Untitled185

August 15, 2024

1 INTERVIEW QUESTIONS

[5]:
```
import pandas as pd
```

[7]:
```
import pandas as pd

# Sample DataFrames
db_employee = pd.DataFrame({
    'employee_id': [1, 2, 3, 4],
    'name': ['Alice', 'Bob', 'Charlie', 'David'],
    'department_id': [10, 20, 30, 40],
    'salary': [30000,40000,50000,60000]

})

db_dept = pd.DataFrame({
    'id': [10, 20, 30, 40],
    'department_name': ['HR', 'IT', 'Finance', 'Marketing'],
    'salary': [30000,40000,50000,70000]
})

# Merge DataFrames
df = db_employee.merge(db_dept, how='left', left_on='department_id',␣
 ↪right_on='id')

# Display the merged DataFrame
print(df)
```

```
   employee_id     name  department_id  salary_x  id department_name  salary_y
0            1    Alice             10     30000  10              HR     30000
1            2      Bob             20     40000  20              IT     40000
2            3  Charlie             30     50000  30         Finance     50000
3            4    David             40     60000  40       Marketing     70000
```

[15]:
```
df.groupby('department_id')['salary_x'].max().reset_index()
```

[15]:
```
   department_id  salary_x
0             10     30000
```

```
1               20      40000
2               30      50000
3               40      60000
```

[16]:
```
## what are the diffirence between the list and tuple
both are the data structure in python list is the mutable tuple is imutable
the item list can be change where as tuple can not be change
```

[19]:
```
## what is the indexing
##indexing is the escending item from the data strure li
list1=['a','b','c','d']
list1[1:3]
```

[19]:
```
['b', 'c']
```

[ ]:
```
## if and else statement
##if statement allows us to execute based on subconditions if the statement is␣
 ↪satified its execute oterwise it not execute
#the else statement can be used with if statement to execute the block of code␣
 ↪when the if condition is not satisfied
```

[65]:
```
## we want to check weather the person is capable of voting or not
age = 20
if age < 18:
    print("you con't vote")
else:
    print("you can vote")
```

```
you can vote
```

[21]:
```
## how do we wright a function
##biggining of line we wright def ofter we put the instance and give the␣
 ↪paranthesis
##inside the parathesis give parameter leave it no problem and give the colon:
def simple():
    print("my first function")
```

[22]:
```
##cal the function
simple()
```

```
my first function
```

[ ]:
```
## what are the libraries used in python.
numpy,pandas,seabon,matplotlib,plotly.sklearn
```

[ ]:
```
## how is numpy and pandas used in analytics
```

```
numpy is used for numerical analysis and linear algebra operation and matrix
 ↪operation
pandas is used for data wrangling,data profiling,data analysis.
```

[ ]:
```
## what is the linear regression and assumption of linear regression
Linear regression is used to predict the numerical and continuous values we fit
 ↪the model with equitiony=mx+c
minimize the sum of squire errors
```

[ ]:
```
## What is the life cycles of data science projects
understand the bussiness requirement from clints
data collection from the various sources
data analysis such as EDA data preparation,data cleaning
feature engineering and feature sellection
hyper parameter tuning
model building and training data
testing data and devolopment
```

[ ]:
```
## what is the data warehouse why do we need it
data warehouse are used store the data for analytics
OLAP AND REAL TIME TRANSACTION
```

[23]:
```
##.Write a program to demonstrate different number data types in Python.
a=10
b=1.5
c=2.05j
```

[24]:
```
## delet one item from list
list=['v','i','k','a','s']
del list[2]
```

[25]:
```
list
```

[25]:
```
['v', 'i', 'a', 's']
```

[28]:
```
## delet multiple item from list
del list[1:5]
```

[29]:
```
list
```

[29]:
```
['v']
```

[30]:
```
##delet entire list
del list
```

[31]:
```
print(list)
```

```
<class 'list'>
```

[34]:
```
##. Write a program to demonstrate working with tuples in python
## empty tuple
tuple=()
print(tuple)
```

```
()
```

[35]:
```
## tuple having integer
tuple=(1,2,3)
print(tuple)
```

```
(1, 2, 3)
```

[36]:
```
## tuple with mixed data type
tuple=(1,"hello",3.5)
print(tuple)
```

```
(1, 'hello', 3.5)
```

[37]:
```
## nested tuple
tuple = ("mouse",[2,3,4],(1,5,7))
print(tuple)
```

```
('mouse', [2, 3, 4], (1, 5, 7))
```

[38]:
```
##. Write a python program to find largest of three numbers
num1=10
num2=14
num3=12
if(num1>=num2)and(num1>=num3):
    lagest=num1
elif(num2>=num1)and(num2>=num3):
    largest=num2
else:
    largest=num3
print("print lagest number is",largest)
```

```
print lagest number is 14
```

[46]:
```
#. Write a Python program to construct the following pattern, using a
#nested for loop
n = 5

# First Part: Increasing Stars
for i in range(n):
    for j in range(i):
        print('* ', end="")
    print("")

# Second Part: Decreasing Stars
```

```
for i in range(n, 0, -1):
    for j in range(i):
        print('* ', end="")
    print("")
```

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

[49]:
```
##Python program to check if a string is palindrome or not
def ispalindrome(s):
    return s == s[::-1]

s = "malayalam"
ans = ispalindrome(s)

if ans:
    print('yes')
else:
    print('no')
```

```
yes
```

[ ]:
```
## what are the class and object in python
class is a blue print and object is houses
cal the class creat an object
```

[51]:
```
class human:
    name=None
    age=None
    def get_name(self):
        print("enter the name")
        self.name=input()
    def get_age(self):
        print("enter the age")
        self.age=input()
    def put_name(self):
        print("your name is",self.name)
    def put_age(self):
        print("your age is",self.age)
```

[52]:
```
person1=human()
```

[53]:
```
person1.get_name()
```

```
enter the name
vikas
```

[54]:
```
person1.get_age()
```

```
enter the age
30
```

[55]:
```
person1.put_age(),person1.put_name()
```

```
your age is 30
your name is vikas
```

[55]:
```
(None, None)
```

[ ]:
```
## what do we understand by __init__()methode in python,give an example of it.
--init__ is the special methode in python classes
it is the constructor methode of the python classes
__init is when ever object of the class is constructor
it is used to initialize the variables
```

[66]:
```
class student:
    def __init__(self,name,age,branch):
        self.name=name
        self.age=age
        self.branch=branch
    def print_student(self):
        print("name",self.name)
        print("age",self.age)
        print("branch",self.branch)
```

[67]:
```
student1 = student("vikas",30,"engineering")
```

[68]:
```
student1.print_student()
```

```
name vikas
age 30
branch engineering
```

[70]:
```
## what do you understand by inherritence in python,give example of it
##one class is inheriting the properties of another class
class fruit:
    def __init__(self):
        print('IAM A FRUIT')

class citrus(fruit):
```

```
    def __init__(self):
        super().__init__()
        print("iam a citrus")

lemon = citrus()
```

```
IAM A FRUIT
iam a citrus
```

[ ]:
```
## whst is the numoy how can you create 1darray 2d array
numpy is widly used python library and linear algebra and used to performing␣
 ↪mathemetical& logical functions
```

[71]:
```
import numpy as np
```

[72]:
```
a=np.array([1,2,3])
```

[73]:
```
a
```

[73]:
```
array([1, 2, 3])
```

[74]:
```
b=np.array([[1,2,3],[4,5,6]])
```

[75]:
```
b
```

[75]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

[76]:
```
## how can we initialize 5*5 numpy array comparising of all zeros
a=np.zeros((5,5))
```

[77]:
```
a
```

[77]:
```
array([[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.]])
```

[78]:
```
## add the two numpy array
a=np.array([1,2,3])
b=np.array([4,5,6])
np.sum((a,b),axis=0)
```

[78]:
```
array([5, 7, 9])
```

[79]:
```
a=np.array([1,2,3])
b=np.array([4,5,6])
```

```
np.sum((a,b),axis=1)
```

[79]:
```
array([ 6, 15])
```

[86]:
```
## in numpy array find the 2lagest values
x=np.array([12,43,2,100,54,5,68])
```

[87]:
```
x
```

[87]:
```
array([ 12,  43,   2, 100,  54,   5,  68])
```

[88]:
```
x[np.argsort(x)[-2:][::1]]
```

[88]:
```
array([ 68, 100])
```

[89]:
```
import pandas as pd
## give  an examples  creating a dataframe list and dictionary.
data=[1,2,3,4]
df=pd.DataFrame(data)
```

[90]:
```
df
```

[90]:

	0
0	1
1	2
2	3
3	4

[92]:
```
dict1 = {'fruit':['apple','mango','banana'],'count':[10,20,30]}
df=pd.DataFrame(dict1)
```

[93]:
```
df
```

[93]:

	fruit	count
0	apple	10
1	mango	20
2	banana	30

[94]:
```
## extract the sapel lenght greater then 5 amd sapel widthgreater then 3,
iris =pd.read_csv('iris.csv')
```

[95]:
```
iris.head()
```

[95]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa

```
4           5.0          3.6           1.4          0.2  setosa
```

[96]:
```
iris[(iris['sepal_length']>5)&(iris['sepal_width']>3)]
```

[96]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
10	5.4	3.7	1.5	0.2	setosa
14	5.8	4.0	1.2	0.2	setosa
15	5.7	4.4	1.5	0.4	setosa
16	5.4	3.9	1.3	0.4	setosa
17	5.1	3.5	1.4	0.3	setosa
18	5.7	3.8	1.7	0.3	setosa
19	5.1	3.8	1.5	0.3	setosa
20	5.4	3.4	1.7	0.2	setosa
21	5.1	3.7	1.5	0.4	setosa
23	5.1	3.3	1.7	0.5	setosa
27	5.2	3.5	1.5	0.2	setosa
28	5.2	3.4	1.4	0.2	setosa
31	5.4	3.4	1.5	0.4	setosa
32	5.2	4.1	1.5	0.1	setosa
33	5.5	4.2	1.4	0.2	setosa
36	5.5	3.5	1.3	0.2	setosa
39	5.1	3.4	1.5	0.2	setosa
44	5.1	3.8	1.9	0.4	setosa
46	5.1	3.8	1.6	0.2	setosa
48	5.3	3.7	1.5	0.2	setosa
50	7.0	3.2	4.7	1.4	versicolor
51	6.4	3.2	4.5	1.5	versicolor
52	6.9	3.1	4.9	1.5	versicolor
56	6.3	3.3	4.7	1.6	versicolor
65	6.7	3.1	4.4	1.4	versicolor
70	5.9	3.2	4.8	1.8	versicolor
85	6.0	3.4	4.5	1.6	versicolor
86	6.7	3.1	4.7	1.5	versicolor
100	6.3	3.3	6.0	2.5	virginica
109	7.2	3.6	6.1	2.5	virginica
110	6.5	3.2	5.1	2.0	virginica
115	6.4	3.2	5.3	2.3	virginica
117	7.7	3.8	6.7	2.2	virginica
120	6.9	3.2	5.7	2.3	virginica
124	6.7	3.3	5.7	2.1	virginica
125	7.2	3.2	6.0	1.8	virginica
131	7.9	3.8	6.4	2.0	virginica
136	6.3	3.4	5.6	2.4	virginica
137	6.4	3.1	5.5	1.8	virginica
139	6.9	3.1	5.4	2.1	virginica

```
140           6.7          3.1           5.6          2.4   virginica
141           6.9          3.1           5.1          2.3   virginica
143           6.8          3.2           5.9          2.3   virginica
144           6.7          3.3           5.7          2.5   virginica
148           6.2          3.4           5.4          2.3   virginica
```

[97]:
```
## how can you intruduce the Nan values in the first ten rows sapel length  and␣
 ↪sapel width
iris.head()
```

[97]:
```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
```

[98]:
```
iris1=iris
```

[99]:
```
iris1.iloc[0:10,1]=np.NAN
```

[100]:
```
iris1.head()
```

[100]:
```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          NaN           1.4          0.2  setosa
1           4.9          NaN           1.4          0.2  setosa
2           4.7          NaN           1.3          0.2  setosa
3           4.6          NaN           1.5          0.2  setosa
4           5.0          NaN           1.4          0.2  setosa
```

[101]:
```
iris1.iloc[0:10,2]=np.NAN
```

[102]:
```
iris1.head()
```

[102]:
```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          NaN           NaN          0.2  setosa
1           4.9          NaN           NaN          0.2  setosa
2           4.7          NaN           NaN          0.2  setosa
3           4.6          NaN           NaN          0.2  setosa
4           5.0          NaN           NaN          0.2  setosa
```

[104]:
```
## how can i count NAN values
iris1.isna().sum()
```

[104]:
```
sepal_length     0
sepal_width     10
petal_length    10
petal_width      0
```

```
species          0
dtype: int64
```

[121]:
```
## how can i read the file   ## "r"represent opening the file read mode
f=open("C:/Users/Vikas/Desktop/iris.csv","r")
```

[122]:
```
f.read()
```

[122]:
```
'sepal_length,sepal_width,petal_length,petal_width,species\n5.1,3.5,1.4,0.2,seto
sa\n4.9,3.0,1.4,0.2,setosa\n4.7,3.2,1.3,0.2,setosa\n4.6,3.1,1.5,0.2,setosa\n5.0,
3.6,1.4,0.2,setosa\n5.4,3.9,1.7,0.4,setosa\n4.6,3.4,1.4,0.3,setosa\n5.0,3.4,1.5,
0.2,setosa\n4.4,2.9,1.4,0.2,setosa\n4.9,3.1,1.5,0.1,setosa\n5.4,3.7,1.5,0.2,seto
sa\n4.8,3.4,1.6,0.2,setosa\n4.8,3.0,1.4,0.1,setosa\n4.3,3.0,1.1,0.1,setosa\n5.8,
4.0,1.2,0.2,setosa\n5.7,4.4,1.5,0.4,setosa\n5.4,3.9,1.3,0.4,setosa\n5.1,3.5,1.4,
0.3,setosa\n5.7,3.8,1.7,0.3,setosa\n5.1,3.8,1.5,0.3,setosa\n5.4,3.4,1.7,0.2,seto
sa\n5.1,3.7,1.5,0.4,setosa\n4.6,3.6,1.0,0.2,setosa\n5.1,3.3,1.7,0.5,setosa\n4.8,
3.4,1.9,0.2,setosa\n5.0,3.0,1.6,0.2,setosa\n5.0,3.4,1.6,0.4,setosa\n5.2,3.5,1.5,
0.2,setosa\n5.2,3.4,1.4,0.2,setosa\n4.7,3.2,1.6,0.2,setosa\n4.8,3.1,1.6,0.2,seto
sa\n5.4,3.4,1.5,0.4,setosa\n5.2,4.1,1.5,0.1,setosa\n5.5,4.2,1.4,0.2,setosa\n4.9,
3.1,1.5,0.1,setosa\n5.0,3.2,1.2,0.2,setosa\n5.5,3.5,1.3,0.2,setosa\n4.9,3.1,1.5,
0.1,setosa\n4.4,3.0,1.3,0.2,setosa\n5.1,3.4,1.5,0.2,setosa\n5.0,3.5,1.3,0.3,seto
sa\n4.5,2.3,1.3,0.3,setosa\n4.4,3.2,1.3,0.2,setosa\n5.0,3.5,1.6,0.6,setosa\n5.1,
3.8,1.9,0.4,setosa\n4.8,3.0,1.4,0.3,setosa\n5.1,3.8,1.6,0.2,setosa\n4.6,3.2,1.4,
0.2,setosa\n5.3,3.7,1.5,0.2,setosa\n5.0,3.3,1.4,0.2,setosa\n7.0,3.2,4.7,1.4,vers
icolor\n6.4,3.2,4.5,1.5,versicolor\n6.9,3.1,4.9,1.5,versicolor\n5.5,2.3,4.0,1.3,
versicolor\n6.5,2.8,4.6,1.5,versicolor\n5.7,2.8,4.5,1.3,versicolor\n6.3,3.3,4.7,
1.6,versicolor\n4.9,2.4,3.3,1.0,versicolor\n6.6,2.9,4.6,1.3,versicolor\n5.2,2.7,
3.9,1.4,versicolor\n5.0,2.0,3.5,1.0,versicolor\n5.9,3.0,4.2,1.5,versicolor\n6.0,
2.2,4.0,1.0,versicolor\n6.1,2.9,4.7,1.4,versicolor\n5.6,2.9,3.6,1.3,versicolor\n
6.7,3.1,4.4,1.4,versicolor\n5.6,3.0,4.5,1.5,versicolor\n5.8,2.7,4.1,1.0,versicol
or\n6.2,2.2,4.5,1.5,versicolor\n5.6,2.5,3.9,1.1,versicolor\n5.9,3.2,4.8,1.8,vers
icolor\n6.1,2.8,4.0,1.3,versicolor\n6.3,2.5,4.9,1.5,versicolor\n6.1,2.8,4.7,1.2,
versicolor\n6.4,2.9,4.3,1.3,versicolor\n6.6,3.0,4.4,1.4,versicolor\n6.8,2.8,4.8,
1.4,versicolor\n6.7,3.0,5.0,1.7,versicolor\n6.0,2.9,4.5,1.5,versicolor\n5.7,2.6,
3.5,1.0,versicolor\n5.5,2.4,3.8,1.1,versicolor\n5.5,2.4,3.7,1.0,versicolor\n5.8,
2.7,3.9,1.2,versicolor\n6.0,2.7,5.1,1.6,versicolor\n5.4,3.0,4.5,1.5,versicolor\n
6.0,3.4,4.5,1.6,versicolor\n6.7,3.1,4.7,1.5,versicolor\n6.3,2.3,4.4,1.3,versicol
or\n5.6,3.0,4.1,1.3,versicolor\n5.5,2.5,4.0,1.3,versicolor\n5.5,2.6,4.4,1.2,vers
icolor\n6.1,3.0,4.6,1.4,versicolor\n5.8,2.6,4.0,1.2,versicolor\n5.0,2.3,3.3,1.0,
versicolor\n5.6,2.7,4.2,1.3,versicolor\n5.7,3.0,4.2,1.2,versicolor\n5.7,2.9,4.2,
1.3,versicolor\n6.2,2.9,4.3,1.3,versicolor\n5.1,2.5,3.0,1.1,versicolor\n5.7,2.8,
4.1,1.3,versicolor\n6.3,3.3,6.0,2.5,virginica\n5.8,2.7,5.1,1.9,virginica\n7.1,3.
0,5.9,2.1,virginica\n6.3,2.9,5.6,1.8,virginica\n6.5,3.0,5.8,2.2,virginica\n7.6,3
.0,6.6,2.1,virginica\n4.9,2.5,4.5,1.7,virginica\n7.3,2.9,6.3,1.8,virginica\n6.7,
2.5,5.8,1.8,virginica\n7.2,3.6,6.1,2.5,virginica\n6.5,3.2,5.1,2.0,virginica\n6.4
,2.7,5.3,1.9,virginica\n6.8,3.0,5.5,2.1,virginica\n5.7,2.5,5.0,2.0,virginica\n5.
8,2.8,5.1,2.4,virginica\n6.4,3.2,5.3,2.3,virginica\n6.5,3.0,5.5,1.8,virginica\n7
```

```
.7,3.8,6.7,2.2,virginica\n7.7,2.6,6.9,2.3,virginica\n6.0,2.2,5.0,1.5,virginica\n
6.9,3.2,5.7,2.3,virginica\n5.6,2.8,4.9,2.0,virginica\n7.7,2.8,6.7,2.0,virginica\
n6.3,2.7,4.9,1.8,virginica\n6.7,3.3,5.7,2.1,virginica\n7.2,3.2,6.0,1.8,virginica
\n6.2,2.8,4.8,1.8,virginica\n6.1,3.0,4.9,1.8,virginica\n6.4,2.8,5.6,2.1,virginic
a\n7.2,3.0,5.8,1.6,virginica\n7.4,2.8,6.1,1.9,virginica\n7.9,3.8,6.4,2.0,virgini
ca\n6.4,2.8,5.6,2.2,virginica\n6.3,2.8,5.1,1.5,virginica\n6.1,2.6,5.6,1.4,virgin
ica\n7.7,3.0,6.1,2.3,virginica\n6.3,3.4,5.6,2.4,virginica\n6.4,3.1,5.5,1.8,virgi
nica\n6.0,3.0,4.8,1.8,virginica\n6.9,3.1,5.4,2.1,virginica\n6.7,3.1,5.6,2.4,virg
inica\n6.9,3.1,5.1,2.3,virginica\n5.8,2.7,5.1,1.9,virginica\n6.8,3.2,5.9,2.3,vir
ginica\n6.7,3.3,5.7,2.5,virginica\n6.7,3.0,5.2,2.3,virginica\n6.3,2.5,5.0,1.9,vi
rginica\n6.5,3.0,5.2,2.0,virginica\n6.2,3.4,5.4,2.3,virginica\n5.9,3.0,5.1,1.8,v
irginica\n'
```

[126]:
```
## use lambdafunction print 10 numbers
x=lambda a: a+10
```

[128]:
```
x(100)
```

[128]:
```
110
```

[129]:
```
## below mentioned points used create line plot
x=np.arange(0,10,1)
```

[130]:
```
x
```

[130]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[131]:
```
y=np.arange(0,10,1)
```

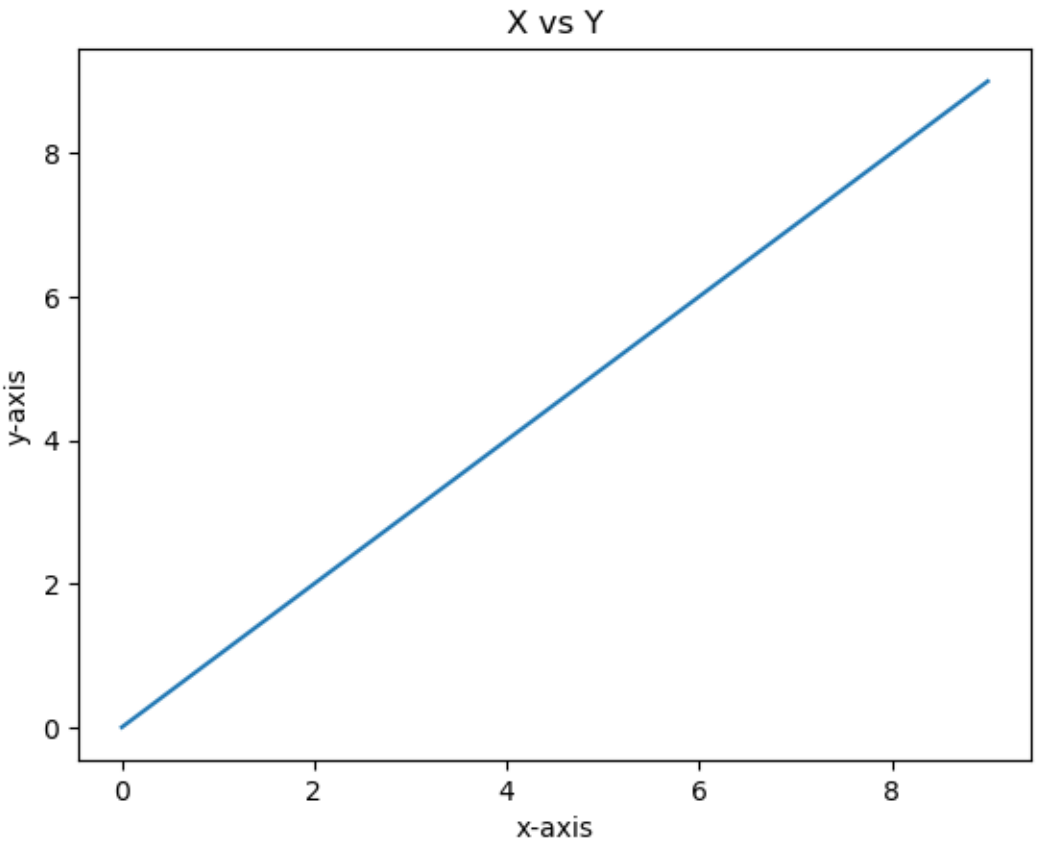
[132]:
```
y
```

[132]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[133]:
```
import matplotlib.pyplot as plt
```

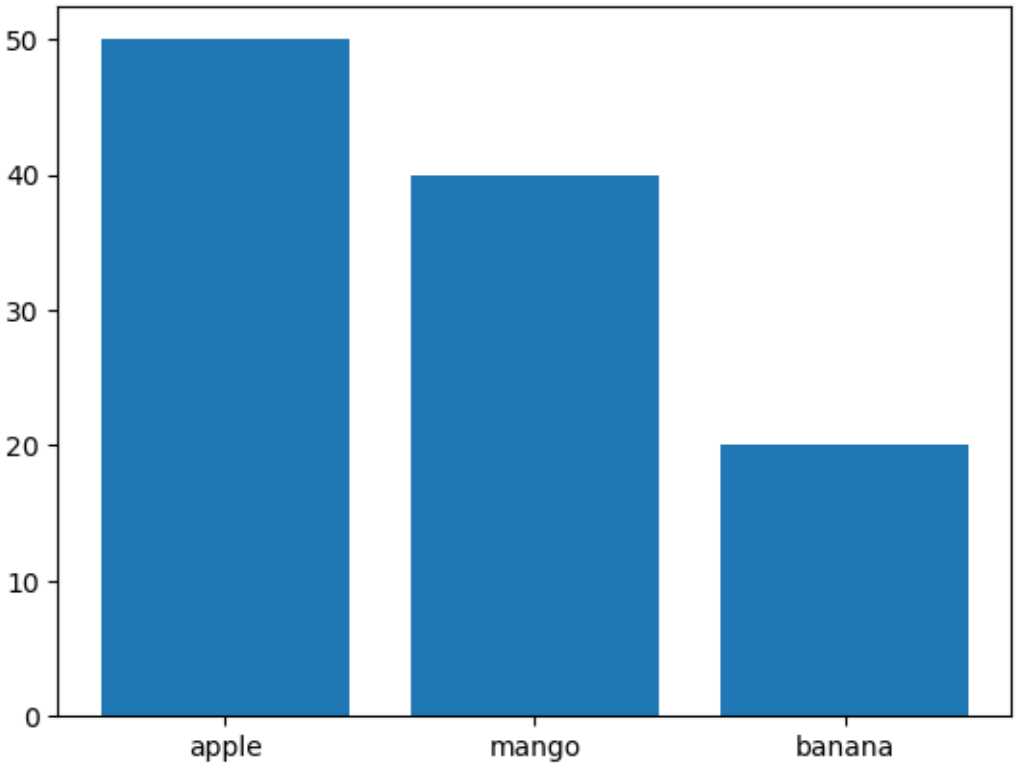
[136]:
```
plt.plot(x,y)
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("X vs Y")
plt.show
```

[136]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



[141]:
```
## CREATE A SIMPLE BAR PLOT WHERE X-AXIS NAMES HAS FRUITS Y-AXIS COST OF FRUITS
data = {'apple':50,'mango':40,'banana':20}
names=list(data.keys())
values=list(data.values())
plt.bar(names,values)
```

[141]: <BarContainer object of 3 artists>



[142]:
```
from random import shuffle
```

[143]:
```
x=['mary','had','a','little','lamb']
```

[144]:
```
shuffle(x)
```

[145]:
```
x
```

[145]:
```
['had', 'mary', 'a', 'little', 'lamb']
```

[146]:
```
## find the length string
a = "vikas"
count=0
for i in a:
    count=count+1
print(count)
```

```
5
```

[147]:
```
len("vikas")
```

[147]:
```
5
```

[151]:
```
## replace all odd value with -1 the numbers in numpy array -1.
arr=np.arange(0,10)
```

[152]:
```
arr
```

[152]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[153]:
```
arr[arr%2==1] =-1

###0 divided by2 check remender wherever remender is equal to 1 changing to  -1
```

[154]:
```
arr
```

[154]:
```
array([ 0, -1,  2, -1,  4, -1,  6, -1,  8, -1])
```

[155]:
```
## how can you get common items between two numpy array.
a=np.array([1,2,3,4,5,6])
b=np.array([2,3,5,6,7,9,0])
np.intersect1d(a,b)
```

[155]:
```
array([2, 3, 5, 6])
```

[156]:
```
## how can you convert first character of each element ipandas series to␣
 ↪uppercase
ser=pd.Series(['marry','had','a','little','lamb'])
```

[157]:
```
ser
```

[157]:
```
0     marry
1       had
2         a
3    little
4      lamb
dtype: object
```

[158]:
```
ser.map(lambda x: x.title())
```

[158]:
```
0     Marry
1       Had
2         A
3    Little
4      Lamb
dtype: object
```

[159]:
```
##how woild  you caluclate the number of character in each word in series
ser = pd.Series(['maryy','had','a','little','lamb'])
ser.map(lambda a:len(a))
```

[159]:
```
0    5
1    3
2    1
3    6
4    4
dtype: int64
```

[160]:
```
## in iris dataset change the col name sepel legth to s length
iris
```

[160]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	NaN	NaN	0.2	setosa
1	4.9	NaN	NaN	0.2	setosa
2	4.7	NaN	NaN	0.2	setosa
3	4.6	NaN	NaN	0.2	setosa
4	5.0	NaN	NaN	0.2	setosa
..	…	…	…	…	…
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

[150 rows x 5 columns]

[162]:
```
iris1=iris.rename(columns={'sepal_length':'s_legth'})
```

[164]:
```
iris1.head()
```

[164]:

	s_legth	sepal_width	petal_length	petal_width	species
0	5.1	NaN	NaN	0.2	setosa
1	4.9	NaN	NaN	0.2	setosa
2	4.7	NaN	NaN	0.2	setosa
3	4.6	NaN	NaN	0.2	setosa
4	5.0	NaN	NaN	0.2	setosa

[ ]:
```
## Build a lenear regression model on this Boston dataset where the independent␣
 ↪vareable is 'rm' dependent varieble is 'medv'
## the train and test set need to be 80:20
```

[165]:
```
## Loading the required package
import pandas as pd
```

[166]:
```
## load the boston dataset
boston = pd.read_csv('Boston1.csv')
```

[167]:
```
boston.head()
```

[167]:
```
      crim    zn  indus  chas    nox     rm   age     dis  rad  tax  ptratio  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296     15.3
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222     18.7

    black  lstat  medv
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90   5.33  36.2
```

[168]:
```
## getting features from the dataset
x= pd.DataFrame(boston['rm'])
y=pd.DataFrame(boston['medv'])
```

[169]:
```
## split the train and test
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20)
```

[171]:
```
## build the model
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(x_train,y_train)
```

[171]:
```
LinearRegression()
```

[172]:
```
## Predicting the value
y_pred = regressor.predict(x_test)
```

[175]:
```
y_pred
```

[175]:
```
array([[24.54588025],
       [22.84168803],
       [27.95426471],
       [31.05444418],
       [14.36605115],
       [21.62699782],
       [24.05637823],
       [17.5478143 ],
       [21.18282006],
       [30.89127684],
       [15.79829781],
       [19.83215707],
       [17.5478143 ],
       [36.4842907 ],
```

```
[24.25580498],
[28.11743205],
[18.45429953],
[ 3.27067197],
[26.03251603],
[17.9466678 ],
[23.61220047],
[29.05111183],
[27.61886517],
[22.22527807],
[19.22481197],
[36.64745804],
[26.340721  ],
[23.10456874],
[21.25533888],
[24.63652878],
[18.98912581],
[33.64699193],
[22.23434292],
[26.10503484],
[10.23247852],
[19.93187045],
[17.09457169],
[23.27680093],
[ 7.93000604],
[23.39464401],
[28.9242039 ],
[25.36171696],
[38.58733642],
[18.85315303],
[22.22527807],
[20.72957745],
[28.77010141],
[35.47809209],
[25.543014  ],
[20.52108584],
[29.34118711],
[12.77063715],
[19.97719471],
[21.66325723],
[21.01058787],
[23.91134059],
[22.19808351],
[32.45949629],
[23.18615241],
[19.5602115 ],
[19.69618429],
```

```
[21.01965272],
[21.7448409 ],
[23.25867123],
[32.05157793],
[28.28059939],
[24.1198322 ],
[21.52728445],
[24.00198912],
[ 6.72438069],
[14.24820807],
[22.04398103],
[30.15702381],
[22.77823406],
[31.6980487 ],
[21.91707309],
[26.7395745 ],
[20.28539969],
[27.22907653],
[25.75150561],
[23.24054153],
[18.45429953],
[19.56927636],
[19.02538522],
[17.67472223],
[18.24580792],
[18.82595847],
[21.66325723],
[22.42470482],
[23.5306168 ],
[23.82069207],
[31.39890857],
[21.31879284],
[25.1260308 ],
[29.76723516],
[20.18568631],
[21.52728445],
[20.88367994],
[18.66279113],
[24.44616688],
[18.18235396],
[16.59600481]])
```

2 TOP 20 FUNCTION IN PANDAS

[177]:
```
## Reading a file
data =pd.read_csv('iris.csv')
```

[178]:
```
## get the top and bottom of dataset
data.head()
```

[178]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

[179]:
```
data.tail()
```

[179]:

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

[180]:
```
## get the complete information about all the columns in dataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

[181]:
```
## see the shape of the dataset
data.shape
```

[181]: (150, 5)

[182]:
```
data.size
```

[182]: 750

[183]:
```
## get the standard methematical analysis each columns
data.describe()
```

[183]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

[184]:
```
data.describe().T                          ## T means "transpose" row to col␣
 ↪col to row convert
```

[184]:

	count	mean	std	min	25%	50%	75%	max
sepal_length	150.0	5.843333	0.828066	4.3	5.1	5.80	6.4	7.9
sepal_width	150.0	3.054000	0.433594	2.0	2.8	3.00	3.3	4.4
petal_length	150.0	3.758667	1.764420	1.0	1.6	4.35	5.1	6.9
petal_width	150.0	1.198667	0.763161	0.1	0.3	1.30	1.8	2.5

[185]:
```
## find the no of distinct value in categorical col
data.nunique()
```

[185]:
```
sepal_length    35
sepal_width     23
petal_length    43
petal_width     22
species          3
dtype: int64
```

[187]:
```
## how to find the missing values
data.isnull().sum()
```

[187]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

[188]:
```
data.isnull().any()
```

[188]:
```
sepal_length    False
sepal_width     False
petal_length    False
petal_width     False
```

```
species         False
dtype: bool
```

[189]:
```
## find the column names
data.columns
```

[189]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

[191]:
```
## get the nsmalest and nlargest values in column
data.nsmallest(10,'petal_length')
```

[191]:

	sepal_length	sepal_width	petal_length	petal_width	species
22	4.6	3.6	1.0	0.2	setosa
13	4.3	3.0	1.1	0.1	setosa
14	5.8	4.0	1.2	0.2	setosa
35	5.0	3.2	1.2	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
16	5.4	3.9	1.3	0.4	setosa
36	5.5	3.5	1.3	0.2	setosa
38	4.4	3.0	1.3	0.2	setosa
40	5.0	3.5	1.3	0.3	setosa
41	4.5	2.3	1.3	0.3	setosa

[192]:
```
data.nlargest(10,'petal_length')
```

[192]:

	sepal_length	sepal_width	petal_length	petal_width	species
118	7.7	2.6	6.9	2.3	virginica
117	7.7	3.8	6.7	2.2	virginica
122	7.7	2.8	6.7	2.0	virginica
105	7.6	3.0	6.6	2.1	virginica
131	7.9	3.8	6.4	2.0	virginica
107	7.3	2.9	6.3	1.8	virginica
109	7.2	3.6	6.1	2.5	virginica
130	7.4	2.8	6.1	1.9	virginica
135	7.7	3.0	6.1	2.3	virginica
100	6.3	3.3	6.0	2.5	virginica

[194]:
```
## loc and iloc
data.loc[1:6,['sepal_length','sepal_width']]
 ↪## lloc 6 is inclussive
```

[194]:

	sepal_length	sepal_width
1	4.9	3.0
2	4.7	3.2
3	4.6	3.1
4	5.0	3.6

```
5           5.4          3.9
6           4.6          3.4
```

[197]:
```
data.iloc[1:6,[0,1]]                                    ## iloc 6 is exclussive
```

[197]:
```
   sepal_length  sepal_width
1           4.9          3.0
2           4.7          3.2
3           4.6          3.1
4           5.0          3.6
5           5.4          3.9
```

[ ]:
```
### groupby it means we can group the perticular columns
data[['age','fare']].groupby(data['pclass']).mean()
data[['age','fare']].groupby(data['pclass']).sum()
```

[198]:
```
## sort values
data.sort_values(by='sepal_length',ascending =False)
```

[198]:
```
     sepal_length  sepal_width  petal_length  petal_width    species
131           7.9          3.8           6.4          2.0  virginica
135           7.7          3.0           6.1          2.3  virginica
122           7.7          2.8           6.7          2.0  virginica
117           7.7          3.8           6.7          2.2  virginica
118           7.7          2.6           6.9          2.3  virginica
..            …            …             …            …          …
41            4.5          2.3           1.3          0.3     setosa
42            4.4          3.2           1.3          0.2     setosa
38            4.4          3.0           1.3          0.2     setosa
8             4.4          2.9           1.4          0.2     setosa
13            4.3          3.0           1.1          0.1     setosa

[150 rows x 5 columns]
```

[203]:
```
## query in dataframe###                                                ␣
 ↪[columns:rows]  [1:2]
data.query('sepal_length > 4.4')[:4]
```

[203]:
```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
```

[204]:
```
## unique in dataframe
data['sepal_length'].unique()
```

[204]:
```
array([5.1, 4.9, 4.7, 4.6, 5. , 5.4, 4.4, 4.8, 4.3, 5.8, 5.7, 5.2, 5.5,
       4.5, 5.3, 7. , 6.4, 6.9, 6.5, 6.3, 6.6, 5.9, 6. , 6.1, 5.6, 6.7,
       6.2, 6.8, 7.1, 7.6, 7.3, 7.2, 7.7, 7.4, 7.9])
```

[206]:
```
data.memory_usage()
```

[206]:
```
Index            128
sepal_length    1200
sepal_width     1200
petal_length    1200
petal_width     1200
species         1200
dtype: int64
```

[ ]:
```
## what is the difference between the  python array and list
array in python can only contain element of same data type
dtatype of array should be homogeneous.
list in python can contain element ofdifferent datatype
datatype of list  it is heterogeneous
```

[ ]:
```
## what is the lambda in python
lambda is anonymous function in python that can accept any number of␣
 ↪orguments,can only have a single expression
lambda function can be used either two way
```

[209]:
```
mul = lambda a, b: a * b

# Use the lambda function to multiply 2 and 5, then print the result
print(mul(2, 5))
```

```
10
```

[210]:
```
def mywrapper(n):
    return lambda a : a * n
mulfive=mywrapper(5)
print(mulfive(2))
```

```
10
```

[ ]:
```
## explain split(),join()function in python
split can used  split the string base don the delimeters.
join the list of string based o the delimeters
```

[213]:
```
string="this is a string"
string_list = string.split(' ')
print(string_list)
```

```
['this', 'is', 'a', 'string']
```

[214]:
```
print(' '.join(string_list))
```

```
this is a string
```

[ ]:
```
## what are the built in data types in python,
Numbers:-they includs the integers,floating point numbers, and complex numbers.
List   :-An order sequence of items is called list  the element of list may␣
 ↪belong to  different datatype
Tuple  :-it is also an order sequence of element,unlikelist,immutable,means it␣
 ↪canot be change
String ;-A sequence of character is called string they are declared within␣
 ↪single, double quote " "
Set    :it is set of unique item, that are not in order,
Dictionary :- Dictionary is a stored the value keys and values, each value can␣
 ↪eccess though its keys.
Boolean  :-there are two boolean value True and False
```

[216]:
```
class employees:

    def __init__(self,firstname,lastname,salary):
        self.firstname=firstname
        self.lastname=lastname
        self.salary=salary
        self.email=self.firstname + "." + self.lastname + "@gmail.com"
```

[219]:
```
## reverse list
my_list = [2, 4, 6, 7, 8]
reversed_list = my_list[::-1]

print(reversed_list)
```

```
[8, 7, 6, 4, 2]
```

[220]:
```
import re                                ### re means regex
```

[221]:
```
def add(a,b):                    ## here we used  2 line code
    return a+b
print(add(4,5))
```

```
9
```

[222]:
```
add=lambda  a,b:a+b        ## hre we used one line code
print(add(4,5))
```

```
9
```

[223]:
```
twice = lambda x:x*2
print(twice(6))
```

12

[ ]:
```
## what is the OS modulu
it is stands for operating system OS module provides various function for␣
 ↪interacting
import os in anaconda prompt
os.name
it gives 'nt'is os name
os.getcwd()   it means cwd  'current working dirrectory'
## i want to change my dirrectory
os.chdir('c://')
os.getcwd()
os.chdir('c://projects//')
## i want to make dirrectory
os.mkdir('new')
## i want to remove the dirrectory
os.rmdir('new')
## i want folder within folder
os.makedirs('new//new2')
## i want to know the directory perticulat folder
os.listdir()
#3 i want clear my current screen
clear=lambda:os.system('cls')
clear()
## i want to know all dirrectory
dir(os)
## i want to know weather this dirrectory exist or not
os.path.exists('c://')
```

[224]:
```
## wap to read afile with multiple line& split data bylines and also split into␣
 ↪words
fb = open("C:\\Users\\Vikas\\Desktop\\iris.csv")
data = fb.read()
print(data)
```

```
sepal_length,sepal_width,petal_length,petal_width,species
5.1,3.5,1.4,0.2,setosa
4.9,3.0,1.4,0.2,setosa
4.7,3.2,1.3,0.2,setosa
4.6,3.1,1.5,0.2,setosa
5.0,3.6,1.4,0.2,setosa
5.4,3.9,1.7,0.4,setosa
4.6,3.4,1.4,0.3,setosa
5.0,3.4,1.5,0.2,setosa
4.4,2.9,1.4,0.2,setosa
4.9,3.1,1.5,0.1,setosa
5.4,3.7,1.5,0.2,setosa
4.8,3.4,1.6,0.2,setosa
```

```
4.8,3.0,1.4,0.1,setosa
4.3,3.0,1.1,0.1,setosa
5.8,4.0,1.2,0.2,setosa
5.7,4.4,1.5,0.4,setosa
5.4,3.9,1.3,0.4,setosa
5.1,3.5,1.4,0.3,setosa
5.7,3.8,1.7,0.3,setosa
5.1,3.8,1.5,0.3,setosa
5.4,3.4,1.7,0.2,setosa
5.1,3.7,1.5,0.4,setosa
4.6,3.6,1.0,0.2,setosa
5.1,3.3,1.7,0.5,setosa
4.8,3.4,1.9,0.2,setosa
5.0,3.0,1.6,0.2,setosa
5.0,3.4,1.6,0.4,setosa
5.2,3.5,1.5,0.2,setosa
5.2,3.4,1.4,0.2,setosa
4.7,3.2,1.6,0.2,setosa
4.8,3.1,1.6,0.2,setosa
5.4,3.4,1.5,0.4,setosa
5.2,4.1,1.5,0.1,setosa
5.5,4.2,1.4,0.2,setosa
4.9,3.1,1.5,0.1,setosa
5.0,3.2,1.2,0.2,setosa
5.5,3.5,1.3,0.2,setosa
4.9,3.1,1.5,0.1,setosa
4.4,3.0,1.3,0.2,setosa
5.1,3.4,1.5,0.2,setosa
5.0,3.5,1.3,0.3,setosa
4.5,2.3,1.3,0.3,setosa
4.4,3.2,1.3,0.2,setosa
5.0,3.5,1.6,0.6,setosa
5.1,3.8,1.9,0.4,setosa
4.8,3.0,1.4,0.3,setosa
5.1,3.8,1.6,0.2,setosa
4.6,3.2,1.4,0.2,setosa
5.3,3.7,1.5,0.2,setosa
5.0,3.3,1.4,0.2,setosa
7.0,3.2,4.7,1.4,versicolor
6.4,3.2,4.5,1.5,versicolor
6.9,3.1,4.9,1.5,versicolor
5.5,2.3,4.0,1.3,versicolor
6.5,2.8,4.6,1.5,versicolor
5.7,2.8,4.5,1.3,versicolor
6.3,3.3,4.7,1.6,versicolor
4.9,2.4,3.3,1.0,versicolor
6.6,2.9,4.6,1.3,versicolor
5.2,2.7,3.9,1.4,versicolor
```

```
5.0,2.0,3.5,1.0,versicolor
5.9,3.0,4.2,1.5,versicolor
6.0,2.2,4.0,1.0,versicolor
6.1,2.9,4.7,1.4,versicolor
5.6,2.9,3.6,1.3,versicolor
6.7,3.1,4.4,1.4,versicolor
5.6,3.0,4.5,1.5,versicolor
5.8,2.7,4.1,1.0,versicolor
6.2,2.2,4.5,1.5,versicolor
5.6,2.5,3.9,1.1,versicolor
5.9,3.2,4.8,1.8,versicolor
6.1,2.8,4.0,1.3,versicolor
6.3,2.5,4.9,1.5,versicolor
6.1,2.8,4.7,1.2,versicolor
6.4,2.9,4.3,1.3,versicolor
6.6,3.0,4.4,1.4,versicolor
6.8,2.8,4.8,1.4,versicolor
6.7,3.0,5.0,1.7,versicolor
6.0,2.9,4.5,1.5,versicolor
5.7,2.6,3.5,1.0,versicolor
5.5,2.4,3.8,1.1,versicolor
5.5,2.4,3.7,1.0,versicolor
5.8,2.7,3.9,1.2,versicolor
6.0,2.7,5.1,1.6,versicolor
5.4,3.0,4.5,1.5,versicolor
6.0,3.4,4.5,1.6,versicolor
6.7,3.1,4.7,1.5,versicolor
6.3,2.3,4.4,1.3,versicolor
5.6,3.0,4.1,1.3,versicolor
5.5,2.5,4.0,1.3,versicolor
5.5,2.6,4.4,1.2,versicolor
6.1,3.0,4.6,1.4,versicolor
5.8,2.6,4.0,1.2,versicolor
5.0,2.3,3.3,1.0,versicolor
5.6,2.7,4.2,1.3,versicolor
5.7,3.0,4.2,1.2,versicolor
5.7,2.9,4.2,1.3,versicolor
6.2,2.9,4.3,1.3,versicolor
5.1,2.5,3.0,1.1,versicolor
5.7,2.8,4.1,1.3,versicolor
6.3,3.3,6.0,2.5,virginica
5.8,2.7,5.1,1.9,virginica
7.1,3.0,5.9,2.1,virginica
6.3,2.9,5.6,1.8,virginica
6.5,3.0,5.8,2.2,virginica
7.6,3.0,6.6,2.1,virginica
4.9,2.5,4.5,1.7,virginica
7.3,2.9,6.3,1.8,virginica
```

```
6.7,2.5,5.8,1.8,virginica
7.2,3.6,6.1,2.5,virginica
6.5,3.2,5.1,2.0,virginica
6.4,2.7,5.3,1.9,virginica
6.8,3.0,5.5,2.1,virginica
5.7,2.5,5.0,2.0,virginica
5.8,2.8,5.1,2.4,virginica
6.4,3.2,5.3,2.3,virginica
6.5,3.0,5.5,1.8,virginica
7.7,3.8,6.7,2.2,virginica
7.7,2.6,6.9,2.3,virginica
6.0,2.2,5.0,1.5,virginica
6.9,3.2,5.7,2.3,virginica
5.6,2.8,4.9,2.0,virginica
7.7,2.8,6.7,2.0,virginica
6.3,2.7,4.9,1.8,virginica
6.7,3.3,5.7,2.1,virginica
7.2,3.2,6.0,1.8,virginica
6.2,2.8,4.8,1.8,virginica
6.1,3.0,4.9,1.8,virginica
6.4,2.8,5.6,2.1,virginica
7.2,3.0,5.8,1.6,virginica
7.4,2.8,6.1,1.9,virginica
7.9,3.8,6.4,2.0,virginica
6.4,2.8,5.6,2.2,virginica
6.3,2.8,5.1,1.5,virginica
6.1,2.6,5.6,1.4,virginica
7.7,3.0,6.1,2.3,virginica
6.3,3.4,5.6,2.4,virginica
6.4,3.1,5.5,1.8,virginica
6.0,3.0,4.8,1.8,virginica
6.9,3.1,5.4,2.1,virginica
6.7,3.1,5.6,2.4,virginica
6.9,3.1,5.1,2.3,virginica
5.8,2.7,5.1,1.9,virginica
6.8,3.2,5.9,2.3,virginica
6.7,3.3,5.7,2.5,virginica
6.7,3.0,5.2,2.3,virginica
6.3,2.5,5.0,1.9,virginica
6.5,3.0,5.2,2.0,virginica
6.2,3.4,5.4,2.3,virginica
5.9,3.0,5.1,1.8,virginica
```

[225]:
```
lines = data.split("\n")
print(lines)
```

```
['sepal_length,sepal_width,petal_length,petal_width,species',
```

```
'5.1,3.5,1.4,0.2,setosa', '4.9,3.0,1.4,0.2,setosa', '4.7,3.2,1.3,0.2,setosa',
'4.6,3.1,1.5,0.2,setosa', '5.0,3.6,1.4,0.2,setosa', '5.4,3.9,1.7,0.4,setosa',
'4.6,3.4,1.4,0.3,setosa', '5.0,3.4,1.5,0.2,setosa', '4.4,2.9,1.4,0.2,setosa',
'4.9,3.1,1.5,0.1,setosa', '5.4,3.7,1.5,0.2,setosa', '4.8,3.4,1.6,0.2,setosa',
'4.8,3.0,1.4,0.1,setosa', '4.3,3.0,1.1,0.1,setosa', '5.8,4.0,1.2,0.2,setosa',
'5.7,4.4,1.5,0.4,setosa', '5.4,3.9,1.3,0.4,setosa', '5.1,3.5,1.4,0.3,setosa',
'5.7,3.8,1.7,0.3,setosa', '5.1,3.8,1.5,0.3,setosa', '5.4,3.4,1.7,0.2,setosa',
'5.1,3.7,1.5,0.4,setosa', '4.6,3.6,1.0,0.2,setosa', '5.1,3.3,1.7,0.5,setosa',
'4.8,3.4,1.9,0.2,setosa', '5.0,3.0,1.6,0.2,setosa', '5.0,3.4,1.6,0.4,setosa',
'5.2,3.5,1.5,0.2,setosa', '5.2,3.4,1.4,0.2,setosa', '4.7,3.2,1.6,0.2,setosa',
'4.8,3.1,1.6,0.2,setosa', '5.4,3.4,1.5,0.4,setosa', '5.2,4.1,1.5,0.1,setosa',
'5.5,4.2,1.4,0.2,setosa', '4.9,3.1,1.5,0.1,setosa', '5.0,3.2,1.2,0.2,setosa',
'5.5,3.5,1.3,0.2,setosa', '4.9,3.1,1.5,0.1,setosa', '4.4,3.0,1.3,0.2,setosa',
'5.1,3.4,1.5,0.2,setosa', '5.0,3.5,1.3,0.3,setosa', '4.5,2.3,1.3,0.3,setosa',
'4.4,3.2,1.3,0.2,setosa', '5.0,3.5,1.6,0.6,setosa', '5.1,3.8,1.9,0.4,setosa',
'4.8,3.0,1.4,0.3,setosa', '5.1,3.8,1.6,0.2,setosa', '4.6,3.2,1.4,0.2,setosa',
'5.3,3.7,1.5,0.2,setosa', '5.0,3.3,1.4,0.2,setosa',
'7.0,3.2,4.7,1.4,versicolor', '6.4,3.2,4.5,1.5,versicolor',
'6.9,3.1,4.9,1.5,versicolor', '5.5,2.3,4.0,1.3,versicolor',
'6.5,2.8,4.6,1.5,versicolor', '5.7,2.8,4.5,1.3,versicolor',
'6.3,3.3,4.7,1.6,versicolor', '4.9,2.4,3.3,1.0,versicolor',
'6.6,2.9,4.6,1.3,versicolor', '5.2,2.7,3.9,1.4,versicolor',
'5.0,2.0,3.5,1.0,versicolor', '5.9,3.0,4.2,1.5,versicolor',
'6.0,2.2,4.0,1.0,versicolor', '6.1,2.9,4.7,1.4,versicolor',
'5.6,2.9,3.6,1.3,versicolor', '6.7,3.1,4.4,1.4,versicolor',
'5.6,3.0,4.5,1.5,versicolor', '5.8,2.7,4.1,1.0,versicolor',
'6.2,2.2,4.5,1.5,versicolor', '5.6,2.5,3.9,1.1,versicolor',
'5.9,3.2,4.8,1.8,versicolor', '6.1,2.8,4.0,1.3,versicolor',
'6.3,2.5,4.9,1.5,versicolor', '6.1,2.8,4.7,1.2,versicolor',
'6.4,2.9,4.3,1.3,versicolor', '6.6,3.0,4.4,1.4,versicolor',
'6.8,2.8,4.8,1.4,versicolor', '6.7,3.0,5.0,1.7,versicolor',
'6.0,2.9,4.5,1.5,versicolor', '5.7,2.6,3.5,1.0,versicolor',
'5.5,2.4,3.8,1.1,versicolor', '5.5,2.4,3.7,1.0,versicolor',
'5.8,2.7,3.9,1.2,versicolor', '6.0,2.7,5.1,1.6,versicolor',
'5.4,3.0,4.5,1.5,versicolor', '6.0,3.4,4.5,1.6,versicolor',
'6.7,3.1,4.7,1.5,versicolor', '6.3,2.3,4.4,1.3,versicolor',
'5.6,3.0,4.1,1.3,versicolor', '5.5,2.5,4.0,1.3,versicolor',
'5.5,2.6,4.4,1.2,versicolor', '6.1,3.0,4.6,1.4,versicolor',
'5.8,2.6,4.0,1.2,versicolor', '5.0,2.3,3.3,1.0,versicolor',
'5.6,2.7,4.2,1.3,versicolor', '5.7,3.0,4.2,1.2,versicolor',
'5.7,2.9,4.2,1.3,versicolor', '6.2,2.9,4.3,1.3,versicolor',
'5.1,2.5,3.0,1.1,versicolor', '5.7,2.8,4.1,1.3,versicolor',
'6.3,3.3,6.0,2.5,virginica', '5.8,2.7,5.1,1.9,virginica',
'7.1,3.0,5.9,2.1,virginica', '6.3,2.9,5.6,1.8,virginica',
'6.5,3.0,5.8,2.2,virginica', '7.6,3.0,6.6,2.1,virginica',
'4.9,2.5,4.5,1.7,virginica', '7.3,2.9,6.3,1.8,virginica',
'6.7,2.5,5.8,1.8,virginica', '7.2,3.6,6.1,2.5,virginica',
'6.5,3.2,5.1,2.0,virginica', '6.4,2.7,5.3,1.9,virginica',
```

```
'6.8,3.0,5.5,2.1,virginica', '5.7,2.5,5.0,2.0,virginica',
'5.8,2.8,5.1,2.4,virginica', '6.4,3.2,5.3,2.3,virginica',
'6.5,3.0,5.5,1.8,virginica', '7.7,3.8,6.7,2.2,virginica',
'7.7,2.6,6.9,2.3,virginica', '6.0,2.2,5.0,1.5,virginica',
'6.9,3.2,5.7,2.3,virginica', '5.6,2.8,4.9,2.0,virginica',
'7.7,2.8,6.7,2.0,virginica', '6.3,2.7,4.9,1.8,virginica',
'6.7,3.3,5.7,2.1,virginica', '7.2,3.2,6.0,1.8,virginica',
'6.2,2.8,4.8,1.8,virginica', '6.1,3.0,4.9,1.8,virginica',
'6.4,2.8,5.6,2.1,virginica', '7.2,3.0,5.8,1.6,virginica',
'7.4,2.8,6.1,1.9,virginica', '7.9,3.8,6.4,2.0,virginica',
'6.4,2.8,5.6,2.2,virginica', '6.3,2.8,5.1,1.5,virginica',
'6.1,2.6,5.6,1.4,virginica', '7.7,3.0,6.1,2.3,virginica',
'6.3,3.4,5.6,2.4,virginica', '6.4,3.1,5.5,1.8,virginica',
'6.0,3.0,4.8,1.8,virginica', '6.9,3.1,5.4,2.1,virginica',
'6.7,3.1,5.6,2.4,virginica', '6.9,3.1,5.1,2.3,virginica',
'5.8,2.7,5.1,1.9,virginica', '6.8,3.2,5.9,2.3,virginica',
'6.7,3.3,5.7,2.5,virginica', '6.7,3.0,5.2,2.3,virginica',
'6.3,2.5,5.0,1.9,virginica', '6.5,3.0,5.2,2.0,virginica',
'6.2,3.4,5.4,2.3,virginica', '5.9,3.0,5.1,1.8,virginica', '']
```

[226]:
```
lines = data.split("\n")
for line in lines:

    print(line)
```

```
sepal_length,sepal_width,petal_length,petal_width,species
5.1,3.5,1.4,0.2,setosa
4.9,3.0,1.4,0.2,setosa
4.7,3.2,1.3,0.2,setosa
4.6,3.1,1.5,0.2,setosa
5.0,3.6,1.4,0.2,setosa
5.4,3.9,1.7,0.4,setosa
4.6,3.4,1.4,0.3,setosa
5.0,3.4,1.5,0.2,setosa
4.4,2.9,1.4,0.2,setosa
4.9,3.1,1.5,0.1,setosa
5.4,3.7,1.5,0.2,setosa
4.8,3.4,1.6,0.2,setosa
4.8,3.0,1.4,0.1,setosa
4.3,3.0,1.1,0.1,setosa
5.8,4.0,1.2,0.2,setosa
5.7,4.4,1.5,0.4,setosa
5.4,3.9,1.3,0.4,setosa
5.1,3.5,1.4,0.3,setosa
5.7,3.8,1.7,0.3,setosa
5.1,3.8,1.5,0.3,setosa
5.4,3.4,1.7,0.2,setosa
5.1,3.7,1.5,0.4,setosa
```

```
4.6,3.6,1.0,0.2,setosa
5.1,3.3,1.7,0.5,setosa
4.8,3.4,1.9,0.2,setosa
5.0,3.0,1.6,0.2,setosa
5.0,3.4,1.6,0.4,setosa
5.2,3.5,1.5,0.2,setosa
5.2,3.4,1.4,0.2,setosa
4.7,3.2,1.6,0.2,setosa
4.8,3.1,1.6,0.2,setosa
5.4,3.4,1.5,0.4,setosa
5.2,4.1,1.5,0.1,setosa
5.5,4.2,1.4,0.2,setosa
4.9,3.1,1.5,0.1,setosa
5.0,3.2,1.2,0.2,setosa
5.5,3.5,1.3,0.2,setosa
4.9,3.1,1.5,0.1,setosa
4.4,3.0,1.3,0.2,setosa
5.1,3.4,1.5,0.2,setosa
5.0,3.5,1.3,0.3,setosa
4.5,2.3,1.3,0.3,setosa
4.4,3.2,1.3,0.2,setosa
5.0,3.5,1.6,0.6,setosa
5.1,3.8,1.9,0.4,setosa
4.8,3.0,1.4,0.3,setosa
5.1,3.8,1.6,0.2,setosa
4.6,3.2,1.4,0.2,setosa
5.3,3.7,1.5,0.2,setosa
5.0,3.3,1.4,0.2,setosa
7.0,3.2,4.7,1.4,versicolor
6.4,3.2,4.5,1.5,versicolor
6.9,3.1,4.9,1.5,versicolor
5.5,2.3,4.0,1.3,versicolor
6.5,2.8,4.6,1.5,versicolor
5.7,2.8,4.5,1.3,versicolor
6.3,3.3,4.7,1.6,versicolor
4.9,2.4,3.3,1.0,versicolor
6.6,2.9,4.6,1.3,versicolor
5.2,2.7,3.9,1.4,versicolor
5.0,2.0,3.5,1.0,versicolor
5.9,3.0,4.2,1.5,versicolor
6.0,2.2,4.0,1.0,versicolor
6.1,2.9,4.7,1.4,versicolor
5.6,2.9,3.6,1.3,versicolor
6.7,3.1,4.4,1.4,versicolor
5.6,3.0,4.5,1.5,versicolor
5.8,2.7,4.1,1.0,versicolor
6.2,2.2,4.5,1.5,versicolor
5.6,2.5,3.9,1.1,versicolor
```

```
5.9,3.2,4.8,1.8,versicolor
6.1,2.8,4.0,1.3,versicolor
6.3,2.5,4.9,1.5,versicolor
6.1,2.8,4.7,1.2,versicolor
6.4,2.9,4.3,1.3,versicolor
6.6,3.0,4.4,1.4,versicolor
6.8,2.8,4.8,1.4,versicolor
6.7,3.0,5.0,1.7,versicolor
6.0,2.9,4.5,1.5,versicolor
5.7,2.6,3.5,1.0,versicolor
5.5,2.4,3.8,1.1,versicolor
5.5,2.4,3.7,1.0,versicolor
5.8,2.7,3.9,1.2,versicolor
6.0,2.7,5.1,1.6,versicolor
5.4,3.0,4.5,1.5,versicolor
6.0,3.4,4.5,1.6,versicolor
6.7,3.1,4.7,1.5,versicolor
6.3,2.3,4.4,1.3,versicolor
5.6,3.0,4.1,1.3,versicolor
5.5,2.5,4.0,1.3,versicolor
5.5,2.6,4.4,1.2,versicolor
6.1,3.0,4.6,1.4,versicolor
5.8,2.6,4.0,1.2,versicolor
5.0,2.3,3.3,1.0,versicolor
5.6,2.7,4.2,1.3,versicolor
5.7,3.0,4.2,1.2,versicolor
5.7,2.9,4.2,1.3,versicolor
6.2,2.9,4.3,1.3,versicolor
5.1,2.5,3.0,1.1,versicolor
5.7,2.8,4.1,1.3,versicolor
6.3,3.3,6.0,2.5,virginica
5.8,2.7,5.1,1.9,virginica
7.1,3.0,5.9,2.1,virginica
6.3,2.9,5.6,1.8,virginica
6.5,3.0,5.8,2.2,virginica
7.6,3.0,6.6,2.1,virginica
4.9,2.5,4.5,1.7,virginica
7.3,2.9,6.3,1.8,virginica
6.7,2.5,5.8,1.8,virginica
7.2,3.6,6.1,2.5,virginica
6.5,3.2,5.1,2.0,virginica
6.4,2.7,5.3,1.9,virginica
6.8,3.0,5.5,2.1,virginica
5.7,2.5,5.0,2.0,virginica
5.8,2.8,5.1,2.4,virginica
6.4,3.2,5.3,2.3,virginica
6.5,3.0,5.5,1.8,virginica
7.7,3.8,6.7,2.2,virginica
```

```
7.7,2.6,6.9,2.3,virginica
6.0,2.2,5.0,1.5,virginica
6.9,3.2,5.7,2.3,virginica
5.6,2.8,4.9,2.0,virginica
7.7,2.8,6.7,2.0,virginica
6.3,2.7,4.9,1.8,virginica
6.7,3.3,5.7,2.1,virginica
7.2,3.2,6.0,1.8,virginica
6.2,2.8,4.8,1.8,virginica
6.1,3.0,4.9,1.8,virginica
6.4,2.8,5.6,2.1,virginica
7.2,3.0,5.8,1.6,virginica
7.4,2.8,6.1,1.9,virginica
7.9,3.8,6.4,2.0,virginica
6.4,2.8,5.6,2.2,virginica
6.3,2.8,5.1,1.5,virginica
6.1,2.6,5.6,1.4,virginica
7.7,3.0,6.1,2.3,virginica
6.3,3.4,5.6,2.4,virginica
6.4,3.1,5.5,1.8,virginica
6.0,3.0,4.8,1.8,virginica
6.9,3.1,5.4,2.1,virginica
6.7,3.1,5.6,2.4,virginica
6.9,3.1,5.1,2.3,virginica
5.8,2.7,5.1,1.9,virginica
6.8,3.2,5.9,2.3,virginica
6.7,3.3,5.7,2.5,virginica
6.7,3.0,5.2,2.3,virginica
6.3,2.5,5.0,1.9,virginica
6.5,3.0,5.2,2.0,virginica
6.2,3.4,5.4,2.3,virginica
5.9,3.0,5.1,1.8,virginica
```

[236]:
```
lines = data.split("\n")
for line in lines:
    words = line.split(" ")
    for word in words:
        print(word)
```

```
sepal_length,sepal_width,petal_length,petal_width,species
5.1,3.5,1.4,0.2,setosa
4.9,3.0,1.4,0.2,setosa
4.7,3.2,1.3,0.2,setosa
4.6,3.1,1.5,0.2,setosa
5.0,3.6,1.4,0.2,setosa
5.4,3.9,1.7,0.4,setosa
4.6,3.4,1.4,0.3,setosa
```

```
5.0,3.4,1.5,0.2,setosa
4.4,2.9,1.4,0.2,setosa
4.9,3.1,1.5,0.1,setosa
5.4,3.7,1.5,0.2,setosa
4.8,3.4,1.6,0.2,setosa
4.8,3.0,1.4,0.1,setosa
4.3,3.0,1.1,0.1,setosa
5.8,4.0,1.2,0.2,setosa
5.7,4.4,1.5,0.4,setosa
5.4,3.9,1.3,0.4,setosa
5.1,3.5,1.4,0.3,setosa
5.7,3.8,1.7,0.3,setosa
5.1,3.8,1.5,0.3,setosa
5.4,3.4,1.7,0.2,setosa
5.1,3.7,1.5,0.4,setosa
4.6,3.6,1.0,0.2,setosa
5.1,3.3,1.7,0.5,setosa
4.8,3.4,1.9,0.2,setosa
5.0,3.0,1.6,0.2,setosa
5.0,3.4,1.6,0.4,setosa
5.2,3.5,1.5,0.2,setosa
5.2,3.4,1.4,0.2,setosa
4.7,3.2,1.6,0.2,setosa
4.8,3.1,1.6,0.2,setosa
5.4,3.4,1.5,0.4,setosa
5.2,4.1,1.5,0.1,setosa
5.5,4.2,1.4,0.2,setosa
4.9,3.1,1.5,0.1,setosa
5.0,3.2,1.2,0.2,setosa
5.5,3.5,1.3,0.2,setosa
4.9,3.1,1.5,0.1,setosa
4.4,3.0,1.3,0.2,setosa
5.1,3.4,1.5,0.2,setosa
5.0,3.5,1.3,0.3,setosa
4.5,2.3,1.3,0.3,setosa
4.4,3.2,1.3,0.2,setosa
5.0,3.5,1.6,0.6,setosa
5.1,3.8,1.9,0.4,setosa
4.8,3.0,1.4,0.3,setosa
5.1,3.8,1.6,0.2,setosa
4.6,3.2,1.4,0.2,setosa
5.3,3.7,1.5,0.2,setosa
5.0,3.3,1.4,0.2,setosa
7.0,3.2,4.7,1.4,versicolor
6.4,3.2,4.5,1.5,versicolor
6.9,3.1,4.9,1.5,versicolor
5.5,2.3,4.0,1.3,versicolor
6.5,2.8,4.6,1.5,versicolor
```

```
5.7,2.8,4.5,1.3,versicolor
6.3,3.3,4.7,1.6,versicolor
4.9,2.4,3.3,1.0,versicolor
6.6,2.9,4.6,1.3,versicolor
5.2,2.7,3.9,1.4,versicolor
5.0,2.0,3.5,1.0,versicolor
5.9,3.0,4.2,1.5,versicolor
6.0,2.2,4.0,1.0,versicolor
6.1,2.9,4.7,1.4,versicolor
5.6,2.9,3.6,1.3,versicolor
6.7,3.1,4.4,1.4,versicolor
5.6,3.0,4.5,1.5,versicolor
5.8,2.7,4.1,1.0,versicolor
6.2,2.2,4.5,1.5,versicolor
5.6,2.5,3.9,1.1,versicolor
5.9,3.2,4.8,1.8,versicolor
6.1,2.8,4.0,1.3,versicolor
6.3,2.5,4.9,1.5,versicolor
6.1,2.8,4.7,1.2,versicolor
6.4,2.9,4.3,1.3,versicolor
6.6,3.0,4.4,1.4,versicolor
6.8,2.8,4.8,1.4,versicolor
6.7,3.0,5.0,1.7,versicolor
6.0,2.9,4.5,1.5,versicolor
5.7,2.6,3.5,1.0,versicolor
5.5,2.4,3.8,1.1,versicolor
5.5,2.4,3.7,1.0,versicolor
5.8,2.7,3.9,1.2,versicolor
6.0,2.7,5.1,1.6,versicolor
5.4,3.0,4.5,1.5,versicolor
6.0,3.4,4.5,1.6,versicolor
6.7,3.1,4.7,1.5,versicolor
6.3,2.3,4.4,1.3,versicolor
5.6,3.0,4.1,1.3,versicolor
5.5,2.5,4.0,1.3,versicolor
5.5,2.6,4.4,1.2,versicolor
6.1,3.0,4.6,1.4,versicolor
5.8,2.6,4.0,1.2,versicolor
5.0,2.3,3.3,1.0,versicolor
5.6,2.7,4.2,1.3,versicolor
5.7,3.0,4.2,1.2,versicolor
5.7,2.9,4.2,1.3,versicolor
6.2,2.9,4.3,1.3,versicolor
5.1,2.5,3.0,1.1,versicolor
5.7,2.8,4.1,1.3,versicolor
6.3,3.3,6.0,2.5,virginica
5.8,2.7,5.1,1.9,virginica
7.1,3.0,5.9,2.1,virginica
```

```
6.3,2.9,5.6,1.8,virginica
6.5,3.0,5.8,2.2,virginica
7.6,3.0,6.6,2.1,virginica
4.9,2.5,4.5,1.7,virginica
7.3,2.9,6.3,1.8,virginica
6.7,2.5,5.8,1.8,virginica
7.2,3.6,6.1,2.5,virginica
6.5,3.2,5.1,2.0,virginica
6.4,2.7,5.3,1.9,virginica
6.8,3.0,5.5,2.1,virginica
5.7,2.5,5.0,2.0,virginica
5.8,2.8,5.1,2.4,virginica
6.4,3.2,5.3,2.3,virginica
6.5,3.0,5.5,1.8,virginica
7.7,3.8,6.7,2.2,virginica
7.7,2.6,6.9,2.3,virginica
6.0,2.2,5.0,1.5,virginica
6.9,3.2,5.7,2.3,virginica
5.6,2.8,4.9,2.0,virginica
7.7,2.8,6.7,2.0,virginica
6.3,2.7,4.9,1.8,virginica
6.7,3.3,5.7,2.1,virginica
7.2,3.2,6.0,1.8,virginica
6.2,2.8,4.8,1.8,virginica
6.1,3.0,4.9,1.8,virginica
6.4,2.8,5.6,2.1,virginica
7.2,3.0,5.8,1.6,virginica
7.4,2.8,6.1,1.9,virginica
7.9,3.8,6.4,2.0,virginica
6.4,2.8,5.6,2.2,virginica
6.3,2.8,5.1,1.5,virginica
6.1,2.6,5.6,1.4,virginica
7.7,3.0,6.1,2.3,virginica
6.3,3.4,5.6,2.4,virginica
6.4,3.1,5.5,1.8,virginica
6.0,3.0,4.8,1.8,virginica
6.9,3.1,5.4,2.1,virginica
6.7,3.1,5.6,2.4,virginica
6.9,3.1,5.1,2.3,virginica
5.8,2.7,5.1,1.9,virginica
6.8,3.2,5.9,2.3,virginica
6.7,3.3,5.7,2.5,virginica
6.7,3.0,5.2,2.3,virginica
6.3,2.5,5.0,1.9,virginica
6.5,3.0,5.2,2.0,virginica
6.2,3.4,5.4,2.3,virginica
5.9,3.0,5.1,1.8,virginica
```

[237]:
```
## how to merge one dictionary to another dictionary
dict1 = {1:'apple',2:'banana'}
dict2 = {3:'orange',4:'pomogranate'}
dict1.update(dict2)
print(dict1)
```

```
{1: 'apple', 2: 'banana', 3: 'orange', 4: 'pomogranate'}
```

[238]:
```
dict3 = {**dict1,**dict2}
print(dict3)
```

```
{1: 'apple', 2: 'banana', 3: 'orange', 4: 'pomogranate'}
```

[240]:
```
## Remove duplicates from list
## remove duplicate by useing 5 methodes

arr=[1,2,2,3,3,4,4,5,6,7,8,9,9,]
```

[241]:
```
## remove the duplicates using set function
arr2=set(arr)
print(arr2)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

[244]:
```
## remove the duplicate useing lambda
re_duplicate_fun=lambda arr:set(arr)
print(re_duplicate_fun(arr))
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

[245]:
```
4+3%5
```

[245]: 7

[ ]:
```
## Why is the python called interpreted language
execute instruction dirrectly by line by line
does not compile and run a complete programe at once
this make the programm esier to debug
we can just wright a python code and run it dirrectly
this makes python fast to develop
```

[ ]:
```
## do you need a indedation in python
yes indendation is a must in python and is part of the languages syntax
indedation provides the readability to the code
```

[ ]:
```
## what are the classes and object in python
class is the blueprint creating a object
object is instance of class
```

[ ]:
```
## diffirence between the list and tuple
list are mutable they can be modified and alter and creation(adding ,changing␣
 ↪item)
list  are defined useing squire bracket
list take up more memory due to there ability to change and grow
tuple is immutable once it created content can not be changed
tuple are defined using parenthesis()
tuple are more memory  efficient usually faster because they are immutable.
```

[ ]:
```
# what is the shallow copy and deep copy in python
a shallow copy is a copy of object that stored the reference of the original␣
 ↪element
so if we make any changes to the copy of the objects,it wiil reflected in␣
 ↪original object
```

[ ]:
```
### fibonecci generator
def fibo_gen():
    a = 0
    b = 1
    while True:
        c = a
        a = b
        b = c + a
        yield c

# Create a generator instance
f = fibo_gen()

# Generate and print the first two numbers of the Fibonacci sequence
print(next(f))  # Output: 0
print(next(f))  # Output: 1
```

[ ]:
```
## how to reverse list in python
list1 = [4, 5, 6, 7]
list1.reverse()
print(list1)
```

[ ]:
```
## slicing methode      ##[start vlue:end value:sep]
list2=list1[::-1]
print(list2)
```

[ ]:
```
list2=[]
list1=[1,2,3,4]
for i in range(len(list1)-1,-1-1):
    list2.append(list1[i])
print(list2)
```

[ ]:
```
## find the current date time

(base) C:\Users\Vikas>python
Python 3.10.9 | packaged by Anaconda, Inc. | (main, Mar  1 2023, 18:18:15) [MSC
 ↪v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import datetime
>>> datetime.datetime.now()
datetime.datetime(2024, 8, 12, 19, 53, 27, 237943)
>>> today=datetime.datetime.now()
>>> today.strftime("%y-%m-%d %H:%M:%S")
'24-08-12 19:54:52'
>>>
```

[ ]:
```
## check weather the email id is  correct or wrong.
import re
```

[ ]:
```
import re

def isvalidemail(email):
    regex = r"^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$"
    if len(email) > 7:
        if re.match(regex, email, re.IGNORECASE) is not None:
            return True
    return False

# Test the function
if isvalidemail("netsetos@gmail.com"):
    print("Valid email address")
else:
    print("Invalid email address")
```

[ ]:
```
def add(a,b):
    return a+b
print(add(4,5))
```

[ ]:
```
Add =lambda a,b:a+b
print(Add(4,5))
```

[ ]:
```

```