DAY-1

Data Analysis with Python

1. Concept Overview-Python

1.1 Basics

1.1.1. Variable

Container to store values

Code:

A=5

1.1.2. Print

It is a function used to display

Code:

```
a=5
print("I am",a,"years old")
```

Output:

```
I am 5 years old
```

1.3 Operators

1.3.1 Arithmetic operators

1.Addition '+'

```
a=5
b=25
print(a+b)
```

Output

```
30
```

2.Subtration '-'

```
print(20-10)
```

Output

```
10
```

3.Power

```
print(2**5)
```

Output

32

4.Multiplication

```
print(2*10)
```

Output

20

5.Floor Division

```
print(25//2)
```

Output

12

1.3.2 Relational Operator :These operators give relation between two operands

1.Equal to "**==**"

```
a=5
b=5
print(a==b)
```

```
Output
True
```

2. Not Equal to “!=”

```
a=5
b=5
print(a!=b)
```

Output

False

3. Greater than or equal to “>=”

```
a=5
b=5
print(a>=b)
```

Output

True

4.Less than or equal to “<=”

```
a=5
b=5
print(a<=b)
```

Output

True

5.Greater than “<”

```
a=5
b=5
print(a<b)
```

Output

```
False
```

6.Less than “>”

```
a=5
b=5
print(a>b)
```

Output

```
False
```

1.3.3 Logical Operators

1. Logical AND: It gives TRUE statement if both conditions are true or it gives FALSE statement

```
a=9
b=25
print((a>b) and (a<b))
Output
False
```

2.Logical OR:It gives TRUE either of the statement is true

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

1.3.4 Membership Operators: Membership operators in Python are operators used to test whether a value exists in a sequence, such as a list, tuple, or string.

```
a= "Pirate"
```

```
print("a" not in a)
print("a" in a)
Output
False
True
```

1.4 Control Flow Conditional Statements:Conditional statements (if, else, and elif) are fundamental programming constructs that allow you to control the flow of your program based on conditions that you specify

1.41 if statement:This condition statement is used to check a condition,and execute it if the condition holds true

```
a=int(input("Enter a number"))
if(a>0):
  print("Given",a,"is positive")
Output
Enter a number 45
Given 45 is positive
```

1.4.2 Nested if: You can have if statements inside if statements

```
a=int(input("enter a"))
b=int(input("enter b"))
c=int(input("enter c"))
if(a>b):
  if(a>c):
    print("a is greater")
  else:
    print("c is greater")
else:
  if(b>a):
    print("b is greater")
  else:
    print("a is greater")
  Output
  enter a 30
  enter b 69
  enter c 54
  b is greater
```

1.4.3 if else ladder: No of if and else statements

```
b=int(input("enter a number"))
if(b>0):
  print(b,"is positive")
elif(b<0):
  print(b,"is negative")
else:
  print(b,"is either positive or negative")
```

```
Output
  enter a number 0
  0 is either positive or negative
```

1.5 Conditional Looping Statements: Python programming language provides two types of loops – **For loop** and **While loop** to handle looping requirements.

1.5.1 For Loop: In Python, a For loop is used to iterate over sequences such as lists, strings, tuples, etc.

```
n=5
for i in range(1,11,1):
  a=n*i
  print(a)
Output
 5
 10
 15
 20
 25
 30
 35
 40
 45
 50
```

1.5.2 While Loop: **It** is used to execute a block of statements repeatedly until a given condition is satisfied. When the condition becomes false, the line immediately after the loop in the program is executed.

```
 n=5
 i=0
 while(i<=10):
   a=n*i
   i=i+1
   print(a)
 Output
 0
 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.

```
a="Python is easy"
print(a[0:6])
Output
Python
```

1.7 Type Casting : Casting, also known as type conversion, is a process that converts a variable's data type into another data type. These conversions can be implicit (automatically interpreted) or explicit (using built-in functions).

```
a= "45"
a=int(a)
print(type(a))
Output
<class 'int'>
```

1.8 Collection Of Lists :List is a collection of elements

Heterogeneous

Mutable(Modified)

```
l1=[34,"Ojas Ghambir",90.8,45]
print(l1)
Output
[34, 'Ojas Ghambir', 90.8, 45]
for i in l1:
print(i)
34
Ojas Ghambir
90.8
45
```

1.8.1 append(): It appends element at the end of the list

```
l1.append("pk")
print(l1)
Output
[34, 'Ojas Ghambir', 90.8, 45, 'pk']
```

1.8.2 insert(): It inserts the elements at the position where ever you want

```
l1.insert(1,"pavan")
print(l1)
Output
[34, 'pavan', 'Ojas Ghambir', 90.8, 45, 'pk']
```

1.8.3 pop(): It deletes the elements at the position you given

```
l1.pop(2)
print(l1)
 Output
 [34, 'pavan', 90.8, 45, 'pk', 54, 69.9, 'venkata']
```

1.8.4 remove(): It removes the what ever element you want

```
l1.remove("pavan")
print(l1)
```

Output

```
[34, 90.8, 45, 'pk', 54, 69.9, 'venkata']
```

1.9 List Comprehension: Iterartors

applies some function on every element

conditions

Output:list

```
l1=[45,69,99,54]
l2=[i**2 for i in l1]
print(l2)
Output
[2025, 4761, 9801, 2916]
```

Example 2:

```
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. NUMPY: Numpy is a python library used for working with numerical data in python.

There are different sub packages in numpy.

Install:

!pip install np

#importing:

import numpy as np

#import ram

#random module is subpackage of numpy

#creating 1-D array

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

print(type(A))

Output:

<class 'numpy.ndarray'>

#creating 2-D array

```
b=np.array([[2,3,4],[4,5,6]])
print(b)
Output:
[[2 3 4]
[4 5 6]]
#creating 3-D array
C=np.array([[[2,3,4],[5,6,7]],[[1,8,9],[0,4,5]]])
print(C)
Output:
[[[2 3 4]
[5 6 7]]
[[1 8 9]
[0 4 5]]]
#checking dimensions
print(A.ndim)
print(B.ndim)
print(C.ndim)
Output:
1
2
3
#ones #groups,rows,columns,2D
e=np.ones((3,2))
print(e)
Output:
[[1. 1.]
[1. 1.]
[1. 1.]]
#zeros
f=np.zeros((4,4))
print(f)
Output:
[[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]]
#arange
b=np.arange(3,31,3)
print(b)
Output:
```

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

Linspace:It is a function that generates a sequence of evenly spaced numbers over specified range

```
#linspace
r=np.linspace(24,10,3)
print(r)
```

Output:

[24. 17. 10.]

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

```
y=np.arange(1,7).reshape(2,3)
print(y)
q=np.arange(9,15).reshape(2,3)
print(q)
```

Output:

[[1 2 3]
[4 5 6]]
[[9 10 11]
[12 13 14]]

```
#sum of arrays
print(L+R)
```

Output:

[[10 12 14]
[16 18 20]]

```
#sum function
G=np.sum((L,R))
print(G)
```

OUtput:

90

```
#sum function using axis=1
g=np.sum((A,B),axis=1)#ROWS FIRST
print(g)
```

Output:

[[4 4]
[6 6]]

```
#sum function using axis=0
g=np.sum((A,B),axis=0)#columns first
print(g)
```

Output:

[[3 3]

```
[7 7]]
b=np.array(25,289,361,81)
#find square rooots and iterate through the results values. output:5 square is 25
b=np.array([25,289,361,81])
for i in b:
print(np.sqrt(i),"square is",i)
Output:
5.0 square is 25
17.0 square is 289
19.0 square is 361
9.0 square is 81
#array joins
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")
output:[[34 35 36]
[37 38 39]
[ 4 5 6]
[ 7 8 9]]
#array joins
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.hstack((a,b))) #rows
print("\n")
Output:
[[34 35 36 4 5 6]
[37 38 39 7 8 9]]
#array joins
a=np.arange(30).reshape(2,3,5)
print(a)
print("output of dstack")
print(np.dstack(a))
#noof rows becomes noof groups
#colums 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]]]
```

output of dstack

```
[[[ 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]]]
```

```
#creating an array of size(4,8)
b=np.arange(2,34).reshape(8,4)
print(b)
```

Output:

```
[[ 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 33]]
```

```
#random array
c=np.random.rand(8,4) #randrange is between 0 1nd 1
print(c)
```

Output:

```
[[0.41710006 0.0.2466779 0.42689413 0.37424698]
```

[0.00386473 0.75278467 0.64264913 0.53306426]
[0.37000206 0.49605354 0.70199896 0.30336513]
[0.33565892 0.42109433 0.76963036 0.55384553]
[0.39601839 0.72165219 0.87305478 0.37336139]
[0.80691092 0.20345904 0.57198638 0.32002396]
[0.35827043 0.66107693 0.97707299 0.35375784]
[0.63987243 0.06149391 0.00472188 0.13097827]]

```
c=10*np.random.rand(8,4) #randrange is between 0 1nd 1
print(c)
#random array
c=10*np.random.rand(8,4) #randrange is between 0 1nd 1
print(c)
```

Output:

[[4.26000300e+00 4.00481007e+00 4.97987841e+00 5.46368804e+00]
[1.65131660e+00 6.05527964e+00 3.76533188e+00 1.90646048e+00]
[1.69403655e+00 5.79061343e-03 9.63423375e+00 5.62560210e+00]
[4.45537268e+00 6.36689456e-01 1.76221251e+00 5.36279923e+00]
[6.27725604e+00 4.09478447e+00 4.43877068e+00 5.19779819e+00]
[5.43818322e+00 3.53624451e-01 5.58745598e+00 1.75436914e-02]
[5.06053128e+00 6.40999956e+00 1.02408020e+00 2.69838000e+00]
[6.28502095e+00 5.65419310e-01 3.70093726e+00 3.31217035e+00]]

```
#random array
c=np.floor(10*np.random.rand(8,4)) #randrange is between 0 1nd 1
print(c)
```

Output:

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

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**

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.

```
np.vsplit(c,4)
Output:
[array([[4., 6., 7., 2.],
[3., 1., 5., 1.]]),
array([[2., 4., 9., 5.],
[9., 8., 1., 4.]]),
array([[7., 8., 2., 3.],
[8., 8., 6., 5.]]),
array([[5., 4., 5., 3.],
[6., 3., 6., 5.]])]
np.vsplit(c,8)
Output:
[array([[4., 6., 7., 2.]]),
array([[3., 1., 5., 1.]]),
array([[2., 4., 9., 5.]]),
array([[9., 8., 1., 4.]]),
array([[7., 8., 2., 3.]]),
array([[8., 8., 6., 5.]]),
array([[5., 4., 5., 3.]]),
array([[6., 3., 6., 5.]])]
#splitting at particular rows 1st and 5th row
np.vsplit(c,(1,5))
Output:
[array([[4., 6., 7., 2.]]),
array([[3., 1., 5., 1.],
[2., 4., 9., 5.],
[9., 8., 1., 4.],
[7., 8., 2., 3.]]),
array([[8., 8., 6., 5.],
[5., 4., 5., 3.],
[6., 3., 6., 5.]])]
#create an array with 4 rows and 8 columns
d=np.arange(2,34).reshape(4,8)
print(d)
Output:
[[ 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 33]]

Horizontal split:

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

np.hsplit(d,4)

Output:

[array([[2, 3],
[10, 11],
[18, 19],
[26, 27]]),
array([[4, 5],
[12, 13],
[20, 21],
[28, 29]]),
array([[6, 7],
[14, 15],
[22, 23],
[30, 31]]),
array([[8, 9],
[16, 17],
[24, 25],
[32, 33]])]

np.hsplit(d,2)

Output:

[array([[2, 3, 4, 5],
[10, 11, 12, 13],
[18, 19, 20, 21],
[26, 27, 28, 29]]),
array([[6, 7, 8, 9],
[14, 15, 16, 17],
[22, 23, 24, 25],
[30, 31, 32, 33]])]

#splitting at particular position

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

np.hsplit(d,(3,7))

Output:

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

[10, 11, 12],
[18, 19, 20],
[26, 27, 28]]),
array([[5, 6, 7, 8],
[13, 14, 15, 16],
[21, 22, 23, 24],
[29, 30, 31, 32]]),
array([[9],
[17],
[25],
[33]])]
np.hsplit(d,(2,6))
Output:
[array([[2, 3],
[10, 11],
[18, 19],
[26, 27]]),
array([[4, 5, 6, 7],
[12, 13, 14, 15],
[20, 21, 22, 23],
[28, 29, 30, 31]]),
array([[8, 9],
[16, 17],
[24, 25],
[32, 33]])]
Trigonometry Functions:
np.pi
Output:
3.141592653589793
#creating radians
e=[np.pi/4,np.pi/4,np.pi]
print(e)
Output:
[0.7853981633974483, 0.7853981633974483, 3.141592653589793]
#converting radians into degree
np.rad2deg(e)
Output:
array([45., 45., 180.])
#converting radians into degree
e=([45., 45., 180.])

np.deg2rad(e)

Output:

array([0.78539816, 0.78539816, 3.14159265])

Trigonometry values:

1.np.cos(1)

Output:

0.5403023058681398

2.np.sin(1)

Output:

0.8414709848078965

3.np.cos(45)

Output:

0.5253219888177297

4.np.sin(45)

Output:

0.8509035245341184

Statistics:

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

#creating an array

1.calculating mean

p=np.array([25,35,45,55,65,75])

np. mean(p)

Output:

50.0

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.

2.calculating median

p=np.array([25,35,45,55,65,75])

np.median(p)

3.calculating variance

Formula:

$$\sigma^2 = \frac{\sum(xi - \bar{x})^2}{N}$$

np.var(p) Output: 291.6666666666667

4.calculating standard deviation:

Formula:

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$$

np.std(p) Output: 17.07825127659933

Creatingmatrix: g=np.arange(1,5).reshape(2,2) print(g)

Output: [[1 2] [3 4]]

Inverseofmatrixinlinearalgebra: np.linalg.inv(g)

Output: array([[-2. , 1.], [1.5,-0.5]])

#Createanarray q=np.arange(1,21).reshape(5,4)

print(q)

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

#find max number print the max number position

print(np.argmax(q))

Output: 19

#createanarray c=np.floor(10*np.random.rand(8,4))

print(c)

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

#find max number print the max number position

print(np.argmax(c))

Output: 26

#createanarray

c=10*np.random.rand(8,4)

print(c)

Output: [[9.712836854.969201850.899651475.63287243]

[1.414420997.612997491.795137094.35663633]

[5.781061297.966964790.212921527.34638472]

[1.818767595.990184316.617766144.47232379]

[1.358411252.050557141.334424222.57001196]

[4.461957630.795858536.658233378.55746628]

[7.266102732.296027063.618100292.64112403]

[3.682492373.089001813.609520062.56567894]]

#usingargmax

1.print(np.argmax(c,axis=1)) # it will print every row highest number position

Output: [0 1 1 2 3 3 0 0]

2.print(np.argmax(c,axis=0)) #print highest value column position

Output: [0 2 5 5]

#usingargmin

1.print(np.argmin(c,axis=0)) #it will print every column lowest number

#usingargmin 1.print(np.argmin(c,axis=0)) it will print every column lowest number position

Output: [4527]

2.print(np.argmin(c,axis=1)) it will print every row lowest number position

Output: [2 0 2 0 2 1 1 3]

#usingwherefunction #creating an array

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

print(h)

Output: [10 20 30 40 50 60]

1.print numbers greater than 20 using where function

print(np.where(h>20))

Output: (array([2, 3, 4, 5]),)

2.#numbers divisible by 6

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

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

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

#searchsort #creating an array

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

print(h)

Output: [10 20 30 40 50 60]

1.search element s=np.searchsorted(h,20)

print(s)

Output: 1

2.# for unsorted elements s1=np.searchsorted(w,2)

print(s1)

Output: 0

#sorting:

1.sorting o=np.array(['banana','cherry','apple'])

print(np.sort(o))

Output: ['apple' 'banana' 'cherry']

2.sorting with in the row 2d array

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

print(np.sort(k))

Output: [[246] [357]]

#usingflit 1.l=np.array([32,55,72,89,36])

flit=np.where(l%2==0)

print(flit)

Output: (array([0,2,4]),)

2.l[flit] Output: array([32, 72, 36]

Iterating through arrays

```
names=np.array(["Pavan","Anil","Harsha",])
initials=np.array(["A","K","P"])
for i,j in zip(initials,names):
    print(i,":",j)
```

Output:

```
A : Pavan
K : Anil
P : Harsha
```

Logarithms

```
print(np.log10(a))
```

Output:

```
[0.         0.30103    0.47712125 0.60205999 0.69897    0.77815125
 0.84509804 0.90308999 0.95424251]
```

Other Mathematical Functions

```
a2=np.array([5,6,7,10])
x=np.cumprod(a2)
print(x)
```

Output

```
[   5   30  210 2100]
```

```
a=np.array([10,15,25,15])
```

```
x=np.diff(a)
print(x)
```

Output

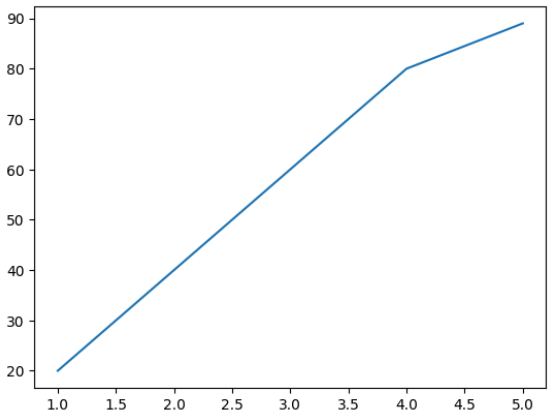
```
[  5  10 -10]
```

DAY-2

Plotting

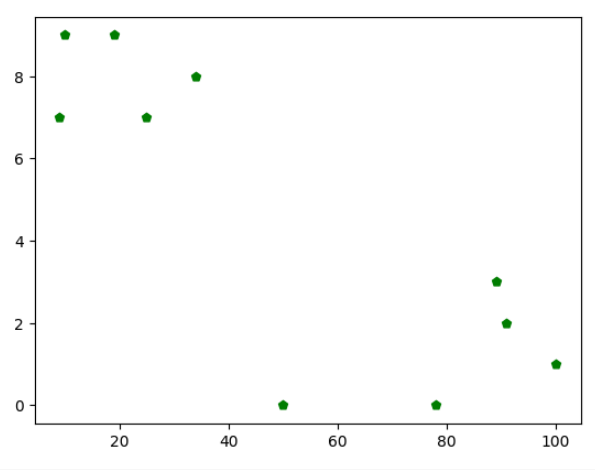
```
A=[20,40,60,80,89]
B=[1,2,3,4,5]
plt.plot(B,A)
plt.show
```

Output



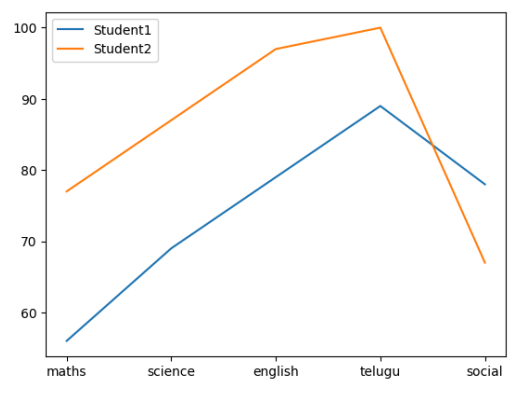
Runs scored by 10 new players[100,50,91,78,89,25,34,19,9,10] wickets taken by same 10 new players[1,0,2,0,3,7,8,9,7,9] from clusters for batsmen and bowlers

```
runs=np.array([100,50,91,78,89,25,34,19,9,10])
wickets=np.array([1,0,2,0,3,7,8,9,7,9])
plt.scatter(runs,wickets,color="green",marker="p")
Plt.show
Output
```




```
stu1=[56,69,79,89,78]
stu2=[77,87,97,100,67]
sub=["maths","science","english","telugu","social"]
plt.plot(sub,stu1,label="Student1")
plt.plot(sub,stu2,label="Student2")
plt.legend()
```

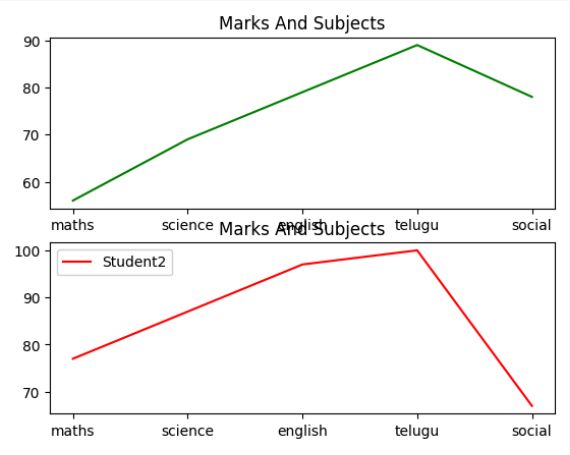
Output




```
stu1=[56,69,79,89,78]
stu2=[77,87,97,100,67]
sub=["maths","science","english","telugu","social"]
plt.subplot(2,1,1)
plt.title("Marks And Subjects")
plt.plot(sub,stu1,label="Student1",color="green")
plt.subplot(2,1,2)
plt.title("Marks And Subjects")
plt.plot(sub,stu2,label="Student2",color="red")
```

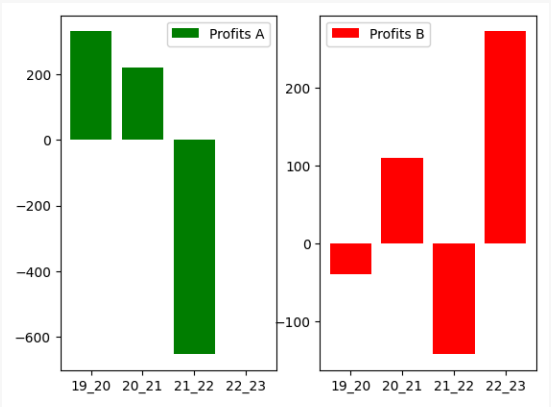
```
plt.legend()
```

Output




```
A=np.array([230,560,780,127,128])
B=np.array([200,160,270,127,400])
n1=np.diff(A)
n2=np.diff(B)
sub=np.array(["2019_2020","2020_2021","2021_2022","2022_2023"])
plt.subplot(1,2,1)
plt.bar(sub,n1,label="Profits A",color="green")
plt.legend(loc="best")
plt.subplot(1,2,2)
plt.bar(sub,n2,label="Profits B",color="red")
plt.legend(loc="best")
```

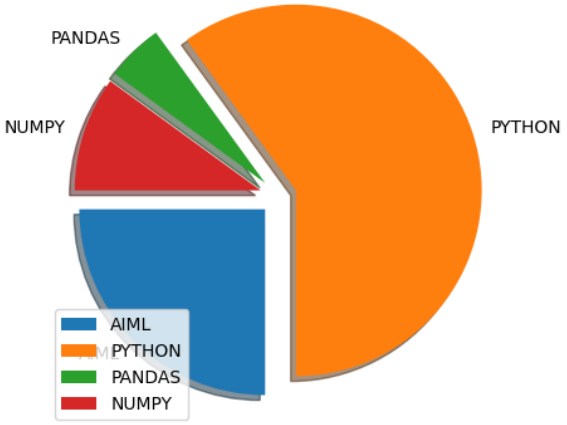
Output




```
a=np.array([25,60,5,10])
labe=["AIML","PYTHON","PANDAS","NUMPY"]
```

```
explo=[0.1,0.1,0.1,0.1]
plt.pie(a,labels=labe,explode=explo,startangle=180,shadow=True)
plt.legend()
plt.show
```

Output



DAY-3

PANDAS

Used for data manipulation(Data cleaning,Data Organization) Creates Dataframes from excel,csv,txt,DBs Dataframes(Rows and Columns readable by Python) Data cleaning by dropping or replacing with mean Visualize the data

```
import pandas as pd
```

Series

```
names=["Pavan","Shannu","Anil","Sai","Harsha"]
index=[40,42,43,44,45]
ser1=pd.Series(names,index)
print(ser1)
```

Output

```
40      Pavan
42     Shannu
43       Anil
```

```
44        Sai
45     Harsha
dtype: object
```

Importing Files . for csv and txt : read_csv("file path") . for excel : read_excel("file path")

```
df=pd.read_csv("/content/diabetcsv.csv",)
```

```
df.head(10)
```

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
5	5	116	74	0	0	25.6	0.201	30	tested_negative
6	3	78	50	32	88	31.0	0.248	26	tested_positive
7	10	115	0	0	0	35.3	0.134	29	tested_negative
8	2	197	70	45	543	30.5	0.158	53	tested_positive
9	8	125	96	0	0	0.0	0.232	54	tested_positive

```
df.tail()
```

Output

```
dft=pd.read_csv("/content/grades.txt",sep=" ")
```

```
dft.head()
```

Output

	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+

```
dfe=pd.read_excel("/content/diabetes.xlsx")
```

```
dfe.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

```
print(df.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]>
```

1. Accessing
2. loc - accepts column names and index
3. iloc - accepts only index

```
first_row=dft.iloc[0]

print(first_row)
```

Output

```
Names         Joe
Initials        K
SEM1          9.8
SEM2         10.0
SEM3          9.9
Grade          A+
Name: 0, dtype: object
```

```
print(dft.loc[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+

5 Raj M 8.9 9.1 9.3 A

```
print(dft.loc[2:5,"Names"]) #rows of specified columns
```

Output

```
2    Kissan
3      Mary
4      Jeen
5       Raj
Name: Names, dtype: object
```

```
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

isnull():this function shows the null value as true

```
dfn.isnull().head(7)
```

Output

	Names	Initials	SEM1	SEM2	SEM3	Grade	Placed
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	True	False	False
4	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False

dropna():this function drops all null rows in table

```
dfn.dropna()
```

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

fillna():this function fills the null value to one value which is given by user

```
dfn.fillna(5)
```

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

Cleaning with Mean

This means we fill the null values with mean value of the data in table

```
me=dfn['SEM3'].mean()
```

```
print(me)
```

Output

```
9.100000000000001
```

```
dfc2=dfn.fillna(me)
```

```
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

You can also change the columns headings with another names as you want

Example

```
dfc2.rename(columns={"Grade":"CGPA"},inplace=True)
```

```
dfc2.head()
```

Output

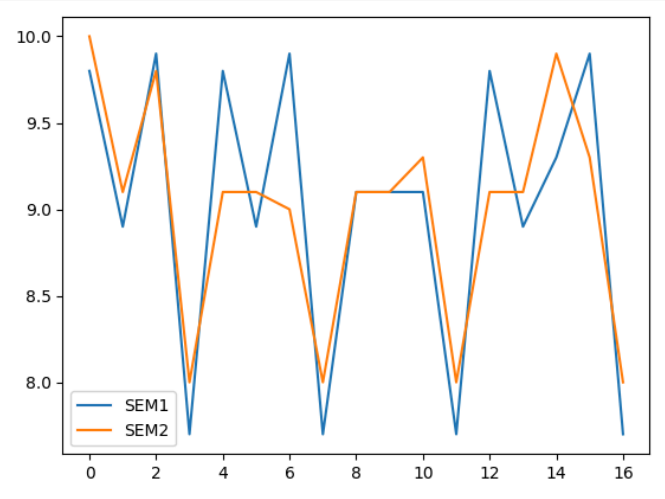
	Names	Initials	SEM1	SEM2	SEM3	CGPA	Placed
0	Joe	K	9.8	10.0	9.9	A+	1
1	Rajesh	M	8.9	9.1	9.3	A	1
2	Kissan	V	9.9	9.8	10.0	A	0
3	Mary	N	7.7	8.0	9.1	B	0
4	Jeen	K	9.8	9.1	9.9	A+	1

plot.line()

It shows the data as visual representation

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

Output



Seaborn

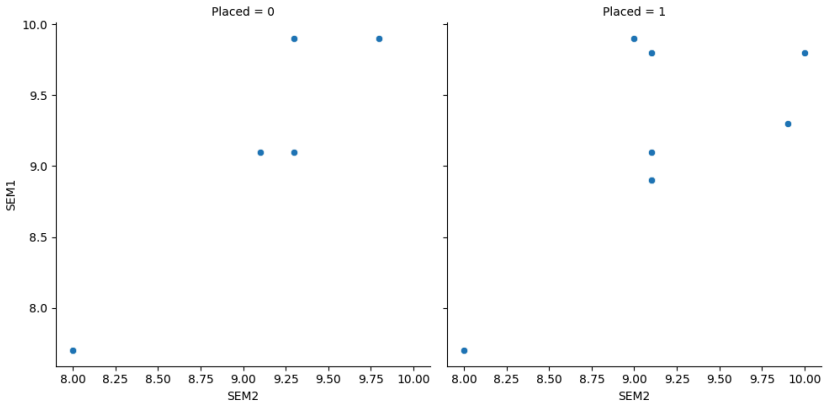
- Inbuilt data sets seaborn
  - Dowjones
    - fmri
    - dots
  - healthexp
- to load dataset use:load_dataset("tips")
- hue - creating difference based on a column via colors
  - style - different color marking
- color palettes - pastel,bright,dark,muted,color_blind,
  - List item

```
import seaborn as sns
```

```
pl=sns.lineplot(x="Names",y="SEM2",data=dfc2)
```

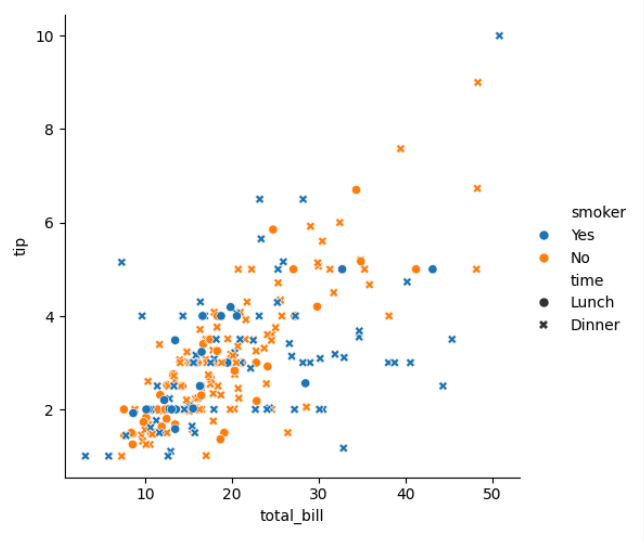
Output


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



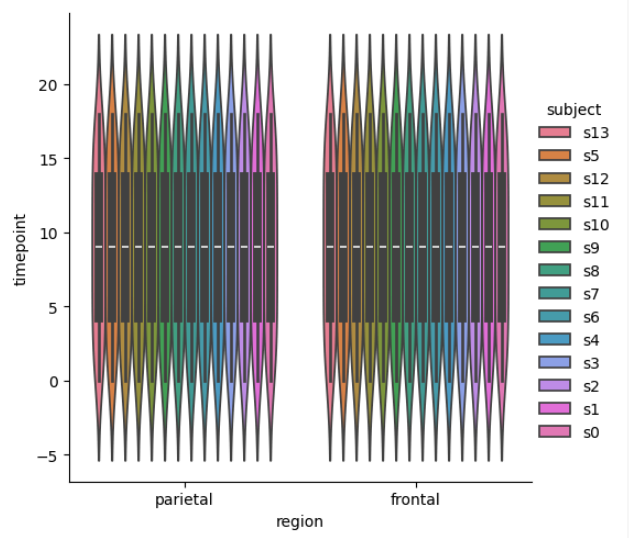
```
sns.relplot(data=tips, x="total_bill",
y="tip",hue="smoker",style="time")

#hue - creating difference based on a column via colors
```

Output




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

Output



DAY-4

Web scrapping

- loading data from websites
- unstructed in HTML
- convertible into spreadsheets/DB
- major websites have their API's for web scrapping 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 the required format(csv,xlsx,Json)
- Applications:
- Email Marketing
- Sentiments Analysis
- News Monitoring
- Market Research
- Price Monitor

LIBRARIES

- **Beautiful soup**
- **Requests**
- **Selenium**
- **Pandas**
- **webdriver**
- **webdriver_manager**
- **To install any library**
- **!pip install library**
- **!pip install --upgrade library**
- **Importing subpackage**

- **from parent import child**
- 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)**

-

```
#extracting HTML code of any website

import requests
```

```
from bs4 import BeautifulSoup
```

Get the url from kaggle website

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

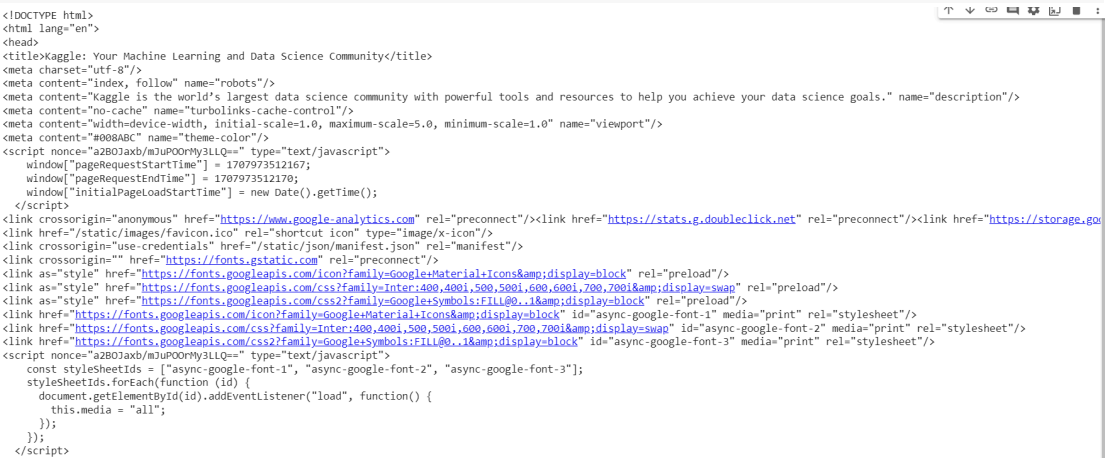
Access the request of url

```
a=requests.get(url)
```

Request is accessed

```
p=BeautifulSoup(a.text,"html")
print(p)
```

Output



```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Kaggle: Your Machine Learning and Data Science Community</title>
<meta charset="utf-8"/>
<meta content="index, follow" name="robots"/>
<meta content="Kaggle is the world's largest data science community with powerful tools and resources to help you achieve your data science goals." name="description"/>
<meta content="no-cache" name="turbolinks-cache-control"/>
<meta content="width=device-width, initial-scale=1.0, maximum-scale=5.0, minimum-scale=1.0" name="viewport"/>
<meta content="#008ABC" name="theme-color"/>
<script nonce="a2BOJaxb/mJuPOOrMy3LLQ==" type="text/javascript">
    window["pageRequestStartTime"] = 1707973512167;
    window["pageRequestEndTime"] = 1707973512170;
    window["initialPageLoadStartTime"] = new Date().getTime();
  </script>
<link crossorigin="anonymous" href="https://www.google-analytics.com" rel="preconnect"/><link href="https://stats.g.doubleclick.net" rel="preconnect"/><link href="https://storage.go
<link href="/static/images/favicon.ico" rel="shortcut icon" type="image/x-icon"/>
<link crossorigin="use-credentials" href="/static/json/manifest.json" rel="manifest"/>
<link crossorigin="" href="https://fonts.gstatic.com" rel="preconnect"/>
<link as="style" href="https://fonts.googleapis.com/icon?family=Google+Material+Icons&amp;display=block" rel="preload"/>
<link as="style" href="https://fonts.googleapis.com/css?family=Inter:400,400i,500,500i,600,600i,700,700i&amp;display=swap" rel="preload"/>
<link as="style" href="https://fonts.googleapis.com/css2?family=Google+Symbols:FILL@0..1&amp;display=block" rel="preload"/>
<link href="https://fonts.googleapis.com/icon?family=Google+Material+Icons&amp;display=block" id="async-google-font-1" media="print" rel="stylesheet"/>
<link href="https://fonts.googleapis.com/css?family=Inter:400,400i,500,500i,600,600i,700,700i&amp;display=swap" id="async-google-font-2" media="print" rel="stylesheet"/>
<link href="https://fonts.googleapis.com/css2?family=Google+Symbols:FILL@0..1&amp;display=block" id="async-google-font-3" media="print" rel="stylesheet"/>
<script nonce="a2BOJaxb/mJuPOOrMy3LLQ==" type="text/javascript">
    const styleSheetIds = ["async-google-font-1", "async-google-font-2", "async-google-font-3"];
    styleSheetIds.forEach(function (id) {
      document.getElementById(id).addEventListener("load", function() {
        this.media = "all";
      });
    });
  </script>
```

```
#problem2 scrapping table
import requests
from bs4 import BeautifulSoup
import pandas as pd
url="https://www.forbesindia.com/article/explainers/top-10-richest-people-india/85909/1"
a=requests.get(url)
P=BeautifulSoup(a.text,"html")
print(P)
```

Output

```
<head>
<meta charset="utf-8"/>
<meta content="IE=edge" http-equiv="X-UA-Compatible"/>
<meta content="width=device-width, initial-scale=1" name="viewport"/>
<title>Top 10 Richest People In India In 2024 | Who Is The Richest Man And Woman In India - Forbes India</title>
<meta content="Gain insights into the extraordinary wealth and remarkable achievements of the ten wealthiest individuals in India, spanning various industries" name="description"/>
<link href="https://www.forbesindia.com/images/forbesicon.ico" rel="shortcut icon" type="image/x-icon"/>
<link href="https://www.forbesindia.com/media/images/2023/Jun/img_210953_top10.jpg" rel="image_src"/>
<link href="https://www.forbesindia.com/article/explainers/top-10-richest-people-india/85909/1" rel="canonical"/>
<link href="https://www.forbesindia.com/amp/article/explainers/top-10-richest-people-india/85909/1" rel="amphtml"/>
<!-- for Facebook -->
<meta content="97368389993" property="fb:pages"/>
<meta content="en_US" property="og:locale"/>
<meta content="The Top 10 Richest People In India In 2024 - Forbes India" property="og:title"/>
<meta content="Gain insights into the extraordinary wealth and remarkable achievements of the ten wealthiest individuals in India, spanning various industries" property="og:descript
<meta content="https://www.forbesindia.com/article/explainers/top-10-richest-people-india/85909/1" property="og:url"/>
<meta content="Forbes India" property="og:site_name"/>
<meta content="https://www.forbesindia.com/media/images/2023/Jun/img_210953_top10.jpg" property="og:image"/>
<meta content="800" property="og:image:width"/>
<meta content="600" property="og:image:height"/>
<!-- for Twitter -->
<meta content="summary_large_image" name="twitter:card"/>
<meta content="@forbesindia" name="twitter:site"/>
<meta content="The Top 10 Richest People In India In 2024 - Forbes India" name="twitter:title"/>
<meta content="Gain insights into the extraordinary wealth and remarkable achievements of the ten wealthiest individuals in India, spanning various industries" name="twitter:descrip
<meta content="@forbesindia" name="twitter:creator"/>
<meta content="https://www.forbesindia.com/media/images/2023/Jun/img_210953_top10.jpg" name="twitter:image"/>
<meta content="forbesindia.com" name="twitter:domain"/>
<!-- link for skeleton -->
<link href="https://cdnjs.cloudflare.com/ajax/libs/meyer-reset/2.0/reset.min.css" rel="stylesheet"/>
<!-- link for skeleton -->
<style type="text/css">
            @font-face{font-family:'MerriweatherBlack';src:url(https://www.forbesindia.com/includes/fonts/Merriweather_Black.woff) format("woff")}@font-face{font-family:'Merriweath
```

#td=table_data

#th=table_head

#tr=table_row

#tbody=table_body

```
table=P.find('table')
```

```
print(table)
```

Output

```
<table style="border-collapse: collapse;">
<tbody>
<tr>
<th style="border: 1px solid black; padding: 8px;"><strong>Name &amp;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>
</tr>
<tr>
<td style="border: 1px solid black; padding: 8px;">#1 Mukesh Ambani <br/></td>
<td style="border: 1px solid black; padding: 8px;">11</td>
<td style="border: 1px solid black; padding: 8px;">$113.0 B</td>
<td style="border: 1px solid black; padding: 8px;">Reliance Industries</td>
</tr>
<tr>
<td style="border: 1px solid black; padding: 8px;">#2 Gautam Adani <br/></td>
<td style="border: 1px solid black; padding: 8px;">16<br/></td>
<td style="border: 1px solid black; padding: 8px;">$81.2 B</td>
<td style="border: 1px solid black; padding: 8px;">Adani Group</td>
</tr>
<tr>
<td style="border: 1px solid black; padding: 8px;">#3 Shiv Nadar <br/></td>
<td style="border: 1px solid black; padding: 8px;">37<br/></td>
<td style="border: 1px solid black; padding: 8px;">$37.1 B</td>
<td style="border: 1px solid black; padding: 8px;">HCL Technologies   <br/></td>
</tr>
<tr>
<td style="border: 1px solid black; padding: 8px;">#4 Savitri Jindal &amp;amp; family <br/></td>
<td style="border: 1px solid black; padding: 8px;">58<br/></td>
<td style="border: 1px solid black; padding: 8px;">$28.9 B</td>
<td style="border: 1px solid black; padding: 8px;">JSW Group<br/></td>
</tr>
<tr>
<td style="border: 1px solid black; padding: 8px;">#5 Cyrus Poonawalla <br/></td>
```

```
for i in tablerow[1:9]:
    er=i.find_all('td')
    eachrow=[j.text for j 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']
```

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

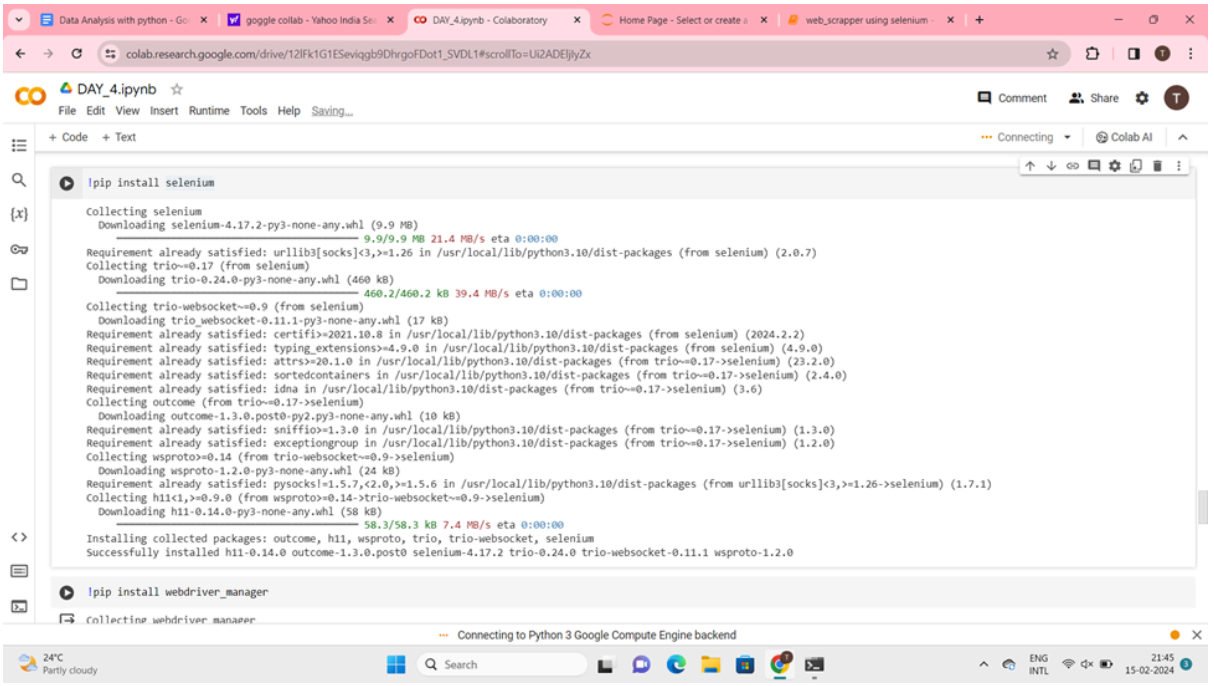
#Program flow

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:**



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:

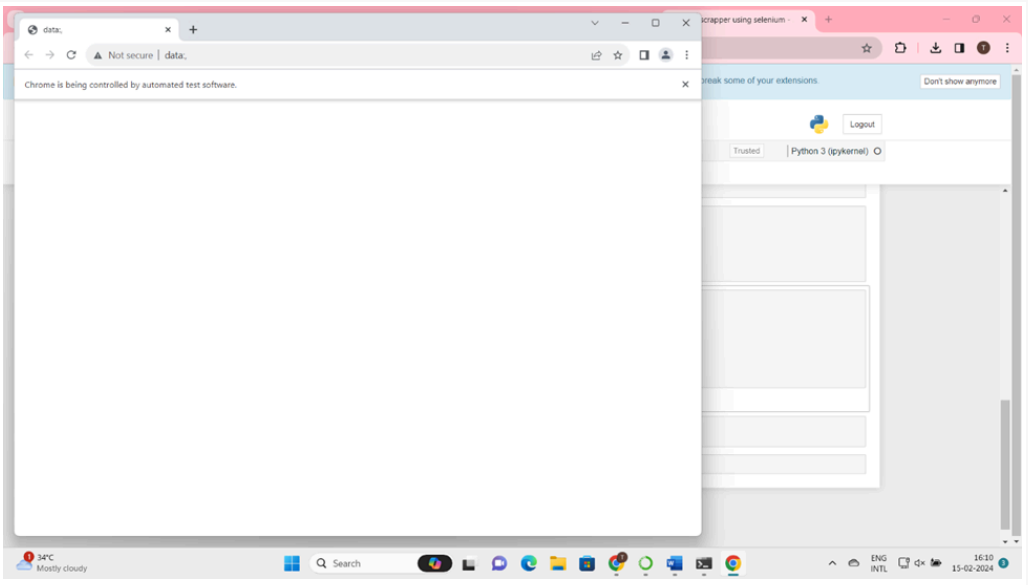
```
from selenium import webdriver
from time import sleep
from selenium.webdriver.chrome.options import Options
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
```

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:

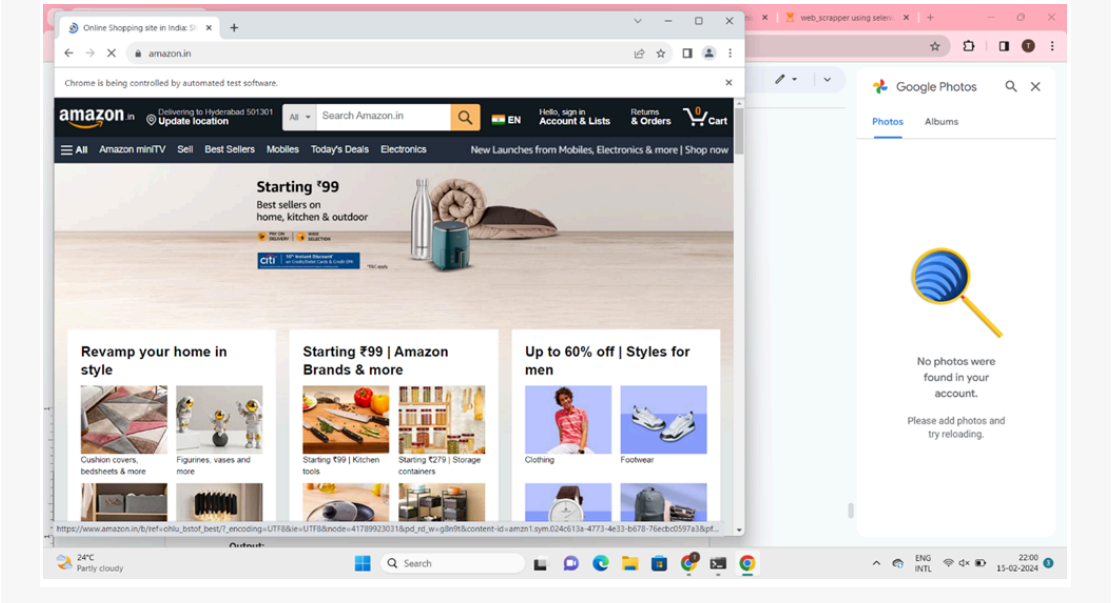
Chrome



Code:

#navigate to a website
driver.get('https://www.amazon.in')

Output:



Code:

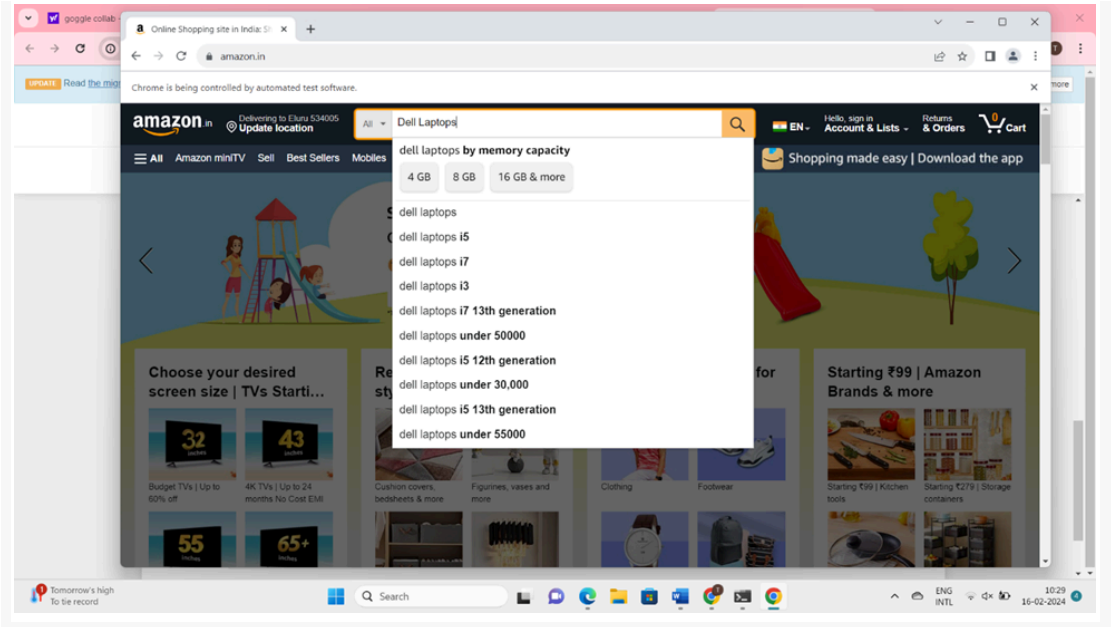
search=driver.find_element(By.ID,"twotabsearchtextbox")

Output:

Code:

search.send_keys("Dell Laptops")

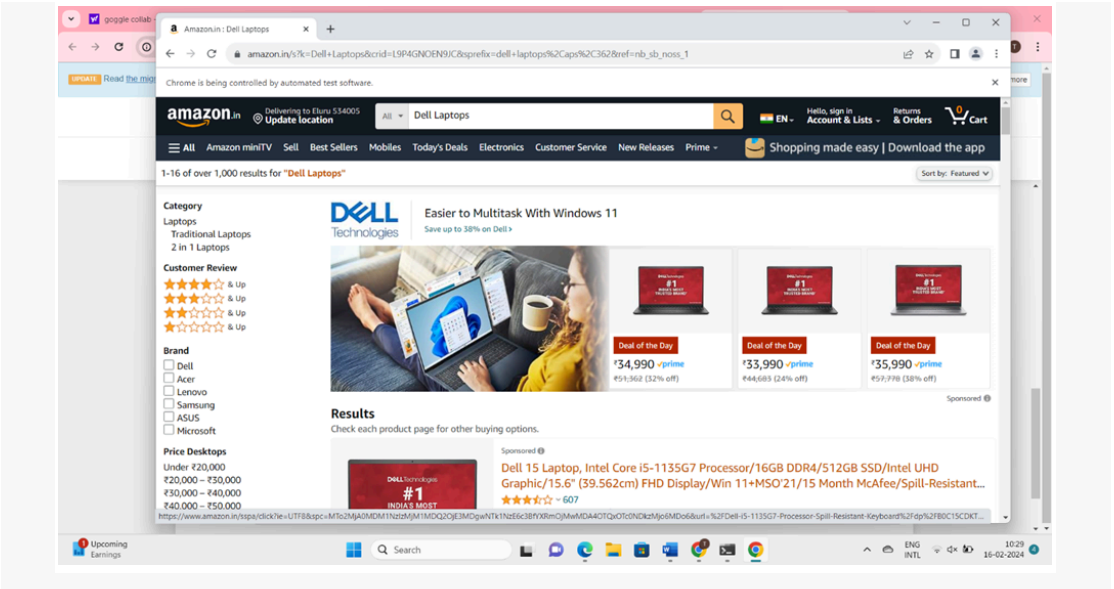
Output:



Code:

driver.find_element(By.ID,"nav-search-submit-button").click()

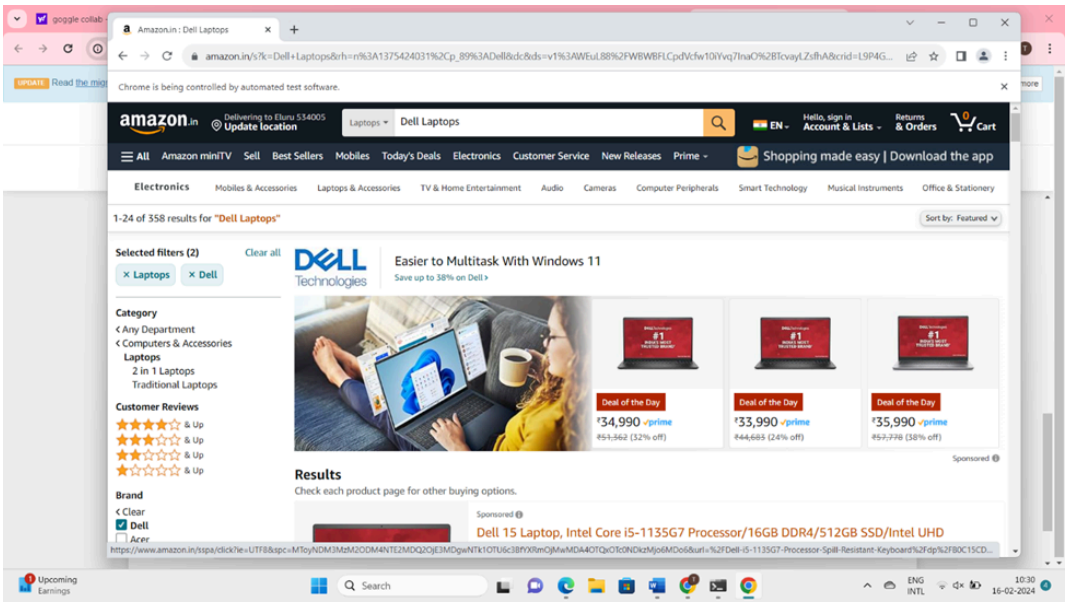
Output:



Code:

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

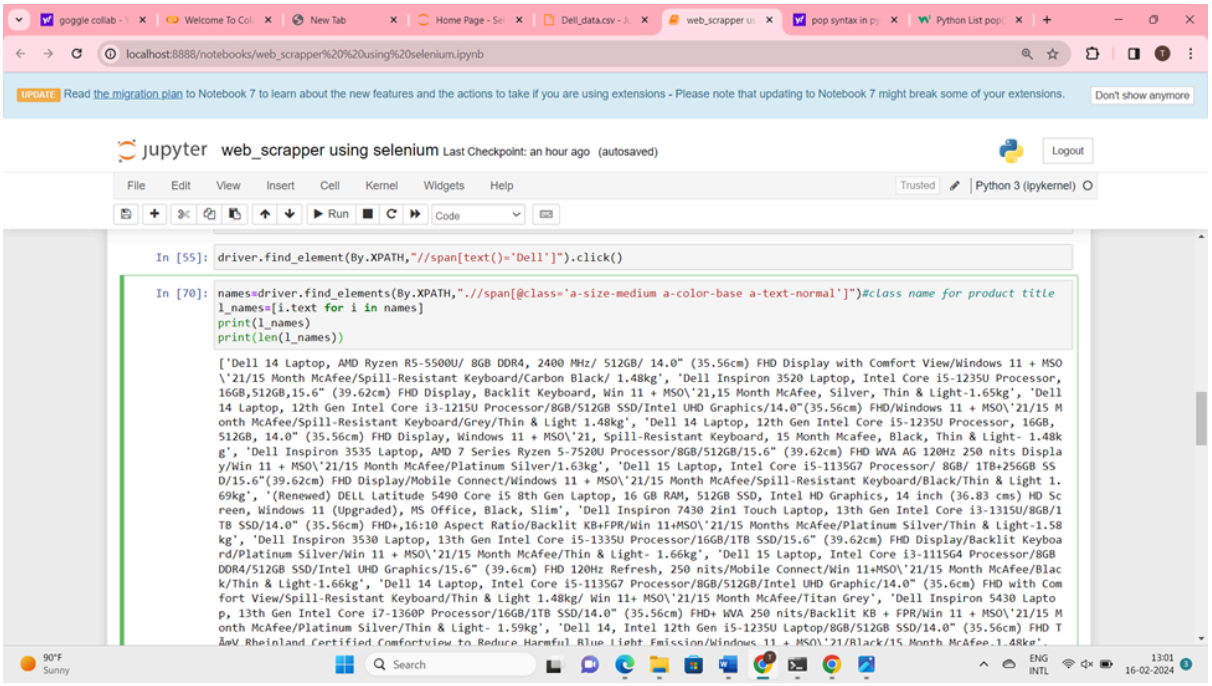
Output:



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']
```

```
['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)

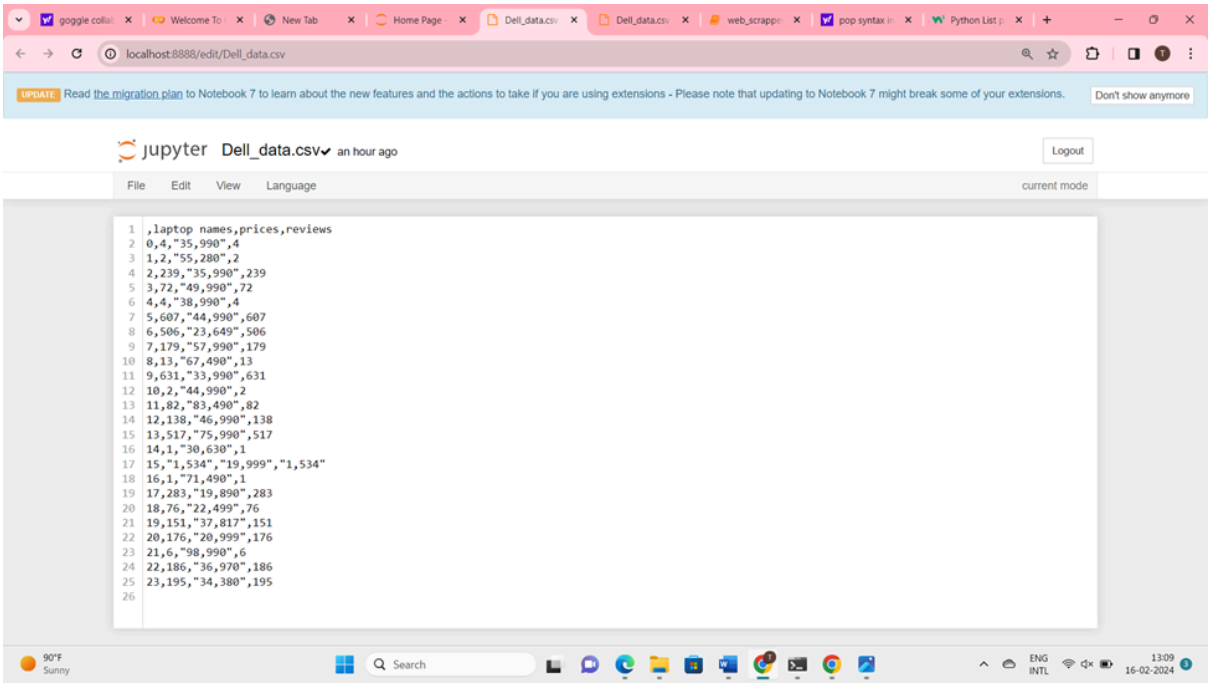
Output:

```
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:

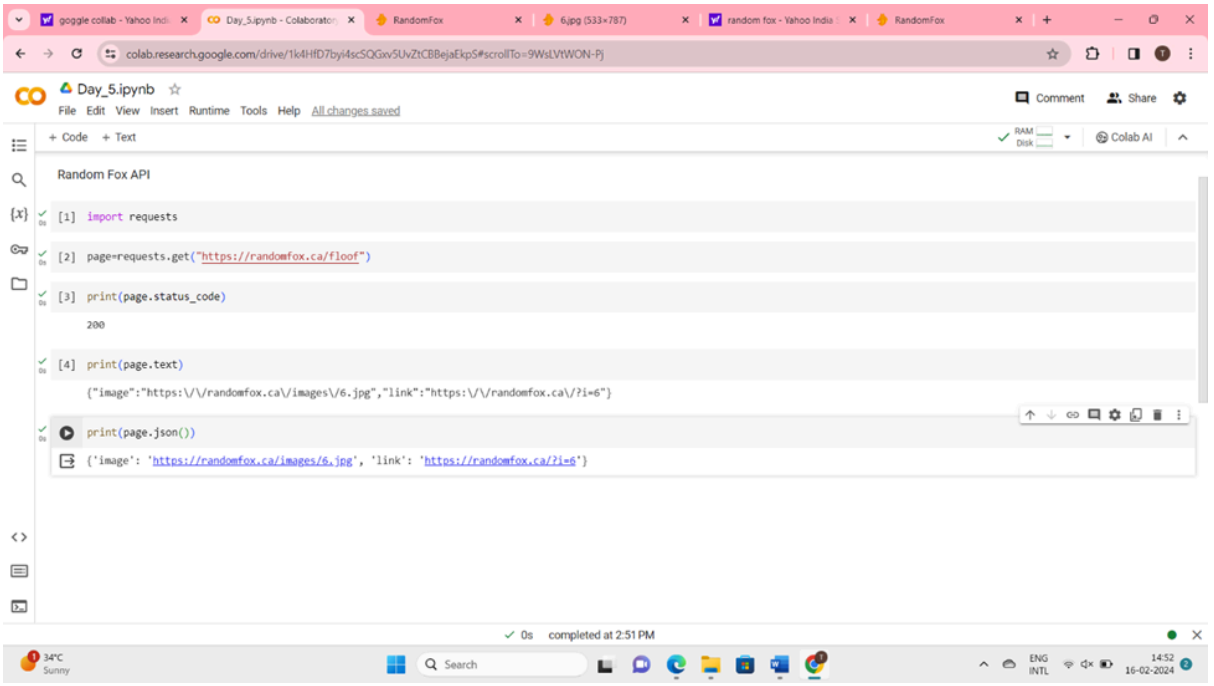
```
{"image":"https:\/\/randomfox.ca\/images\/6.jpg","link":"https:\/\/rand
omfox.ca\/?i=6"}
```

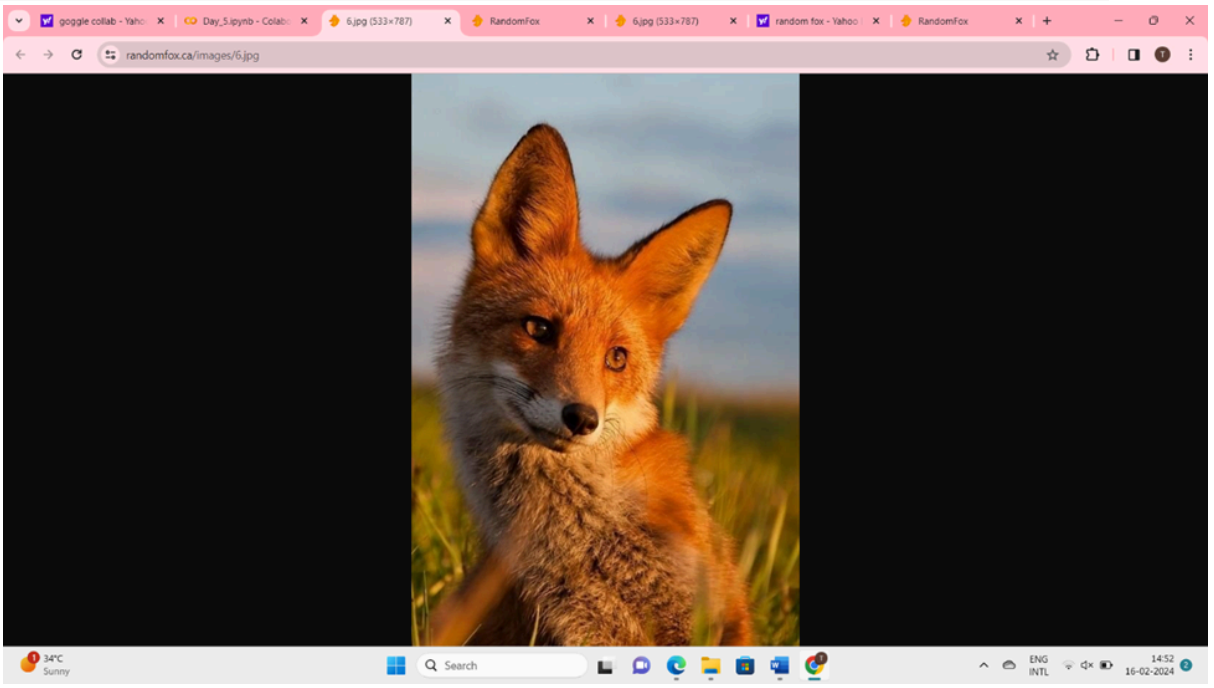
Code:

```
print(page.json())
```

Output:

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



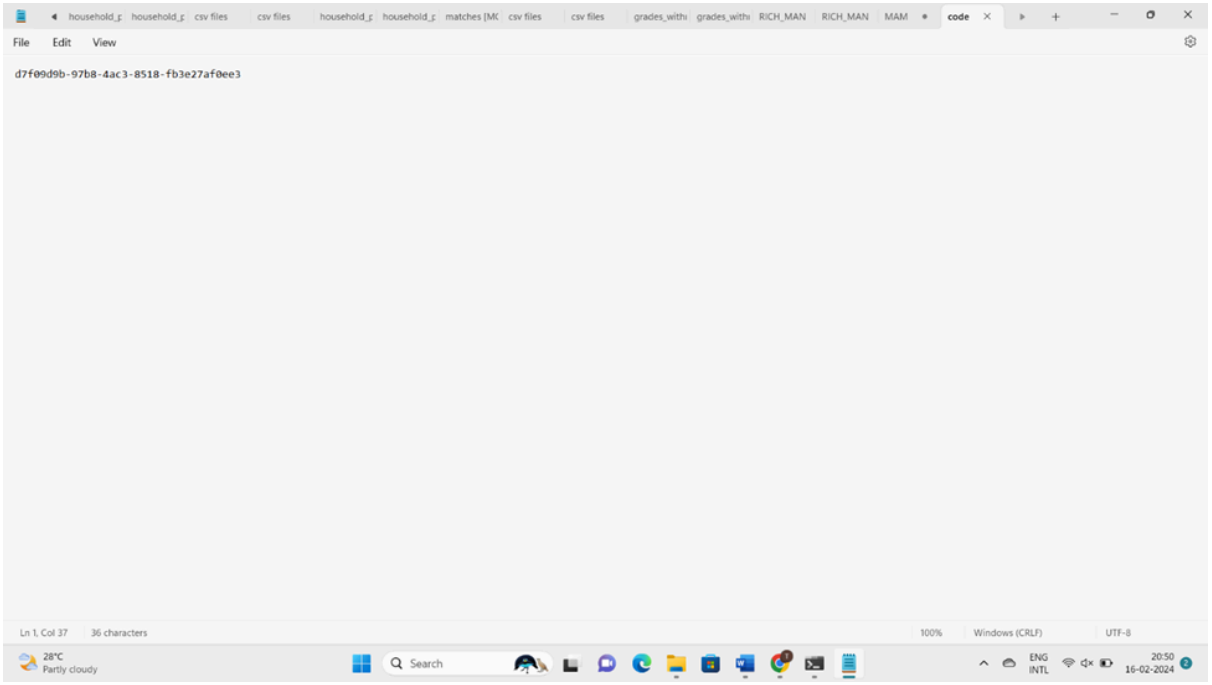


- search **coinmarketcap** and enter

- And scroll down that page at the bottom there is a Crypto API in products
- Go to Crypto API

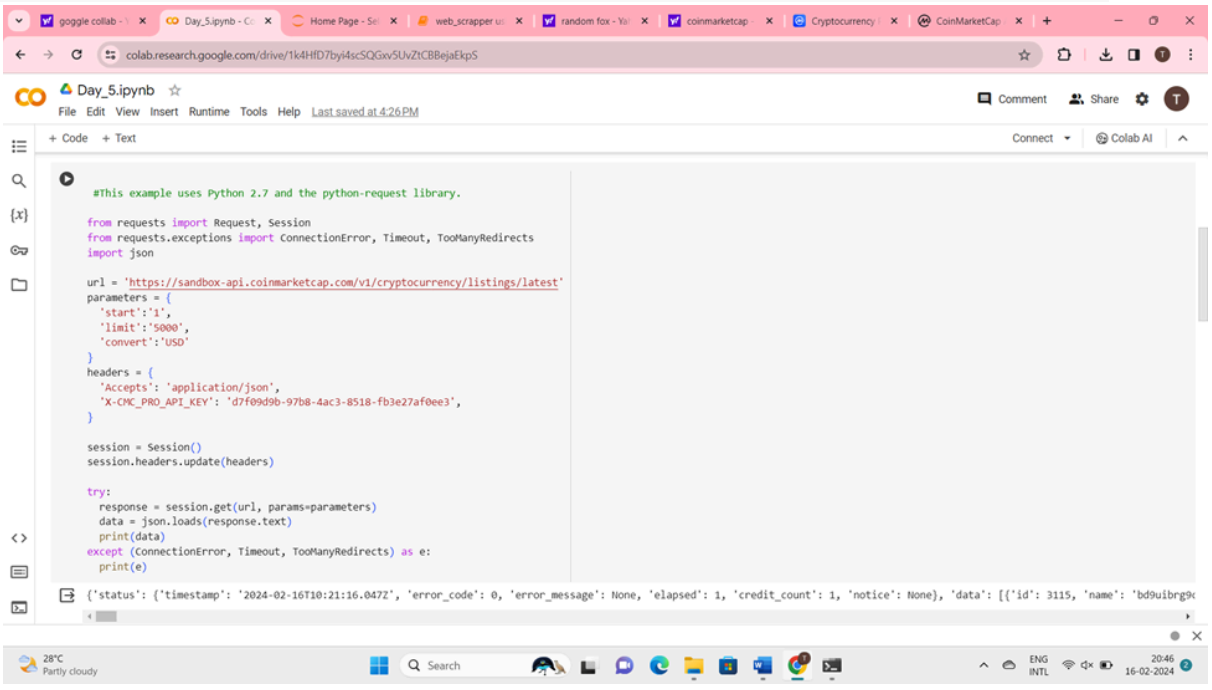
- Next **login**
- After login to that page,you get the **key value**

- 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**

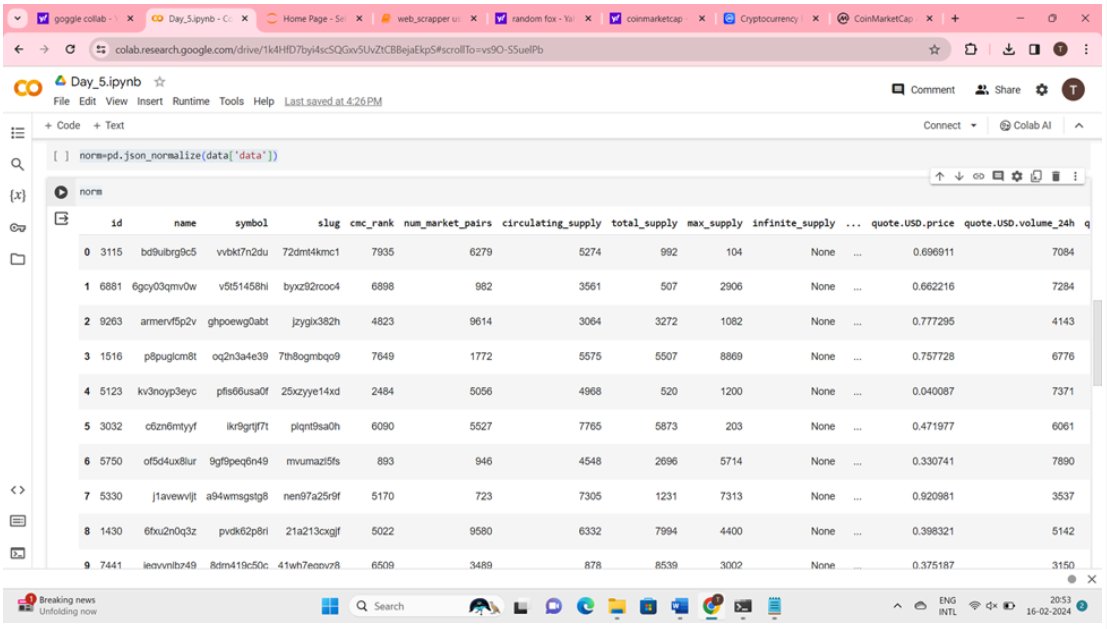
- Then replace the key with your **original key** given below,




```
import pandas as pd
```

```
norm=pd.json_normalize(data['data'])

norm
```

Output:



	id	name	symbol	slug	cmc_rank	num_market_pairs	circulating_supply	total_supply	max_supply	infinite_supply	...	quote.USD.price	quote.USD.volume_24h	q
0	3115	bd9uibrg9c5	vvbkt7n2du	72dmt4kmc1	7935	6279	5274	992	104	None	...	0.696911	7084	
1	6881	6gcy03qmv0w	v5t51458hi	byxz92rcoc4	6898	982	3561	507	2906	None	...	0.662216	7284	
2	9263	armervf5p2v	ghpoewg0abt	jzyglx382h	4823	9614	3064	3272	1082	None	...	0.777295	4143	
3	1516	p8puglcm8t	oq2n3a4e39	7th8ogmbqo9	7649	1772	5575	5507	8869	None	...	0.757728	6776	
4	5123	kv3noyp3eyc	pfis66usa0f	25xzyye14xd	2484	5056	4968	520	1200	None	...	0.040087	7371	
5	3032	c6zn6mtyyf	ikr9grtjf7t	plqnt9sa0h	6090	5527	7765	5873	203	None	...	0.471977	6061	
6	5750	of5d4ux8lur	9gf9peq6n49	mvumazi5fs	893	946	4548	2696	5714	None	...	0.330741	7890	
7	5330	j1avewvljt	a94wmsgstg8	nen97a25r9f	5170	723	7305	1231	7313	None	...	0.920981	3537	
8	1430	6fxu2n0q3z	pvdk62p8ri	21a213cxgjf	5022	9580	6332	7994	4400	None	...	0.398321	5142	
9	7441	iegvvnlbz49	8drn419c50c	41wh7eqpvz8	6509	3489	878	8539	3002	None	...	0.375187	3150	

MACHINE LEARNING

There are 3 types of machine learnings

1)Supervised Machine Learning

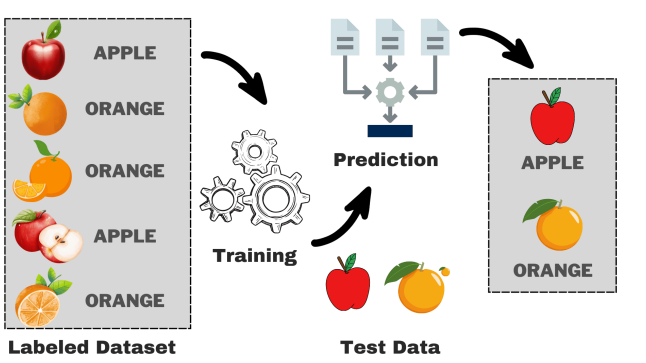
2)Unsupervised Machine Learning

3)Reinforcement Machine Learning

1.Supervised Machine Learning :

In supervised machine learning technique,we trained machines using the “labelled” dataset. So first, we will provide the training to machine to understand the images such as shape,size,color etc.

Example:



When you proved a new label dataset to machine it search for features in the machine

and it will look for the feature,if the feature value matches with label then it will give that feature as the output

Machine Learning Algorithm:

1) Linear

2)Forests

3)Logistic Regression

4)Decision tree etc.

Linear Regression:

> Learns from the labelled datasets and maps the data points to the most optimized linear functions

>These points can be used for prediction new datasets

Dependent and Independent Variable:

>Independent:

The independent variable is the cause. Its value is independent of other variable 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(gm)	40	120	180	210	240

Time-Independent->Feature

Mass-Dependent->Lable

Linear Algebra Calculation

Find the mean of dependent and independent variable

Linear regression line is

$$Y=a+bx$$

Decision Trees:

>Decision trees in machine learning provide an effective method for making decisions because they lay out the problem and all the 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

Case Study:

• Taking a dataset of 26 states with features like Literacy, Cleanliness,

Crime Rate and targetting (predicting) Good or Bad state!

• Good is called Target variable here, it has values of Os and 1s

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

>Decision tree recurrently (continously) 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,Z=0](pure leaf)
- False=[A,E,F,G,I,K,L,P,R,U,V=0];[B,D,H,J,M,O,S,T,W,Y,Z=1]
- Its a mix a 0s and 1s
- Reason:The mixed leaf has target variable with both 0s and 1s.Hence this data is splitted once again
- NODE2:
- CR>50
- True=[A,E,F,G,I,K,L,P,R,U,V=0](ALL ARE 0s,It is pure leaf)
- False=[B,D,H,J,M,O,S,T,W,Y,Z=1](ALL ARE 1s,It is Pure leaf)

Condition	Good
CR>60	0
CR<60	Cannot be determine
CR<50	1

Random Forest:The collection of trees

>Case study

.x0,x1,x2,x3,x4 are features

Bootstrapping: Splitting the parent dataset into child

>Having same no of rows in child ,should have different combination

id	x0	x1	x2	x3	x4	y
1	4.3	4.9	4.4	4.7	5.5	0
2	3.9	6.1	5.9	5.5	5.9	0
3	2.7	4.8	4.1	5.0	5.6	0
4	6.6	4.4	4.5	3.9	5.9	1
5	6.5	2.7	4.7	4.6	6.1	1
6	2.9	6.7	4.2	5.3	4.8	1