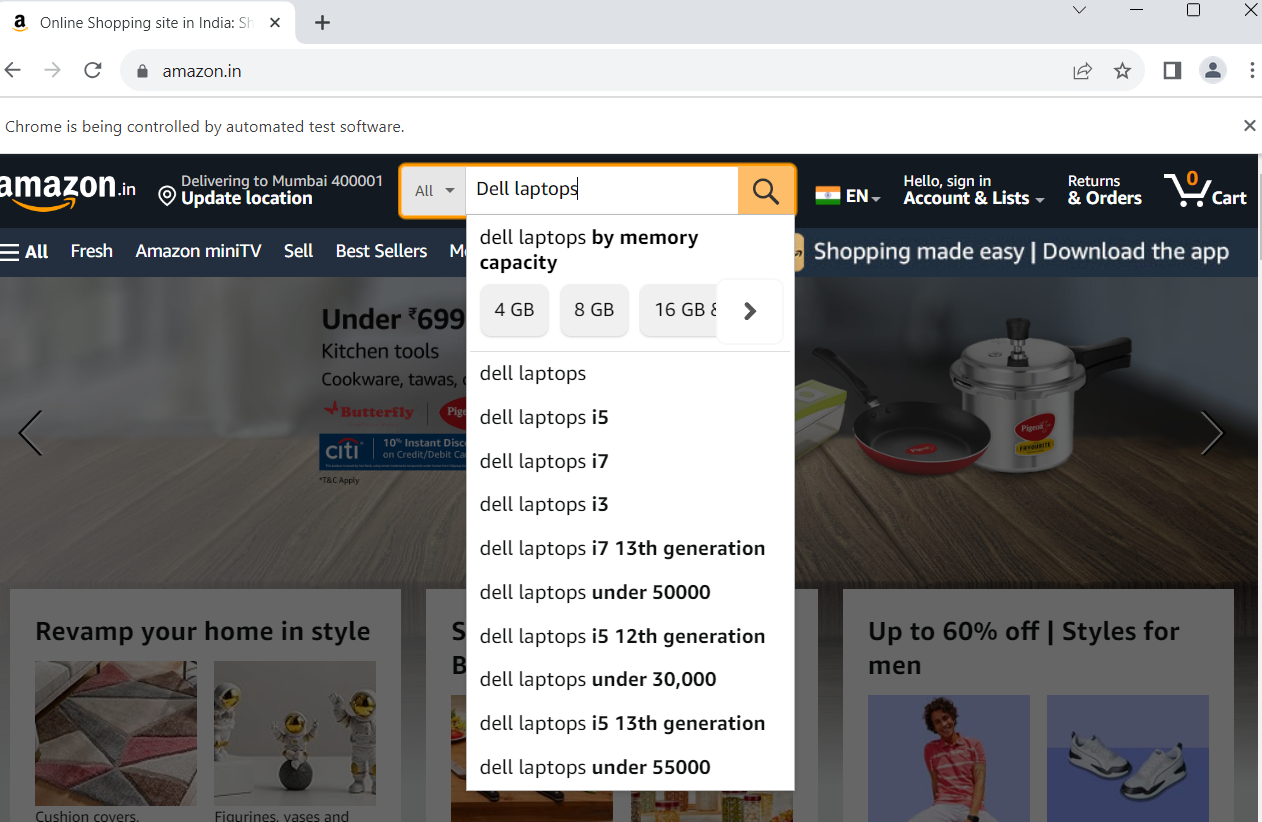
Step-4

#navigate to a website

driver.get("https://www.amazon.in/") 

Step-5

search=driver.find\_element(By.ID,"twotabsearchtextbox")

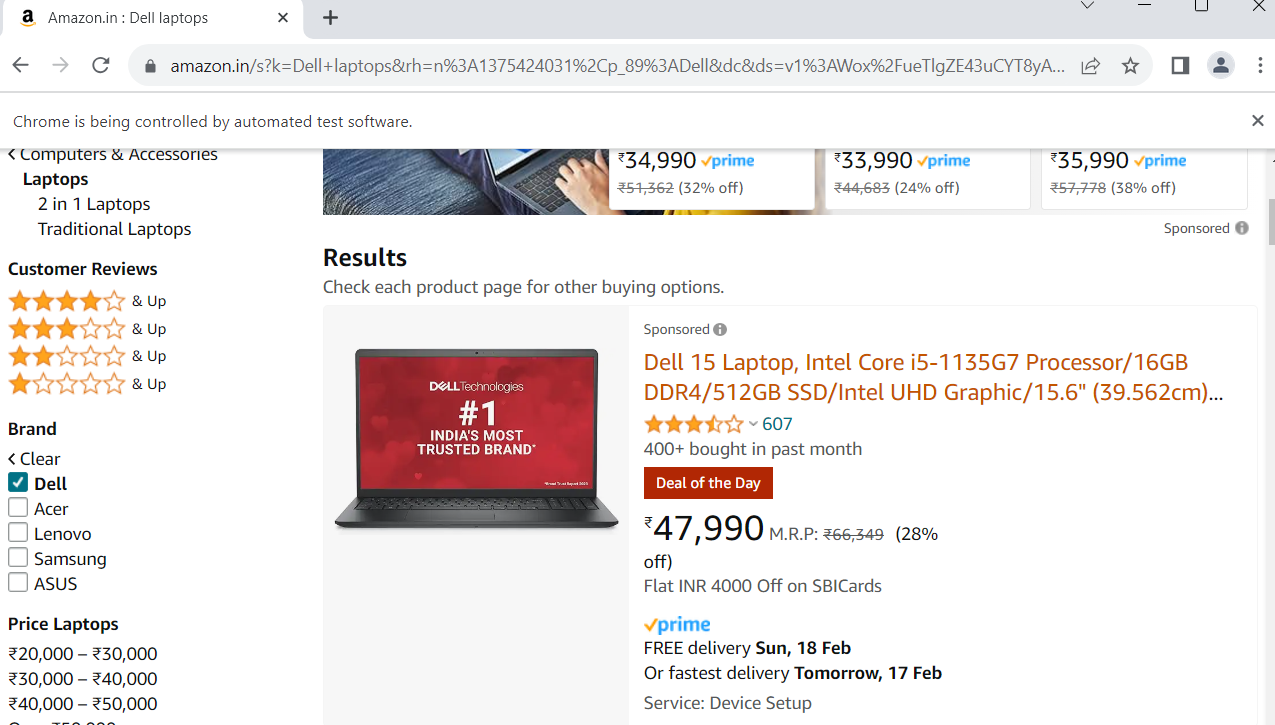


Step-6

driver.find\_element(By.ID,"nav-search-submit-button").click()



Step-7



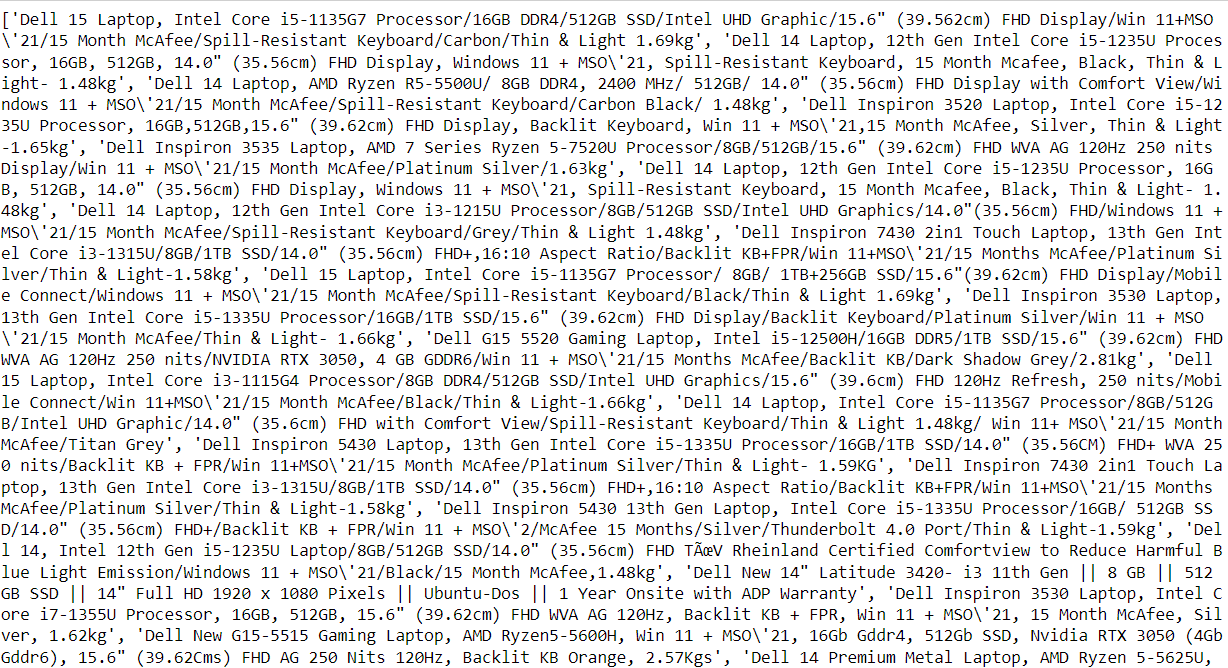
Step-8

names=driver.find\_elements(By.XPATH,"//span[@class='a-size-medium a-color-base a-text-normal']")

l\_names=[i.text for i in names]

print(l\_names)

print(len(l\_names))



=> 30

Step-9

price=driver.find\_elements(By.XPATH,"//span[@class='a-price-whole']")

l\_price=[i.text for i in price]

print(l\_price)

['34,990', '33,990', '35,990', '47,990', '49,990', '35,990', '55,280', '38,990', '49,990', '35,990', '57,990', '44,990', '67,490', '75,990', '33,990', '44,990', '71,290', '57,990', '68,990', '46,990', '30,630', '71,490', '78,490', '50,490', '34,380', '81,490', '82,990', '87,490', '38,090', '75,490', '37,990', '72,990', '56,990']

#To remove the elements that are shown as extra in case of offers

=>l\_price.pop(0)

print(l\_price)

['47,990', '49,990', '35,990', '55,280', '38,990', '49,990', '35,990', '57,990', '44,990', '67,490', '75,990', '33,990', '44,990', '71,290', '57,990', '68,990', '46,990', '30,630', '71,490', '78,490', '50,490', '34,380', '81,490', '82,990', '87,490', '38,090', '75,490', '37,990', '72,990', '56,990']

Step-10

reviews=driver.find\_elements(By.XPATH,"//span[@class='a-size-base s-underline-text']")

l\_reviews=[i.text for i in reviews]

print(l\_reviews)

['607', '72', '4', '2', '4', '72', '239', '179', '607', '13', '517', '631', '2', '82', '179', '82', '138', '1', '1', '71', '1', '195', '11', '27', '7', '3', '11', '3', '517', '34']

Step-11

import pandas as pd

headings=['laptop\_name','prices','Reviews']

df=pd.DataFrame(columns=headings)

print(df)

=> Empty DataFrame

Columns: [laptop\_name, prices, Reviews]

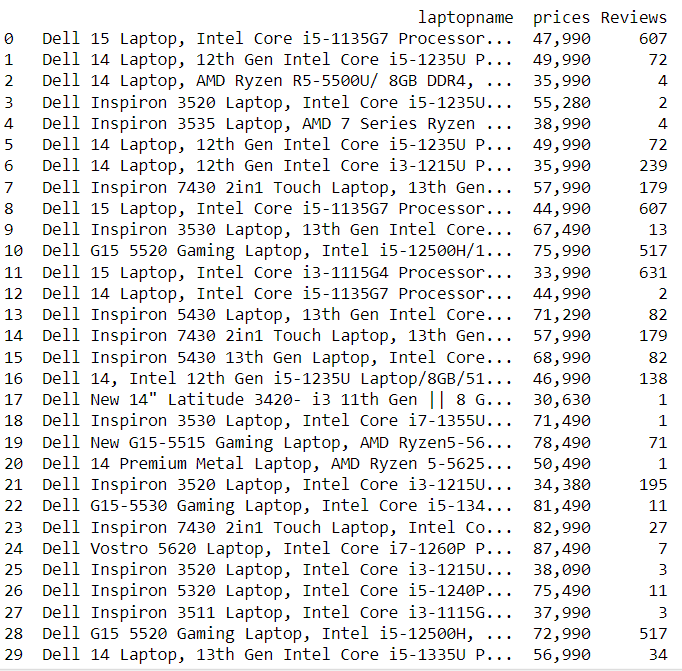
Index: []

df['laptop\_name']=l\_names

df['prices']=l\_price

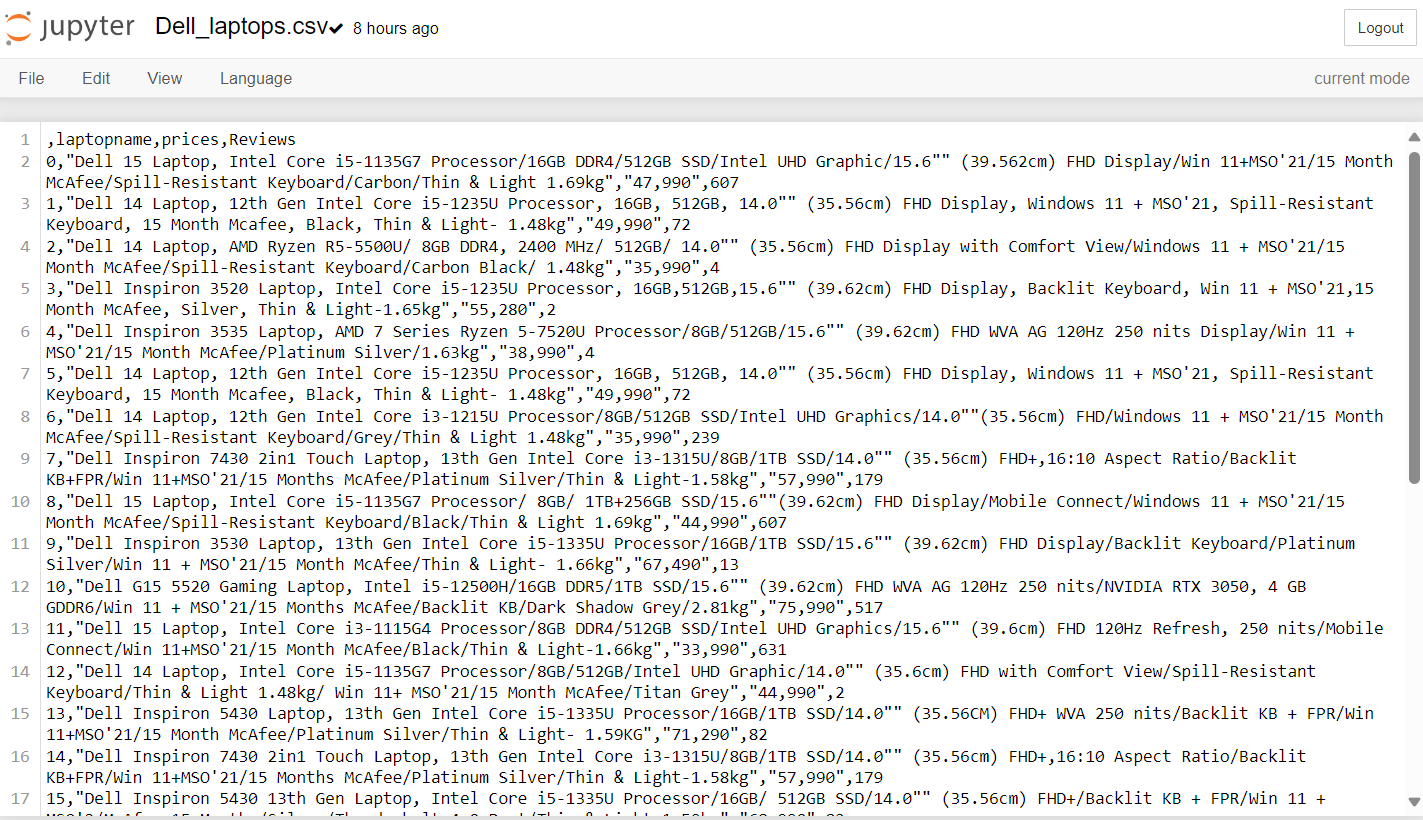
df['Reviews']=l\_reviews

print(df)



Step-12

df.to\_csv("Dell\_laptops.csv")

****

**API -> Application Program Interference**

**→** It is a collection of apps,the API of each required app is combined together to form a big app.

→ It contains the protocols.

→ A server that you can use to retrieve and send data using code.

* One website fetching the data from one database through direct connection.
* One website fetching the data from different databases belonging to different companies requires API’s.

**Public key:Accessible to everyone.**

**Private key:Accessible to subscribers.**

**–>API:**

**connect apps together.**

Eg:Ola

APP1:Log in --

APP2:Location --

APP3:Payment --

And so on....

* All these small apps are connected together into a large app called Ola.
* These small apps are called APIs and they are building blocks of so many Bigger apps,hence reusable.
* So if 1f 1000+ apps use these APIs,what happens!
* API crashes...
* Hence API keys are created.

**Ex:**

**Random Fox API**

Step-1

import requests

Step-2

page=requests.get("https://randomfox.ca/floof")

Step-3

print(page.status\_code)

200

Step-4

print(page.text)

Output:

{"image":"https:\/\/randomfox.ca\/images\/73.jpg","link":"https:\/\/randomfox.ca\/?i=73"}

Step-5

print(page.json())

Output:

{'image': '<https://randomfox.ca/images/73.jpg>', 'link': '<https://randomfox.ca/?i=73>'}



**PROJECT:**

**Extracting data from coinmarketcap API**

Program flow:

1.Go to Coinmarket cap website.

2.Go to Products -->API and obtain API key

3.Go to documentation to obtain API code!

4.Know your code!

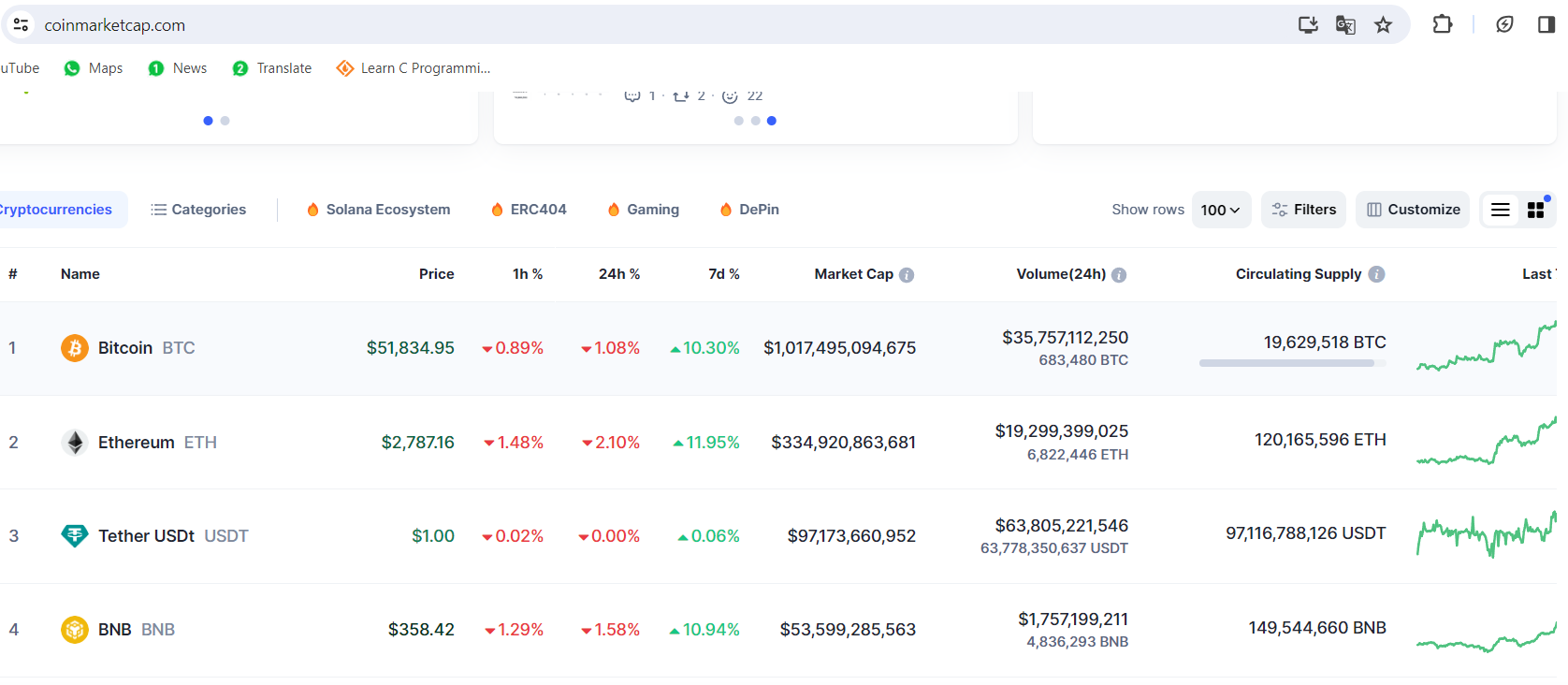
5.Run it in any python environment

6.Obtain data in the form of json

7.Normalize the data into a dataframe

8.save it if you needed

Step-1



Step-2 API key → c3b7dd81-b6d3-4ef8-a269-a036313f0a41

Step-3

#This example uses Python 2.7 and the python-request library.

from requests import Request, Session

from requests.exceptions import ConnectionError, Timeout, TooManyRedirects

import json

url = 'https://sandbox-api.coinmarketcap.com/v1/cryptocurrency/listings/latest'

parameters = {

'start':'1',

'limit':'5000',

'convert':'USD'

}

headers = {

'Accepts': 'application/json',

'X-CMC\_PRO\_API\_KEY': '54ed5df8-6a1f-4448-8b70-def508eff113',

}

session = Session()

session.headers.update(headers)

try:

response = session.get(url, params=parameters)

data = json.loads(response.text)

print(data)

except (ConnectionError, Timeout, TooManyRedirects) as e:

print(e)

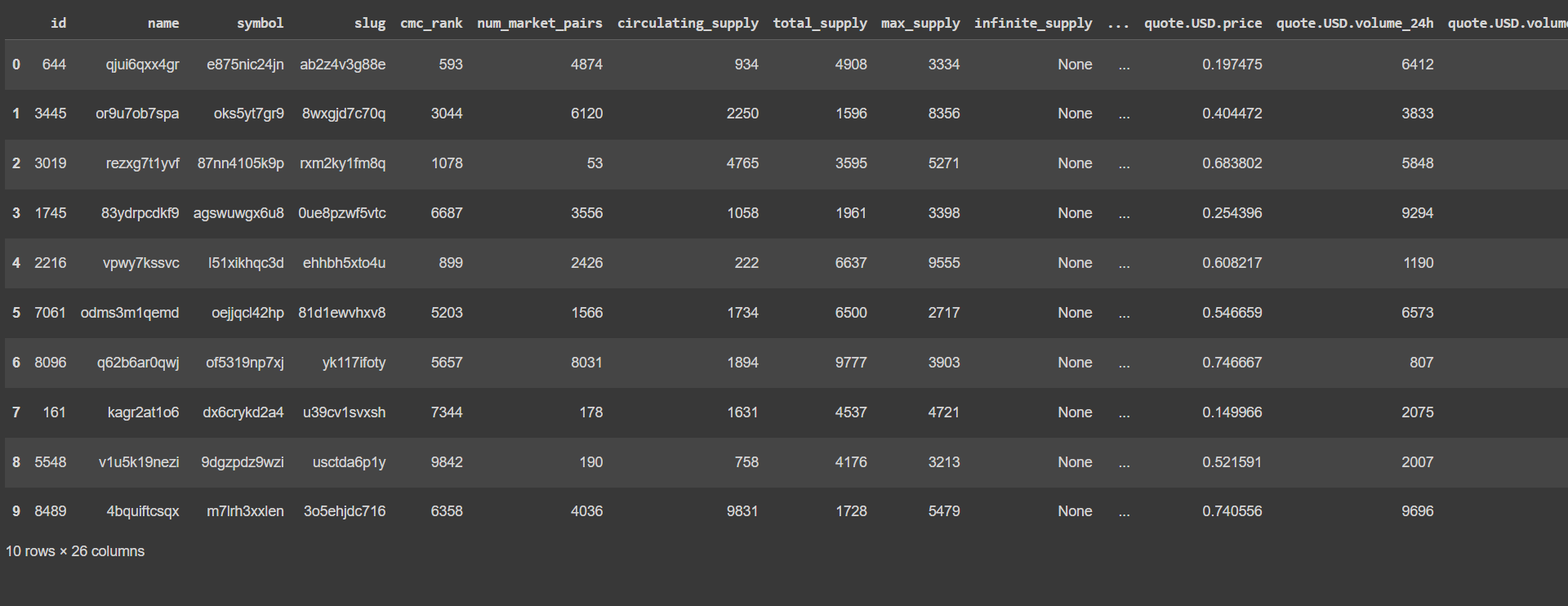
'data': [{'id': 4367, 'name': 'up8ropnbxv', 'symbol': 'g219vxa5yqi', 'slug': 'wf5bij2xf5', 'cmc\_rank': 3192, 'num\_market\_pairs': 4272, 'circulating\_supply': 2850

Step-5

import pandas as pd

norm=pd.json\_normalize(data['data'])

norm



**Machine learning:**

* Like humans learn from their past experiences,machines learn from past data.
* Training a machine on various things is called machine learning.

**Example :Case Study on Tesla:**

Tesla is now a big player in the electric automobile industry. It is widely known for its advanced and futuristic cars. The company says that the cars have their own AI hardware. Tesla is using AI for making self-driving cars. At the moment, cars are not completely autonomous. The company is working on the thinking algorithm for cars. It is currently working with NVIDIA on an unsupervised ML algorithm.

**Case study:**

A machine will suggest the songs by liked songs and the high rated song will be played to the listener .

**Types of Machine Learning:**

* Supervised
* Unsupervised
* Reinforcement

**Supervised learning:**

Training the machine with some labelled data.

Example;

* Assuming 3 types of coins(1 rupee,1,euro,1 dirham)
* Takes a feature for every label.(features=weight,currency=Label)
* It takes inputs as a label gives output as any one of the labels.
* When you provide a new coin to the machine it will look for the feature value.
* If the feature value matches, and gives the label as output.

**Supervised Algorithm:**

1. Linear Regression
2. Logistic Regression
3. Decision Tree
4. Random Forest

**Linear regression:**

* It is also a type of machine learning -algorithm.
* Supervised machine learning algorithm.
* Learns from the labelled datasets and maps the data points to the most optimised linear functions.
* These points can be used for prediction on new datasets.

**Dependent and Independent Variable:**

* Independent:

The independent Variable is the cause.Its value is independent of other variables in your study.

* Dependent:

The dependent variable is the effect .its value depends on changes in the

independent variable.

**Case Study:**

Consider measurements of a chemical reaction.

The mass of the product increases with time.

The observations are:

| Time(m) | 5 | 7 | 12 | 16 | 20 |
| --- | --- | --- | --- | --- | --- |
| Mass(gms) | 40 | 120 | 180 | 210 | 240 |

Time-Independent

Mass-Dependent

Problem:

Find the mean of dependent and independent Variable

Regression line is y=a+bx

**Regression:**

* training->fit
* algorithm->model

Code:

from sklearn.linear\_model import LinearRegression

LR=LinearRegression()

Code:

#Time is independent and it is in 2D array

t=[[5],[7],[12],[16],[20]]

#Mass is dependent and it is in 1D array

m=[40,120,180,210,240]

LR.fit(t,m)

Output:

LinearRegression

LinearRegression()

Code:

LR.predict([[5.5]])

Output:

array([78.64935065])

**Logistic Regression**:

* It is used for binary classification.
* Supervised learning Algorithm.

**Binary Classification:**

Example Segregating Food:

Y=0,Class A-Healthy food

Y=1, Class B -Unhealthy food

**Case Study:**

* Let us take data from Football match.
* Based on the distance between the player and the goal post,we are going to predict whether it is a goal or not.!
* Let us plot some trails now.
* When distance =2m,goal is scored,Y=1
* When distance =4m,goal is scored ,Y=1
* For 5,7,10,20,22m,it is always a goal, Y=1
* When distance=23m,for 15 trails,

few are goals Y=1, few are failures Y=0

Code:

from logging import LogRecord

import numpy as np

from sklearn.linear\_model import LogisticRegression

import matplotlib.pyplot as plt

Code:

#Distance and corresponding probability data

distances=np.array([1,2,5,10,15,20,21,22,23,24,25,26,27,28,29,30,35,40,41,47,50]).reshape(-1,1)

probabilities=np.array([1,1,1,1,1,1,0.9,0.85,0.73,0.67,0.5,0.47,0.39,0.31,0.25,0.15,0,0,0,0,0])

#convert probabilities to binary labels

threshold=0.5

binary\_labels=(probabilities>threshold).astype(int)

Code:

#create and fit logistic regression model

logr=LogisticRegression()

logr.fit(distances,binary\_labels)

Output:

LogisticRegression

LogisticRegression()

Code:

p=logr.predict([[10]]) #distance

Code:

if p==[1]:

print("goal")

else:

print("not goal")

Output:

goal

**Sigmoid Function:**

* Sigmoid function is a s-shaped function with peaks at 1 and 0 at valleys.
* When model is very confident, Narrow DB
* When model is not confident ,Wide DB

**Problem:**

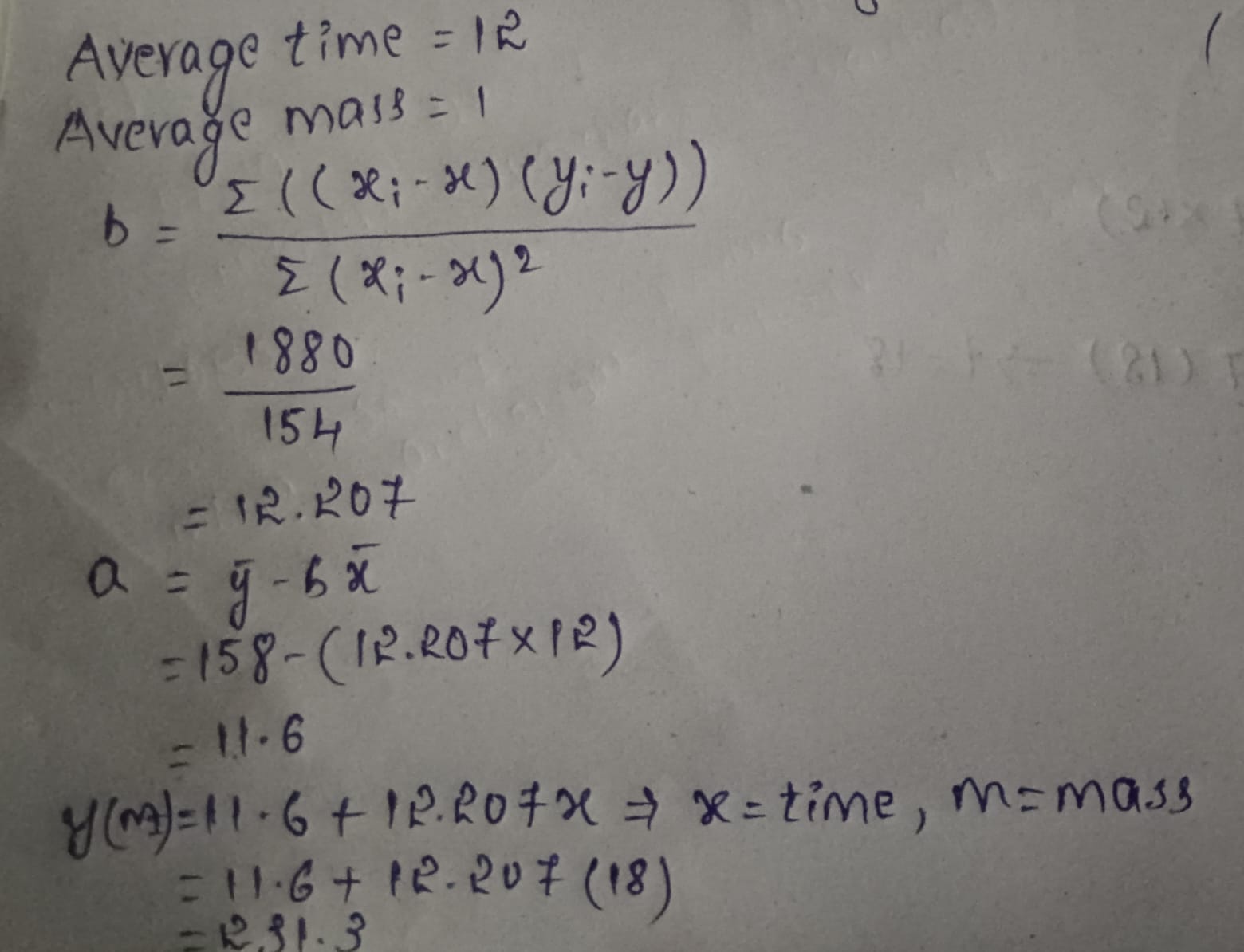
After plotting the distance vs goal between 20 and 30 metres the probability of goal gradually reduces from 1 to 0.

But logistic regression accepts only two classes.

So a threshold variable(0.5) is set

→The model Prediction is:

* P>0.5, It's a Goal!- class A (Y=1)
* P=<0.5, It's a Miss!- class B (Y=0)



Code:

#predict 100 distance between 1 and 50

#generate distance for prediction

dist=np.linspace(1,50,100).reshape(-1,1)

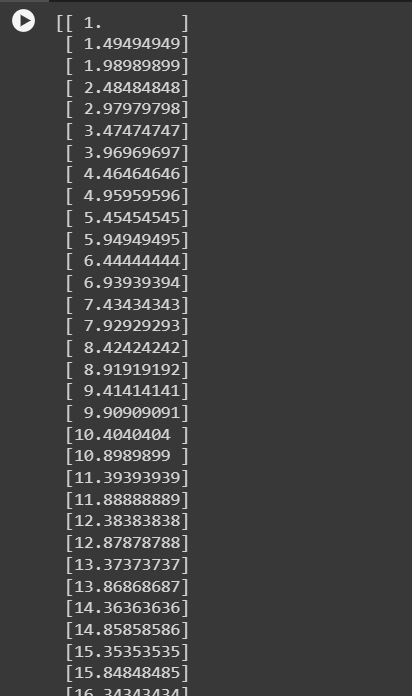
print(dist) #distance

#make prediction using the model

prob=logr.predict\_proba(dist)[:,1] #predictions

print(prob)

Output:



Code:

#plotting actual data - train

plt.scatter(distances,binary\_labels,color='black',label='Data')

#plotting test data with prediction - valid/test

plt.plot(dist,prob,color='blue',label='Logistic Regression')

plt.title('Distance vs Probability of scoring a Goal')

plt.xlabel('Distance')

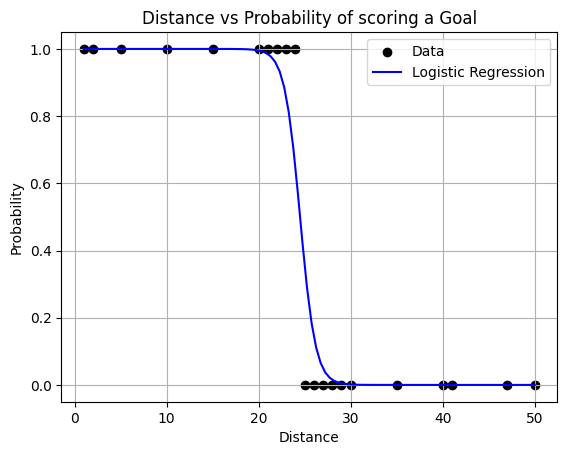
plt.ylabel('Probability')

plt.legend()

plt.grid(True)

plt.show()

Output:



**DECISION TREES:**

Decision trees in machine learning provide an effective method for decision because they lay out the problem and all possible outcomes.

* Have nodes and leaves
* Node: Condition having True and False Branches
* Leaf:Result-Showing the dataset that is True and False to the condition
* Decision tree continuously splits the data until it gets the pure leaves.

**Case Study:**

* Taking a dataset of 26 states with features like literacy,Cleanliness,Crime Rate and targeting(predicting) Good or Bad state!
* Good is called Target variable here,it has values of 0s and 1s
* View**:**

| **state** | **literacy** | **cleanliness** | **Crime Rate** | **Good** |
| --- | --- | --- | --- | --- |
| A | 92 | 90 | 54 | 0 |
| B | 56 | 67 | 50 | 1 |
| C | 78 | 85 | 62 | 0 |
| D | 63 | 72 | 48 | 1 |
| E | 85 | 79 | 55 | 0 |

Now let us create a Decision tree

* Decision tree recurrently(continuously) splits the data until it gets pure leaves.
* Let us view a DT based on Crime Rate

**→BUILDING DECISION TREE**

**NODE1:**

CR>60

True=[C,Q,X=0](Pure leaf)

False=[A,E,F,G,I,K,L,N,P,R,U,V=0],[B,D,H,J,M,O,S,T,W,Y,Z=1] (Mixed leaf)

Mixed leaf has target variable with both 1’s and 0’s .Hence this data is splitted once again.

**NODE2:**

CR>50

True=[A,E,F,G,I,K,L,N,P,R,U,V=0] (Pure leaf)

False=[B,D,H,J,M,O,S,T,W,Y,Z=1] (Pure leaf)

**Results:**

| Node | Good |
| --- | --- |
| CR >60 | 0 |
| CR<60 | Cannot be determined |
| CR<50 | 1 |

Predict:

* CR=63 ,Good=0
* CR=45, Good=1

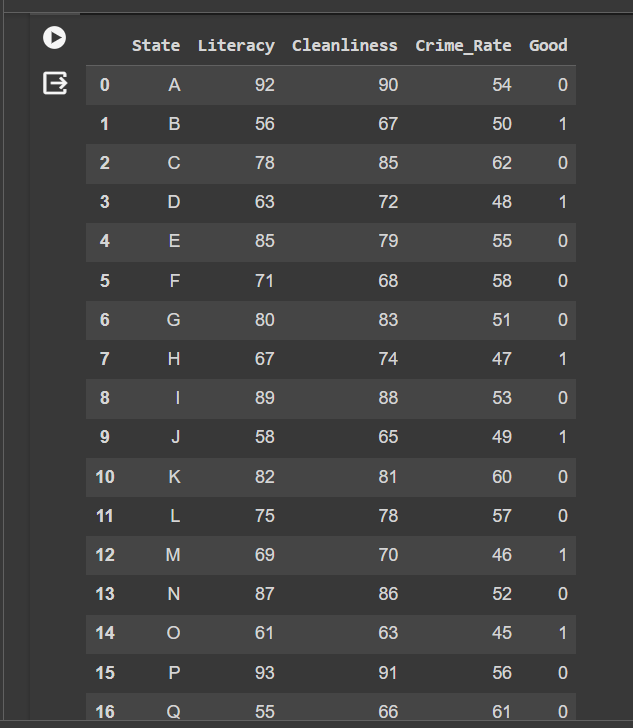
Code:

import pandas as pd

dft=pd.read\_csv("/content/demodt.txt",sep=",")

dft

Output:



Code:

from sklearn.tree import DecisionTreeClassifier

DT=DecisionTreeClassifier()

target=dft.Good

Feat\_list=["Literacy","Cleanliness","Crime\_Rate"]

Feat=dft[Feat\_list]

DT.fit(Feat,target)

Output:

DecisionTreeClassifier

DecisionTreeClassifier()

Code:

pred=DT.predict([[90,90,63]])

print(pred)

Output:

[0]

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names

warnings.warn(

Code:

if pred==1:

print("Good")

else:

print("Not Good"**)**

Output:

**Not Good**

Code:

import numpy as np

Lit=int(input("Enter Literacy"))

Cle=int(input("Enter Cleanliness"))

Cmr=int(input("Enter Crime rate"))

d=np.array([Lit,Cle,Cmr]).reshape(-1,1)

print(d)

e=DT.predict([[Lit,Cle,Cmr]

])

if e==1:

print("Good")

else:

print("Not Good")

Output:

Enter Literacy90

Enter Cleanliness90

Enter Crime rate53

[[90]

[90]

[53]]

Not Good

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names

warnings.warn(

**Random Forest:**

* Collection of decision tree.
* Keywords:bootstrapping,aggregation.

**Case study:**

Why we need RFs and Dts

Observe below data

Y-target

X0,X1,X2,X3,X4-features

**Bootstrapping:**

Splitting the parent dataset into child datasets having the same number of rows and should have different row combinations.

**Convolutional Neural Networking:**

* A convolutional NeuralNetwork (CNN) is a type of Deep Learning neural network architecture commonly used in Computer Vision
* It consists of Three layers

1. **Input Layers**

It's the layer in which we give input to our model.In CNN generally the input will be a image this layer holds the raw data

1. **Hidden layers**

→**Convolutional layer:**

* This is the layer ,which is used to extract features from the input dataset .It applies a set of learnable filters known as the kernel to input images.The filters/kernel are smaller matrices usually 2x2, 3x3and 5x5 shape, it slides over the input image data and compute the dot product between kernel weight and the corresponding input image patch.
* The output of this layer is referred to as featured maps.Suppose we use a total of 12 filters for this layer we will get an output volume of dimensions 32x32x12.

→**Activation Layer :**

* By adding an activation function to the output of the preceding layer,activation layers add non linearly to the network.It i will apply an element -wise activation function to the output to the convolutional
* The volume remains unchanged hence output volume will have dimensions 32\*332\*12.

→**Pooling Layer:**

* This layer is periodically inserted in the converts and its main function is to reduce the size of volume which makes the computation fast reduces memory and also
* If we use a max pool with 2x2 filters and stride 2,the resultant volume will be of dimension 16x16x12.

1. **Output layers:**

The output from the fully connected layers is then fed into a logistic .

classification tasks like sigmoid or softmax which converts the output**.**

**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 bias of the error at the output. This process is known as back propagation. Activation functions make the back propagation possible since the gradients are supplied along with the error to update the weights and biases.
* **Tanh :**

=>Tanh Function

1. Valin Range-1 to 0.

* **Sigmoid :**

=>Sigmoid Function

1. It's a function which is plotted as a ‘s’ shaped graph.
2. Equation A=1/(1+ex)
3. Nature: Non-linear that X values lie between -2 to 2, Y values are very steep. This means small changes in x would also bring about large changes in the value Y.
4. Value Range: 0 to 1.

* **Relu :**

1. It stands for Rectified linear unit.It is the most widely used activation function.Chiefly implemented in hidden layers of Neural network.
2. Equation:-A(x)=max(0,x).It gives an output x if x is positive and 0 otherwise.
3. Value range:-[0,inf)
4. Nature:-Nonlinear which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the Relu function.
5. 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.
6. In other words,RELU learns much faster than sigmoid and Tanh functions.

* **Softmax :**

1. The softmax function is also a type of sigmoid function but id handy where we are trying to handle multi class classification problems.
2. Nature: non-linear
3. Uses:- The softmax function was commonly found in the output layer of the image classification problems.the softmax function would squeeze the outputs for each class between 0 and 1 and would also dividenby the sum of the outputs.
4. Output:

The softmax function is ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input.

If your output is for binary classification then,sigmoid function is a very natural choice for the output layer.

If your output is multi-class classification then ,Softmax is very useful to predict the probabilities of each .

**Code for Brain Tumor:**

Step1:

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

Step2:

Code:

from google.colab import drive

drive.mount('/content/drive')

Output:

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force\_remount=True).

Step3:

Code:

#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/Brain\_Tumor\_Detection/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/Brain\_Tumor\_Detection/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/Brain\_Tumor\_Detection/Brain\_Tumor\_Detection/test',

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

batch\_size=BATCH\_SIZE,

class\_mode='binary'

)

Output:

Found 2408 images belonging to 2 classes.

Found 602 images belonging to 2 classes.

Found 60 images belonging to 1 classes.

Step4:

Code:

#define the model

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

layers.MaxPool2D((2,2)),

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

layers.MaxPool2D((2,2)),

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

layers.MaxPool2D((2,2)),

layers.Flatten(),

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

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

])

Step5:

Code:

#compile the model

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

Step6:

history=model.fit(train\_generator,validation\_data=val\_generator,epochs=5)

Output:

Epoch 1/5

76/76 [==============================] - 655s 9s/step - loss: 0.5669 - accuracy: 0.7458 - val\_loss: 0.3734 - val\_accuracy: 0.8173

Epoch 2/5

76/76 [==============================] - 290s 4s/step - loss: 0.2555 - accuracy: 0.9003 - val\_loss: 0.1763 - val\_accuracy: 0.9219

Epoch 3/5

76/76 [==============================] - 287s 4s/step - loss: 0.1312 - accuracy: 0.9560 - val\_loss: 0.1838 - val\_accuracy: 0.9169

Epoch 4/5

76/76 [==============================] - 287s 4s/step - loss: 0.0882 - accuracy: 0.9672 - val\_loss: 0.0496 - val\_accuracy: 0.9817

Epoch 5/5

76/76 [==============================] - 290s 4s/step - loss: 0.0422 - accuracy: 0.9892 - val\_loss: 0.0205 - val\_accuracy: 0.9950

Step7:

Code:

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

Output:

/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my\_model.keras')`.

saving\_api.save\_model(

Step8:

Code:

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/Brain\_Tumor\_Detection/Brain\_Tumor\_Detection/test/pred/pred1.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) #batch dimension

img\_array /=255. #normalise 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],")")

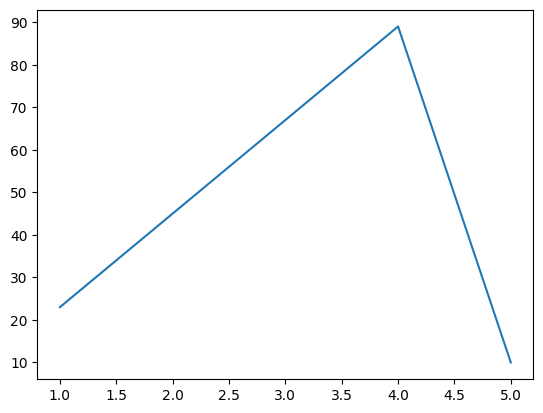
else:

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

Output:

1/1 [==============================] - 0s 118ms/step

Prediction: No tumor(Probability: 2.3448249e-05 )



Code: