Data Analysis with Python

1. Concept Overview - Python
   1. Basics
      1. Variable:

Container to store values Code:

A=5

* + 1. Print:

It is a function which is used to display Code:

#var and print a=5

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

output

i am 5 years old

* 1. Operators
     1. Arithmetic: #def

Code:

1.2.2 Power:

1.2.3 Relational

a=2

b=2

print(a==b)

print(2\*\*6) OUTPUT: 64

a=5

b=20

print(a+b)

OUTPUT: 5

print(a<b)

print(a>=b)

print(a<=b)

print(a!=b)

OUTPUT:

TRUE

FALSE

TRUE

TRUE

FALSE

1.2.4 Logical

a=9 b=25

print((a>b) and (a<b))

OUTPUT:

False

print((a>b) or (a<b))

OUTPUT:

True

1.2.5 Membership a="pirate"

print("a" not in a) print("a" in a)

OUTPUT:

False True

* 1. Control flow - Conditional statements

|  |  |  |
| --- | --- | --- |
| n=int(input()) | | |
| if (n>=0): |  | |
| print("positive")  else: print("negative") | | |
|  | |  |

OUTPUT: 5

Positive

OUTPUT:

n=int(input()) if(n>0): print("positive") elif(n<0):

print("negative") else:

print("neutral")

0

neutral

* 1. Control flow - Looping statements

OUTPUT:

n=int(input())

for i in range(1,11,1): print(n,'\*',i,'=',n\*i)

7

7 \* 1 = 7

7 \* 2 = 14

7 \* 3 = 21

7 \* 4 = 28

7 \* 5 = 35

7 \* 6 = 42

7 \* 7 = 49

7 \* 8 = 56

7 \* 9 = 63

7 \* 10 = 70

* 1. Data slicing

OUTPUT:

a="python is easy" print(a[::-1])

print(a[::1])

print(a[::2])

print(a[::-2])

ysae si nohtyp python is easy pto ses

ya inhy

* 1. Type Casting

OUTPUT:

print(a\*b)

#typecasting. Implicit=done by interpretor, Explicit=forceful conversion by user a=25.5

b=6

153.0

OUTPUT:

a=25.5

b=6 print(int(a\*b))

153

* 1. Collections - List

.list is collection of elements .heterogeneous .mutable

l=[]

a=int(input()) for i in range(a):

x=input()

l.append(x) print(l) l.insert(2,6) print(l)

OUTPUT:

4

34

54

32

15

['34', '54', '32', '15']

['34', '54', 6, '32', '15']

|  |  |  |
| --- | --- | --- |
| l=[4,53,25,62,'4',1,'jelo'] | | |
| l.pop(2) | |  |
| l.remove('4') | | |
| print(l) |  | |
|  |

OUTPUT:

[4, 53, 62, 1, 'jelo']

l=[4,6,3.53,32]

print(max(l))

print(len(l))

OUTPUT:

32

4

List comprehension -iterates -applies some function on every element -conditions -output:list

l=[3,5,2]

l2=[i\*\*2 for i in l if i>=3] print(l2)

print(l)

OUTPUT:

[9, 25]

[3, 5, 2]

The salaries of 5 employees in a company is taken as a list. The tax is 10% if the salary is less than or equal to 50,000 or it is 15% Create a new list with tax amount [67000,45000,89000,34000,50000]

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

tax=[]

for i in sal:

if (i<=50000):

t=0.1\*i

tax.append(t)

else:

t=0.15\*i

OUTPUT:

#list=[(body of if) if (condition) else (body of else) iterate]

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

print(tax)

tax.append(t)

print(tax)

[10050.0, 4500.0, 13350.0, 3400.0, 5000.0]

[10050.0, 4500.0, 13350.0, 3400.0, 5000.0]

2 Numpy

OUTPUT:

#importing

import numpy as np

2.1 Arrays #creating 1D array

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

<class 'numpy.ndarray'>

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

print(B)

OUTPUT: [[2 3 4]

[7 8 9]]

OUTPUT:

#creating a 3D array - rows,columns, groups C=np.array([[[1,2,3],[4,5,6]],[[9,8,7],[6,5,4]]])

print(C)

[[[1 2 3]

[4 5 6]]

[[9 8 7]

[6 5 4]]]

* + 1. Checking Dimensions

Use ndim attribute available with the NumPy array as numpy\_array\_name. ndim to get the number of dimensions. Alternatively, we can use the shape attribute to get the size of each dimension and then use len() function for the number of dimensions

OUTPUT:

print(A.ndim) print(B.ndim) print(C.ndim)

1

2

3

* + 1. Ones

Python numpy. ones() function returns a new array of given shape and data type, where the element's value is set to 1. This function is very similar to numpy zeros() function

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

print(D)

OUTPUT:

[[[1. 1.]

[1. 1.]

[1. 1.]]

[[1. 1.]

[1. 1.]

[1. 1.]]]

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

OUTPUT: [[1. 1.]

[1. 1.]

[1. 1.]]

* + 1. Zeros

To create an array of zeros in Python, you can use the np. zeros function from the numpy library. Here's a simple example: import numpy as np arr = np

OUTPUT:

F=np.zeros((4,5)) print(F)

[[0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0.]]

G=np.zeros((4,5,6)) print(G)

OUTPUT:

[[[0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0.]]

[[0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0.]]

[[0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0.]]

[[0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0.]]]

* + 1. eye

OUTPUT:

J=np.eye(4) print(J)

[[1. 0. 0. 0.]

[0. 1. 0. 0.]

[0. 0. 1. 0.]

[0. 0. 0. 1.]]

J=np.eye(4,3) print(J)

OUTPUT:

[[1. 0. 0.]

[0. 1. 0.]

[0. 0. 1.]

[0. 0. 0.]]

* 1. Arange

arange() function in Python is a powerful tool that allows you to create arrays with evenly spaced values. It is a versatile function used in various scenarios, from simple arithmetic to complex mathematical operations. This blog will explore the various applications of numpy

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

OUTPUT:

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

* + 1. Arrange with reshape

arange() This is particularly suitable when you want to create a plot in Matplotlib. That's how you can obtain the ndarray instance with the elements [0, 1, 2, 3, 4, 5] and reshape it to a

two-dimensional array.

OUTPUT:

H=np.arange(3,31,3).reshape(5,2 print(H)

[[ 3 6]

[ 9 12]

[15 18]

[21 24]

[27 30]]

# 5 table upto 1000

H=np.arange(5,1001,5).reshape(2,100) print(H)

[[ 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]] |  |

* 1. Linspace

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

OUTPUT;

print(V)

#13.3-12 = 1.3

#14.6-13.3 = 1.3 10 parts

|  |  |  |
| --- | --- | --- |
| [12. | 13.33333333 14.66666667 16. | 17.33333333 18.66666667 |
| 20. | 21.33333333 22.66666667 24. | ] |

V=np.linspace(1,20,20) print(V)

OUTPUT:

[ 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20.]

|  |  |
| --- | --- |
| 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]]

print(L+R)

OUTPUT: [[10 12 14]

[16 18 20]]

|  |  |
| --- | --- |
| G=np.sum((L,R)) | |
| print(G) |  |
|  |

OUTPUT: 90

* 1. Axis

|  |  |
| --- | --- |
| G=np.sum((L,R),axis=0) | |
| print(G) |  |
| OUTPUT:  [[10 12 14] | |
| [16 18 20]] | |

[21 23 25]]

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

OUTPUT:

OUTPUT:

[[ 5 7 9]

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

[[ 6 15]

[30 39]]

EXAMPLE:

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

OUTPUT:

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

for i in b:

print(np.sqrt(i),'square is',i)

5.0 square is 25

17.0 square is 289

19.0 square is 361

9.0 square is 81

* 1. Array Joints

Joining means putting contents of two or more arrays in a single array. In SQL we join tables based on a key

**print("\n") print(np.vstack((a,b))) #columns print("\n") print(np.hstack((a,b))) #rows**

**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]]

[[34 35 36]

[37 38 39]

[ 4 5 6]

[ 7 8 9]]

[[34 35 36 4 5 6]

[37 38 39 7 8 9]]

* 1. Random and Rand

NumPy random. rand() function in Python is used to return random values from a uniform distribution in a specified shape. This function creates an array of the given shape and it fills with random samples from the uniform distribution.

### EXAMPLE:

#### 1)

A=np.random.rand(1) print(A)

**OUTPUT: [0.46584502]**

## 2)

A=np.random.rand(8,4)#rand range is b/w 0 and 1 print(A)

##### OUTPUT:

[[0.09907899 0.1598259 0.71504942 0.34729956]

[0.84639197 0.14582284 0.68671143 0.42395321]

[0.45224266 0.99970248 0.04310802 0.58339377]

[0.542621 0.91284503 0.40066622 0.60989855]

[0.19722872 0.21018408 0.32868864 0.53519215]

[0.46477612 0.43323041 0.49110677 0.14027071]

[0.62703084 0.12187961 0.10166665 0.30419094]

[0.86395801 0.93728822 0.1887755 0.18869618]]

#### 3)

a=np.floor (10\*np.random.rand(8,4))#rand range is b/w 0 and 1 print(a)

###### OUTPUT: [[6. 6. 8. 8.]

[3. 6. 9. 4.]

[7. 1. 7. 1.]

[7. 7. 0. 4.]

[2. 3. 1. 5.]

[8. 1. 2. 7.]

[3. 4. 7. 5.]

[0. 6. 9. 0.]]

**SPLITTING AN ARRAY:Splitting is reverse operation of Joining.Joining merges multiple arrays into one and Splitting breaks one array into multiple.We use array\_split() for splitting arrays, we pass it the array we want to split and the number of splits**

### EXAMPLE:

#### 1)

np.split(a,(2,7))#split after 2nd line and again split after 7th line

###### OUTPUT:

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

[5, 6, 7, 8]]),

array([[ 9, 10, 11, 12],

[13, 14, 15, 16],

[17, 18, 19, 20],

[21, 22, 23, 24],

[25, 26, 27, 28]]),

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

## VSPLIT:

vsplit. numpy. vsplit is a special case of split() function where axis is 1 indicating a vertical split regardless of the dimension of the input array.

### EXAMPLE:

np.vsplit(a,4)#split can be done only when the given num is a multiple of number of rows

###### 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]])]

### EXAMPLE QUESTION:

creating an array with four rows and 8 coloumns:

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

###### 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]]

### HSPLIT:

**hsplit() function split an array into multiple sub-arrays horizontally (column-wise). hsplit is equivalent to split with axis=1, the array is always split along the second axis regardless of the array dimension**

### EXAMPLES:

#### 1)

np.hsplit(b,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]])]

#### 2)

np.hsplit(b,(2,6))#splits after second and fifth

###### OUTPUT:

[array([[ 1, 2],

[ 9, 10],

[17, 18],

[25, 26]]),

array([[ 3, 4, 5, 6],

[11, 12, 13, 14],

[19, 20, 21, 22],

[27, 28, 29, 30]]),

array([[ 7, 8],

[15, 16],

[23, 24],

[31, 32]])]

#### 3)

np.hsplit(b,(2,3,6))#splits after 2nd and 3rd and 6th coloumns

###### OUTPUT:

[array([[ 1, 2],

[ 9, 10],

[17, 18],

[25, 26]]),

array([[ 3],

[11],

[19],

[27]]),

array([[ 4, 5, 6],

[12, 13, 14],

[20, 21, 22],

[28, 29, 30]]),

array([[ 7, 8],

[15, 16],

[23, 24],

[31, 32]])]

### NUMPY PI:

**NumPy constants are the predefined fixed values used for mathematical calculations. For example, np. pi is a mathematical constant that returns the value of pi (π), i.e.**

**3.141592653589793. Using predefined constants makes our code concise and easier to read.**

### EXAMPLES:

#### 1)

np.pi

**OUTPUT: 3.141592653589793**

#### 2)

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

###### OUTPUT:

[0.7853981633974483, 1.0471975511965976, 1.5707963267948966, 3.141592653589793]

### RANDOM:

The random is a module present in the NumPy library. This module contains the functions which are used for generating random numbers. This module contains some simple random data generation methods, some permutation and distribution functions, and random generator functions.

### DEGREE:

numpy. degrees() and rad2deg() in Python

* **Syntax : numpy.degrees(x[, out]) = ufunc 'degrees') Parameters :**
* **array : [array\_like] elements are in radians. out : [ndaaray, optional] Output array of same shape as x. 2pi Radians = 360 degrees.**
* **Return : An array with degree values in place of radian values**

### EXAMPLES:

#### 1)

b=np.rad2deg(A)#converts radians to degree print(b)

###### OUTPUT:

**[0.7853981633974483, 1.0471975511965976, 1.5707963267948966, 3.141592653589793]**

## 2)

b=np.rad2deg(A)#converts radians to degree print(b)

###### OUTPUT:

**[ 45. 60. 90. 180.]**

## 3)

**np.deg2rad(b)**

##### OUTPUT:

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

### SIN:

**sin() in Python. numpy. sin(x[, out]) = ufunc 'sin') : This mathematical function helps user to calculate trigonometric sine for all x(being the array elements).**

### EXAMPLE:

np.sin(1)

###### OUTPUT:

0.8414709848078965

### COS:

**The numpy. cos() method in Python calculates the ratio of the length of the side nearest to the angle to the length of the hypotenuse. This function returns the cosine of a number. To be more precise, it returns the cosine of a radian.**

### EXAMPLE:

np.cos(1)

**OUTPUT: 0.5403023058681398 TAN:**

**tan() in Python. numpy. tan(array[, out]) = ufunc 'tan') : This mathematical function helps user to calculate trigonometric tangent for all x(being the array elements).**

### EXAMPLE:

np.tan(1)

###### OUTPUT:

1.5574077246549023

**MEAN:**

**NumPy array mean() function in Python is used to compute the arithmetic mean or average of the array elements along with the specified axis or multiple axis**

**EXAMPLE:**

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

np.mean(st) OUTPUT: 46.5

**MEDIAN:**

numpy. median(arr, axis=None, out=None)

* arr: array\_like This will be our input array to perform median method.
* axis: None or int or tuple of ints, optional Axis on which we perform the arithmetic median if specified. ...
* out: ndarray(optional) Used for defining an alternative output array in which the result is placed.

**EXAMPLE:**

**np.median(st) OUTPUT:**

**39.5**

**SQUARE ROOT:**

**You can use NumPy to calculate the square root of elements in an array using the numpy. sqrt()**

**function. This function is used to return the non-negative square root of an array element-wise (for each element of the array).**

### EXAMPLE:

np.std(st)#std=sqrt((sigma(xi-x)^2)/n) where xi=individual elementt and x=mean n= no of elements

###### OUTPUT:

24.452334585202017

### VARIABLE(VAR):

var() in Python. numpy. var(arr, axis = None) : Compute the variance of the given data (array elements) along the specified axis(if any). This Result is Variance.

### EXAMPLE:

#### 1)

np.var(st)#var=(std)^2

###### OUTPUT:

**597.9166666666666**

#### 2)

**print(c)**

**c=np.arange(1,5).reshape(2,2)**

###### OUTPUT:

[[1 2]

[3 4]]

#### LINERA ALZEBRA(linalg):

The NumPy linear algebra functions rely on BLAS and LAPACK to provide efficient low level implementations of standard linear algebra algorithms

##### EXAMPLE: 1)

np.linalg.inv(c)#linear alzebra is a subpackage in numpy

###### OUTPUT:

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

[ 1.5, -0.5]])

### ARGUMENT MAXIMUM NUMBER:

NumPy argmax() in Python is used to return the indices of the max elements of the given array along with the specified axis. Using this function gets the indices of maximum elements of single dimensions and multi-dimensional(row-wise or column-wise) of the given array

### EXAMPLE:

**1)**

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

print(np.argmax(c))#argmax will provide u the index of highest number in each row or coloumn in the given matrix

###### OUTPUT:

23

# 2)

**print(np.argmax(c))**

**OUTPUT: 23**

# 3)

**print(d)**

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

##### OUTPUT:

[[4.07751998 8.72834794 8.97873355 5.95557456]

[6.58553538 2.38379297 9.23355193 9.42305738]

[8.50221714 3.01282204 2.4185026 2.45751048]

[6.46190302 9.19893241 4.35219561 3.28589104]

[9.8770008 4.53912116 5.75201058 4.9960637 ]

[0.1713341 3.35536689 2.15546352 3.97426257]]

We use the function argmax() with the parameter axis that can be defined as follows:

* **axis=0 , the index of the max element per column will be returned.**
* **axis=1 the index of the max element per row will be returned**

### EXAMPLE: 1)

[2 3 0 1 0 3]

**number in each individual row in the given matrix**

**print(np.argmax(d,axis=1))#argmax with axis1 will provide u the index of the highest**

#### 2)

**number in each individual coloumn in the given matrix**

**OUTPUT:**

**print(np.argmax(d,axis=0))#argmax with axis 0 will provide u the index off the highest**

[4 3 1 1]

#### ARGUMENT MINIMUM NUMBER(argmin()):

* If axis = None , the array is flattened and the index of the flattened array is returned.
* If axis = 0, the index of the smallest element in each column is returned.
* If axis = 1, the index of the smallest element in each row is returned.

### EXAMPLE:

**1)**

#we can usemin as like argmax

print(np.argmin(d,axis=1)) arg

### OUTPUT:

[0 1 2 3 1 0]

**2)**

Print(np.argmin(d,axis=0))

### OUTPUT:

[5 1 5 2]

# FLOOR:

floor() function in NumPy is used to return the floor values of each element in an input array x such that for each of the elements of x , the floor values are less than or equal to it.

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

###### OUTPUT:

[[4. 8. 6. 3.]

[3. 8. 4. 7.]

[5. 0. 1. 8.]

[8. 3. 2. 0.]

[4. 1. 4. 2.]

[4. 7. 0. 9.]]

### SEARCHING:

numpy.searchsorted(): The function is used to find the indices into a sorted array arr such that, if elements are inserted before the indices,

#searching the indices based on the conditions

#finding indixes where the values are even a=np.array([34,56,17,89,91])

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

(array([0, 1]),)

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

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

### OUTPUT:

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

### SORTED LIST SEARCHING:

Search Sorted

There is a method called searchsorted() which performs a binary search in the array, and returns the index where the specified value would be inserted to maintain the search order. The searchsorted() method is assumed to be used on sorted arrays.

#searching is done only for sorted list a=np.array([6,7,8,9,10]) x=np.searchsorted(a,10)

print(x)

###### OUTPUT: 4

**OUTPUT:**

#unsortedlist b=np.array([34,12,7]) x=np.searchsorted(b,7) print(x)

0

### SORTING ARRAYS:

The NumPy ndarray object has a function called sort() , that will sort a specified array.

* **ExampleGet your own Python Server. Sort the array: import numpy as np. ...**
* **Sort the array alphabetically: import numpy as np. ...**
* **Sort a boolean array: import numpy as np. ...**
* **Sort a 2-D array: import numpy as np.**

arr=np.array(['banana','cherry','apple']) ar=np.array([True,False,True])#value for false=0 and value for true=1 print(np.sort(arr))

print(np.sort(ar))

###### OUTPUT:

['apple' 'banana' 'cherry'] [False True True]

#2d arrayL:sorting happens within a row arr=np.array([[3,2,4],[5,0,1]])

print(np.sort(arr))

###### OUTPUT:

[[2 3 4]

[0 1 5]]

### ARRAY FILTERS:

Getting some elements out of an existing array and creating a new array out of them is called filtering. In NumPy, you filter an array using a boolean index

#array filters 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]

arr=np.array([40,43,50,44,67,78]) filt=np.where(arr%2==0)

print(filt)

OUTPUT: (array([0, 2, 3, 5]),)

arr[filt]

###### OUTPUT:

array([40, 50, 44, 78])

### ITERATING THROUGH 2 ARRAYS:

Suppose the first operand is one dimensional and the second operand is two dimensional. The iterator will have three dimensions, so op\_axes will have two 3-element lists. The first list picks out the one axis of the first operand, and is -1 for the rest of the iterator axes, with a final result of [0, -1,

-1].

#iterating through two arrays

names=np.array([" harsha","sada","rohith","josh","abhi"])

initials=np.array(["b","m","u","v","l"]) for i,j in zip(initials,names):

print(i,".",j)

###### OUTPUT:

b . harsha m . sada

u . rohith v . josh

l . abhi

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

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

n1=np.multiply(a1,a2) print(n1)

###### OUTPUT:

[ 200 420 660 920 1200 1500]

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

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

n1=np.divide(a1,a2) print(n1)

###### OUTPUT:

[0.5 0.95238095 1.36363636 1.73913043 2.08333333 2.4 ]

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

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

n1=np.mod(a1,a2) print(n1)

###### OUTPUT:

[10 20 8 17 2 10]

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

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

n1=np.divmod(a1,a2) print(n1)

###### OUTPUT:

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

### LOGARITHMS:

Python's numpy. log() is a mathematical function that computes the natural logarithm of an input array's elements. The natural logarithm is the inverse of the exponential function, such that log (exp(x)) = x.

a1=1.2

print(np.log(a1))#natural loge

###### OUTPUT: 0.2630344058337938

a1=1.2

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

###### OUTPUT:

0.07918124604762482

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

print(np.log(a))

###### OUTPUT:

[0. 0.69314718 1.09861229 1.38629436 1.60943791 1.79175947

1.94591015]

### CUMULATIVE PRODUCT:

* Overview. The numpy. cumprod() function in NumPy computes the cumulative product of the elements in a given input array over a given axis.
* Syntax. numpy.cumprod(a, axis=None, dtype=None, out=None)
* Parameter values. The numpy. ...
* Return value. The numpy. ...
* Code example.

#cummulative product a1=np.array([5,6,7,10]) x=np.cumprod(a1)

print(x)

###### OUTPUT:

**[ 5 30 210 2100]**

#### 2)

**a1=np.array([5,6,7,10]) x=np.cumsum(a1) print(x)**

###### OUTPUT:

[ 5 11 18 28]

### DIFFERENCE:

diff(arr[, n[, axis]]) function is used when we calculate the n-th order discrete difference along the given axis. The first order difference is given by out[i] = arr[i+1] – arr[i] along the given axis. If we have to calculate higher differences, we are using diff recursively.

#difference

a2=np.array([10,15,25,15]) new=np.diff(a2) print(new)

**OUTPUT: [ 5 10 -10]**

### LCM:

lcm() in Python. numpy. lcm(arr1, arr2, out = None, where = True, casting = 'same\_kind', order = 'K', dtype = None) : This mathematical function helps user to calculate lcm value of

|arr1| and |arr2| elements. Parameters : arr1 / arr2 : [array\_like]Input array

#lcm n1=455 n2=665

x=np.lcm(n1,n2)

print(x)

###### OUTPUT:

8645

### GCD:

**numpy. gcd() returns the greatest common divisor of the absolute values of the input. The return type is either ndarray or scaler , depending on the input type.**

### EXAMPLES:

#### 1)

#gcd n1=455 n2=665

x=np.gcd(n1,n2)

print(x)

**OUTPUT: 35**

#### 2)

#finding gcd for arrays of elements a=np.array([12,15,60])

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

print(gc)

**OUTPUT: 3**

#### 3)

#finding gcd for arrays of elements a=np.array([12,15,60])

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

print(gc)

60

**Plotting:**

Ploting is used to draw diagrams using plot()

import numpy as np

import matplotlib.pyplot as plt #matplotlib is a sub package in numpy

#Problem1:

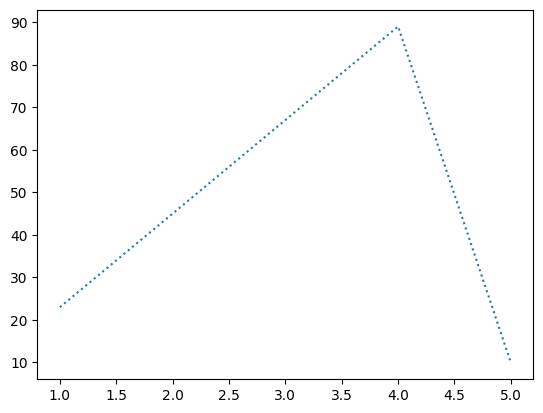
a=[23,45,67,89,10]#carona cases in first days

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

plt.plot(b,a,linestyle=":")#plot(x,y)

plt.show()

Output:



#Problem 2:

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

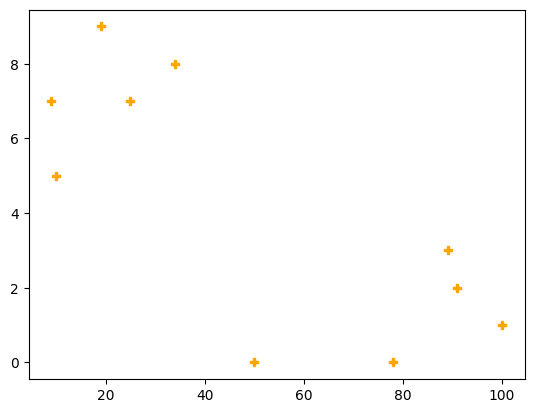
runs=np.array([100,50,91,78,89,25,34,19,9,10])

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

plt.scatter(runs,wickets,color='orange',marker="P")

plt.show()

Output:



COLOUR CODES: \*r-red \*g-green \*b-blue \*c-cyan \*m-magenta \*y-yellow \*k-black \*w-white

MARKER:

o circle, \* star, . point , , pixel,x x,X x(filled),+ plus,P plus(filled),s square,D diamond,d diamond(thin),p pentagon,H hexagon,h hexagon,v triangle down,^ triangle up,< triangle right,>triangle left

LINE STYLES: i)SolId **‘-‘ ii)**Dashed **‘:’ iii)**Dotted **‘–‘ iv)**Dashdot **‘-.’**

#Problem 3

show the scores of 2 students in 5 different subjects telugu,hindi,english,maths,science as x axis and marks as y stu1=[56,78,92,95,78] stu2=[77,89,91,23,45]

#code

stu1=[56,78,92,95,78]

stu2=[77,89,91,23,45]

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

plt.plot(sub,stu1,label="student1",color='b')

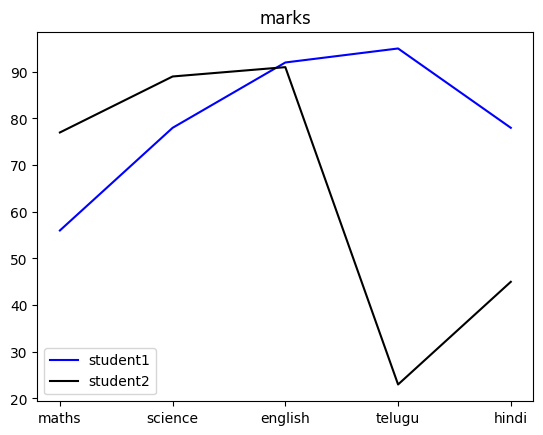
plt.plot(sub,stu2,label="student2",color='k')

plt.title("marks")

plt.legend()#legend() will display the labels in a box

plt.show()

Output:



#code to plot sub graphs in a graph using given data

stu1=[56,78,92,95,78]

stu2=[77,89,91,23,45]

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

plt.subplot(2,1,1)#subplot(no.of.rows,no.of.coloumns,postion of graph)

plt.plot(sub,stu1,label="student1",color='b')

plt.legend()

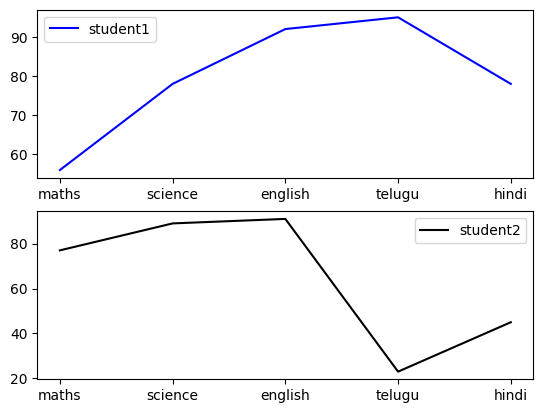
plt.subplot(2,1,2)

plt.plot(sub,stu2,label="student2",color='k')

plt.legend()

plt.show()

Output:



#Problem 4

year=np.array(["2019-2020","2020-2021","2021-2022","2022-2023"])

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

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

profita=np.diff(revenueofa)

profitb=np.diff(revenueofb)

print("revenue of company A",revenueofa)

print("revenue of company B",revenueofb)

print("profit of A",profita)

print("profit of B ",profitb)

plt.subplot(1,2,1)

plt.bar(year,profita,label="company A",color='k')

plt.legend(loc="best")#it will gives you the best location for legend box or we can use left and right options to place

plt.subplot(1,2,2)

plt.bar(year,profitb,label="company B",color='g')

plt.legend(loc="best")

plt.show()

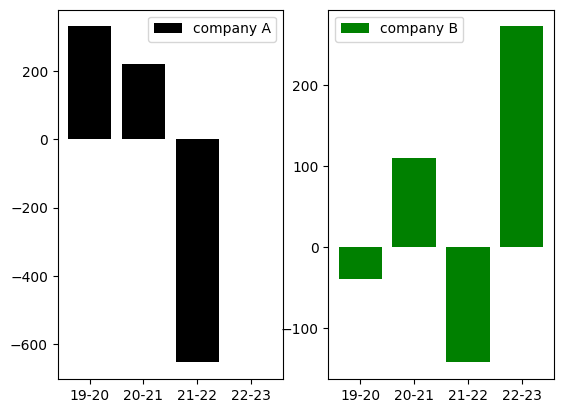
Output:

revenue of company A [230 560 780 127 128]

revenue of company B [200 160 270 127 400]

profit of A [ 330 220 -653 1]

profit of B [ -40 110 -143 273]



#Problem 5

Convert the given data into pie chart

#code1

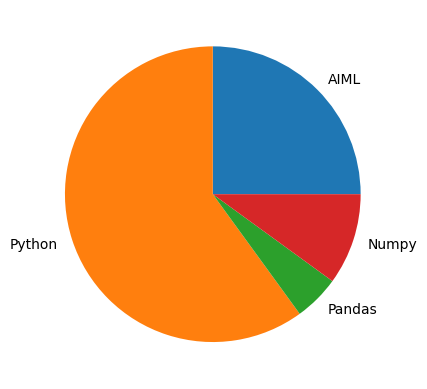
a=np.array([25,60,5,10])

lab=["AIML","PYTHON","Pandas","Numpy"]

plt.pie(a,labels=labe)

plt.show()

Output:



#code2

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

lab=["AIML","PYTHON","Pandas","Numpy"]

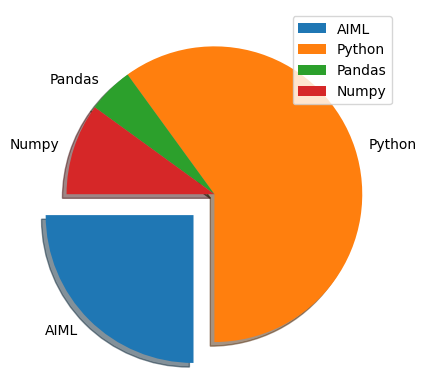
explo=[0.2,0,0,0]#explo helps to extract particular part

plt.pie(a,labels=labe,explode=explo,startangle=180,shadow=True)#division starts from given angle and shadow gives shadow to diagram

plt.legend(loc="best")

plt.show()

Output:



1. PANDAS

Pandas: -

* + Used for data manipulation (data cleaning, organising data)
  + Creates data frames from excel,csv,txt,
  + Dataframes (rows and columns readable by python)
  + Data cleaning by dropping or replacing
  + visualising data

3.1.1 Series

import pandas as pd

Series -1 column with value -2nd column contains index

|  |  |  |
| --- | --- | --- |
| names=['suji','aashi','ramu','pavani','meghana'] | | |
| index=[40,42,42,43,45] | |  |
| seri=pd.Series(names,index)  print(seri) | | |
|  |  | |

40 suji

OUTPUT:

42 aashi

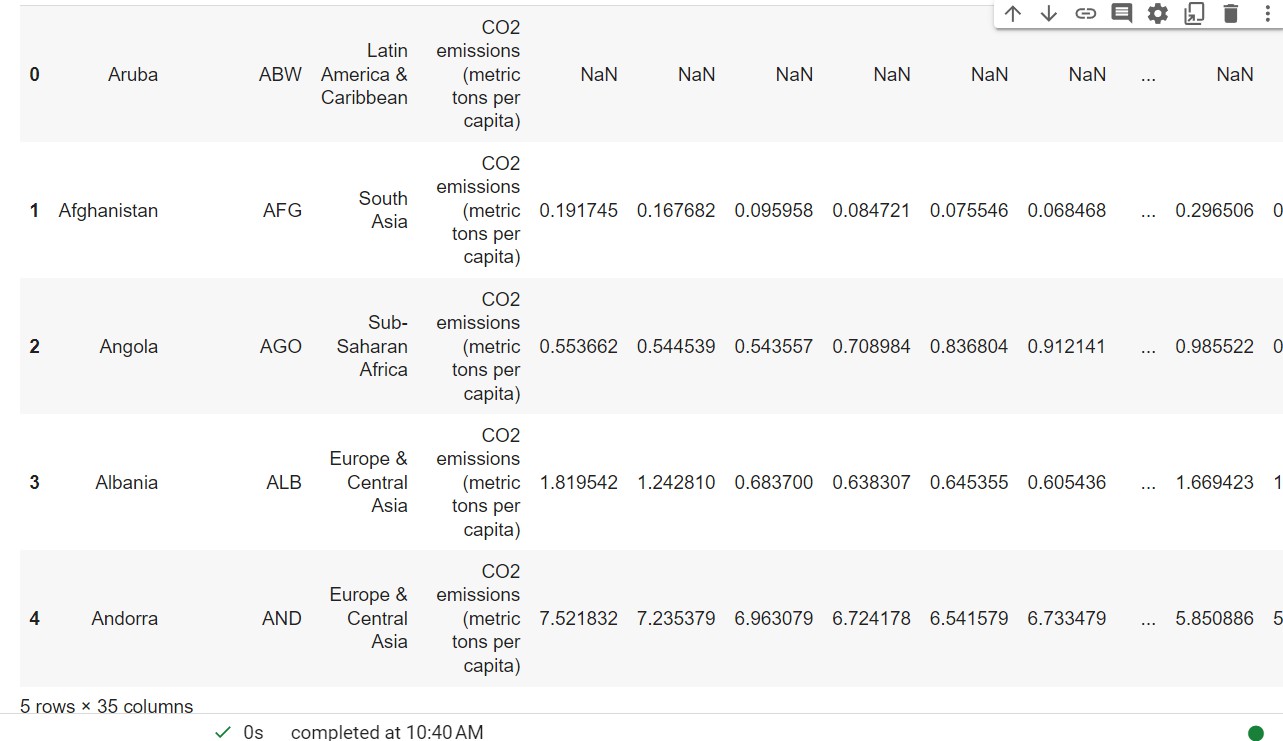
42 ramu

43 pavani

45 meghana dtype: object

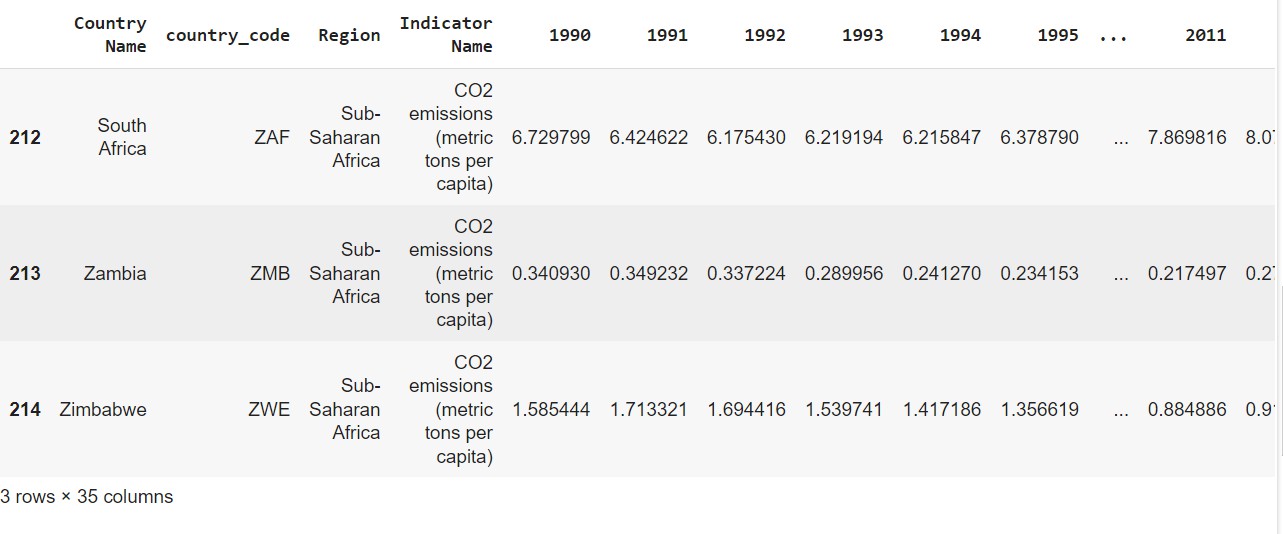
* + 1. Importing ﬁles
       - for csv and txt : read\_csv ('ﬁle path')
       - for excel : read\_excel ('ﬁle path')

df=pd.read\_csv("/content/CO2\_emmision.csv") #df=data frame df.head()



OUTPUT:

df.tail(3)



OUTPUT:

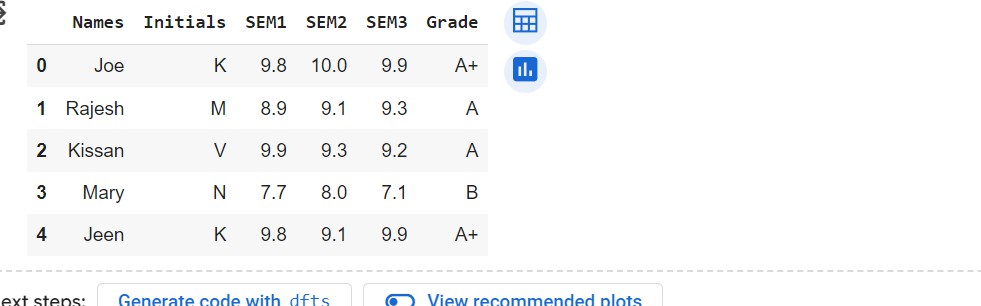
|  |  |
| --- | --- |
| dft =pd.read\_csv("/content/grades\_withnulls.txt") | |
| dft.head() |  |
|  |

OUTPUT:



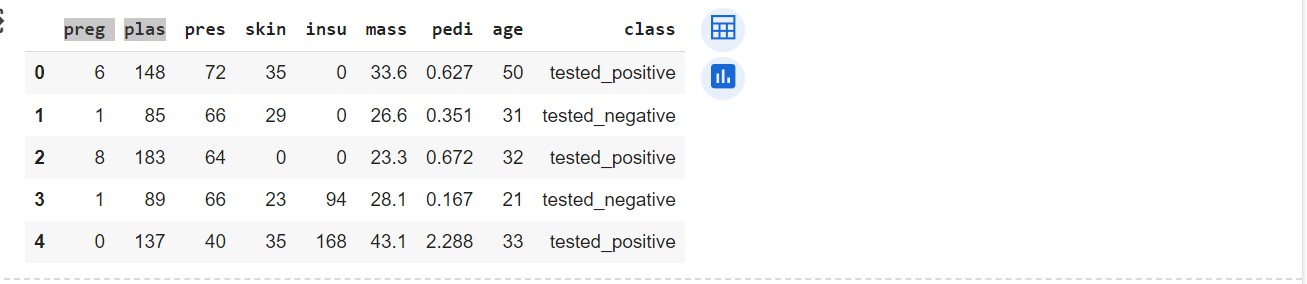
dfts =pd.read\_csv("/content/grades.txt",sep=" ") dfts.head()

OUTPUT:



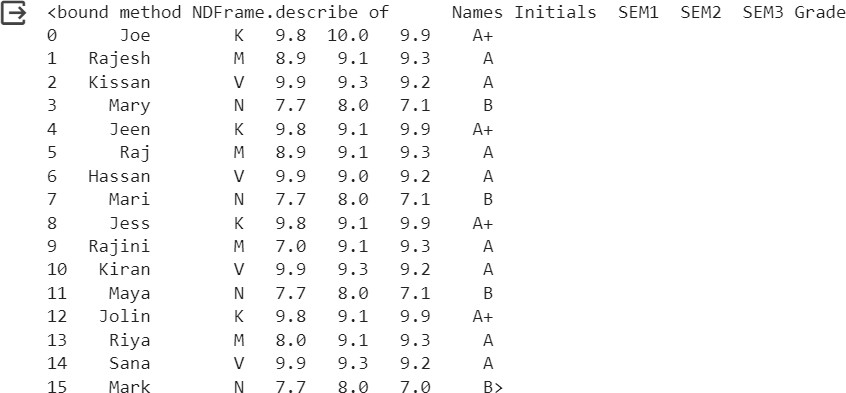
dfe=pd.read\_excel("/content/diabetes.xlsx") dfe.head()

OUTPUT:



print(dfts.describe)

OUTPUT:



* + 1. Shape

print(dfts.shape)

print(dfts.shape[0]) #display no of rows only print(dfts.shape[1]) #displays no of columns

OUTPUT:

(16, 6)

16

6

* + 1. Column

print(dfts.columns) #displays the column names

OUTPUT;

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

* 1. Accessing Data

iloc - accepts only index

loc - accepts column name and index

3.2.1

print(dfts[2:5]) #to access rows 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+ |

|  |  |
| --- | --- |
| print(dfts.loc[2:5,"Names"]) #rows of specified columns | |
| OUTPUT: |  |
|  |

1. Kissan
2. Mary
3. Jeen
4. Raj

Name: Names, dtype: object

print(dfts.iloc[2:5]) #rows of specified columns 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+ |

Names Initials SEM1

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

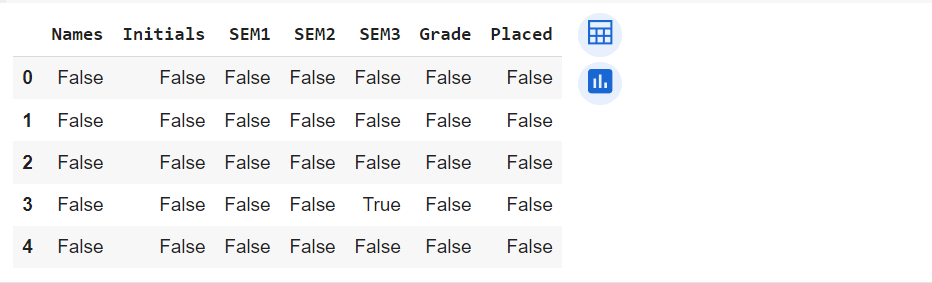
|  |  |  |  |
| --- | --- | --- | --- |
| 2 | Kissan | V | 9.9 |
| 3 | Mary | N | 7.7 |
| 4 | Jeen | K | 9.8 |

3.2.2

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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

dfn.isnull().head()



OUTPUT:

3rd row and 5th column in the data earlier it was given NaN which means Not a Null.But now, executing this command, at that cell it is showing True (it means that at that particular cell we have a null) while keeping the rest as False

dfn.isnull().sum() #to view total no of nulls in each column #NOTE: True=1

OUTPUT:

Names 0

Initials 0

SEM1 3

SEM2 0

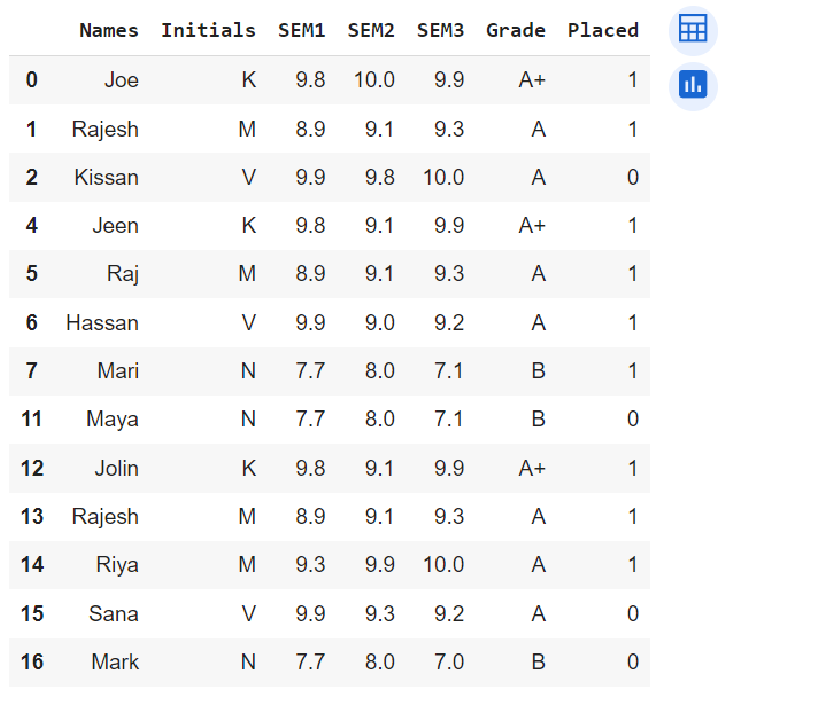
SEM3 1

Grade 0

Placed 0

dtype: int64

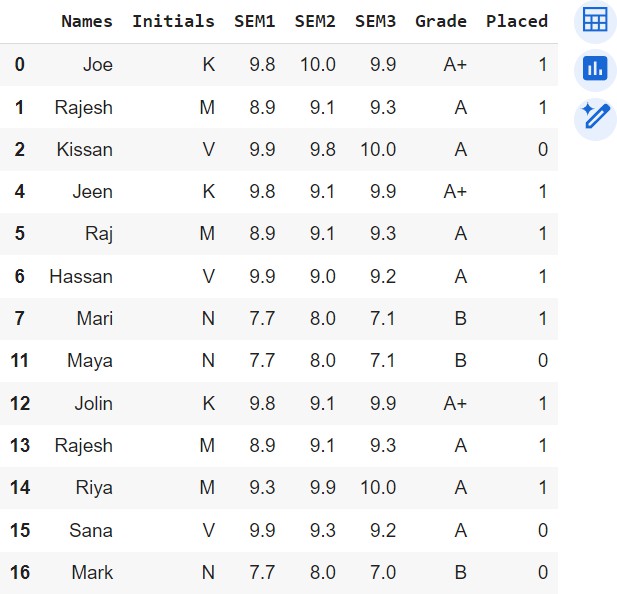
* 1. Drop NULL values

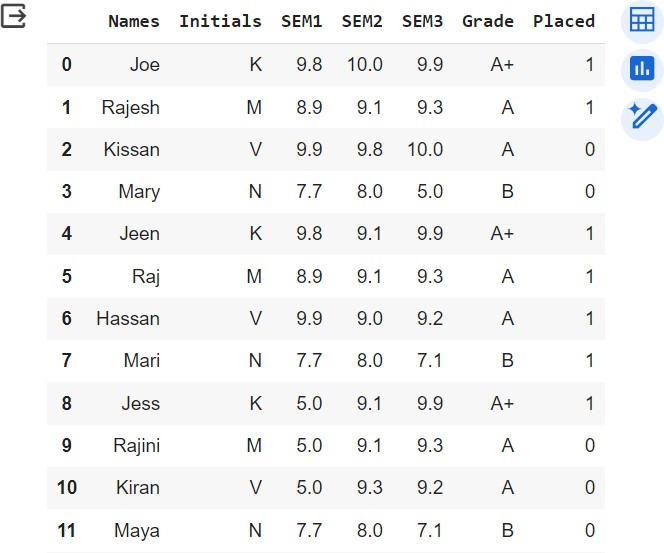


dfn.dropna() #dropping all the rows with nulls OUTPUT:

dfc=dfn.dropna() #to make it permanent dfc

OUTPUT:





dfc

OUTPUT:

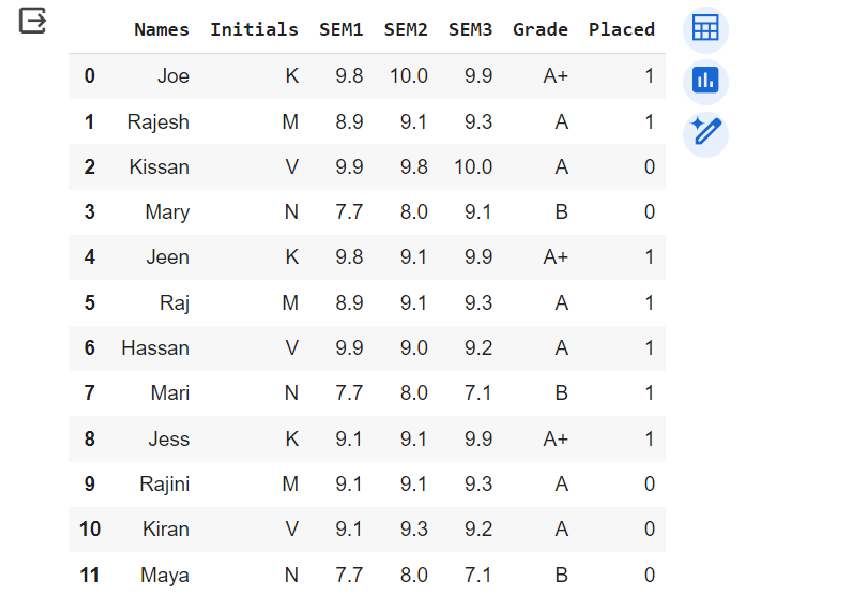
dfc=dfn.fillna(5) #fill null values to 5

* + 1. Cleaning/Replacing with mean

|  |  |  |
| --- | --- | --- |
| m=dfn['SEM3'].mean() | | |
| print(m) | |  |
| OUTPUT: |  |
|  |

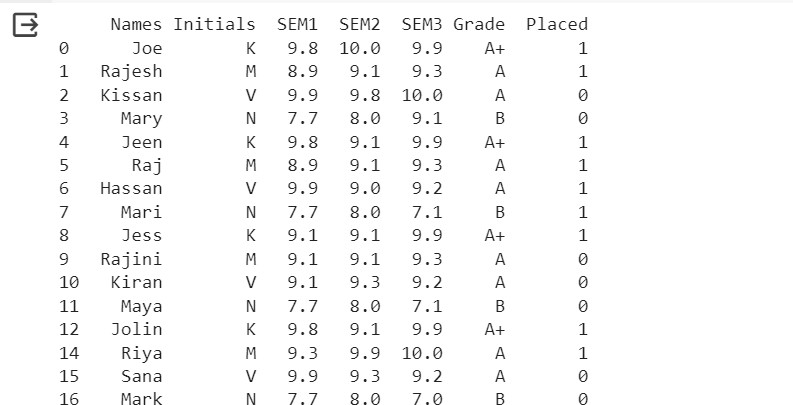
9.100000000000001

dfc2=dfn.fillna(m) dfc2



OUTPUT:

* + 1. To drop duplicates dropped=dfc2.drop\_duplicates()

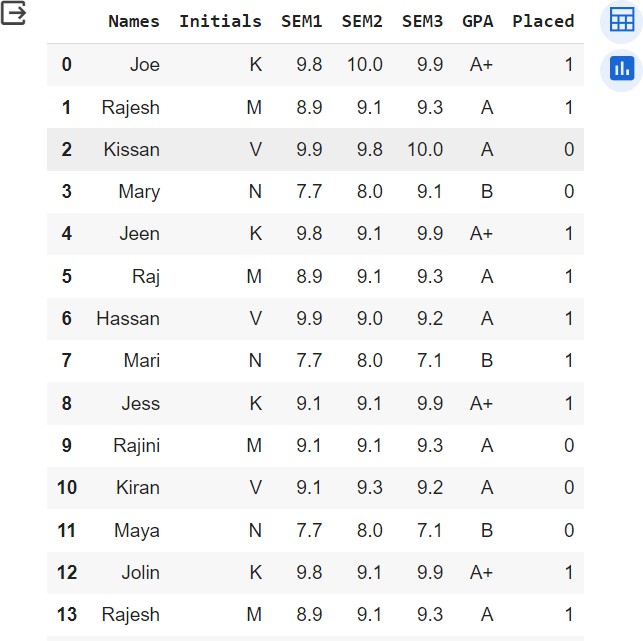


OUTPUT:

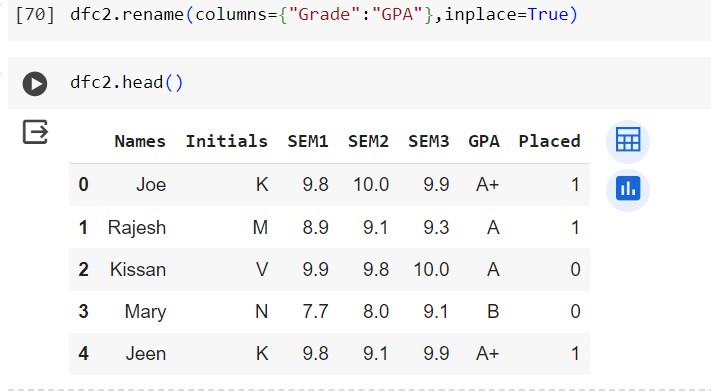
print(dropped)

OUTPUT:

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



In order to save the changes:

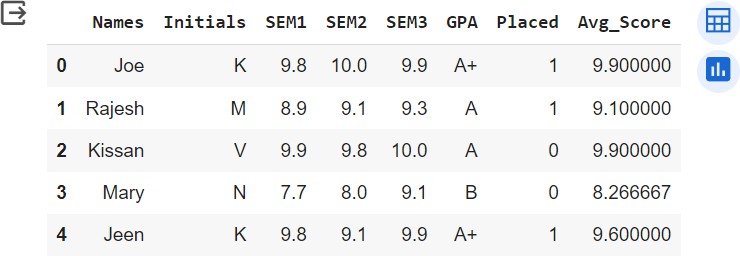


#diff b/w copy and view:

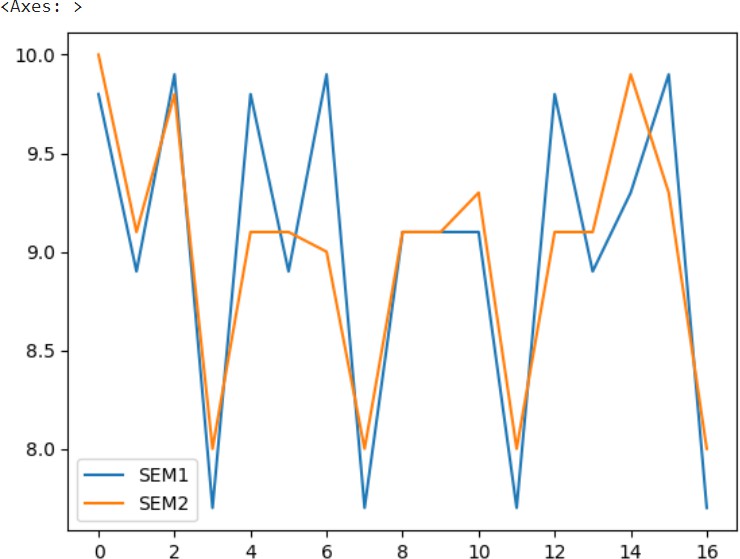
* 1. To add a new column

dfc2['Avg\_Score'] = (dfcs['SEM1'] + dfcs['SEM2'] + dfcs['SEM3'])/3 #ds['new column']=values

dfc2.head() OUTPUT:



* 1. Plotting

3.5.1

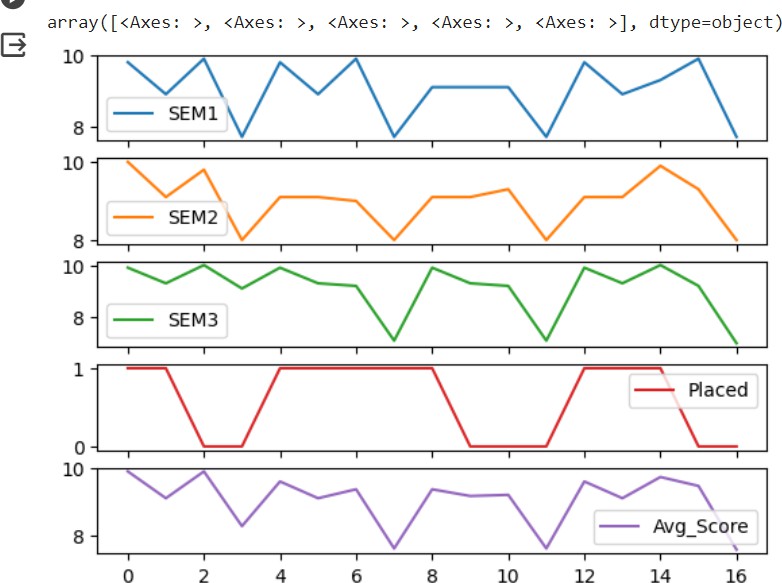
OUTPUT:

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

* + 1. Subplots

dfc2.plot.line(subplots=True)

OUTPUT:



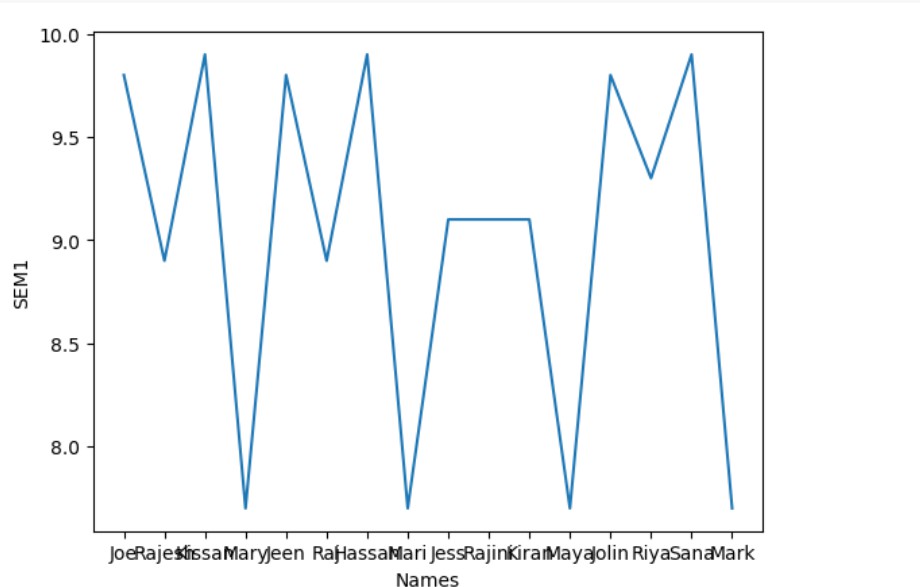
* 1. Seaborn

It is mostly used for statistical plotting

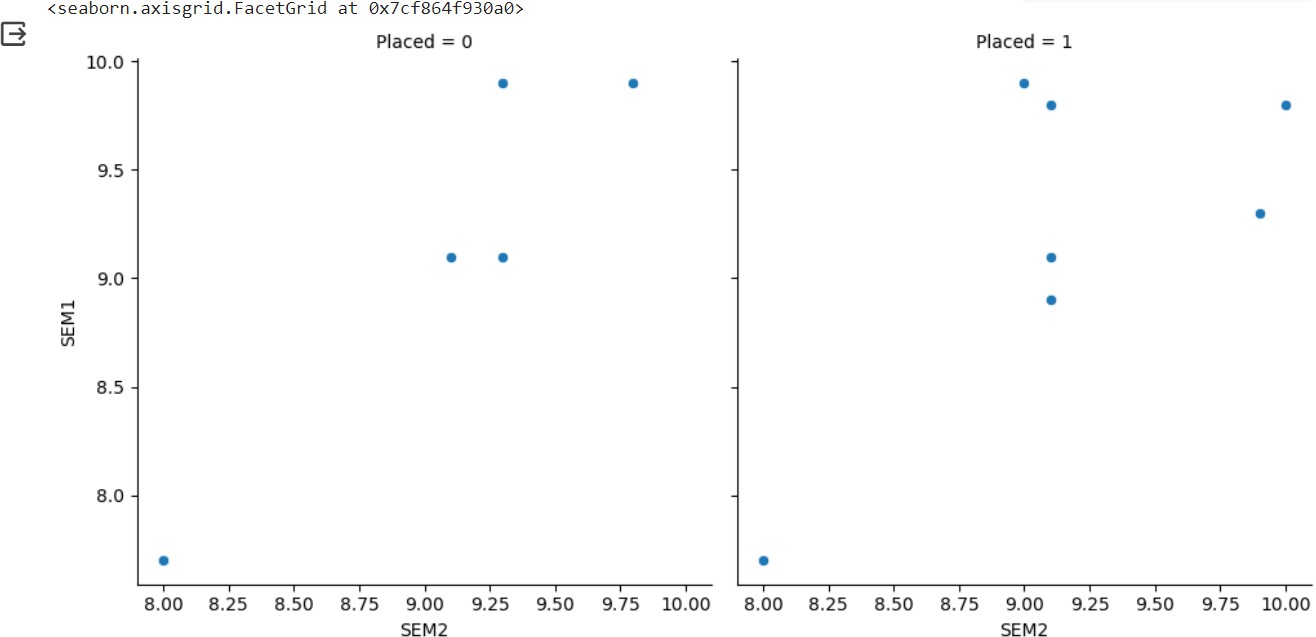
OUTPUT:

import seaborn as sns

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



sns.relplot(data=dfc2,x='SEM2',y='SEM1',col='Placed')



OUTPUT;

Problem:

Load diabetes.csv

create a relplot with age in the x axis and class as columns

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

OUTPUT:

preg plas pres skin insu mass pedi age class

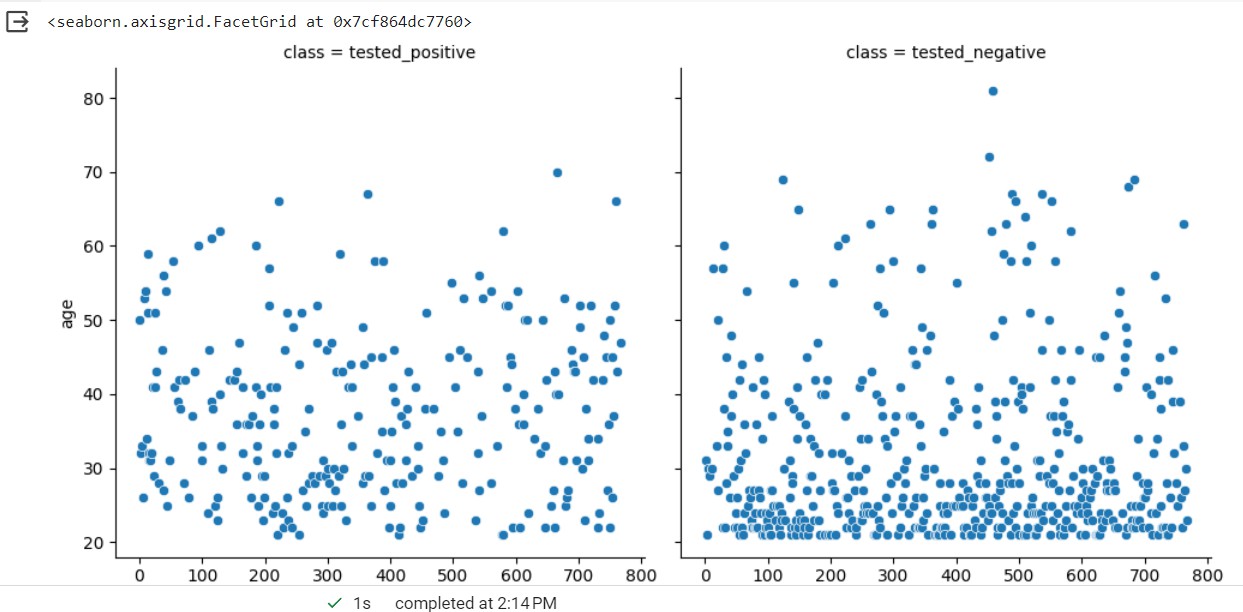
**0** 6 148 72 35 0 33.6 0.627 50 tested\_positive

**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



dfd['Index'] = range(0,768) sns.relplot(

data=dfd,

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

)

OUTPUT:

3.6

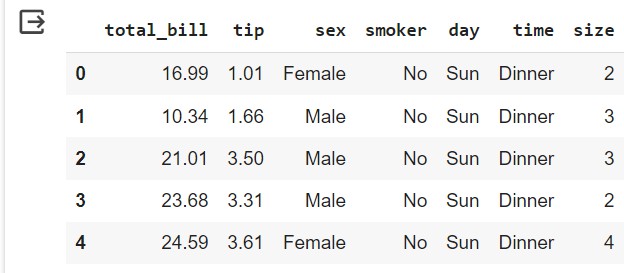
seaborn inbuilt data sets

|  |  |
| --- | --- |
| tips dowjones | |
| fmri |  |
| dots |
| healthexp  load\_dataset("dataset name") | |

3.6.1 tipa

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

OUTPUT:



* + 1. dowjones



OUTPUT:

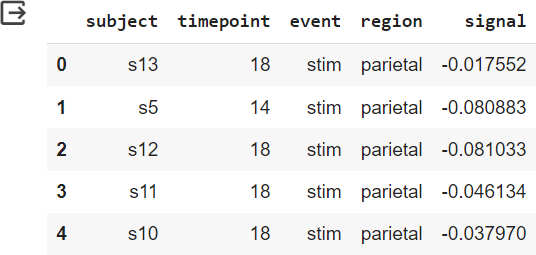
x1=sns.load\_dataset("dowjones")

x1.head()

* + 1. fmri

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

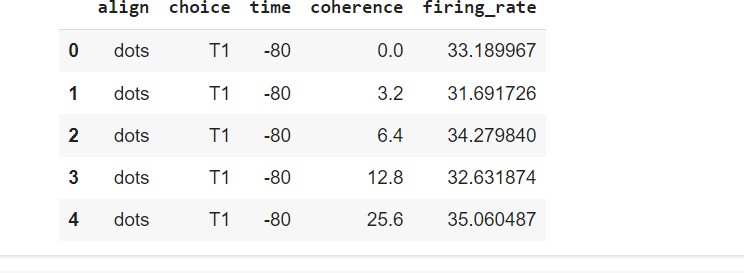
OUTPUT:



* + 1. dots

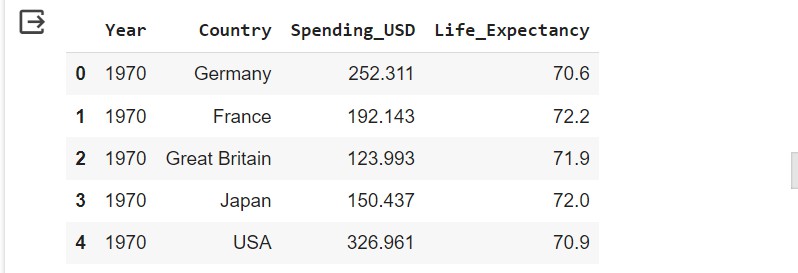
dots=sns.load\_dataset("dots") dots.head()

OUTPUT:



* + 1. healthexp

health=sns.load\_dataset("healthexp") health.head()

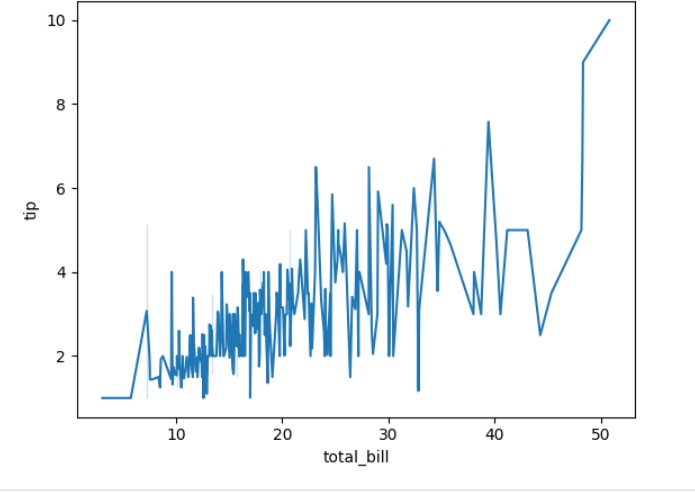


OUTPUT:

* 1. Plotting
     1. Lineplot

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

OUTPUT:

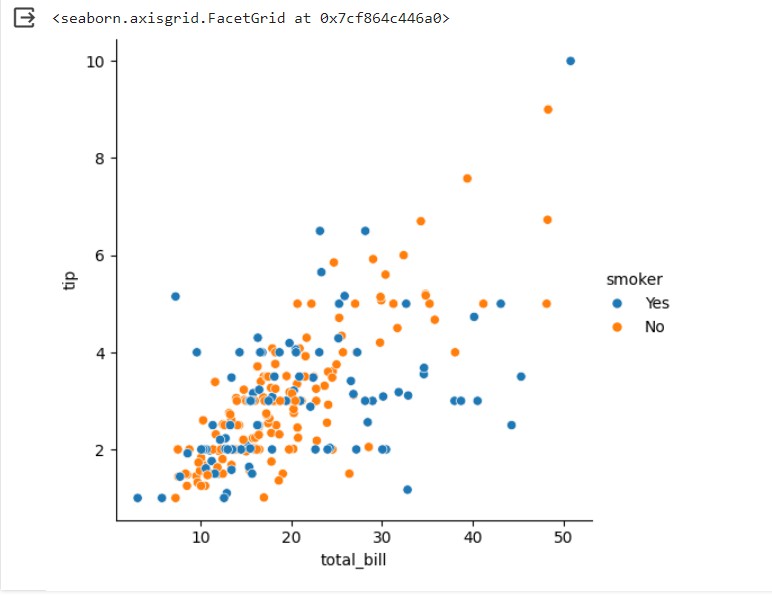


* + 1. Relplot

It is used to create difference based on a column via colors

Example 1:

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

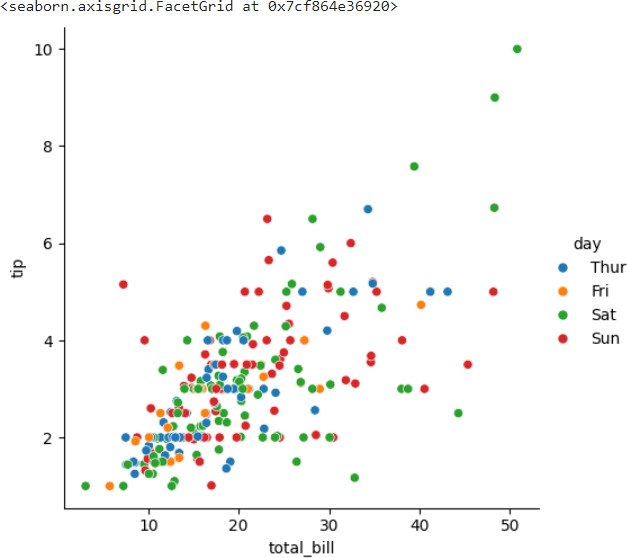


OUTPUT:

Example 2:

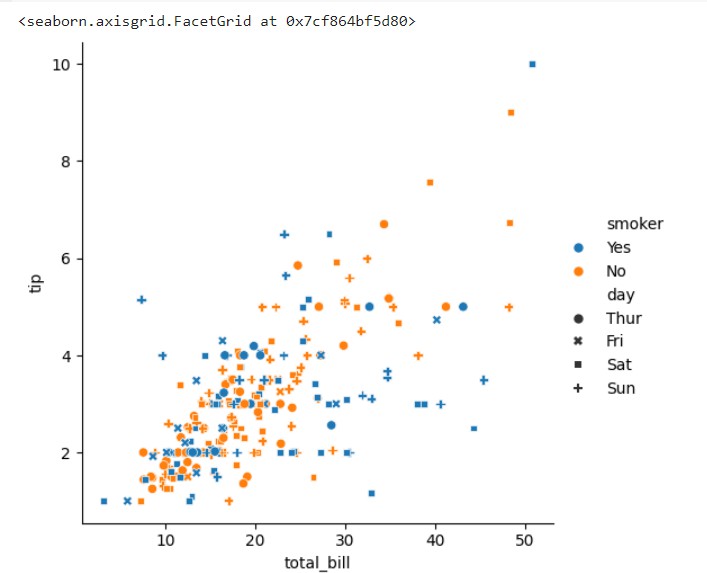
OUTPUT:

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

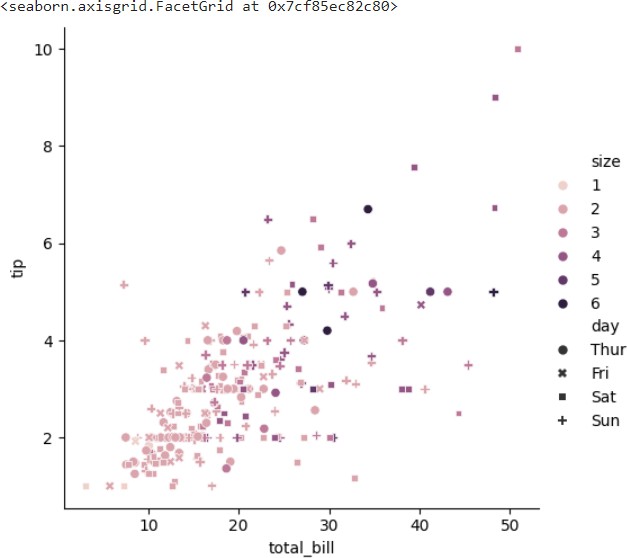


style = difference markers for different category hue = different color for different category

Example 1 sns.relplot(data=tips,x='total\_bill',y='tip',hue='smoker',style='day')



OUTPUT:

Example 2

OUTPUT:

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

* + 1. Color Palatte

Seaborn makes it easy to use colors that are well-suited to the characteristics of your data and your visualization goals.

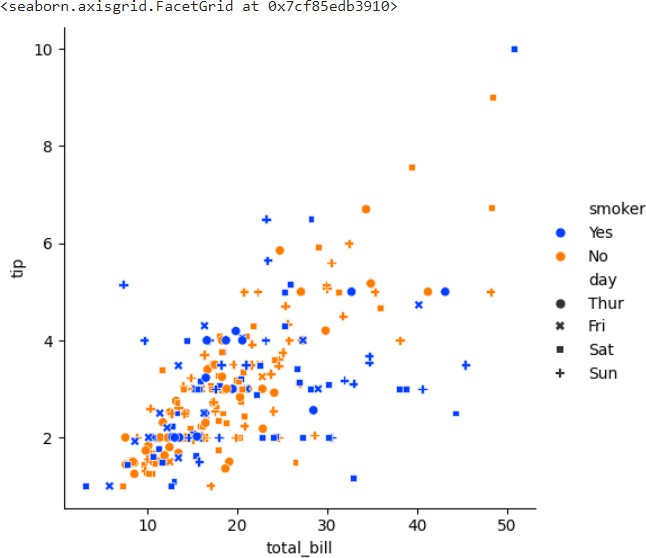
Types:

pastel bright dark

muted color blind deep

sns.relplot(data=tips,x='total\_bill',y='tip',hue='smoker',style='day',p alette='bright')

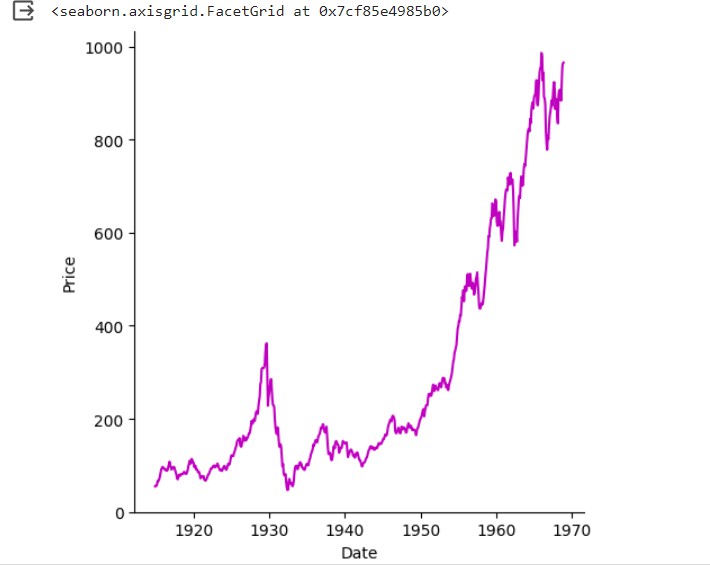
OUTPUT:



Example 1

sns.relplot(data=x1, x='Date', y='Price', kind='line',color='m')

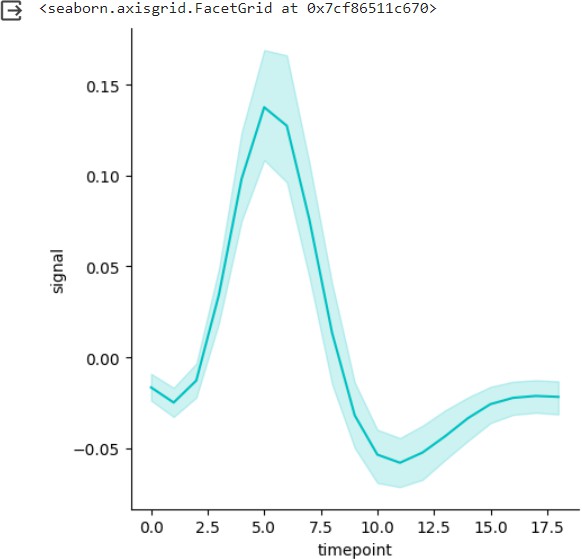
OUTPUT:



Example 2

OUTPUT:

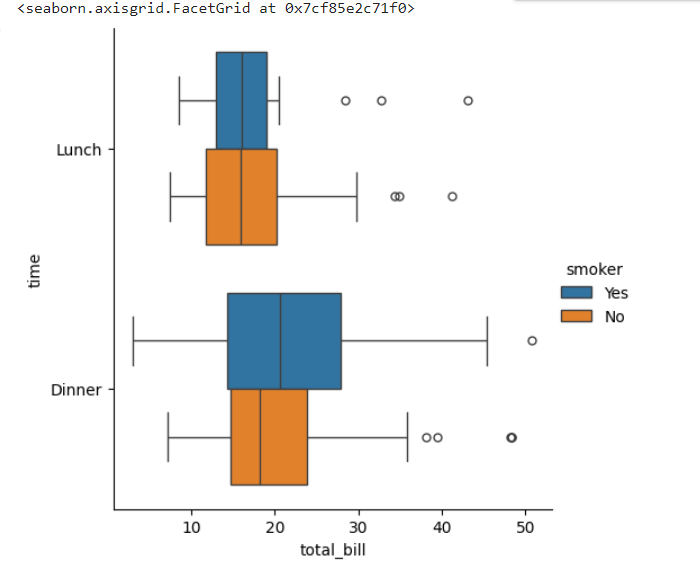
sns.relplot(data=fmri,x='timepoint',y='signal', kind='line',color='c')



Example 3:

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

OUTPUT:

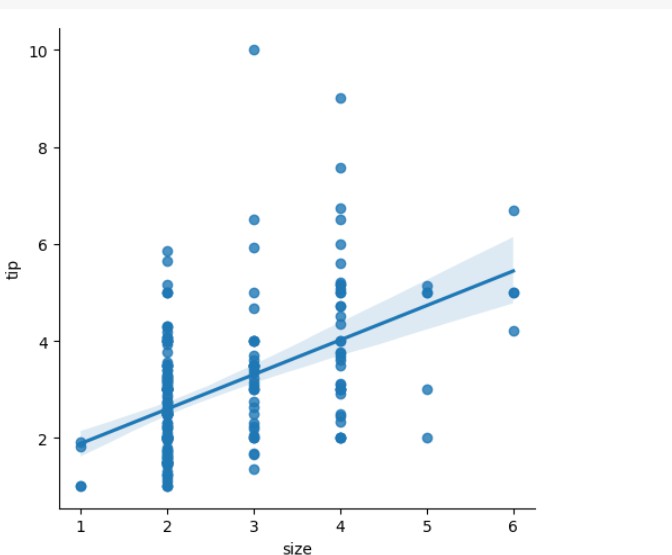


* 1. Linear Fit

Step 1:Plot the points Step 2: Fit the line

Step 3: Select the line with least difference (perfect line) Step 4: lmplot to create a function

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



OUTPUT:

1. Web scraping
   1. Introduction

Web scraping

* loading data from websites
* Unstructured in HTML

- Convertible into spreadsheets/DB

- Major websites have their APs for web scraping

Purpose of scrapper

* to 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 (csv, xlsx, Json)

Applications:

-Email Marketing

-Sentiment Analysis

-News Monitoring

-Market Research

-Price Monitoring

WEB SCRAPPING

\* extracting/scrapping data from websites

\* unstructured in HTML

\* convertable into spreadsheets/DB

\* major websites have their APIs for web scrapping

Scrapper:

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

Process:

* URL
* HTML code
* elements(CSS/IS)
* Scrapes the required data
* Saves it in required format(csv,xlsx,Ison)

Application:

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

ACCESSING A STATIC WEBSITE

Libraries:

* **BeautifulSoup:** used to soup the HTML code from static website

1. Package:bs4: subpackage:BeautifulSoup
2. Functions:

|  |  |  |
| --- | --- | --- |
| **Function** | **Purpose** | **Attributes** |
| Beautifulsoup() | to extract html code from  a webpage | text   html |
| find() | to find first element of a kind | (‘element\_name) |
| findall() | to find all the elements of a kind | (‘element\_name) |

* **Requests**: The requests module allows you to send HTTP requests using Python.The HTTP request returns a [Response Object](https://www.w3schools.com/python/ref_requests_response.asp) with all the response data (content, encoding, status, etc)

1. Used to send request to a webpage
2. get(‘url’)

* **Pandas:Pandas** makes it easy to scrape a table ( <table> tag) on a **web page**.
* selenium:
* Webdriver
* webdriver\_manager

INSTALLING LIBRARY:

           !pip install library

UPGRADING LIBRARY:

           !pip install - -upgrade library  #to upgrade to recent version

IMPORTING SUBPACKAGE

           from parent import child

           Ex:from bs4 import beautifulsoup

problem1

Extracting HTML code of any website

* import libraries
* save the url
* using requests.get('url'),access the web page
* using BeautifulSoup(), access the HTML code

Code

#importing libararies

from bs4 import BeautifulSoup

import requests

#saving url

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

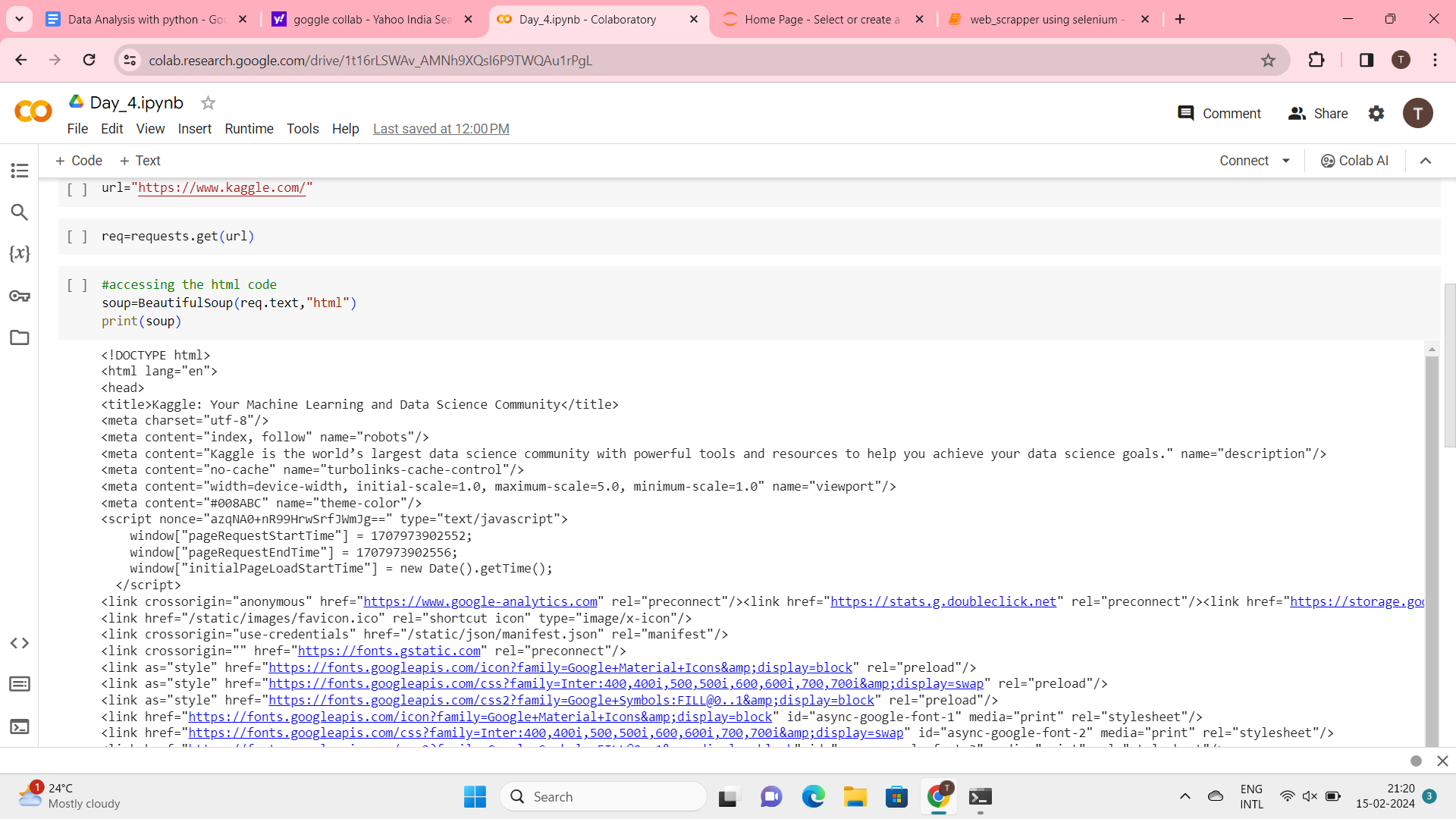
#accessing the webpage

req=requests.get(url)

#accessing the HTML code

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

print(soup)

Output:**Problem 2:**

Extracing a table from HTML code of any website and saving it as csv file\*\*

# program flow

\* import libraries

\* save the url

\* using requests.get(),access the web page

\* using BeautifulSoup(),access the HTML code

\* using find, acess the table

\* using find\_all, access the rows of the table

\* using .text, extract only the text(removing html tags)

\* create a data frame using pandas

\* push all the extracted data into dataframe

\* using to\_csv, save the dataframe in csv format

**Acessing elelments**

**\*** go to webpage

\* right click --> inspect

\* click on

\* now hover on the element you want to acess

\* the html tags/code for that specific elements is highlighted

\* extract the elelment name and import in your program using XPATH or By ID

#importing libararies

from bs4 import BeautifulSoup

import requests

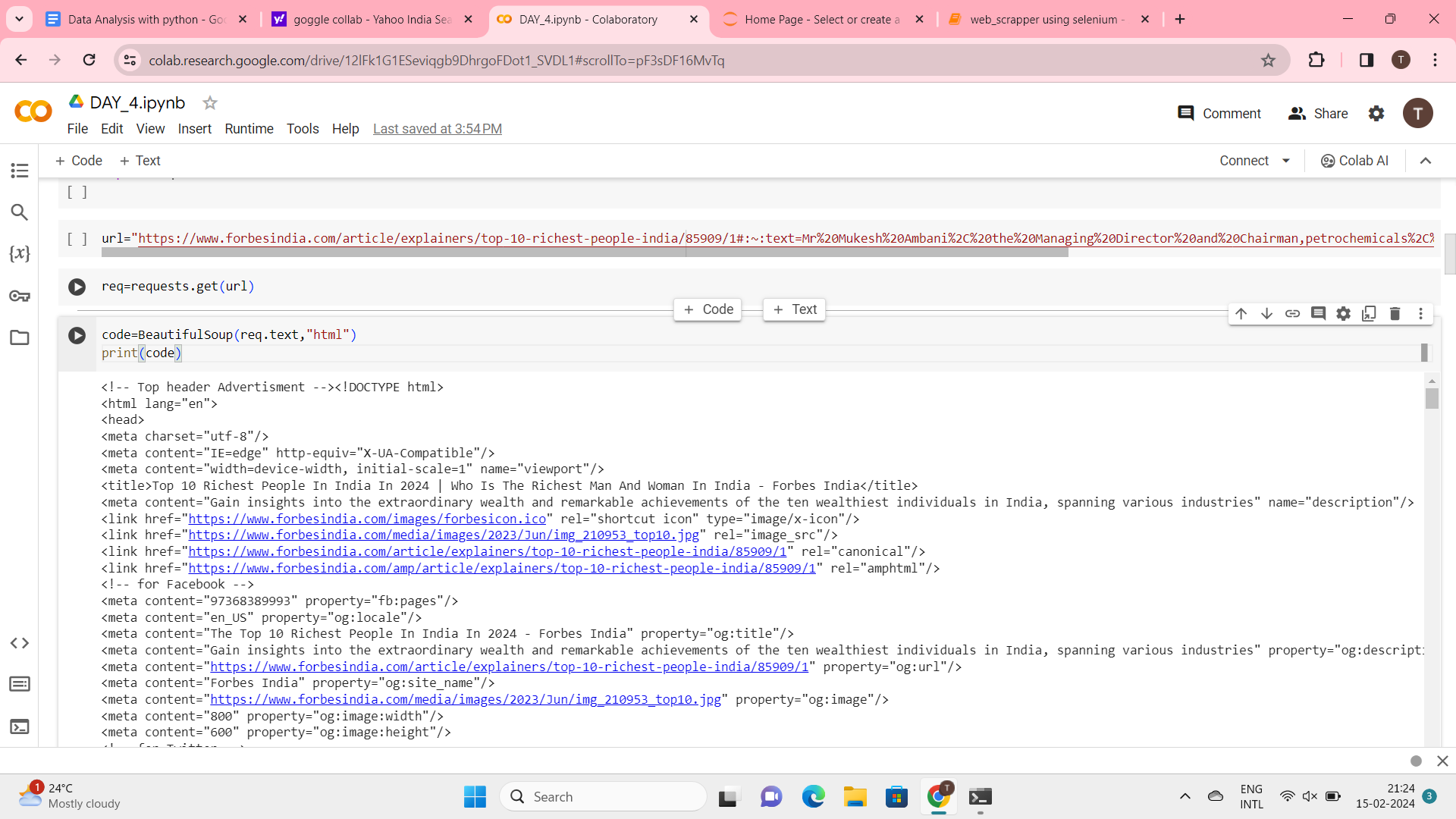
import pandas as pd

url1="<https://www.forbesindia.com/article/explainers/top-10-richest-people-india/85909/1>"

req=requests.get(url)

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

print(code)

Output:

* table-parent
* th-table header
* td-table data
* tr-table rows
* tb-table body

**Code:**

table=code.find("table")

print(table)

**output:**

**Code:**

headings=table.find\_all("th")

Headings

**output:**[<th style="border: 1px solid black; padding: 8px;"><strong>Name &amp; India Rank</strong></th>,

 <th style="border: 1px solid black; padding: 8px;"><strong>Global Rank</strong></th>,

 <th style="border: 1px solid black; padding: 8px;"><strong>Net worth (US$)</strong></th>,

 <th style="border: 1px solid black; padding: 8px;"><strong>Company</strong></th>]

**Code:**

head=[i.text for i in headings]

head

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

**Code:**

**#DATA FRAME**

df=pd.DataFrame(columns=head)

print(df)

**Output:**

Empty DataFrame

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

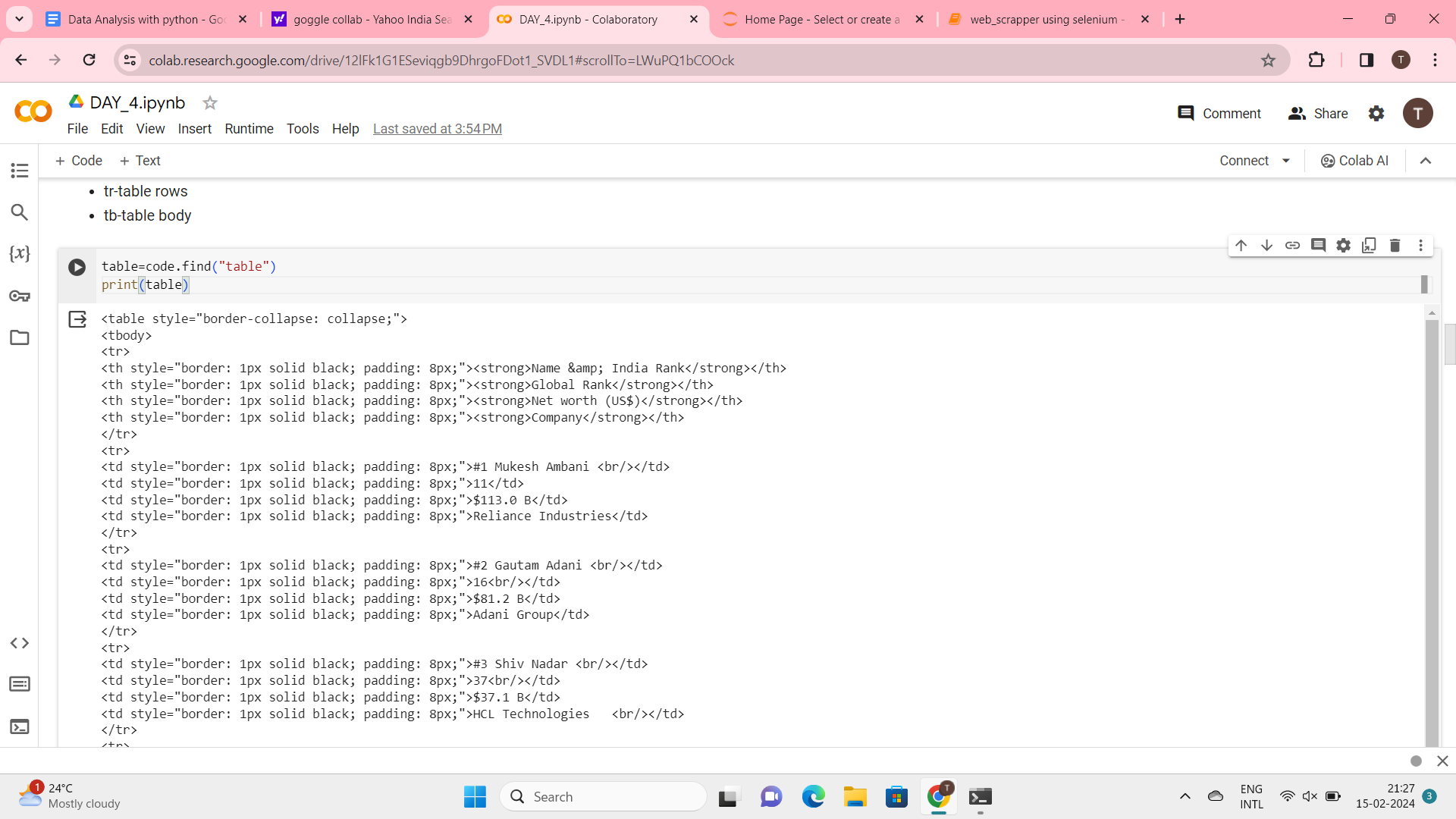
Index: []

**Code:**

rows=table.find\_all("tr")

rows

**Output:**



**Code:**

c=0

for i in rows[1:9]:

  rd=i.find\_all('td')

  eachrow=[i.text for i in rd]

  df.loc[c] = eachrow

  c = c+1

df

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name & India Rank** | **Global Rank** | **Net worth (US$)** | **Company** |  |
| 0 | #1 Mukesh Ambani | 11 | $113.0 B | Reliance Industries |
| 1 | #2 Gautam Adani | 16 | $81.2 B | Adani Group |
| 2 | #3 Shiv Nadar | 37 | $37.1 B | HCL Technologies |
| 3 | #4 Savitri Jindal & family | 58 | $28.9 B | JSW Group |
| 4 | #5 Cyrus Poonawalla | 68 | $25.6 B | Serum Institute of India |
| 5 | #6 Dilip Shanghvi | 69 | $25.5 B | Sun Pharmaceutical Industries Ltd |
| 6 | #7 Kumar Birla | 97 | $18.9 B | Aditya Birla Group |
| 7 | #8 Kushal Pal Singh | 98 | $18.9 B | DLF Limited |

**Code:**

print(df.iloc[0:1])

**Output:**

Name & India Rank Global Rank Net worth (US$)              Company

0  #1 Mukesh Ambani           11        $113.0 B  Reliance Industries

**Code:**

df.to\_csv("RICH\_MAN")

SELENIUM:

* Used to scrape data from dynamic website
* Subpackage:webdiver

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

!pip install selenium

!pip install webdriver\_manager

import selenium

import webdriver\_manager

import pandas as pd

from selenium import webdriver

from time import sleep

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

            #code

#defining 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)

Output: Chrome

            #navigate to a website

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

**Amazon Access:**

* Amazon Search box id:twotabsearchtextbox
* Searching id:nav-search-submit-button

# Project 3

* Extracting dell laptops data from amazon.in website and saving it as.csv Name of the laptop,Price,Number of Reviews

#Program flow

Phase 1-launching

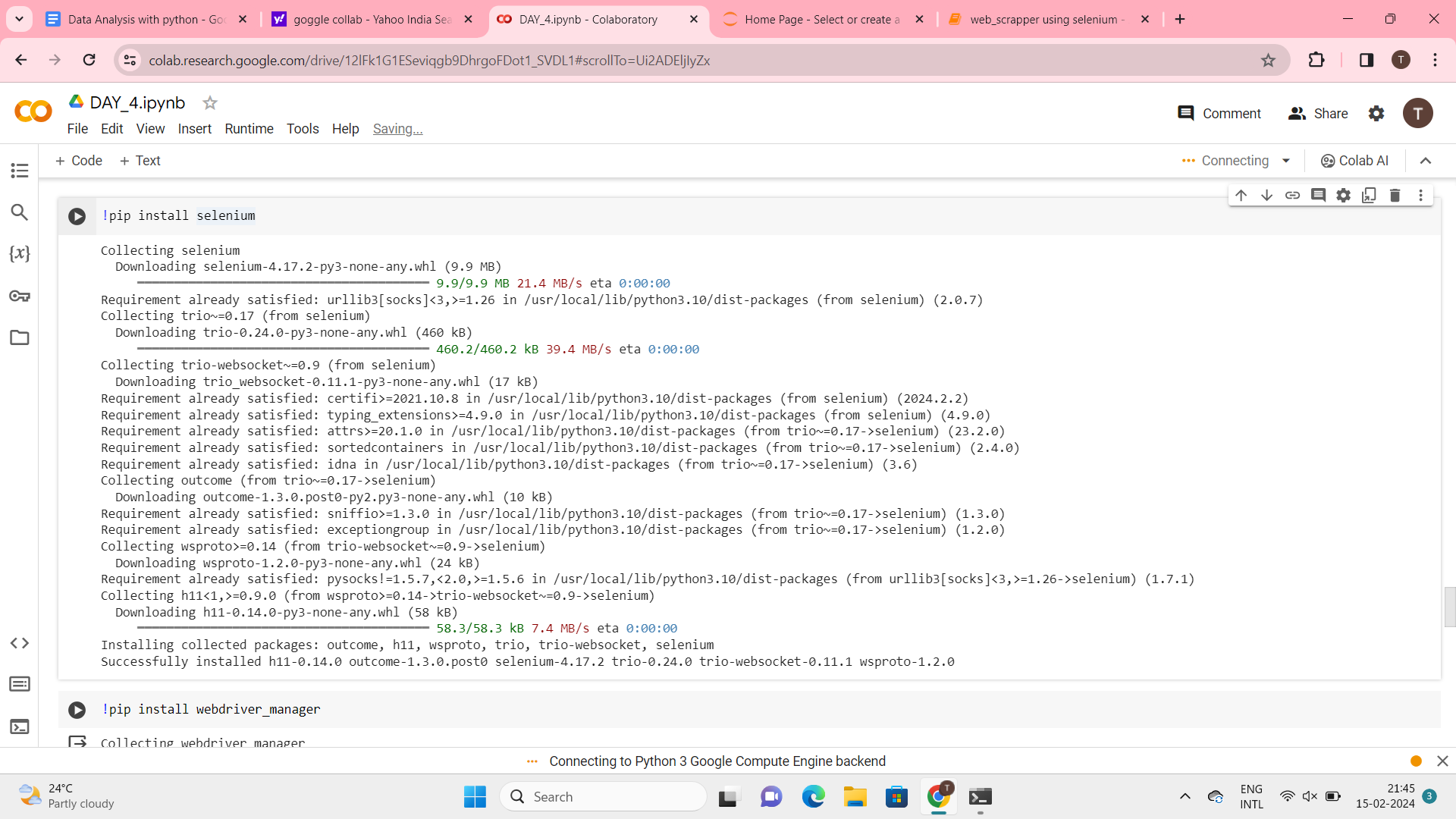
* 1.install libraries-selenium,web\_manager
* 2

Phase 2

* 1.find the search box
* 2.Push the text ‘dell laptops’
* 3.Click the search button
* 4.Select only ‘’dell”**Code:**

!pip install selenium

**Output:**



**Code:**

!pip install webdriver\_manager

**Output:**

Collecting webdriver\_manager

  Downloading webdriver\_manager-4.0.1-py2.py3-none-any.whl (27 kB)

Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from webdriver\_manager) (2.31.0)

Collecting python-dotenv (from webdriver\_manager)

  Downloading python\_dotenv-1.0.1-py3-none-any.whl (19 kB)

Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from webdriver\_manager) (23.2)

Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->webdriver\_manager) (3.3.2)

Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->webdriver\_manager) (3.6)

Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->webdriver\_manager) (2.0.7)

Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->webdriver\_manager) (2024.2.2)

Installing collected packages: python-dotenv, webdriver\_manager

Successfully installed python-dotenv-1.0.1 webdriver\_manager-4.0.1

**Code:**

import selenium

import webdriver\_manager

import pandas as pd

**Code:**

from selenium import webdriver

from time import sleep

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

**Code:**

#define options and set browser capabilities

options=webdriver.ChromeOptions()

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

#create webdriver instance with options

driver=webdriver.Chrome(options=options)

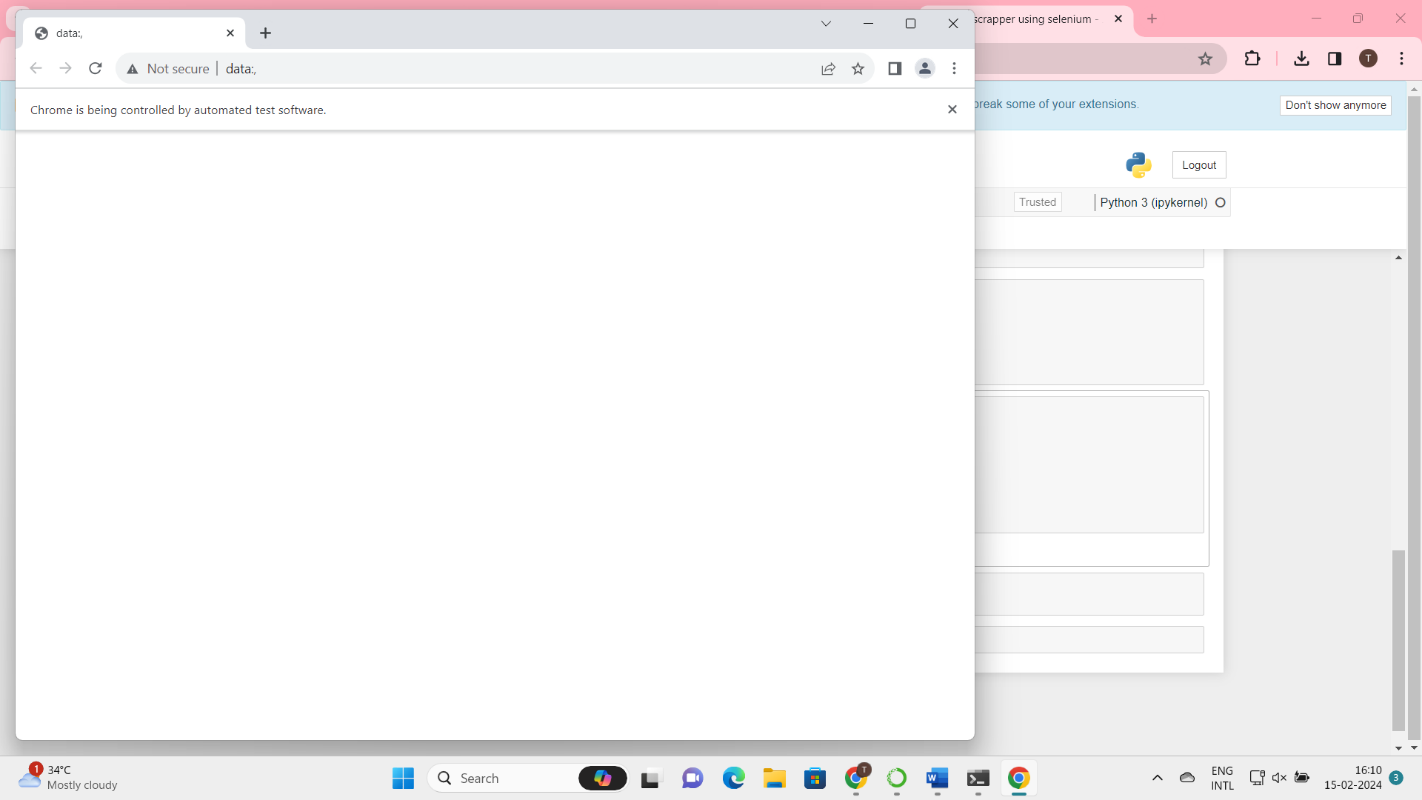
#access browser capabilities

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

print(browser\_name)

**Output:**

Chrome

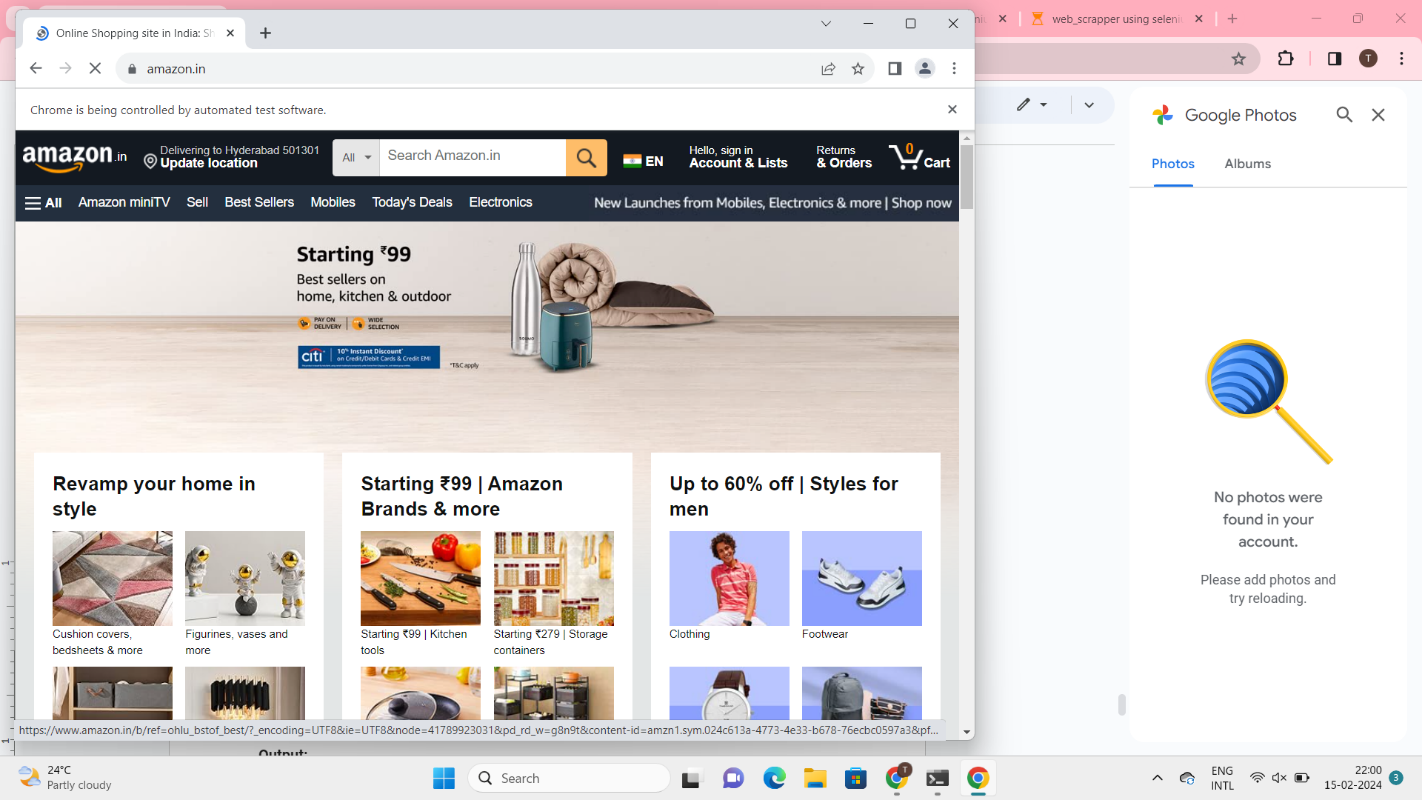


**Code:**

#navigate to a website

driver.get('<https://www.amazon.in>')

**Output:**



**Code:**

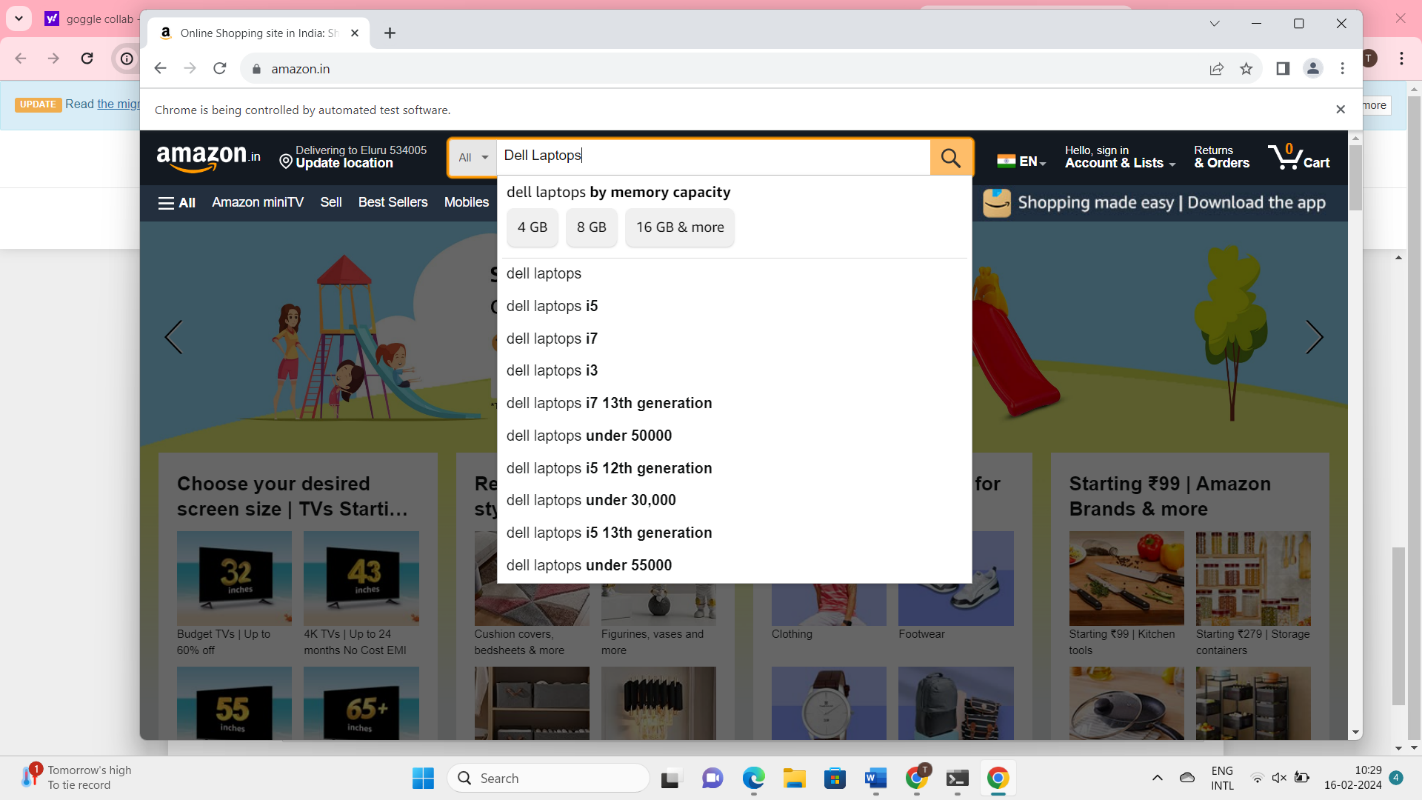
search=driver.find\_element(By.ID,"twotabsearchtextbox")

**Output:**

**Code:**

search.send\_keys("Dell Laptops")

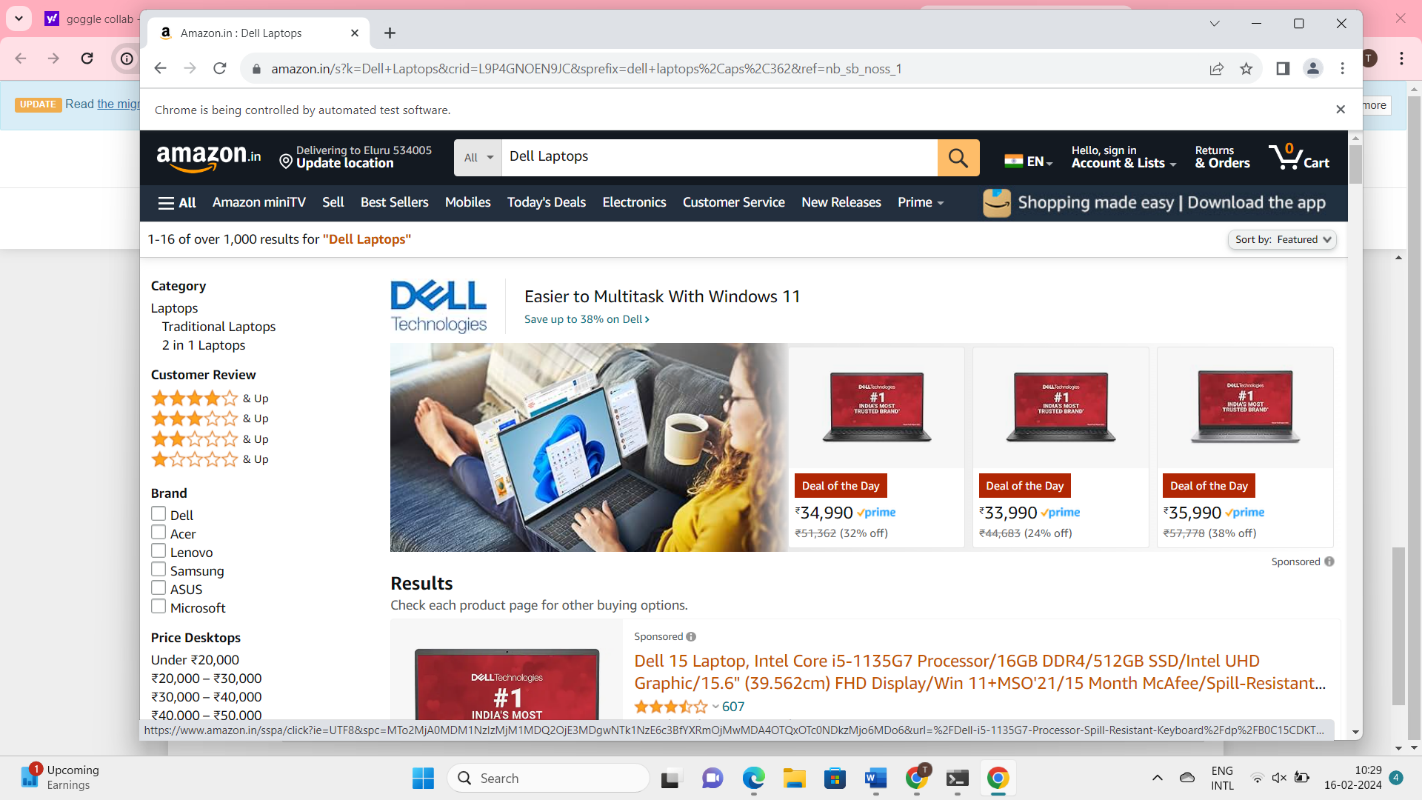
**Output:**



**Code:**

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

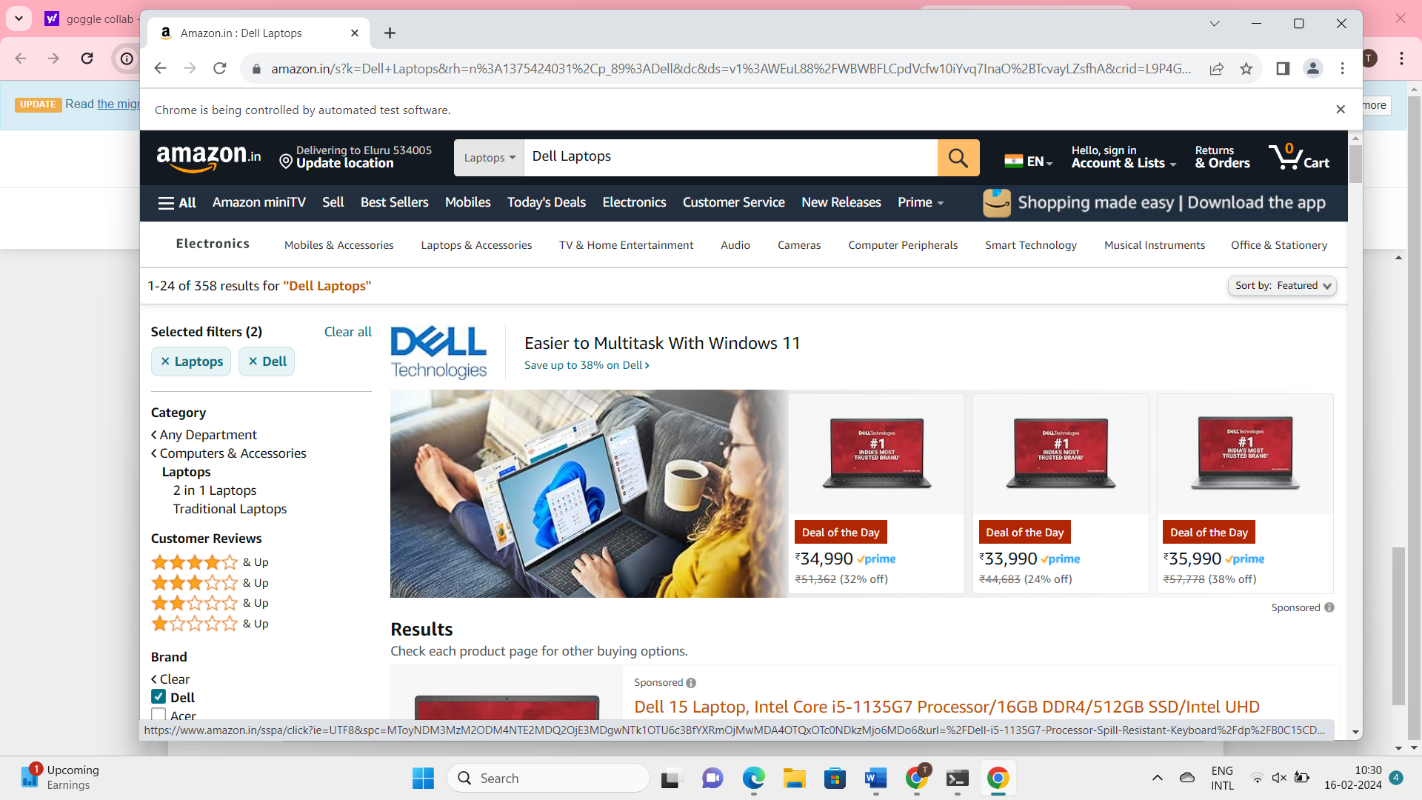
**Output:**



**Code:**

driver.find\_element(By.XPATH,"//span[text()='Dell']").click()

**Output:**



**Code:**

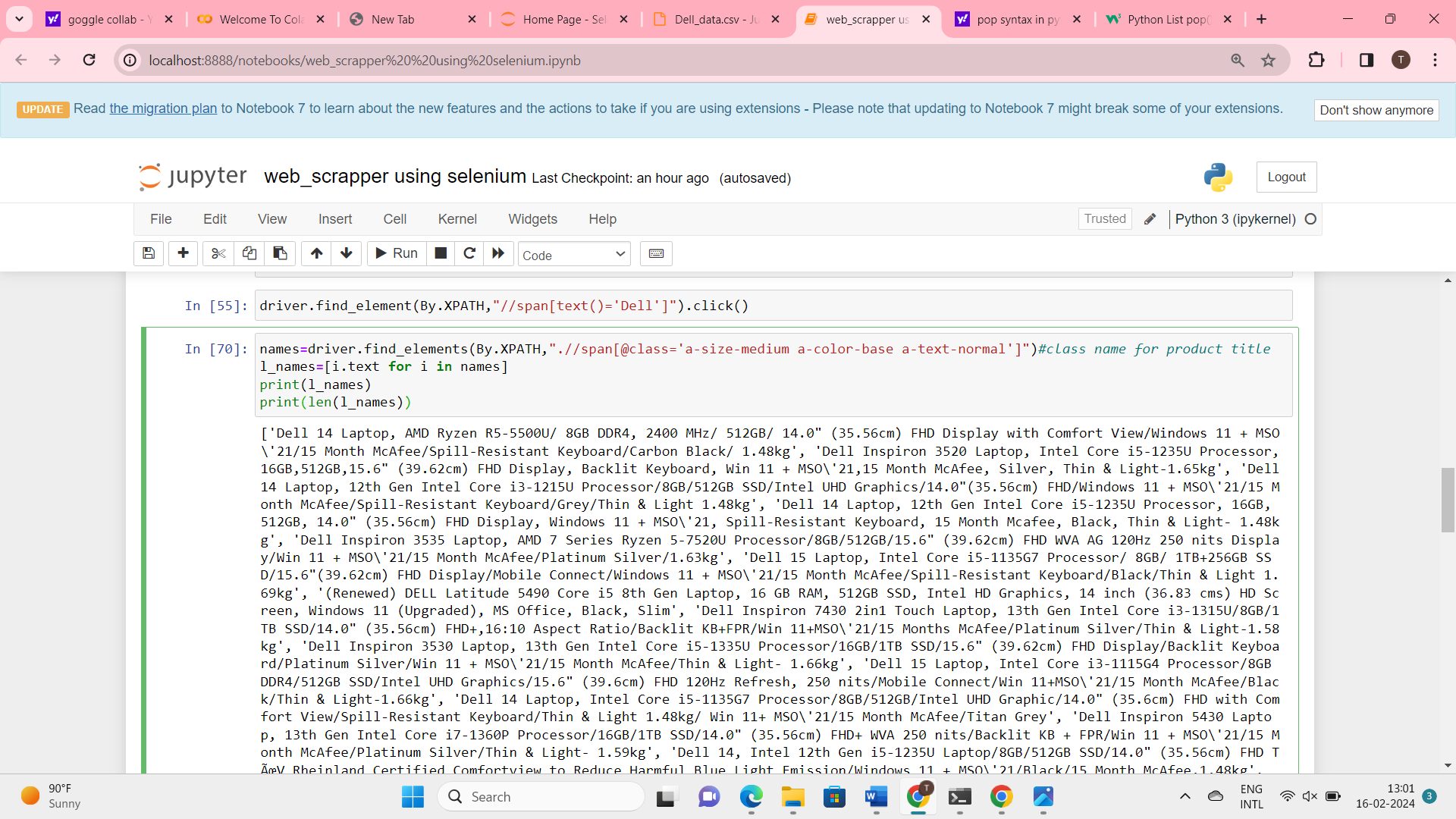
names=driver.find\_elements(By.XPATH,".//span[@class='a-size-medium a-color-base a-text-normal']")#class name for product title

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

print(l\_names)

print(len(l\_names))

**Output:**

****

**Code:**

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

l1\_prices=[i.text for i in prices]

print(l\_prices)

print(len(l1\_prices))

l1\_prices.pop(0)

l1\_prices.pop(0)

l1\_prices.pop(0)

print(l1\_prices)

print(len(l1\_prices))

**Output:**

['34,990', '33,990', '35,990', '35,990', '55,280', '35,990', '49,990', '38,990', '44,990', '23,649', '57,990', '67,490', '33,990', '44,990', '83,490', '46,990', '75,990', '30,630', '19,999', '71,490', '19,890', '22,499', '37,817', '20,999', '98,990', '36,970', '34,380']

27

['35,990', '55,280', '35,990', '49,990', '38,990', '44,990', '23,649', '57,990', '67,490', '33,990', '44,990', '83,490', '46,990', '75,990', '30,630', '19,999', '71,490', '19,890', '22,499', '37,817', '20,999', '98,990', '36,970', '34,380']

24

**Code:**

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

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

print(l\_reviews)

print(len(l\_reviews))

**Output:**

['4', '2', '239', '72', '4', '607', '506', '179', '13', '631', '2', '82', '138', '517', '1', '1,534', '1', '283', '76', '151', '176', '6', '186', '195']

24

**Code:**

headings=["laptop names","prices","reviews"]

Code:

df=pd.DataFrame(columns=headings)

print(df)

**Output:**

Empty DataFrame

Columns: [laptop names, prices, reviews]

Index: []

**Code:**

df["laptop names"]=l\_names

df["prices"]=l1\_prices

df["reviews"]=l\_reviews

print(df)

**Output:**

laptop names prices reviews

0 4 35,990 4

1 2 55,280 2

2 239 35,990 239

3 72 49,990 72

4 4 38,990 4

5 607 44,990 607

6 506 23,649 506

7 179 57,990 179

8 13 67,490 13

9 631 33,990 631

10 2 44,990 2

11 82 83,490 82

12 138 46,990 138

13 517 75,990 517

14 1 30,630 1

15 1,534 19,999 1,534

16 1 71,490 1

17 283 19,890 283

18 76 22,499 76

19 151 37,817 151

20 176 20,999 176

21 6 98,990 6

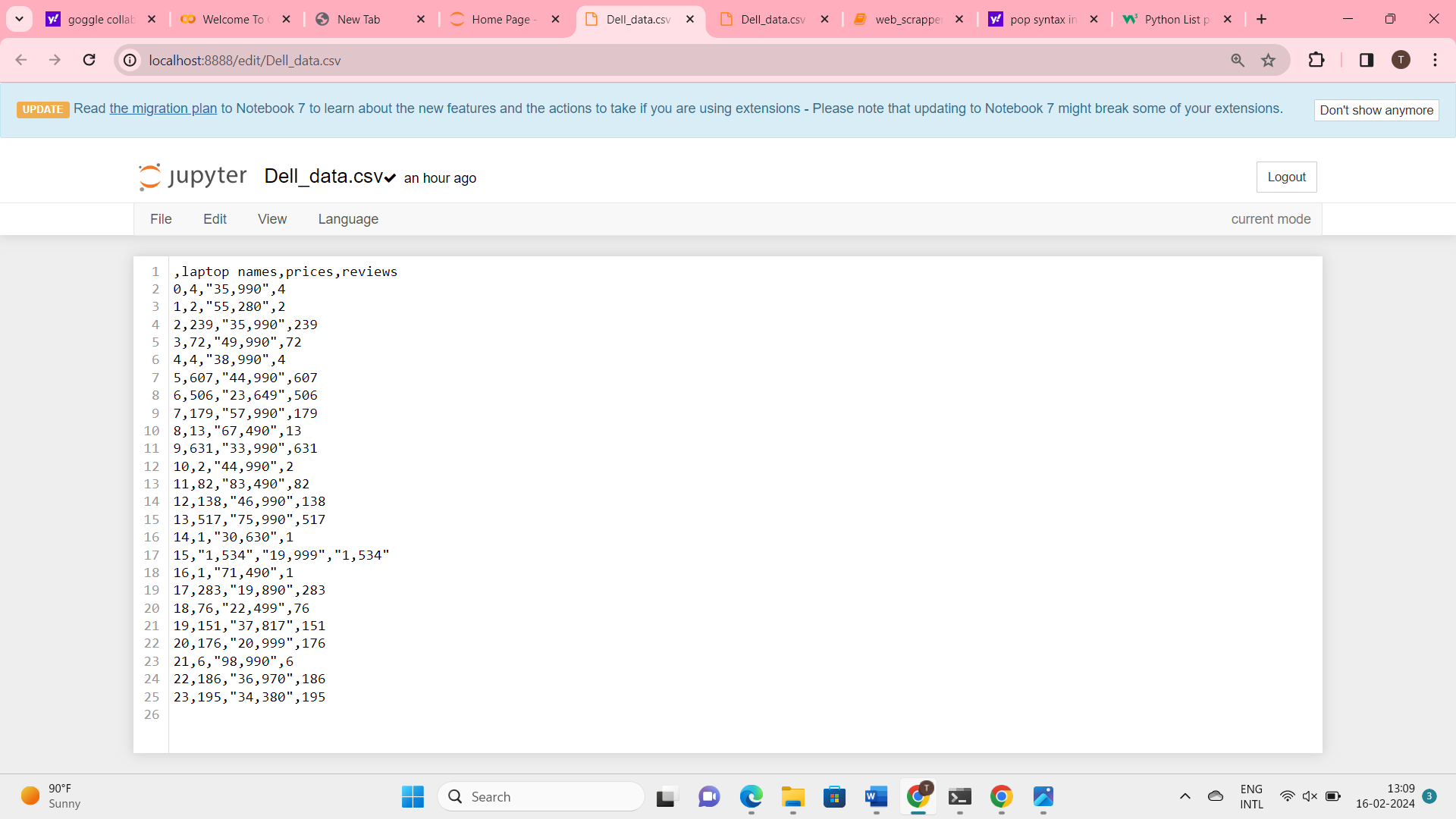
22 186 36,970 186

23 195 34,380 195

**Code:**

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

**Output:**it the creates dell data file in home page

****

**What is API?**

\*API-application programming interface

\*Connecting small apps together create bigger apps

\*API is a building part of any bigger applications

\*one website accessing data from same data bases it works straight forward

\*one website accessing data from different data bases belong to different companies we need API’s for connection

**Case Study:**Flight booking

**Ex:**ola

\*App1=login

\*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 API’s and they are building blocks of so many bigger apps,hence reusable
* so if 1000+ apps use these APIs,what happens!
* API crashes…….
* hence API keys are creayed!

**Random Fox API:**

**Code:**

import requests

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

**Code:**

print(page.status\_code)

Output:

200

**Code:**

print(page.text)

**Output:**

{"image":"https:\/\/randomfox.ca\/images\/6.jpg","link":"https:\/\/randomfox.ca\/?i=6"}

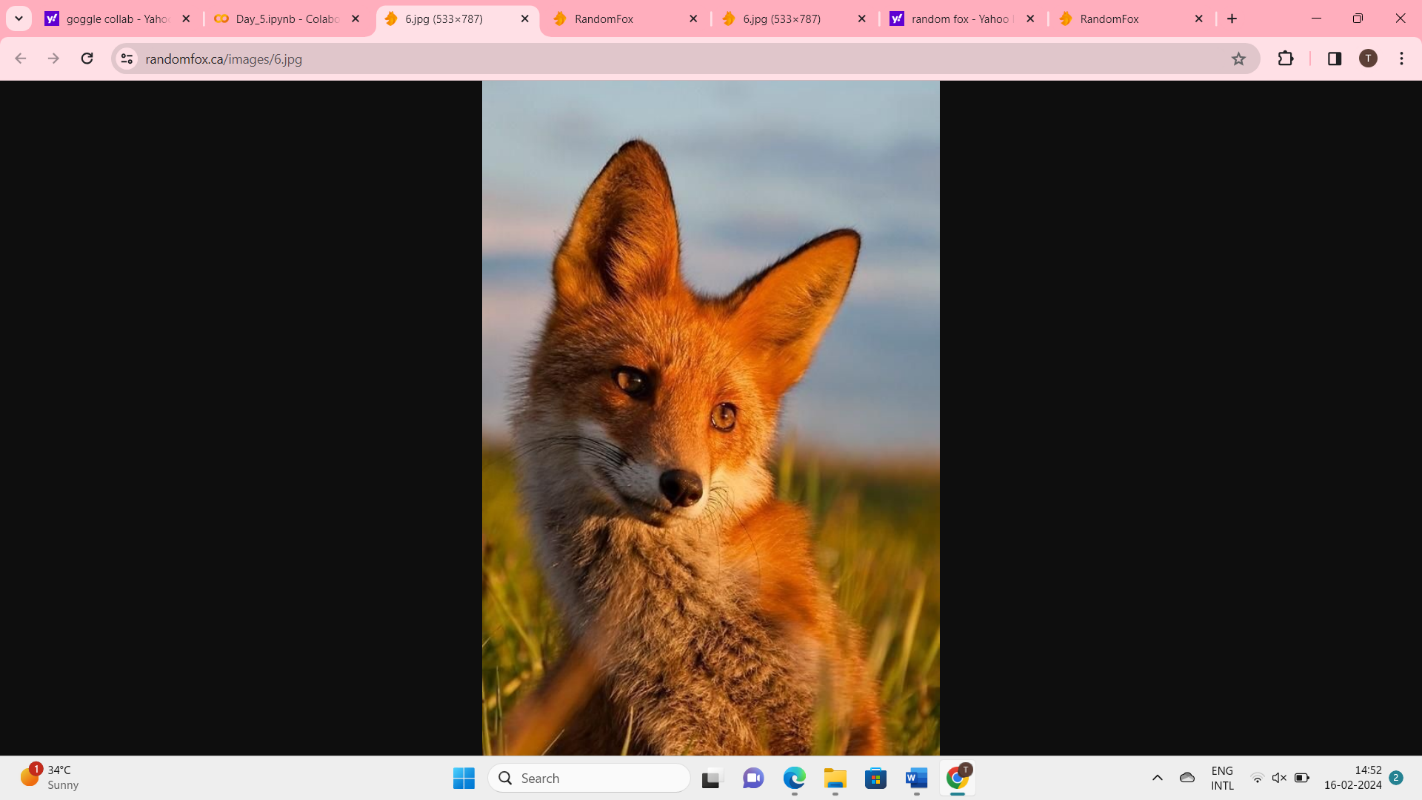
**Code:**

print(page.json())

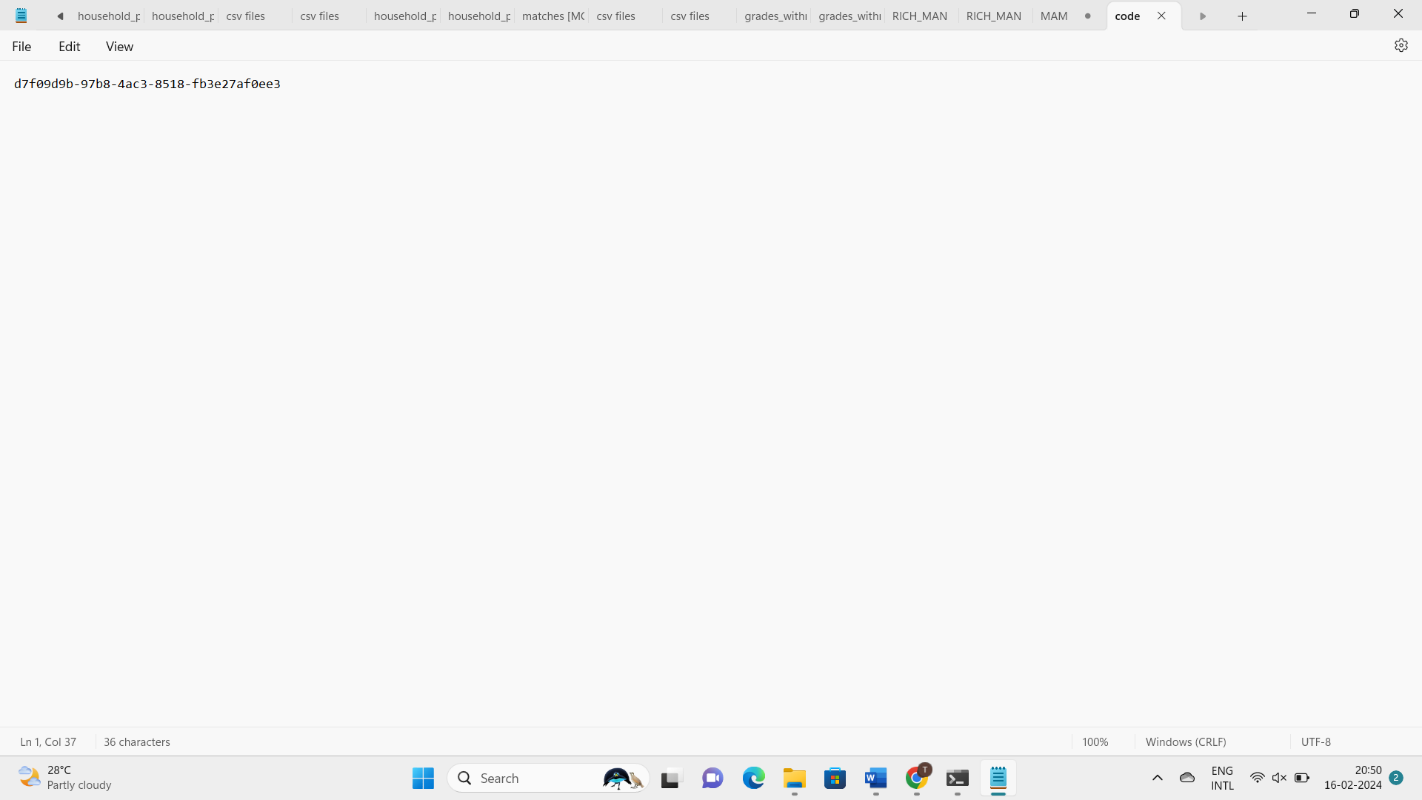
**Output:**

{'image': '[https://randomfox.ca/images/6.jpg](https://randomfox.ca/images/6.jpg" \t "_blank)', 'link': '[https://randomfox.ca/?i=6](https://randomfox.ca/?i=6" \t "_blank)'}

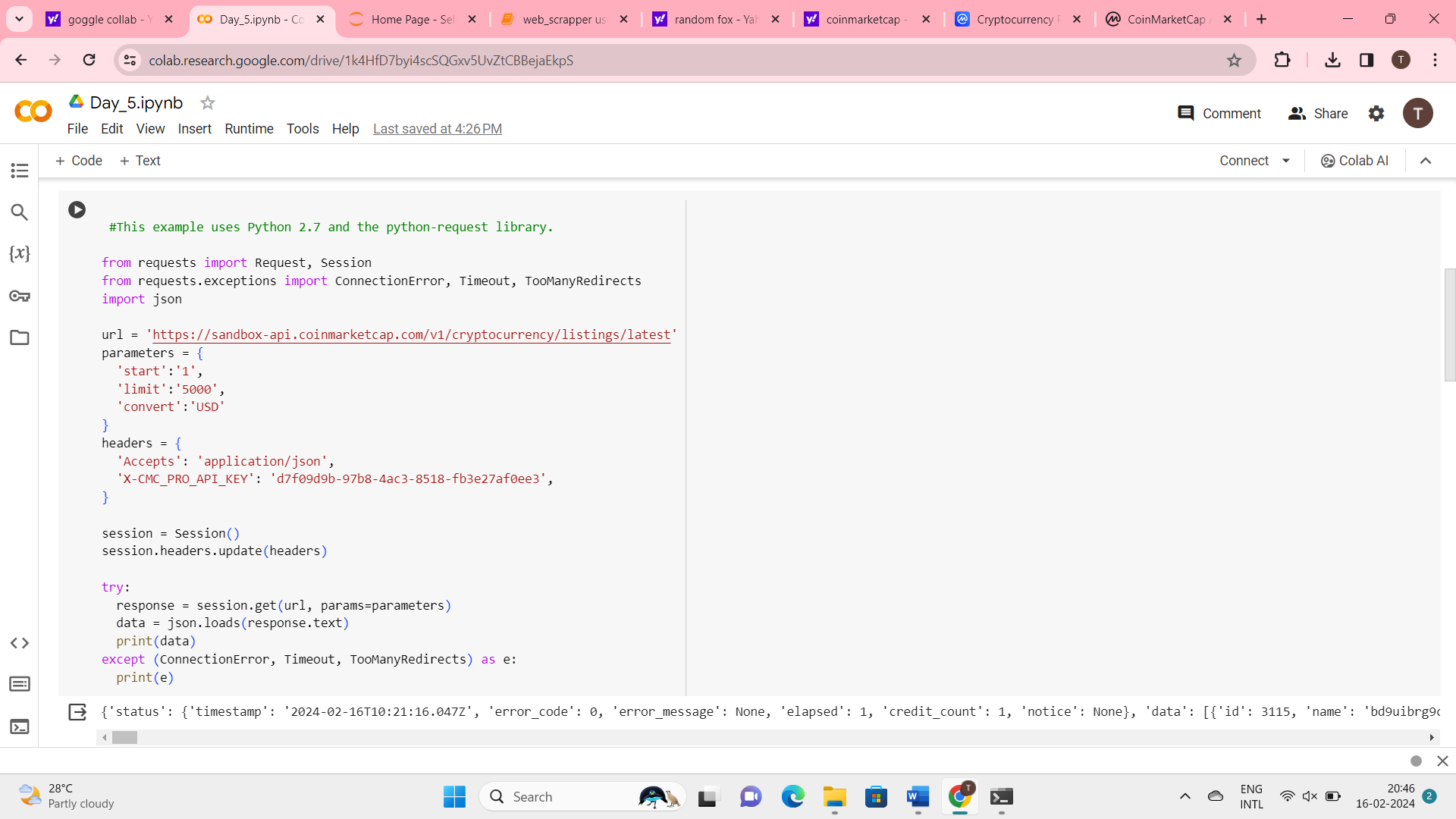




* search **coinmarketcap** and enter
* And scroll down that page at the bottom there is a[Crypto API](https://coinmarketcap.com/api/" \t "_blank) in products
* Go to[Crypto API](https://coinmarketcap.com/api/" \t "_blank)
* Next **login**
* After login to that page,you get the **key value**
* That key value save into the **note pad**

****

* After that,go to **“API Documentation”**
* Then click the **python** and copy that code and pate that code in **goggle collub**
* Then replace the key with your **original key** given below,

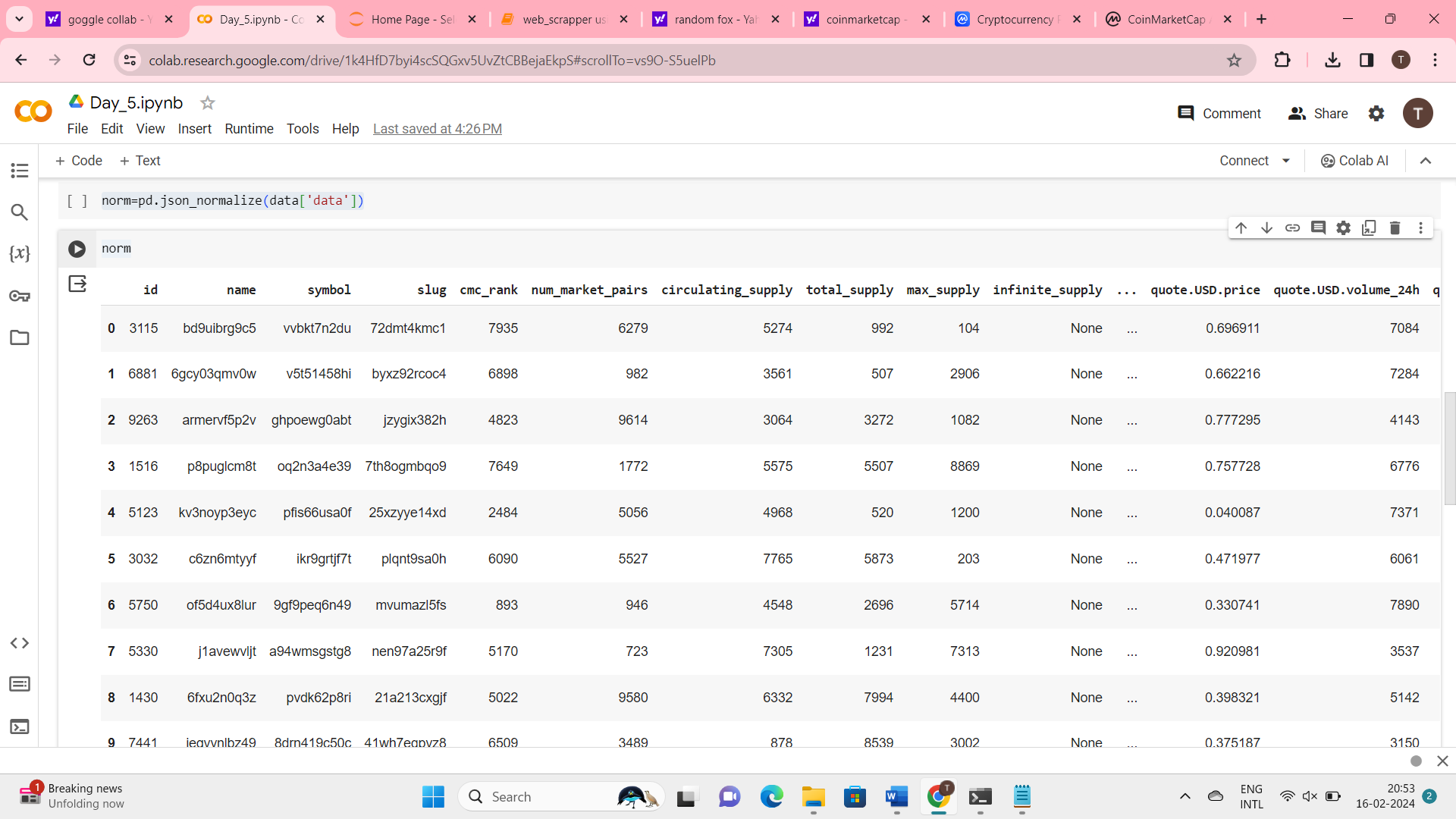
****

import pandas as pd

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

norm

Output:



**Program**

# extracting data from coinMarket cap 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 the data in the form of json
7. normalize the data into a dataframe
8. save if you needed

**Machine Learning:**

* like human learns from their past experience,machines learn from past data!
* train a machine on various things is called Machine Learning!

**Types Of Machine Learning:**

1. Supervised Machine Learning
2. Unsupervised Machine Learning
3. Reinforcement Learning

**Case Study:**

* based on the tempo,machine will suggest you the songs which contain same tempo as previous songs that you heared and liked