1. **Write a program to perform arithmetic operations in python.**

Program:

```
a=int(input("Enter a value:"))
b=int(input("Enter b value:"))
print("add:",a+b)
print("Sub:",a-b)
print("Mul:",a*b)
print("Div:",a/b)
print("Mod:",a%b)
a=float(input("Enter a value:"))
b=float(input("Enter b value:"))
print("add:",a+b)
print("Sub:",a-b)
print("Mul:",a*b)
print("Div:",a/b)
print("Mod:",a%b)
```

Output:

```
Enter a value:10
Enter b value:22
add: 32
Sub: -12
Mul: 220
Div: 0.45454545454545453
Mod: 10
Enter a value:1.2
Enter b value:3.5
add: 4.7
Sub: -2.3
Mul: 4.2
Div: 0.34285714285714286
Mod: 1.2
```

2.Write a program to check whether the string is symmetrical/palindrome.

Program:

```
st=input("Enter a string:")
def palindrome(st):
    m=(len(st)-1)//2
    l=0
    h=len(st)-1
    flag=0
    while(l<=m):
        if st[l]==st[h]:
            l=l+1
            h=h-1
        else:
            flag=1
            break;
    if flag==0:
        print(st,"is palindrome")
    else:
        print(st,"is not palindrome")
def symmetrical(st):
    n=len(st)
    flag=0
    if n%2==0:
        mid=n//2
    else:
        mid=n//2+1
    l=0
    m=mid
    while(l<mid and m<n):
        if(st[l]==st[m]):
            l=l+1
            m=m+1
        else:
            flag=1
            break
    if flag==0:
        print(st,"is symmetrical")
    else:
        print(st,"is not symmetrical")
palindrome(st)
symmetrical(st)
```

Output:

```
Enter a string:khokho
khokho is not palindrome
khokho is symmetrical
```

3.Write a program to print the multiplication table of a given number.

Program:

```
n=int(input("enter a number:"))
for i in range(1,11):
    print(n,"*",i,"=",n*i)
```

Output:

```
enter a number:5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

4. Write a program to print largest of three numbers using conditional statements.

Program:

```
a=int(input("Enter a value:"))
b=int(input("Enter b value:"))
c=int(input("Enter c value:"))
if a>=b and a>=c:
    print(a,"is largest")
elif b>=a and b>=c:
    print(b,"is largest")
else:
    print(c,"is largest")
```

Output:

```
Enter a value:12
Enter b value:13
Enter c value:45
45 is largest
```

5. Write a python program to count the occurrences of each word in a text file.

Program:

```
fd=open("demo.txt",'r')
d=fd.read()
count=dict()
words=d.split()
for i in words:
    if i in count:
        count[i]+=1
    else:
        count[i]=1
print(count)
```

Output:

```
{'python': 1, 'is': 2, 'a': 1, 'simple': 2, 'programming': 1, 'language.': 1, 'it': 1, 'very': 1, 'and': 1, 'easier': 1, 'to': 1, 'understand.': 1}
```

6. Write a program to perform the different basic operations on a tuple.

Program:

```
t1=('akhila','aakanksha','sai')
t2=(10,22,3,1)
print(len(t1))
print(max(t2))
print(min(t2))
t=t1+t2
print(t)
print(t1[-2])
print(t2[::-1])
print(list(t))
```

Output:

```
3
22
1
('akhila', 'aakanksha', 'sai', 10, 22, 3, 1)
aakanksha
(1, 3, 22, 10)
['akhila', 'aakanksha', 'sai', 10, 22, 3, 1]
```

7. Write a program to get the largest number from a list.

Program:

```
l=[22,56,23,45,87,10,108,46,67,82]
mx=l[1]
for i in range(0,len(l)-1):
    if l[i]>l[i+1]:
        mx=l[i]
print("Largest number from list is",mx)
```

Output:

```
Largest number from list is 108
```

8. Write a program to get the maximum and minimum value in a dictionary.

Program:

```
d={'akhila':92,'anusha':96,'pavani':85}
mx=max(d.keys(),key=(lambda k:d[k]))
mn=min(d.keys(),key=(lambda k:d[k]))
print("Maximum value:",d[mx])
print("Minimum value:",d[mn])
```

Output:

```
Maximum value: 96
Minimum value: 85
```

9. Explore python classes, packages related to Machine Learning (numpy, pandas, matplotlib)

Numpy

```
import numpy as np
a=np.array([5,10,22,15,13,60])
```

```
a
```

```
array([ 5, 10, 22, 15, 13, 60])
```

```
print(a[0:3])
```

```
[ 5 10 22]
```

```
print(a[:5])
```

```
[ 5 10 22 15 13]
```

```
print(a[3:])
```

```
[15 13 60]
```

```
print(a[::-1])
```

```
[60 13 15 22 10  5]
```

```
print(np.sort(a))
```

```
[ 5 10 13 15 22 60]
```

```
type(a)
```

```
numpy.ndarray
```

```
b=np.array([[1,2,3],[4,5,6],[7,8,9]])
```

```
b
```

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

```
type(b)
```

```
numpy.ndarray
```

```
print(a.dtype)
```

```
int64
```

```
a.shape
```

```
(6,)
```

```
b.shape
```

```
(3, 3)
```

```
a.ndim
```

```
1
```

```
b=np.array([[1,2,3],[4,5,6],[7,8,9]])
```

```
b
```

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

```
type(b)
```

```
numpy.ndarray
```

```
print(a.dtype)
```

```
int64
```

```
a.shape
```

```
(6,)
```

```
b.shape
```

```
(3, 3)
```

```
a.ndim
```

```
1
```

```
b.ndim
```

```
2
```

```
c=np.array([[(1,2,3),(3,3,3)],
            [(7,8,9),(4,4,4)],
            [(1,1,1),(3,3,3)]
           ])
```

```
c.shape
```

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

```
np.zeros((4,3))
```

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

```
np.zeros((4,3),dtype=np.int32)
```

```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]], dtype=int32)
```

```
d=np.arange(1,20,2)
```

```
c=np.array([[(1,2,3),(3,3,3)],
            [(7,8,9),(4,4,4)],
            [(1,1,1),(3,3,3)]
           ])
```

```
[ ] c.shape
```

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

```
[ ] np.zeros((4,3))
```

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

```
[ ] np.zeros((4,3),dtype=np.int32)
```

```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]], dtype=int32)
```

```
[ ] d=np.arange(1,20,2)
```

```
[ ] d
```

```
array([ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19])
```

```
[ ] e=np.full((4,3),10)
```

```
[ ] e
```

```
array([[10, 10, 10],
       [10, 10, 10],
       [10, 10, 10],
       [10, 10, 10]])
```

```
[ ] np.linspace(1,5,10)
```

```
array([1.        , 1.44444444, 1.88888889, 2.33333333, 2.77777778,
```

Pandas

```
[ ] import pandas as pd
```

Creating data frame

```
[ ] df=pd.DataFrame([[4,5,6],[3,4,5],[12,13,14]],index=[1,2,3],columns=["a","b","c"])
```

```
[ ] df
```

	a	b	c
1	4	5	6
2	3	4	5
3	12	13	14

```
[ ] df = pd.DataFrame(
    {"a" : [4 ,5, 6],
    "b" : [7, 8, 9],
    "c" : [10, 11, 12]},
    index = pd.MultiIndex.from_tuples(
    [('d',1),('d',2),('e',2)],
    names=['n','v']))
```

```
[ ] df
```

		a	b	c
n	v			
d	1	4	7	10
	2	5	8	11
e	2	6	9	12

```
[ ] dfn = pd.DataFrame(
        {
          "Name": [
                  "Braund, Mr. Owen Harris",
                  "Allen, Mr. William Henry",
                "Bonnell, Miss. Elizabeth",
              ],
            "Age": [22, 35, 58],
            "Sex": ["male", "male", "female"],
        }
    )
```

```
[ ] dfn
```

	Name	Age	Sex
0	Braund, Mr. Owen Harris	22	male
1	Allen, Mr. William Henry	35	male
2	Bonnell, Miss. Elizabeth	58	female

printing one column of dataframe

```
[ ] dfn["Age"]
```

```
0    22
1    35
2    58
Name: Age, dtype: int64
```

Series is one dimensional data

```
[ ] ages = pd.Series([22, 35, 58], name="Age")
```

```
[ ] ages
```

```
0    22
1    35
2    58
Name: Age, dtype: int64
```

```
[ ] dfn["Age"].max()
```

```
58
```

```
[ ] ages.max()
```

```
58
```

```
[ ] dfn.describe()
```

	Age
count	3.000000
mean	38.333333
std	18.230012
min	22.000000
25%	28.500000
50%	35.000000
75%	46.500000
max	58.000000

Reading csv file into dataframe

```
[ ] titanic = pd.read_csv("C:/Users/Student/Downloads/train.csv")
```

```
[ ] titanic
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

```
[ ] titanic.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
[ ] titanic.tail(3)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

writng dataframe to excel sheet

```
titanic.to_excel("C:/Users/Student/Downloads/titanic.xlsx", sheet_name="passengers", index=False)
```

```
titanic.info() #Gives summary of dataframe
```

Iloc and loc

loc and iloc

```
[ ] import pandas as pd
    import random
    data=pd.read_csv('https://s3-eu-west-1.amazonaws.com/shanebucket/downloads/uk-500.csv')
    data.head(5)
```

	first_name	last_name	company_name	address	city	county	postal	phone1	phone2	email	web
0	Aleshia	Tomkiewicz	Alan D Rosenburg Cpa Pc	14 Taylor St	St. Stephens Ward	Kent	CT2 7PP	01835-703597	01944-369967	atomkiewicz@hotmail.com	http://www.alandrosenburgcpapc.co.uk
1	Evan	Zigomalas	Cap Gemini America	5 Binney St	Abbey Ward	Buckinghamshire	HP11 2AX	01937-864715	01714-737868	evan.zigomalas@gmail.com	http://www.capgeminiamerica.co.uk
2	France	Andrade	Elliott, John W Esq	8 Moor Place	East Southbourne and Tuckton W	Bournemouth	BH6 3BE	01347-368222	01935-821636	france.andrade@hotmail.com	http://www.elliottjohnwesq.co.uk
3	Ulysses	Mcwalters	Mcmahan, Ben L	505 Exeter Rd	Hawerby cum Beesby	Lincolnshire	DN36 5RP	01912-771311	01302-801380	ulysses@hotmail.com	http://www.mcmahanbenl.co.uk
4	Tyisha	Veness	Champagne Room	5398 Forth Street	Greets Green and Lyng Ward	West Midlands	B70 9DT	01547-429341	01290-367248	tyisha.veness@hotmail.com	http://www.champagneroom.co.uk

```
[ ] data.tail(5)
```

	first_name	last_name	company_name	address	city	county	postal	phone1	phone2	email	web
495	Avery	Veit	Plaza Gourmet Delicatessen	9168 Devon St #905	Knightsbridge and Belgravia Wa	Greater London	SW1W 8JY	01748-625058	01389-188737	avery@veit.co.uk	http://www.plazagourmetdelicatessen.co.uk
496	Reid	Euresti	Fitzgerald, Edward J	70 Foster St	Inverness Ness-Side Ward	Highland	IV2 6WT	01916-983261	01370-319414	reuresti@euresti.co.uk	http://www.fitzgeraldedwardj.co.uk
497	Charlette	Brenning	Furey & Associates	714 Fonthill Rd	Darton West Ward	South Yorkshire	S75 5EJ	01888-152110	01301-312487	cbrenning@brenning.co.uk	http://www.fureyassociates.co.uk
498	Celestina	Keeny	Bfg Federal Credit Union	9 Milton St	Consett North ED	County Durham	DH8 5LP	01877-379881	01600-483475	celestina_keeny@gmail.com	http://www.bfgfederalcreditunion.co.uk
499	Mi	Richan	Nelson Wright Haworth Golf Crs	6 Norwood Grove	Tanworth-in-Arden	Warwickshire	B94 5RZ	01451-785624	01202-738408	mi@hotmail.com	http://www.nelsonwrighthaworthgolfcrs.co.uk

```
[ ] data.iloc[0]
```

```
first_name                  Aleshia
last_name                  Tomkiewicz
company_name    Alan D Rosenburg Cpa Pc
address                  14 Taylor St
city                  St. Stephens Ward
county                         Kent
postal                      CT2 7PP
phone1                 01835-703597
phone2                 01944-369967
email            atomkiewicz@hotmail.com
web          http://www.alandrosenburgcpapc.co.uk
Name: 0, dtype: object
```

```
[ ] data.iloc[0][0]
```

```
'Aleshia'
```

```
[ ] data.iloc[0][2]
```

```
'Alan D Rosenburg Cpa Pc'
```

```
[ ] data.iloc[2:4]
```

	first_name	last_name	company_name	address	city	county	postal	phone1	phone2	email	web
2	France	Andrade	Elliott, John W Esq	8 Moor Place	East Southbourne and Tuckton W	Bournemouth	BH6 3BE	01347-368222	01935-821636	france.andrade@hotmail.com	http://www.elliottjohnwesq.co.uk
3	Ulysses	Mcwalters	Mcmahan, Ben L	505 Exeter Rd	Hawerby cum Beesby	Lincolnshire	DN36 5RP	01912-771311	01302-801380	ulysses@hotmail.com	http://www.mcmahanbenl.co.uk

```
data.iloc[0][2]
```

```
'Alan D Rosenburg Cpa Pc'
```

```
data.iloc[2:4]
```

	first_name	last_name	company_name	address	city	county	postal	phone1	phone2	email	web
2	France	Andrade	Elliott, John W Esq	8 Moor Place	East Southbourne and Tuckton W	Bournemouth	BH6 3BE	01347-368222	01935-821636	france.andrade@hotmail.com	http://www.elliottjohnwesq.co.uk
3	Ulysses	Mcwalters	Mcmahan, Ben L	505 Exeter Rd	Hawerby cum Beesby	Lincolnshire	DN36 5RP	01912-771311	01302-801380	ulysses@hotmail.com	http://www.mcmahanbenl.co.uk

```
data.iloc[0:3,0:3]
```

	first_name	last_name	company_name
0	Aleshia	Tomkiewicz	Alan D Rosenburg Cpa Pc
1	Evan	Zigomalas	Cap Gemini America
2	France	Andrade	Elliott, John W Esq

```
data.iloc[:,0:2]
```

	first_name	last_name
0	Aleshia	Tomkiewicz
1	Evan	Zigomalas
2	France	Andrade
3	Ulysses	Mcwalters
4	Tyisha	Veness
...	...	...
495	Avery	Veit
496	Reid	Euresti
497	Charlette	Brenning
498	Celestina	Keeny
499	Mi	Richan

500 rows × 2 columns

```
data['id']=[random.randint(0,1000) for x in range(data.shape[0])]
data.loc[(data['id']>7) & (data['id']<10)]
```

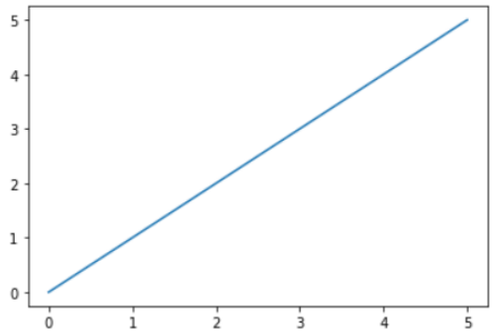
	first_name	last_name	company_name	address	city	county	postal	phone1	phone2	email	web	id
35	Alethea	Mould	Hughlett, Henry T Jr	6305 Elstow St	Castle Ward	West Sussex	BN12 6PR	01862-114247	01351-868986	alethea@hotmail.com	http://www.hughletthenrytjr.co.uk	9
117	Frankie	Marruffo	Paralegal Personnel	52 Morningside Rd #1	Cilybebyll Community	Neath Port Talbot	SA8 3HE	01295-678319	01966-999493	fmarruffo@gmail.com	http://www.paralegalpersonnel.co.uk	9

```
data.loc['Andrade']
```

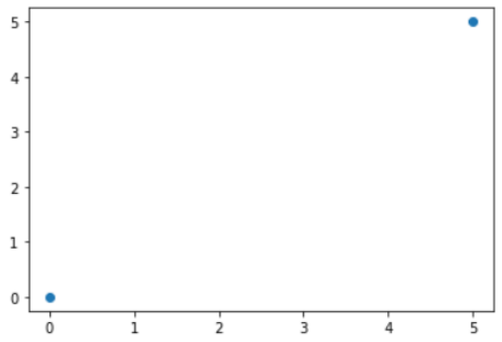
```
first_name                          France
company_name            Elliott, John W Esq
address                       8 Moor Place
city          East Southbourne and Tuckton W
county                        Bournemouth
postal                            BH6 3BE
phone1                        01347-368222
phone2                        01935-821636
email               france.andrade@hotmail.com
web              http://www.elliottjohnwesq.co.uk
id                                      233
Name: Andrade, dtype: object
```

Matplotlib

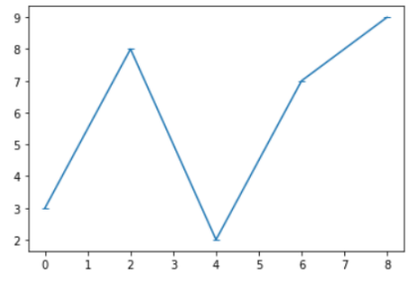
```
import matplotlib.pyplot as plt
import numpy as np
x=np.array([0,5])
y=np.array([0,5])
plt.plot(x,y)
plt.show()
```



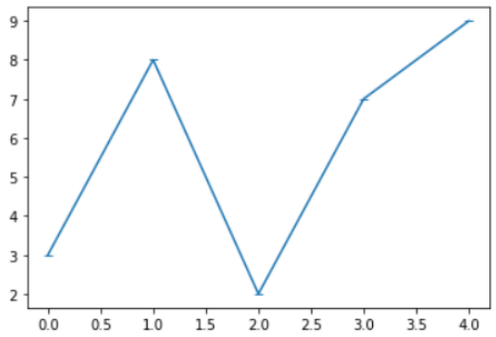
```
x=np.array([0,5])
y=np.array([0,5])
plt.plot(x,y,'o')
plt.show()
```



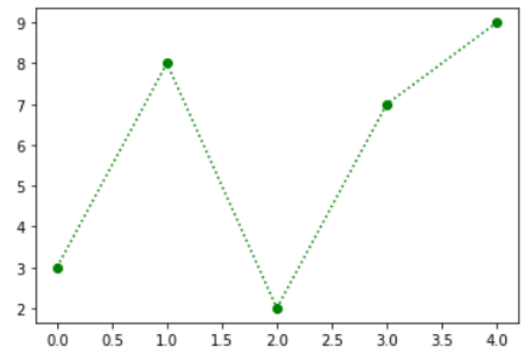
```
x=np.array([0,2,4,6,8])
y=np.array([3,8,2,7,9])
plt.plot(x,y,marker="_")
plt.show()
```



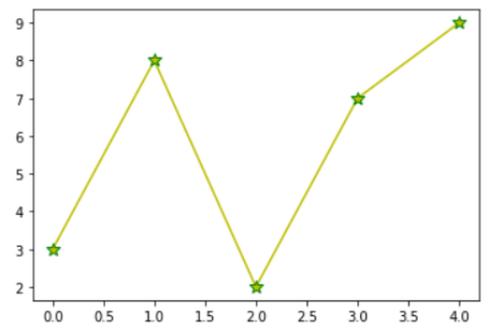
```
y=np.array([3,8,2,7,9])
plt.plot(y,marker="_")
plt.show()
```




```
y=np.array([3,8,2,7,9])
plt.plot(y,'o:g')
plt.show()
```

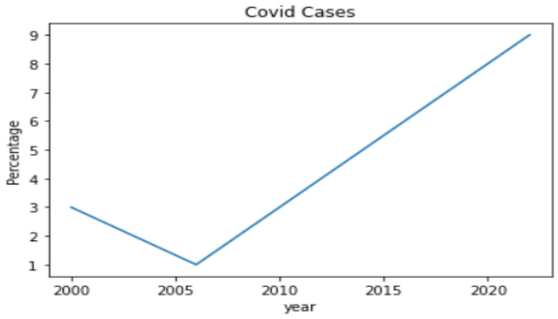



```
y=np.array([3,8,2,7,9])
plt.plot(y,marker="*",ms=10,mec='g',mfc='y',color='y')
plt.show()
```



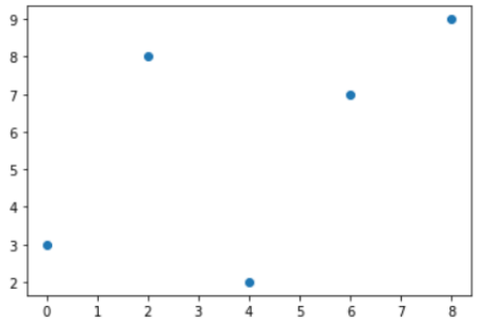

```
x=np.array([2000,2006,2008,2022])
y=np.array([3,1,2,9])
plt.plot(x,y)
```

```
plt.title("Covid Cases")
plt.xlabel("year")
plt.ylabel("Percentage")
plt.show()
```



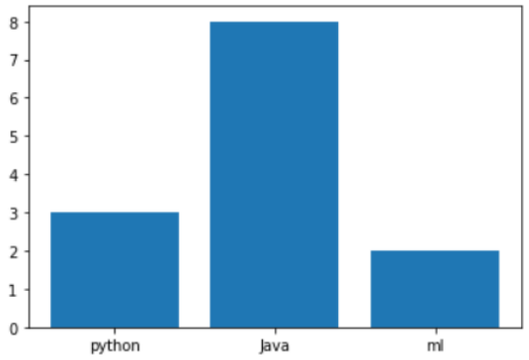
Scatter

```
x=np.array([0,2,4,6,8])
y=np.array([3,8,2,7,9])
plt.scatter(x,y)
plt.show()
```

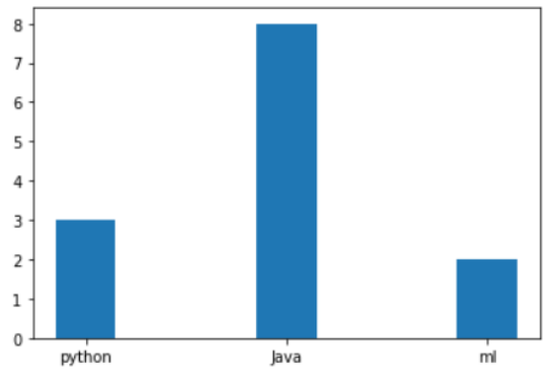


Bar graph

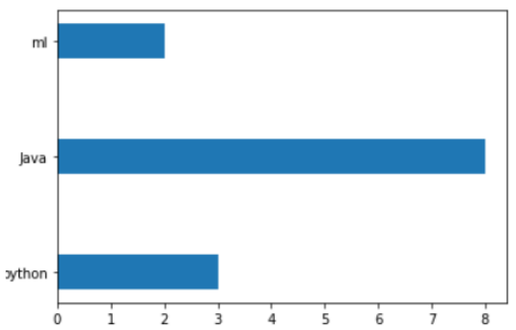
```
x=np.array(['python','Java','ml'])
y=np.array([3,8,2])
plt.bar(x,y)
plt.show()
```



```
x=np.array(['python','Java','ml'])
y=np.array([3,8,2])
plt.bar(x,y,width=0.3)
plt.show()
```

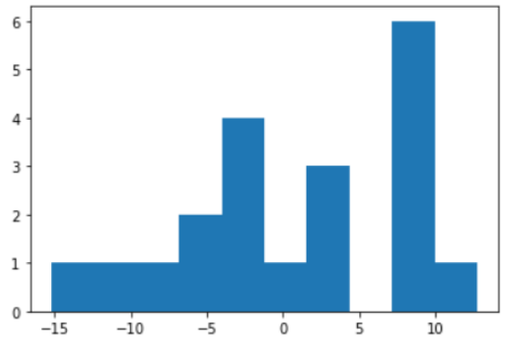



```
x=np.array(['python','Java','ml'])
y=np.array([3,8,2])
plt.barh(x,y,height=0.3)
plt.show()
```



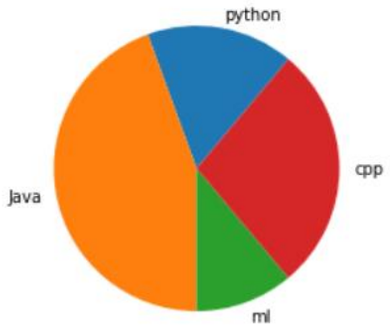
Histogram:

```
y=np.random.normal(3,8,20)
plt.hist(y)
plt.show()
```

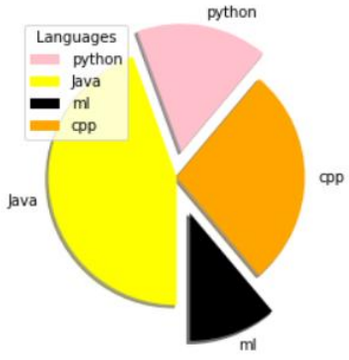


Pie chart

```
x=np.array([3,8,2,5])
label=['python','Java','ml','cpp']
plt.pie(x,labels=label,startangle=50)
plt.show()
```




```
x=np.array([3,8,2,5])
label=['python','Java','ml','cpp']
exp=[0.2,0,0.3,0]
color=['pink','yellow','k','orange']
plt.pie(x,labels=label,startangle=50,explode=exp,shadow=True,colors=color)
plt.legend(title="Languages")
plt.show()
```



10. Implement FIND-S algorithm for finding the most specific hypothesis based on a given set of training data sample.

Program:

```
import pandas as pd
import random
data=pd.read_csv("enjoySport.csv")
print(data)

#making an array of all the attributes
d = np.array(data)[:,:-1]
print("The attributes are:\n",d)

#segragating the target that has positive and negative examples
target = np.array(data)[:,-1]
print("The target is: ",target)

#training function to implement find-s algorithm
def train(c,t):
    for i, val in enumerate(t):
        if val == "Y":
            specific_hypothesis = c[i].copy()
            break

    for i, val in enumerate(c):
        if t[i] == "Y":
            for x in range(len(specific_hypothesis)):
                if val[x] != specific_hypothesis[x]:
                    specific_hypothesis[x] = '?'
                else:
                    pass

    return specific_hypothesis
```

Output:

```
      Sky AirTemp Humidity    Wind Water Forecast EnjoySport
0  Sunny    Warm   Normal  Strong  Warm     Same        Yes
1  Sunny    Warm     High  Strong  Warm     Same        Yes
2  Rainy    Cold     High  Strong  Warm   Change         No
3  Sunny    Warm     High  Strong  Cool   Change        Yes
The attributes are:
 [['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
 ['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
 ['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']
 ['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']]
The target is:  ['Yes' 'Yes' 'No' 'Yes']
```

11. For a given set of training data examples, implement Candidate-Elimination algorithm to output a description of the set of all consistent hypothesis.

Program:

```
import random
import csv
def g_0(n):
    return ("?",)*n

def s_0(n):
    return ('0',)*n
def more_general(h1, h2):
    more_general_parts = []
    for x, y in zip(h1, h2):
        mg = x == "?" or (x != "0" and (x == y or y == "0"))
        more_general_parts.append(mg)
    return all(more_general_parts)
def fulfills(example, hypothesis):
    ### the implementation is the same as for hypotheses:
    return more_general(hypothesis, example)

def min_generalizations(h, x):
    h_new = list(h)
    for i in range(len(h)):
        if not fulfills(x[i:i+1], h[i:i+1]):
            h_new[i] = '?' if h[i] != '0' else x[i]
    return [tuple(h_new)]
def min_specializations(h, domains, x):
    results = []
    for i in range(len(h)):
        if h[i] == "?":
            for val in domains[i]:
                if x[i] != val:
                    h_new = h[:i] + (val,) + h[i+1:]
                    results.append(h_new)
        elif h[i] != "0":
            h_new = h[:i] + ('0',) + h[i+1:]
            results.append(h_new)
    return results
with open('enjoySport.csv')  as csvFile:
        examples = [tuple(line) for line in csv.reader(csvFile)]
examples
def get_domains(examples):
    d = [set() for i in examples[0]]
    for x in examples:
        for i, xi in enumerate(x):
            d[i].add(xi)
    return [list(sorted(x)) for x in d]

get_domains(examples)
def candidate_elimination(examples):
    domains = get_domains(examples)[:-1]

    G = set([g_0(len(domains))])
```

```
    S = set([s_0(len(domains))])
    i=0
    print("\n G[{0}]:".format(i),G)
    print("\n S[{0}]:".format(i),S)
    for xcx in examples:
        i=i+1
        x, cx = xcx[:-1], xcx[-1]  # Splitting data into attributes and decisions
        if cx=='Y': # x is positive example
            G = {g for g in G if fulfills(x, g)}
            S = generalize_S(x, G, S)
        else: # x is negative example
            S = {s for s in S if not fulfills(x, s)}
            G = specialize_G(x, domains, G, S)
        print("\n G[{0}]:".format(i),G)
        print("\n S[{0}]:".format(i),S)
    return
def generalize_S(x, G, S):
    S_prev = list(S)
    for s in S_prev:
        if s not in S:
            continue
        if not fulfills(x, s):
            S.remove(s)
            Splus = min_generalizations(s, x)
            ## keep only generalizations that have a counterpart in G
            S.update([h for h in Splus if any([more_general(g,h)
                                    for g in G])])
            ## remove hypotheses less specific than any other in S
            S.difference_update([h for h in S if
                                 any([more_general(h, h1)
                                     for h1 in S if h != h1])])
    return S
def specialize_G(x, domains, G, S):
    G_prev = list(G)
    for g in G_prev:
        if g not in G:
            continue
        if fulfills(x, g):
            G.remove(g)
            Gminus = min_specializations(g, domains, x)
            ## keep only specializations that have a conuterpart in S
            G.update([h for h in Gminus if any([more_general(h, s)
                                    for s in S])])
            ## remove hypotheses less general than any other in G
            G.difference_update([h for h in G if
                                 any([more_general(g1, h)
                                     for g1 in G if h != g1])])
    return G
candidate_elimination(examples)
```

Output:

```
[('Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same', 'Y'),
 ('Sunny', 'Warm', 'High', 'Strong', 'Warm', 'Same', 'Y'),
 ('Rainy', 'Cold', 'High', 'Strong', 'Warm', 'Change', 'N'),
 ('Sunny', 'Warm', 'High', 'Strong', 'Cool', 'Change', 'Y')]
```

```
[['Rainy', 'Sunny'],
 ['Cold', 'Warm'],
 ['High', 'Normal'],
 ['Strong'],
 ['Cool', 'Warm'],
 ['Change', 'Same'],
 ['N', 'Y']]
```

```
G[0]: {('?', '?', '?', '?', '?', '?')}
```

```
S[0]: {('0', '0', '0', '0', '0', '0')}
```

```
G[1]: {('?', '?', '?', '?', '?', '?')}
```

```
S[1]: {('Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same')}
```

```
G[2]: {('?', '?', '?', '?', '?', '?')}
```

```
S[2]: {('Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same')}
```

```
G[3]: {('Sunny', '?', '?', '?', '?', '?'), ('?', 'Warm', '?', '?', '?', '?'), ('?', '?', '?', '?', '?', 'Same')}
```

```
S[3]: {('Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same')}
```

```
G[4]: {('Sunny', '?', '?', '?', '?', '?'), ('?', 'Warm', '?', '?', '?', '?')}
```

```
S[4]: {('Sunny', 'Warm', '?', 'Strong', '?', '?')}
```

1.Implement Decision tree for classification of any given data

Program:

```
set.import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
#%matplotlib inline#for encoding
from sklearn.preprocessing import LabelEncoder#for train test splitting
from sklearn.model_selection import train_test_split#for decision tree object
from sklearn.tree import DecisionTreeClassifier#for checking testing results
from sklearn.metrics import classification_report, confusion_matrix#for visualiz
ing tree s
from sklearn.tree import plot_tree
#reading the data
df = sns.load_dataset('iris')
df.head()
```

	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

```
#getting information of dataset
df.info()
```

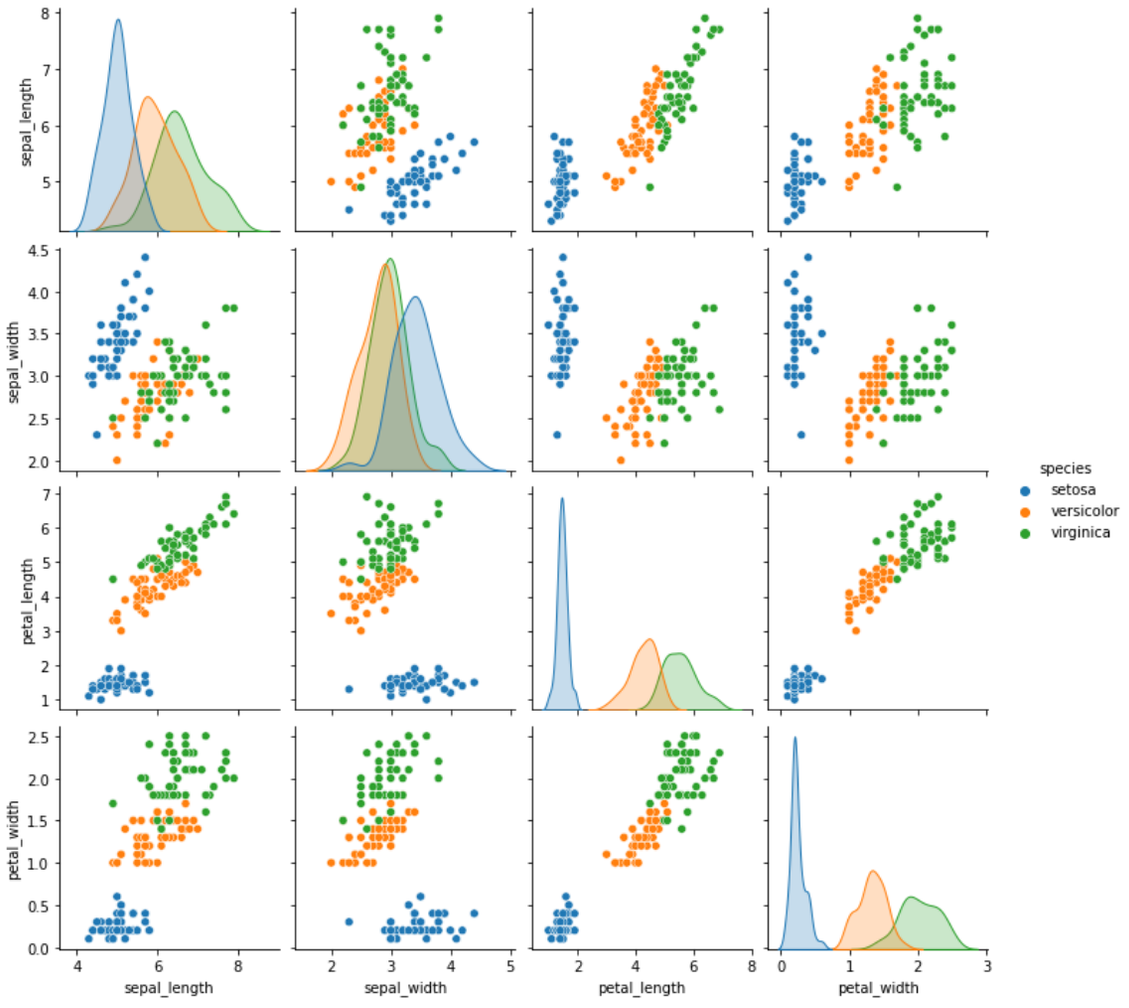
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null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

```
df.shape
(150, 5)
df.isnull().any()
```

```
sepal_length    False
sepal_width     False
petal_length    False
petal_width     False
species         False
dtype: bool
```

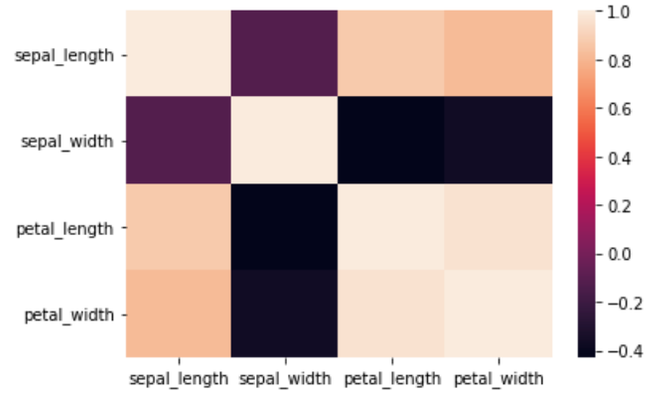
```
# let's plot pair plot to visualise the attributes all at once
sns.pairplot(data=df, hue = 'species')
```

```
<seaborn.axisgrid.PairGrid at 0x7f6ece660190>
```




```
# correlation matrix
sns.heatmap(df.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6ed5505e90>
```




```
target=df['species']
df1=df.copy()
df1=df1.drop('species',axis=1) #removing whole species column
X = df1
target=df['species']
target
```

```
0         setosa
1         setosa
2         setosa
3         setosa
4         setosa
        ...
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Name: species, Length: 150, dtype: object
```

```
#label encoding
le = LabelEncoder()
target = le.fit_transform(target)
target
```

```
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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
y=target
# Splitting the data - 80:20 ratio
X_train, X_test, y_train, y_test = train_test_split(X , y, test_size = 0.2, rand
om_state = 42)
print("Training split input- ", X_train.shape)
print("Testing split input- ", X_test.shape)
Training split input-  (120, 4)
   Testing split input-  (30, 4)
```

```
# Defining the decision tree algorithm
dtree=DecisionTreeClassifier()
dtree.fit(X_train,y_train)
print('Decision Tree Classifier Created')
# Predicting the values of test data
y_pred = dtree.predict(X_test)
print("Classification report - \n", classification_report(y_test,y_pred))
```

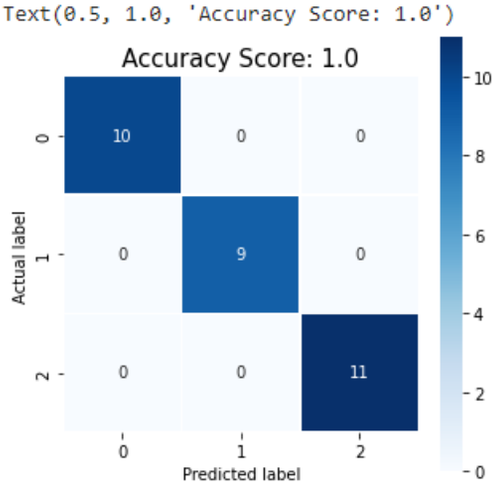
```
Classification report -
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

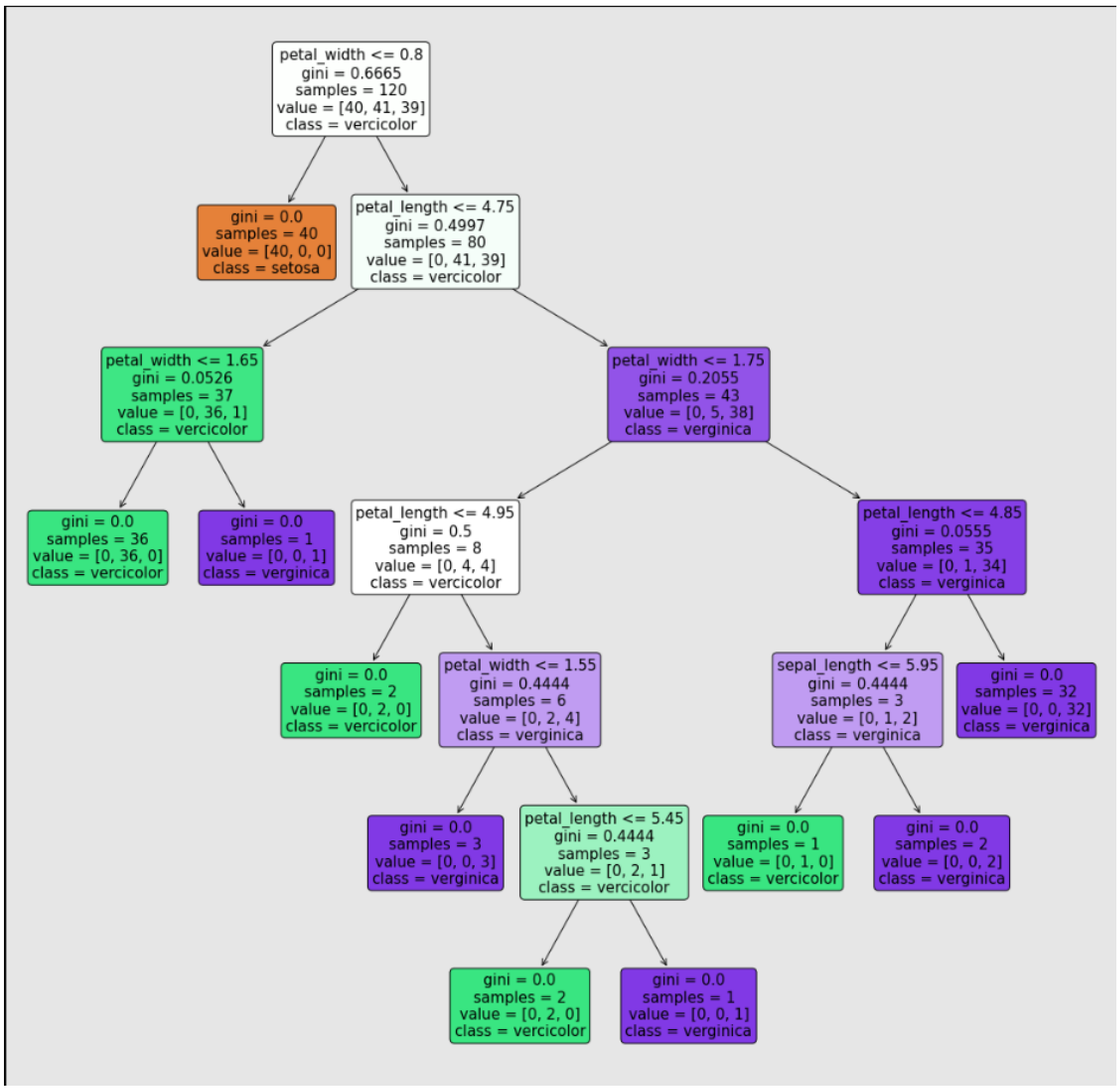
```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, annot=True,square = True,  cmap = 'Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score: {0}'.format(dtree.score(X_test, y_test))
plt.title(all_sample_title, size = 15)
```

```
Text(0.5, 1.0, 'Accuracy Score: 1.0')
```




```
# Visualising the graph without the use of graph
vizplt.figure(figsize = (20,20))
dec_tree = plot_tree(decision_tree=dtree, feature_names = df1.columns,
                     class_names =["setosa", "vercicolor", "verginica"] , filled
 = True , precision = 4, rounded = True)
```

Output:


petal_width <= 0.8
gini = 0.6665
samples = 120
value = [40, 41, 39]
class = vercicolor
gini = 0.0
samples = 40
value = [40, 0, 0]
class = setosa
petal_length <= 4.75
gini = 0.4997
samples = 80
value = [0, 41, 39]
class = vercicolor
petal_width <= 1.65
gini = 0.0526
samples = 37
value = [0, 36, 1]
class = vercicolor
petal_width <= 1.75
gini = 0.2055
samples = 43
value = [0, 5, 38]
class = verginica
gini = 0.0
samples = 36
value = [0, 36, 0]
class = vercicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = verginica
petal_length <= 4.95
gini = 0.5
samples = 8
value = [0, 4, 4]
class = vercicolor
petal_length <= 4.85
gini = 0.0555
samples = 35
value = [0, 1, 34]
class = verginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = vercicolor
petal_width <= 1.55
gini = 0.4444
samples = 6
value = [0, 2, 4]
class = verginica
sepal_length <= 5.95
gini = 0.4444
samples = 3
value = [0, 1, 2]
class = verginica
gini = 0.0
samples = 32
value = [0, 0, 32]
class = verginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = verginica
petal_length <= 5.45
gini = 0.4444
samples = 3
value = [0, 2, 1]
class = vercicolor
gini = 0.0
samples = 1
value = [0, 1, 0]
class = vercicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = verginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = vercicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = verginica