```
x = 10
```

```
x
```

```
10
```

```
mysub = "python"
mysub
```

```
'python'
```

```
print(mysub)
```

```
python
```

```
_mycity = "Solan"
print(_mycity)
```

```
Solan
```

```
my_sub1 = "Generative AI"
print(my_sub1)
```

```
Generative AI
```

```
1my_sub = "Science"
print(1my_sub)
```

```
  File "/tmp/ipython-input-2475929071.py", line 1
    1my_sub = "Science"
    ^
SyntaxError: invalid decimal literal
```

```
import keyword
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def
```

```
x = 0
type(x)
```

```
int
```

```
y = 1.0
type(y)
```

```
float
```

```
z = 3+4j
type(z)
```

```
complex
```

```
#string --data in the form text
a = "solan"
```

```
type(a)
```

```
str
```

```
#boolean -- it is used to store value in the form of True and False
a = True
type(a)
```

```
bool
```

```
#None --it is define value means nothing
e = None
type(e)
```

```
NoneType
```

```
#single quotes
my_sub = 'Generative AI'
type(my_sub)
```

```
str
```

```
#double quotes
my_college = "shoolini university"
type(my_college)
```

```
str
```

```
#triple quotes
my_sentence = ''' this sentence is multiple lines,
this is also a string'''
type(my_sentence)
```

```
str
```

```
#find the length of string
my_sub = "Machine learning"
len(my_sub)
```

```
16
```

```
my_sub[0]
```

```
'M'
```

```
my_sub[-16]
```

```
'M'
```

```
my_sub[-1]
```

```
'g'
```

```
my_sub[4]
```

```
'i'
```

```
my_sub
```

```
'Machine learning'
```

```
my_sub[0:7:1]
```

```
'Machine'
```

```
#concatenation--adding the string
first_name = "Amit"
last_name = "Kumar"
full_name = first_name +" "+ last_name
print(full_name)
```

```
Amit Kumar
```

```
#methods in string
#lower()--this method convert string into lower case
my_sub = "Machine Learning"
my_sub.lower()
```

```
'machine learning'
```

```
lower(my_sub)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-3174124106.py in <cell line: 0>()
----> 1 lower(my_sub)

NameError: name 'lower' is not defined
```

```
#upper()--this method convert string into upper case
my_sub.upper()
```

```
'MACHINE LEARNING'
```

```
#count()--count particular character of the string
my_sub = "Machine Learning"
my_sub.count("a")
```

```
2
```

```
#list--it is sequence of item,every item separate by comma
#in list all item are enclosed by squarebracket
my_list = [98, 34, 56, "apple", True]
print(my_list)
```

```
[98, 34, 56, 'apple', True]
```

```
type(my_list)
```

```
list
```

```
#list homogenius as well hetrogenious
my_list = [10, 4, 5,6 ,7] #homogenius
```

```
type(my_list)
```

```
list
```

```
list1 = ["apple", True, 2, 3.14, None] #hetrogenious
type(list1)
```

```
list
```

```
#indexing in list-- location of item
#positive indexing
my_list =[23,45, 98, 12, 56]
my_list[0]
```

```
23
```

```
len(my_list)
```

```
5
```

```
my_list[4]
```

```
56
```

```
#negative indexing
my_list =[23,45, 98, 12, 56]
my_list[-1]

```

```
56
```

```
print(my_list[-2])
print(my_list[-3])
print(my_list[-4])
print(my_list[-5])
```

```
12
98
45
23
```

```
#slicing in list
my_list = [2, 4, 7, 8, 9, 5, 3]
my_list[0:6:1]
```

```
[2, 4, 7, 8, 9, 5]
```

```
#list print in apposite direction
my_list[: :-1]
```

```
[3, 5, 9, 8, 7, 4, 2]
```

```
my_list[:]
```

```
[2, 4, 7, 8, 9, 5, 3]
```

```
#list is concatenated
list1 = [3, 4,5, 6]
list2 = [6, 7, 8, 9]
```

```
my_list = list1+list2
print(my_list)
```

```
[3, 4, 5, 6, 6, 7, 8, 9]
```

```
#function in list
my_list = [1, 2,3 ,4, 5,6]
min(my_list)
```

```
1
```

```
max(my_list)
```

```
6
```

```
sum(my_list)
```

```
21
```

```
list1 = [8, 6, 9, 3, 10, 5, 2]
sorted(my_list)
```

```
[2, 3, 5, 6, 8, 9, 10]
```

```
list1
```

```
[8, 6, 9, 3, 10, 5, 2]
```

```
my_list = [8, 6, 9, 3, 10, 5, 2]
sorted(my_list, reverse = True)
```

```
[10, 9, 8, 6, 5, 3, 2]
```

```
my_list = [8, 6, 9, 3, 10, 5, 2]
my_list.sort(reverse =True)
```

```
my_list
```

```
[10, 9, 8, 6, 5, 3, 2]
```

```
my_list = [98, 45, 56, 78]
my_list.append(100)
my_list
```

```
[98, 45, 56, 78, 100]
```

```
list1 = [67, 56, 43, 78]
list2 = [90, 89, 76]
list1.extend(list2)
list1
```

```
[67, 56, 43, 78, 90, 89, 76]
```

```
#pop()--
my_list = [78, 45, 35, 78, 90]
my_list.pop()
```

```
90
```

```
my_list
```

```
[78, 45, 35, 78]
```

```
my_list.pop(2)
```

```
35
```

```
my_list
```

```
[78, 45, 78]
```

```
#remove
list1 = [67, 89, 45, 34]
list1.remove(89)
```

```
list1
```

```
[67, 45, 34]
```

```
my_dict = {"Name":"Kritika", "College":"shoolini"}
type(my_dict)
```

```
dict
```

```
print(my_dict)
```

```
{'Name': 'Kritika', 'College': 'shoolini'}
```

```
my_dict["Name"]
```

```
'Kritika'
```

```
my_dict["College"]
```

```
'shoolini'
```

```
#method in dictionary
#keys()--this method return keys of dictionary
list(my_dict.keys())
```

```
['Name', 'College']
```

```
#values()--this method return values of dictionary
list(my_dict.values())
```

```
['Kritika', 'shoolini']
```

```
#items()--this method return item of the dictionary
list(my_dict.items())
```

```
[('Name', 'Kritika'), ('College', 'shoolini')]
```

```
#get()--this method find value and take key as a argument
my_dict.get("Name")
```

```
'Kritika'
```

```
print(my_dict.get("Age", "Age is not present"))
```

```
Age is not present
```

```
#update()--this method update dictionary and take dictionary as argument
my_dict.update({"Age":20})
print(my_dict)
```

```
{'Name': 'Kritika', 'College': 'shoolini', 'Age': 20}
```

```
my_dict.update({"Name":"Jagat"})
print(my_dict)
```

```
{'Name': 'Jagat', 'College': 'shoolini', 'Age': 20}
```

```
#conditional statement
#if --this statement execute when condition is true
#elif--this statement execute when condition is true
#else--this statement execute when if and elif are not execute
age = int(input("enter your age: "))
if age>60:
  print("You are too old for marry")
elif age<18:
  print("You are too young for marry")
else:
  print("we will find perfact match for you")
```

```
enter your age: 12
You are too young for marry
```

```
#loop helps execute block of repeatedly
#for loop--this loop apply on sequence(list, string, tuple)
my_sub = "Python"
print(my_sub[0])
print(my_sub[1])
print(my_sub[2])
print(my_sub[3])
print(my_sub[4])
print(my_sub[5])
```

```
P
y
t
h
o
n
```

```
my_sub = "python"
for i in my_sub:
  print(i)
```

```
p
y
t
h
o
n
```

```
for i in range(len(my_sub)):
  print(my_sub[i], "->", i)
```

```
p -> 0
y -> 1
t -> 2
h -> 3
o -> 4
n -> 5
```

```
my_list = [89, 76, 45, 34]
for i in my_list:
  print(i)
```

```
89
76
45
34
```

```
#while loop--this loop execute until condition is true
i= 0
while i<5:
  print(i)
  i+=1
```

```
0
1
2
3
4
```

```
#loop control statement
#break-- this statement break the loop
i = 0
while i<5:
  i+=1
  if i == 3:
    break
  else:
    print(i)
```

```
1
2
```

```
#continue statement--skip current iteration
i = 0
while i<5:
  i+=1
  if i==3:
    continue
    #print(i)
  else:
    print(i)
```

```
1
2
4
5
```

```
#pass statement --only pass cursor
i = 0
```

```
while i<5:
  i+=1
  if i==3:
    pass
    print(i)
  else:
    print(i)
```

```
1
2
3
4
5
```

```
#user defined function without argument
def add():
  a = int(input("Enter first number: "))
  b = int(input("Enter second number:"))
  return a+b
```

```
print(add())
```

```
Enter first number: 10
Enter second number:5
15
```

```
#positional argument
def Calculation(a, b):
  print("addition: ", a+b)
  print("Subtraction :", a-b)
  print("Multiplication: ", a*b)
  print("Division :", a/b)
  return "Calculation are complete"
```

```
print(Calculation(10, 5))
```

```
addition:  15
Subtraction : 5
Multiplication:  50
Division : 2.0
Calculation are complete
```

```
#default argument
def area_perimeter(width = 8, height = 4):
  area = width*height
  perimeter = 2*(width +height)
  return "Area is", area, "and perimeter is", perimeter
```

```
print(area_perimeter(10, 8))
```

```
('Area is', 80, 'and perimeter is', 36)
```

```
#keyword argument
def interest(p, r, t):
  i = (p*r*t)/100
  return i
```

```
print(interest(t = 2, p = 1000, r= 10))
```

```
200.0
```

```
#mixed argument
print(interest(1000, t = 2, r = 10))
```

```
200.0
```

```
print(interest(p = 1000, r = 10, 2))
```

```
  File "/tmp/ipython-input-1496732972.py", line 1
    print(interest(p = 1000, r = 10, 2))
                                       ^
SyntaxError: positional argument follows keyword argument
```

```
#non keyword variable length argument
def test(*args):
  print(args)
  print(len(args))
  print(type(args))
```

```
test(1, 2,3, 4,5, 6,7)
```

```
(1, 2, 3, 4, 5, 6, 7)
7
<class 'tuple'>
```

```
def sum_number(*args):
  sum = 0
  for num in args:
    sum+=num
  return sum
```

```
sum_number(4, 5, 6, 7)
```

```
22
```

```
#keyword variable length argument
def test(**kwargs):
  print(kwargs)
  print(len(kwargs))
  print(type(kwargs))
```

```
test(a = 10, b = 20, c = 30)
```

```
{'a': 10, 'b': 20, 'c': 30}
3
<class 'dict'>
```

```
import numpy as np
```

```
oned_array = np.array([1, 2, 3])
print(oned_array)
```

```
[1 2 3]
```

```
row_vector = np.array([[1, 2, 3]])
print(row_vector)
```

```
[[1 2 3]]
```

```
column_vector = row_vector.transpose()
print(column_vector)
```

```
[[1]
 [2]
 [3]]
```

```
print(oned_array)
```

```
[1 2 3]
```

```
#ndim--dimension
oned_array.ndim
```

```
1
```

```
row_vector
```

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

```
row_vector.ndim
```

```
2
```

```
print(row_vector)
```

```
[[1 2 3]]
```

```
print(row_vector.ndim)
print(row_vector.shape)
print(row_vector.size)
print(row_vector.transpose())
```

```
2
(1, 3)
3
[[1]
 [2]
 [3]]
```

```
rank_one_vector = np.array([1, 2, 3])
print(rank_one_vector)
```

```
[1 2 3]
```

```
twod_array = np.array([[1, 2, 3],[4, 5, 6]])
print(twod_array)
```

```
[[1 2 3]
 [4 5 6]]
```

```
#indexing
print(twod_array[0][1])
```

```
2
```

```
print(twod_array[1][2])
```

```
6
```

```
print(twod_array)
```

```
[[1 2 3]
 [4 5 6]]
```

```
print(twod_array[0:,0:2])
```

```
[[1 2]
 [4 5]]
```

```
mat1 = np.matrix("1, 2, 3;4, 5, 6;7, 8, 9")
print(mat1)
```

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

```
random_array = np.random.random((3, 3))
print(random_array)
```

```
[[0.63446121 0.0837939  0.15107758]
 [0.82039292 0.91825149 0.47926521]
 [0.02743014 0.28666014 0.40375273]]
```

```
randint_array = np.random.randint(1, 10, (3, 3))
print(randint_array)
```

```
[[1 8 5]
 [4 1 2]
 [8 9 6]]
```

```
data = np.array([1,2, 3, 4])
power_data = np.power(data, 2)
print(power_data)
```

```
[ 1  4  9 16]
```

```
print(data)
```

```
[1 2 3 4]
```

```
exp_data = np.exp(data)
print(exp_data)
```

```
[ 2.71828183  7.3890561  20.08553692 54.59815003]
```

```
log_data = np.log(data)
print(log_data)
```

```
[0.         0.69314718 1.09861229 1.38629436]
```

```
log2_data = np.log2(data)
print(log2_data)
```

```
[0.        1.        1.5849625 2.       ]
```

```
log10_data = np.log10(data)
print(log10_data)
```

```
[0.         0.30103    0.47712125 0.60205999]
```

```
zeros_array = np.zeros((3, 3),dtype =int )
print(zeros_array)
```

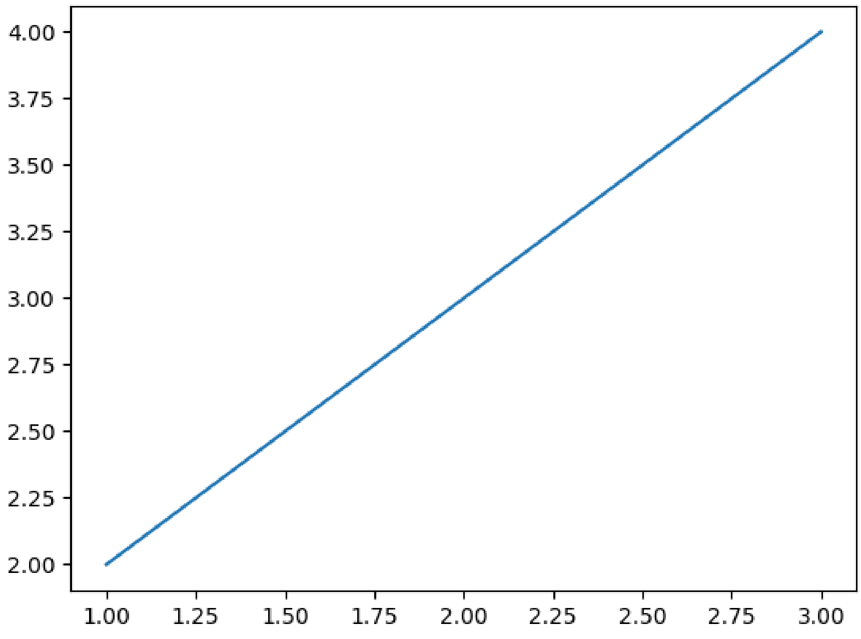
```
[[0 0 0]
 [0 0 0]
 [0 0 0]]
```

```
ones_array = np.ones((2, 3),dtype = int)
print(ones_array)
```

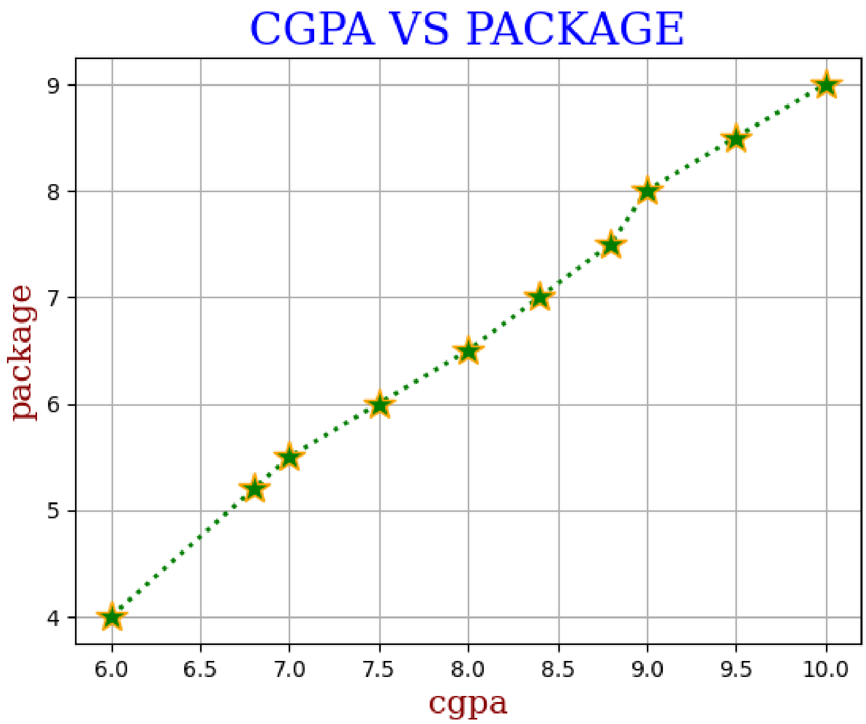
```
[[1 1 1]
 [1 1 1]]
```

```
#matplotlib
import matplotlib.pyplot as plt
import numpy as np
```

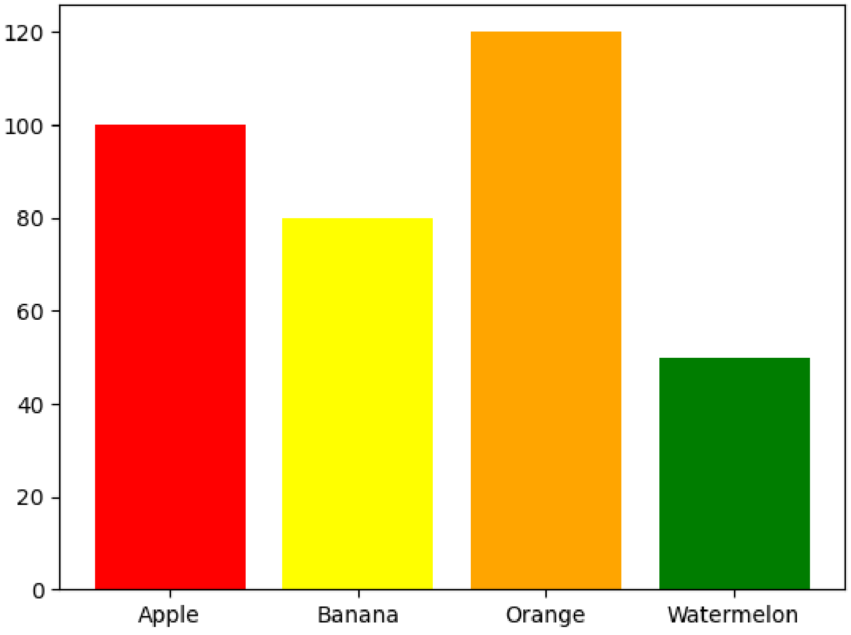
```
#line plot
xpoint = np.array([1, 3])
ypoint = np.array([2, 4])
plt.plot(xpoint, ypoint)
plt.show()
```



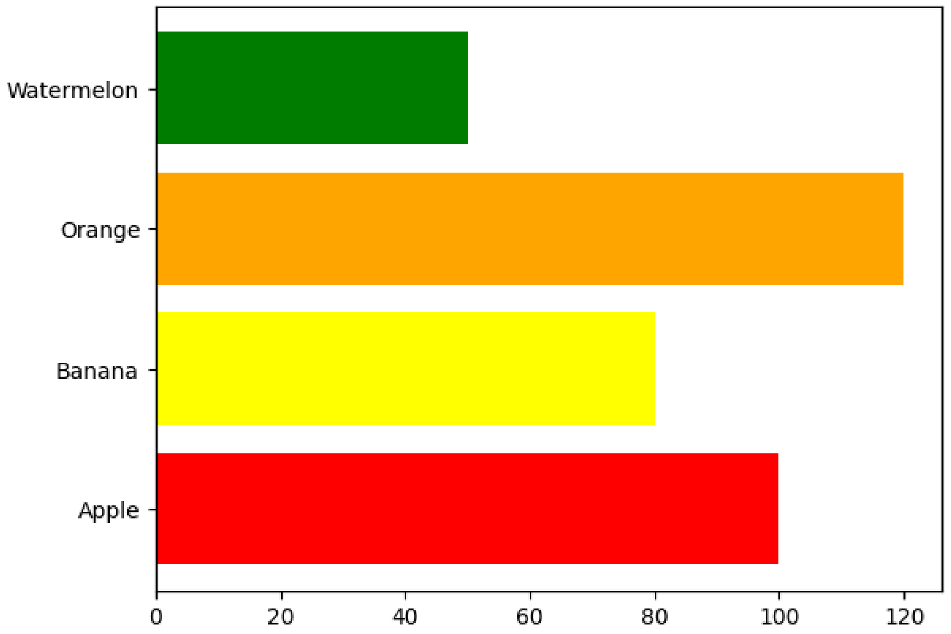
```
cgpa = np.array([6, 6.8, 7, 7.5, 8, 8.4, 8.8, 9, 9.5, 10])
package = np.array([4, 5.2, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9])
font1 = {'family':'serif','color':'blue','size':20}
font2 = {'family':'serif','color':'darkred','size':15}
plt.plot(cgpa,package,"*:g", ms =15,mec = "orange",linewidth = 2)
plt.xlabel("cgpa",fontdict=font2)
plt.ylabel("package",fontdict=font2)
plt.title("CGPA VS PACKAGE",fontdict=font1)
plt.grid()
plt.show()
```



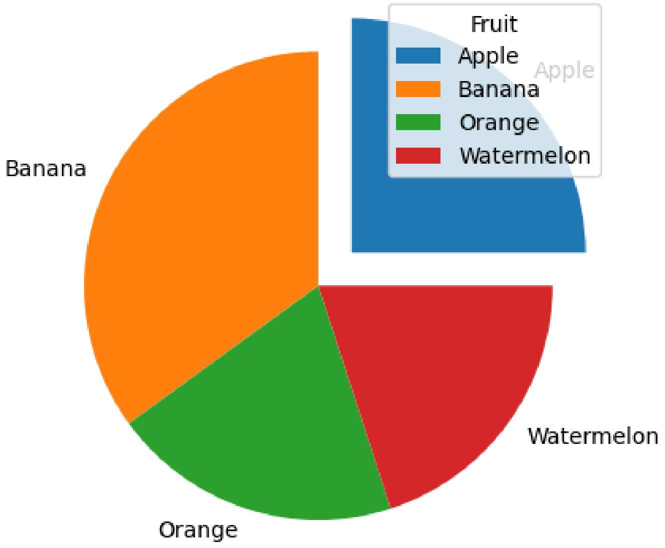
```
#bar plot--it work on numerical as well categorical data
import matplotlib.pyplot as plt
import numpy as np
fruit = np.array(["Apple", "Banana","Orange","Watermelon"])
fruit_number = np.array([100, 80,120, 50])
color = ["red","yellow","orange","green"]
plt.bar(fruit, fruit_number, color = color)
plt.show()
```



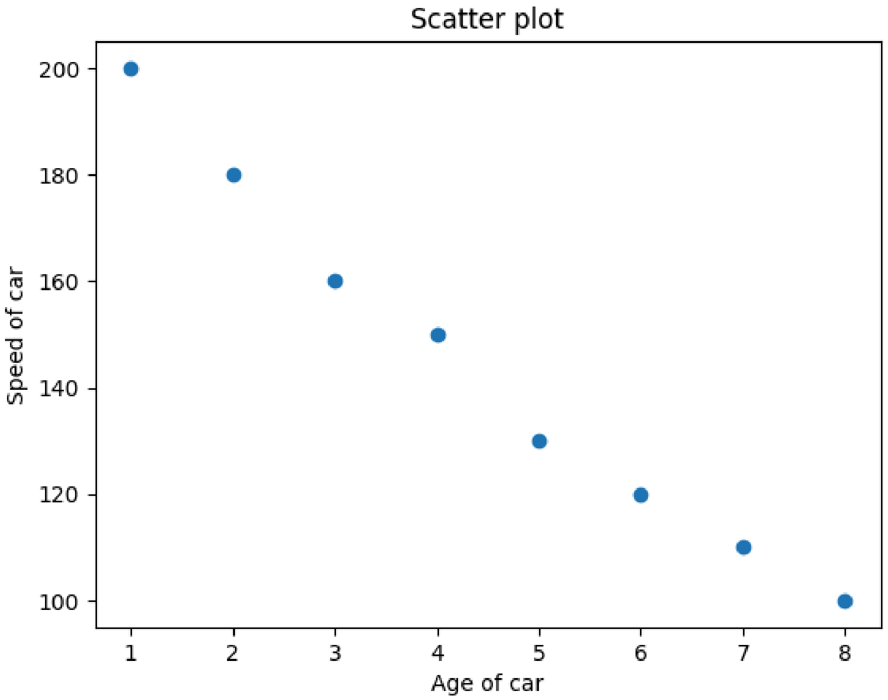
```
fruit = np.array(["Apple", "Banana","Orange","Watermelon"])
fruit_number = np.array([100, 80,120, 50])
color = ["red","yellow","orange","green"]
plt.barh(fruit, fruit_number, color = color)
plt.show()
```



```
#pie plot
x = np.array([25, 35, 20, 20])
my_labels = np.array(["Apple", "Banana", "Orange", "Watermelon"])
my_explode = np.array([0.2, 0, 0, 0])
plt.pie(x, labels = my_labels,explode = my_explode)
plt.legend(title = "Fruit",loc = "upper right")
plt.show()
```

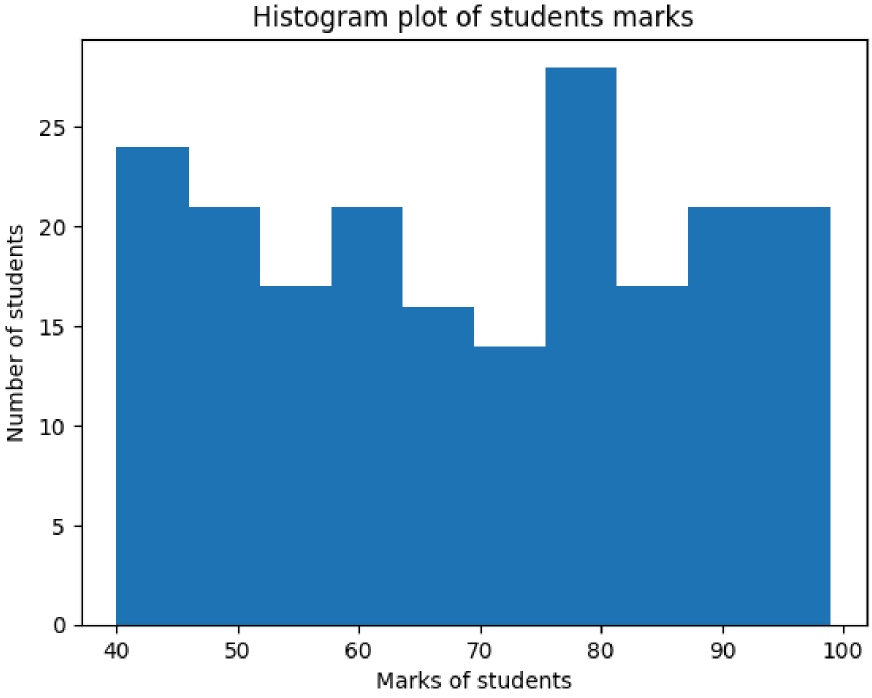


```
#scatter plot--it is work on numerical data
x = np.array([1, 2, 3, 4, 5, 6, 7, 8])
y = np.array([200, 180, 160, 150, 130, 120, 110, 100])
plt.xlabel("Age of car")
plt.ylabel("Speed of car")
plt.title("Scatter plot")
plt.scatter(x, y)
plt.show()
```



```
#histogram --A histogram shows frequency distribution
x = np.random.randint(40, 100, (1, 200))
x = x.flatten()
print(x)
plt.xlabel("Marks of students")
plt.ylabel("Number of students")
plt.title("Histogram plot of students marks")
plt.hist(x,bins = 10)
plt.show()
```

```
[44 60 89 53 97 48 40 74 70 80 61 97 48 60 61 49 53 98 96 51 74 42 86 64
 41 79 75 88 59 92 55 86 79 60 88 62 65 98 70 59 65 51 53 81 99 87 99 91
 46 48 55 62 81 64 78 88 44 61 66 82 95 94 82 79 96 65 98 43 71 83 41 60
 96 77 81 98 48 95 85 40 72 71 61 67 66 44 56 79 67 89 62 56 45 78 57 78
 45 50 91 41 86 47 76 49 77 86 65 46 61 84 80 76 55 74 75 80 43 99 57 84
 83 47 57 49 43 51 89 60 45 92 45 40 51 59 82 78 95 92 89 81 45 48 91 87
 44 79 58 88 67 89 67 48 43 60 75 81 90 43 47 73 93 52 64 88 81 91 83 42
 53 79 94 50 54 44 93 78 97 97 68 96 93 84 79 78 57 71 49 68 41 79 62 56
 62 83 59 78 56 99 65 73]
```



```
import pandas as pd
```

```
my_series = pd.Series(["Apple", "Banana", "Orange", "Watermelon"], name="Fruit
my_series
```

	Fruit
0	Apple
1	Banana
2	Orange
3	Watermelon

dtype: object

```
list_of_list = [["Amar",15],["Akbar",14], ["Anthony",13]]
df = pd.DataFrame(list_of_list, columns = ["Name", "Age"] )
df
```

	Name	Age
0	Amar	15
1	Akbar	14
2	Anthony	13

```
import pandas as pd
df = pd.read_csv("/content/drive/MyDrive/data/imdb_data.csv")
df
```

	id	belongs_to_collection	budget	genres	homepage	imdb_id	origir
0	1	[{'id': 313576, 'name': 'Hot Tub Time Machine ...	14000000	[{'id': 35, 'name': 'Comedy'}]	NaN	tt2637294	
1	2	[{'id': 107674, 'name': 'The Princess Diaries ...	40000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	tt0368933	
2	3	NaN	3300000	[{'id': 18, 'name': 'Drama'}]	http://sonyclassics.com/whiplash/	tt2582802	
3	4	NaN	1200000	[{'id': 53, 'name': 'Thriller'}, {'id': 18, 'n...	http://kahaanithefilm.com/	tt1821480	
4	5	NaN	0	[{'id': 28, 'name': 'Action'}, {'id': 53, 'nam...	NaN	tt1380152	
...	...	...	...	...	...	...	
2995	2996	NaN	0	[{'id': 35, 'name': 'Comedy'}, {'id': 10749, '...	NaN	tt0109403	
2996	2997	NaN	0	[{'id': 18, 'name': 'Drama'}, {'id': 10402, 'n...	NaN	tt2364975	
2997	2998	NaN	65000000	[{'id': 80, 'name': 'Crime'}, {'id': 28, 'name...	NaN	tt0116908	

```
#find the number of rows and columns
df.shape
```

```
(3000, 23)
```

```
#fetch data from top
df.head(2)
```

3000 rows × 23 columns

	id	belongs_to_collection	budget	genres	homepage	imdb_id	original_language	original_tit
0	1	[{'id': 313576, 'name': 'Hot Tub Time Machine ...	14000000	[{'id': 35, 'name': 'Comedy'}]	NaN	tt2637294	en	Hot Tub Tir Machine
1	2	[{'id': 107674, 'name': 'The Princess Diaries ...	40000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	tt0368933	en	The Prince Diaries 2: Roy Engageme

2 rows × 23 columns

```
#fetch data from bottom
df.tail(2)
```

	id	belongs_to_collection	budget	genres	homepage	imdb_id	origina
2998	2999	NaN	42000000	[{'id': 35, 'name': 'Comedy'}, {'id': 10749, '...	http://www.alongcamepolly.com/	tt0343135	
2999	3000	NaN	35000000	[{'id': 53, 'name': 'Thriller'}, {'id': 28, 'n...	http://www.abductionthefilm.com/	tt1600195	

2 rows × 23 columns

```
#fetch data point randomly
df.sample(10)
```

	id	belongs_to_collection	budget	genres	homepage
267	268	[{'id': 182813, 'name': 'Clerks Collection', '...	27000	[{'id': 35, 'name': 'Comedy'}]	http://www.miramax.com/movie/clerks/
2593	2594	[{'id': 937, 'name': 'The Pink Panther (Origin...	25000000	[{'id': 35, 'name': 'Comedy'}]	NaN
75	76	[{'id': 230161, 'name': 'Fright Night (Reboot)...	17000000	[{'id': 27, 'name': 'Horror'}, {'id': 35, 'nam...	http://www.welcometofrightnight.com
1701	1702	NaN	0	[{'id': 99, 'name': 'Documentary'}]	NaN
1699	1700	NaN	20000000	[{'id': 53, 'name': 'Thriller'}, {'id': 18, 'n...	http://www.disturbia.com/
332	333	NaN	5000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	http://www.greenestreetfilms.com/f_bill.html
2182	2183	NaN	70000000	[{'id': 18, 'name': 'Drama'}, {'id': 10752, 'n...	NaN
1770	1771	NaN	0	[{'id': 18, 'name': 'Drama'}, {'id': 14, 'name...	NaN
786	787	[{'id': 437451, 'name': 'Cinderella Story Coll...	19000000	[{'id': 35, 'name': 'Comedy'}]	http://www2.warnerbros.com/acinderellastory/in...
		Collec...		Comedy'}, {'id': 10751, '...	

10 rows × 23 columns

```
#find the information of the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 23 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   id                     3000 non-null   int64 
 1   belongs_to_collection  604 non-null    object
 2   budget                 3000 non-null   int64 
 3   genres                 2993 non-null   object
 4   homepage               946 non-null    object
 5   imdb_id                3000 non-null   object
 6   original_language      3000 non-null   object
 7   original_title         3000 non-null   object
 8   overview               2992 non-null   object 
 9   popularity             3000 non-null   float64
 10  poster_path            2999 non-null   object 
 11  production_companies   2844 non-null   object 
 12  production_countries   2945 non-null   object 
 13  release_date           3000 non-null   object 
 14  runtime                2998 non-null   float64
 15  spoken_languages       2980 non-null   object 
 16  status                 3000 non-null   object 
 17  tagline                2403 non-null   object 
 18  title                  3000 non-null   object 
 19  Keywords               2724 non-null   object 
 20  cast                   2987 non-null   object 
 21  crew                   2984 non-null   object 
 22  revenue                3000 non-null   int64  
dtypes: float64(2), int64(3), object(18)
memory usage: 539.2+ KB
```

```
#How does the data looklike mathematically
df.describe()
```

	id	budget	popularity	runtime	revenue
count	3000.000000	3.000000e+03	3000.000000	2998.000000	3.000000e+03
mean	1500.500000	2.253133e+07	8.463274	107.856571	6.672585e+07
std	866.169729	3.702609e+07	12.104000	22.086434	1.375323e+08
min	1.000000	0.000000e+00	0.000001	0.000000	1.000000e+00
25%	750.750000	0.000000e+00	4.018053	94.000000	2.379808e+06
50%	1500.500000	8.000000e+06	7.374861	104.000000	1.680707e+07
75%	2250.250000	2.900000e+07	10.890983	118.000000	6.891920e+07
max	3000.000000	3.800000e+08	294.337037	338.000000	1.519558e+09

```
#find the number of columns
df.columns
```

```
Index(['id', 'belongs_to_collection', 'budget', 'genres', 'homepage',
       'imdb_id', 'original_language', 'original_title', 'overview',
       'popularity', 'poster_path', 'production_companies',
       'production_countries', 'release_date', 'runtime', 'spoken_languages',
       'status', 'tagline', 'title', 'Keywords', 'cast', 'crew', 'revenue'],
      dtype='object')
```

```
#find the duplicate value
print(df.duplicated().sum())
```

```
0
```

```
#find the null value
df.isnull().sum()
```

	0
id	0
belongs_to_collection	2396
budget	0
genres	7
homepage	2054
imdb_id	0
original_language	0
original_title	0
overview	8
popularity	0
poster_path	1
production_companies	156
production_countries	55
release_date	0
runtime	2
spoken_languages	20
status	0
tagline	597
title	0
Keywords	276
cast	13
crew	16
revenue	0

dtype: int64

```
df.columns
```

```
Index(['id', 'belongs_to_collection', 'budget', 'genres', 'homepage',
       'imdb_id', 'original_language', 'original_title', 'overview',
       'popularity', 'poster_path', 'production_companies',
       'production_countries', 'release_date', 'runtime', 'spoken_languages',
       'status', 'tagline', 'title', 'Keywords', 'cast', 'crew', 'revenue'],
      dtype='object')
```

```
#indexing and slicing
df["title"]
```

	title
0	Hot Tub Time Machine 2
1	The Princess Diaries 2: Royal Engagement
2	Whiplash
3	Kahaani
4	Marine Boy
...	...
2995	Chasers
2996	We Are the Best!
2997	The Long Kiss Goodnight
2998	Along Came Polly
2999	Abduction

3000 rows × 1 columns

dtype: object

```
df[["title", "budget","revenue", "original_language"]]
```

	title	budget	revenue	original_language
0	Hot Tub Time Machine 2	14000000	12314651	en
1	The Princess Diaries 2: Royal Engagement	40000000	95149435	en
2	Whiplash	3300000	13092000	en
3	Kahaani	1200000	16000000	hi
4	Marine Boy	0	3923970	ko
...	...	...	...	...
2995	Chasers	0	1596687	en
2996	We Are the Best!	0	180590	sv
2997	The Long Kiss Goodnight	65000000	89456761	en
2998	Along Came Polly	42000000	171963386	en
2999	Abduction	35000000	82087155	en

3000 rows × 4 columns

```
df[0:4]
```

	id	belongs_to_collection	budget	genres	homepage	imdb_id	original_lan
0	1	[{'id': 313576, 'name': 'Hot Tub Time Machine ...	14000000	[{'id': 35, 'name': 'Comedy'}]	NaN	tt2637294	
1	2	[{'id': 107674, 'name': 'The Princess Diaries ...	40000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	tt0368933	
2	3	NaN	3300000	[{'id': 18, 'name': 'Drama'}]	http://sonyclassics.com/whiplash/	tt2582802	
3	4	NaN	1200000	[{'id': 53, 'name': 'Thriller'}, {'id': 18, 'n...	http://kahaanithefilm.com/	tt1821480	

4 rows × 23 columns

```
df.iloc[0:4,0:3]
```

	id	belongs_to_collection	budget
0	1	[{'id': 313576, 'name': 'Hot Tub Time Machine ...	14000000
1	2	[{'id': 107674, 'name': 'The Princess Diaries ...	40000000
2	3	NaN	3300000
3	4	NaN	1200000

```
df.loc[0:4,["title","budget","revenue"]]
```

	title	budget	revenue
0	Hot Tub Time Machine 2	14000000	12314651
1	The Princess Diaries 2: Royal Engagement	40000000	95149435
2	Whiplash	3300000	13092000
3	Kahaani	1200000	16000000
4	Marine Boy	0	3923970

```
df["original_language"].unique()
```

```
array(['en', 'hi', 'ko', 'sr', 'fr', 'it', 'nl', 'zh', 'es', 'cs', 'ta',
       'cn', 'ru', 'tr', 'ja', 'fa', 'sv', 'de', 'te', 'pt', 'mr', 'da',
       'fi', 'el', 'ur', 'he', 'no', 'ar', 'nb', 'ro', 'vi', 'pl', 'hu',
       'ml', 'bn', 'id'], dtype=object)
```

```
hindi_movie = df[df["original_language"]=="hi"]
hindi_movie
```

	id	belongs_to_collection	budget	genres	homepag
3	4	NaN	1200000	[{'id': 53, 'name': 'Thriller'}, {'id': 18, 'n...	http://kahaanithefilm.com
213	214	NaN	6700000	[{'id': 28, 'name': 'Action'}]	NaN
371	372	NaN	13100000	[{'id': 18, 'name': 'Drama'}, {'id': 35, 'name...	http://www.thankyouthefilm.com
401	402	NaN	3000000	[{'id': 18, 'name': 'Drama'}, {'id': 36, 'name...	NaN
419	420	NaN	3750000	[{'id': 35, 'name': 'Comedy'}, {'id': 10769, '...	http://www.atithithefilm.com
481	482	NaN	10500000	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	http://www.cc2c-thefilm.com
585	586	NaN	10400000	[{'id': 18, 'name': 'Drama'}]	NaN
627	628	NaN	16000000	[{'id': 28, 'name': 'Action'}, {'id': 18, 'nam...	http://www.yashrajfilms.com/Movies/MovieIndivi...
671	672	NaN	6400000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	http://www.anjaanaanjaani.erosentertainment.com
				[{'id': 10749 ... {'id': 53, ...	
771	772	NaN	9000000	[{'id': 28, 'name': 'Action'},	http://kaminey.utvnet.com
848	849	NaN	6400000	'Drama'}, {'id': 10749,	NaN

```
hindi_movie.to_csv("hindi_movie.csv")
```

Linear regression

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

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

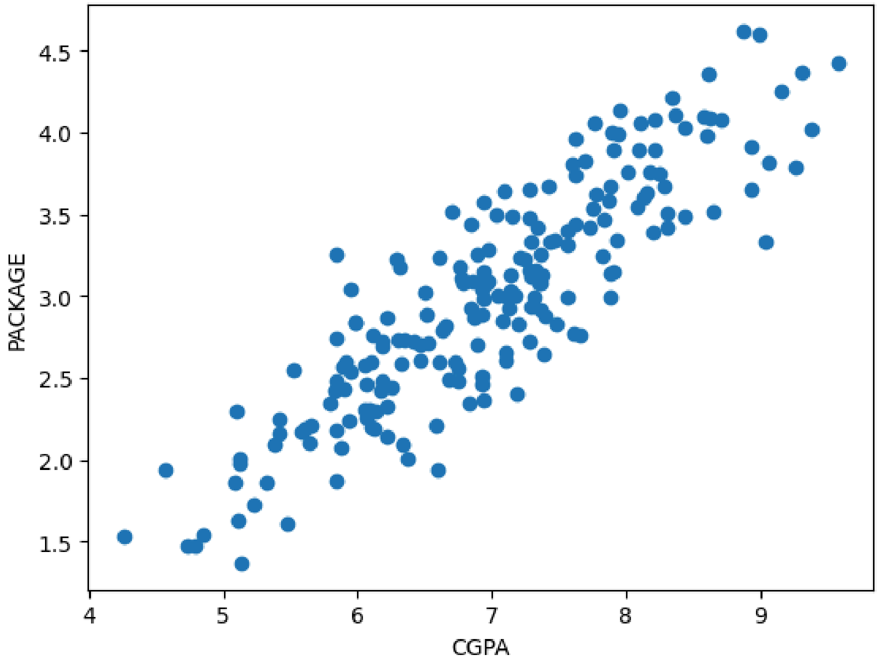
	cgpa	package
0	6.89	3.26
1	5.12	1.98
2	7.82	3.25
3	7.42	3.67
4	6.94	3.57
...	...	...
195	6.93	2.46
196	5.89	2.57
197	7.21	3.24
198	7.63	3.96
199	6.22	2.33

200 rows × 2 columns

```
df.columns
```

```
Index(['cgpa', 'package'], dtype='object')
```

```
plt.scatter(df["cgpa"],df["package"])
plt.xlabel("CGPA")
plt.ylabel("PACKAGE")
plt.show()
```



```
#find the null value
df.isnull().sum()
```

	0
cgpa	0
package	0

dtype: int64

```
#find the independent and dependent feature
x = df.iloc[:,0]
y = df.iloc[:,1]
x
```

```
     cgpa
0    6.89
1    5.12
2    7.82
3    7.42
4    6.94
...   ...
195  6.93
196  5.89
197  7.21
198  7.63
199  6.22

200 rows × 1 columns

dtype: float64
```

```
      [{'id': 44976, 'name':   19500000  [{'id': 28, 'name': 'Action'}, {'id': 80, 'nam...                  NaN
      'Dhoom Series', 'poster...
2251  2252          NaN      130  [{'id': 18, 'name': 'Drama'}, {'id': 28, 'name...                    NaN
                    NaN 18000000  [{'id': 80, 'name': 'Crime'}, {'id': 18, 'name...                    NaN
2353  2354          NaN  8376800  [{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...  http://www.jodhaaakbar.com
                    NaN  7500000  [{'id': 35, 'name': 'Comedy'}, {'id': 18,   http://www.khattameethathefilm.com
```

```
y
```

```
     package
0       3.26
1       1.98
2       3.25
3       3.67
4       3.57
...      ...
195     2.46
196     2.57
197     3.24
198     3.96
199     2.33

200 rows × 1 columns

dtype: float64
```

```
2406  2407  NaN  4300000  [{'id': 18, 'name': 'Drama'}, {'id': 10749, 'n...      NaN
2480  2481  NaN  6400000  [{'id': 10749, 'name': 'Romance'}, {'id': 18, ...    NaN
            NaN  6900000  [{'id': 10749, 'name': 'Romance'}, {'id': 18, ...    NaN
2563  2564  NaN  7400000  [{'id': 18, 'name': 'Drama'}, {'id': 10749, 'n...      NaN
2567  2568  NaN 11000000  [{'id': 18, 'name': 'Drama'}, {'id': 28,              NaN
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,
                                                  random_state = 42 )
x_train
```

```
2828  2829  NaN  19500000  [{'id': 10752, 'name': 'War'}, {'id': 36, 'nam...  NaN
```

	cgpa
79	7.18
197	7.21
38	8.62
24	6.53
122	5.12
...	...
106	6.13
14	7.73
92	7.90

2857	2858	NaN	5400000	[{'id': 53, 'name': 'Thriller'}]	NaN
2859	2860	[{'id': 286951, 'name': 'Singham Collection', ...	3062000	[{'id': 18, 'name': 'Drama'}, {'id': 28, 'name...	NaN
[illegible]	[illegible]	NaN	2300000	[{'id': 18, 'name': 'Drama'}, {'id': 35, 'name...	NaN

columns