In [1]:
```
print("Hello World")
```

```
Hello World
```

In [2]:
```
name=input("Enter ur name:")
print("Welcome",name)
```

```
Enter ur name:James
Welcome James
```

In [3]:
```
print(type(5))
```

```
<class 'int'>
```

In [4]:
```
print(type(2.5))
```

```
<class 'float'>
```

In [5]:
```
print(type("Hello World"))
```

```
<class 'str'>
```

In [6]:
```
print(type(False))
```

```
<class 'bool'>
```

In [7]:
```
x = 1
print(x)
```

```
1
```

In [8]:
```
name = "James"
print(name)
```

```
James
```

In [9]:
```
print(5+3)
```

```
8
```

In [10]:
```
print(6.5-2)
```

```
4.5
```

In [11]:
```
print(2*4)
```

```
8
```

In [12]:
```
print(5/2)
```

```
2.5
```

In [13]:
```
print(5//2)
```

```
2
```

In [14]:
```
print(5%2)
```

```
1
```

In [15]:
```
print(20>3)
```

```
True
```

In [16]:
```
print(100<7)
```

```
False
```

In [17]:
```
print((20>3) and (100<7))
```

```
False
```

In [18]:
```
print((20>3) or (100<7))
```

```
True
```

In [19]:
```
x=True
print(x)
print(not x)
```

```
True
False
```

In [20]:
```
if 5==6:
    print(1)
elif 6==6:
    print(2)
else:
    print(3)
```

```
2
```

In [21]:
```
if 7>3 and 10<5:
    print(1)
elif 7>3 and 5<10:
    print(2)
elif 3>7 and 10<5:
    print(3)
elif 3>7 and 5<10:
    print(4)
else:
    print(5)
```

```
2
```

In [22]:
```
#Print all the even numbers from 0 to 20.
for i in range(0,21,2):
    print(i)
```

```
0
2
4
6
8
10
12
14
16
18
20
```

In [23]:
```
#By default the step value is 1 so if we omi the step value then it will prin
for i in range(0,21):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

In [24]:
```
while True:
    username=input("Enter the username:")
    if username == "James":
        print("Welcome James")
        break
    else:
        print("wrong username")
```

```
Enter the username:Anderson
wrong username
Enter the username:James
Welcome James
```

In [25]:
```
i=1
while i<=10:
    print(i)
    i=i+1
```

```
1
2
3
4
5
6
7
8
9
10
```

In [26]:
```
flowersList = ["Rose","Lily","Tulip"]
firstelement = flowersList[0]
print(firstelement)
```

```
Rose
```

In [27]:
```
lasttwoelements= flowersList[1:3]
print(lasttwoelements)
```

```
['Lily', 'Tulip']
```

In [29]:
```
for i in flowersList:
    print(i)
```

```
Rose
Lily
Tulip
```

In [30]:
```
data = {"Name":"James","Age":27,"Country":"US"}
name = data["Name"]
age= data["Age"]
country = data["Country"]

print(name)
print(age)
print(country)
```

```
James
27
US
```

In [32]:
```
data = {"Name":"James","Age":27,"Country":"US"}
for key in data:
    print(data[key])
```

```
James
27
US
```

In [34]:
```
flowertuple=("Rose","Lily","Tulip")
for item in flowertuple:
    print(item)
```

```
Rose
Lily
Tulip
```

In [35]:
```
data = "Rose",
print(type(data))
```

```
<class 'tuple'>
```

In [36]:
```
data= 1,2,3,4,5,6,7,8,9,10
print(type(data))
```

```
<class 'tuple'>
```

In [37]:
```
data="Rose","Lily","Tulip"
print(type(data))
```

```
<class 'tuple'>
```

In [38]:
```
def AddNumbers(x,y):
    sum = x+y
    return sum
result = AddNumbers(5,10)
print(result)
```

```
15
```

In [39]:
```
def IncrementByOne(x,y):
    x = x+1;
    y=  y+1;
    return x,y

num1, num2=IncrementByOne(8,9)
print(num1)
print(num2)
```

```
9
10
```

In [1]:
```
#Creating a 1-D NumPy Array
import numpy as np
```

In [2]:
```
oneDarray=np.array([1,2,3,4,5])
print(oneDarray)
```

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

In [3]:
```
print(oneDarray.ndim)
```

```
1
```

In [5]:
```
#Creating a 2-D Numpy array
twoDarray= np.array([[1,2,3],[4,5,6]])
print(twoDarray)
```

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

In [6]:
```
print(twoDarray.ndim)
```

```
2
```

In [14]:
```
#Creating a 3-D Numpy array
threeDarray = np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]])
print(threeDarray)
```

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

 [[ 7  8  9]
  [10 11 12]]]
```

In [8]:
```
print(threeDarray.ndim)
```

```
3
```

In [9]:
```
#Indexing a Numpy array
twoDarray= np.array([[1,2,3],[4,5,6]])
print(twoDarray)
```

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

In [10]:
```
print(twoDarray[1])
```

```
[4 5 6]
```

In [12]:
```
print(twoDarray[1][0])
```

```
4
```

In [13]:
```
print(twoDarray[1,0])
```

```
4
```

In [15]:
```
#Array Shape
twoDarray= np.array([[1,2,3],[4,5,6]])
print(twoDarray.shape)
```

```
(2, 3)
```

In [17]:
```
#Create a Numpy array prefilled with 10 zeros.
zerosArray= np.zeros(10)
print(zerosArray)
```

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

In [18]:
```
zerosArray = zerosArray.astype(int)
print(zerosArray)
```

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

In [19]:
```
#Create a Numpy array prefilled with 10 ones.
onesArray=np.ones(10)
print(onesArray)
```

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

In [20]:
```
onesArray= onesArray.astype(int)
print(onesArray)
```

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

In [16]:
```
#To create an array prefilled with some specific number we can use the .full
prefilledarray= np.full(10,5)
print(prefilledarray)
```

```
[5 5 5 5 5 5 5 5 5 5]
```

In [21]:
```
#Adding a scalar
matrix= np.array([[1,2,3],[4,5,6]])
print(matrix)
```

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

In [22]:
```
print(matrix+2)
```

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

In [23]:
```
print(matrix-2)
```

```
[[-1  0  1]
 [ 2  3  4]]
```

In [24]:
```
print(matrix)
```

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

In [25]:
```
print(matrix*2)
```

```
[[ 2  4  6]
 [ 8 10 12]]
```

In [26]:
```
print(matrix/2)
```

```
[[0.5 1.  1.5]
 [2.  2.5 3. ]]
```

In [27]:
```
print(matrix//2)
```

```
[[0 1 1]
 [2 2 3]]
```

In [29]:
```
print(matrix**2)
```

```
[[ 1  4  9]
 [16 25 36]]
```

In [30]:
```
print(matrix)
```

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

In [31]:
```
print(matrix.T)
```

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

In [32]:
```
matrix1=np.array([[1,2,3],[4,5,6]])
print(matrix1)
```

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

In [33]:
```
matrix2=np.array([[7,8,9],[10,11,12]])
print(matrix2)
```

```
[[ 7  8  9]
 [10 11 12]]
```

In [34]:
```
print(matrix1+matrix2)
```

```
[[ 8 10 12]
 [14 16 18]]
```

In [35]:
```
print(matrix1-matrix2)
```

```
[[-6 -6 -6]
 [-6 -6 -6]]
```

In [36]:
```
print(matrix1*matrix2)
```

```
[[ 7 16 27]
 [40 55 72]]
```

In [37]:
```
print(matrix1/matrix2)
```

```
[[0.14285714 0.25       0.33333333]
 [0.4        0.45454545 0.5       ]]
```

In [38]:
```
print(matrix1//matrix2)
```

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

In [39]:
```
matrix1=np.array([[1,2,3],[4,5,6],[0,0,0]])
print(matrix1)
```

```
[[1 2 3]
 [4 5 6]
 [0 0 0]]
```

In [40]:
```
matrix2=np.array([[-1,-2,-3],[-4,-5,-6],[0,0,0]])
print(matrix2)
```

```
[[-1 -2 -3]
 [-4 -5 -6]
 [ 0  0  0]]
```

In [41]:
```
np.matmul(matrix1,matrix2)
```

Out[41]:
```
array([[ -9, -12, -15],
       [-24, -33, -42],
       [  0,   0,   0]])
```

In [42]:
```
mat1=np.array([[1,2,3],[4,5,6]])
print(mat1)
```

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

In [44]:
```
print(np.min(mat1))
```

```
1
```

In [45]:
```
print(np.max(mat1))
```

```
6
```

In [46]:
```
print(np.sum(mat1))
```

```
21
```

In [47]:
```
print(np.mean(mat1))
```

```
3.5
```

In [48]:
```
print(np.std(mat1))
```

```
1.707825127659933
```

In [49]:
```
print(np.median(mat1))
```

```
3.5
```

In [50]:
```
import pandas as pd
myList=[['Apple','Red'],['Banana','Yellow'],['Litchi','Pink']]
myDataFrame=pd.DataFrame(myList)
print(myDataFrame)
```

```
        0       1
0   Apple     Red
1  Banana  Yellow
2  Litchi    Pink
```

In [51]:
```
myList=[['Apple','Red'],['Banana','Yellow'],['Litchi','Pink']]
myDataFrame=pd.DataFrame(myList,columns=['Fruit','Color'])
print(myDataFrame)
```

```
    Fruit   Color
0   Apple     Red
1  Banana  Yellow
2  Litchi    Pink
```

In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv("Top YouTube Channels Data .csv")
df
```

Out[2]:

	rank	youtuber	subscribers	video views	video count	category	started
0	1	T-Series	213000000	188,073,919,029	16708.0	Music	2006
1	2	YouTube Movies	150000000	167,122,746,349	NaN	Film & Animation	2015
2	3	Cocomelon - Nursery Rhymes	133000000	126,822,520,940	751.0	Education	2006
3	4	SET India	131000000	101,541,977,714	78334.0	Shows	2006
4	5	Music	116000000	78,437,871,689	NaN	Music	2013
...	...	...	...	...	...	...	...
95	96	Markiplier	32600000	18,011,837,263	5129.0	Gaming	2012
96	97	Like Nastya ESP	32600000	15,144,858,210	584.0	Entertainment	2017
97	98	Ryan's World	32400000	51,312,603,726	2155.0	Entertainment	2015
98	99	ABP News	32300000	9,850,740,503	209351.0	People & Blogs	2012
99	100	Desi Music Factory	32200000	9,115,577,588	122.0	Music	2014

100 rows × 7 columns

In [5]:
```
df.set_index('youtuber')
```

Out[5]:

	rank	subscribers	video views	video count	category	started
youtuber						
T-Series	1	213000000	188,073,919,029	16708.0	Music	2006
YouTube Movies	2	150000000	167,122,746,349	NaN	Film & Animation	2015
Cocomelon - Nursery Rhymes	3	133000000	126,822,520,940	751.0	Education	2006
SET India	4	131000000	101,541,977,714	78334.0	Shows	2006
Music	5	116000000	78,437,871,689	NaN	Music	2013
...	...	...	...	...	...	...
Markiplier	96	32600000	18,011,837,263	5129.0	Gaming	2012
Like Nastya ESP	97	32600000	15,144,858,210	584.0	Entertainment	2017
Ryan's World	98	32400000	51,312,603,726	2155.0	Entertainment	2015
ABP News	99	32300000	9,850,740,503	209351.0	People & Blogs	2012
Desi Music Factory	100	32200000	9,115,577,588	122.0	Music	2014

100 rows × 6 columns

In [6]:
```
df
```

Out[6]:

	rank	youtuber	subscribers	video views	video count	category	started
0	1	T-Series	213000000	188,073,919,029	16708.0	Music	2006
1	2	YouTube Movies	150000000	167,122,746,349	NaN	Film & Animation	2015
2	3	Cocomelon - Nursery Rhymes	133000000	126,822,520,940	751.0	Education	2006
3	4	SET India	131000000	101,541,977,714	78334.0	Shows	2006
4	5	Music	116000000	78,437,871,689	NaN	Music	2013
...	...	...	...	...	...	...	...
95	96	Markiplier	32600000	18,011,837,263	5129.0	Gaming	2012
96	97	Like Nastya ESP	32600000	15,144,858,210	584.0	Entertainment	2017
97	98	Ryan's World	32400000	51,312,603,726	2155.0	Entertainment	2015
98	99	ABP News	32300000	9,850,740,503	209351.0	People & Blogs	2012
99	100	Desi Music Factory	32200000	9,115,577,588	122.0	Music	2014

100 rows × 7 columns

In [7]:
```
df.head()
```

Out[7]:

	rank	youtuber	subscribers	video views	video count	category	started
0	1	T-Series	213000000	188,073,919,029	16708.0	Music	2006
1	2	YouTube Movies	150000000	167,122,746,349	NaN	Film & Animation	2015
2	3	Cocomelon - Nursery Rhymes	133000000	126,822,520,940	751.0	Education	2006
3	4	SET India	131000000	101,541,977,714	78334.0	Shows	2006
4	5	Music	116000000	78,437,871,689	NaN	Music	2013

In [8]:
```
df.tail()
```

Out[8]:

	rank	youtuber	subscribers	video views	video count	category	started
95	96	Markiplier	32600000	18,011,837,263	5129.0	Gaming	2012
96	97	Like Nastya ESP	32600000	15,144,858,210	584.0	Entertainment	2017
97	98	Ryan's World	32400000	51,312,603,726	2155.0	Entertainment	2015
98	99	ABP News	32300000	9,850,740,503	209351.0	People & Blogs	2012
99	100	Desi Music Factory	32200000	9,115,577,588	122.0	Music	2014

In [9]:
```
#Find the statistical summaryn of the dataframe
df.describe()
```

Out[9]:

	rank	subscribers	video count	started
count	100.000000	1.000000e+02	95.000000	100.000000
mean	50.500000	5.336300e+07	15847.221053	2010.800000
std	29.011492	2.869713e+07	40955.200388	5.504819
min	1.000000	3.220000e+07	45.000000	1970.000000
25%	25.750000	3.620000e+07	393.500000	2007.750000
50%	50.500000	4.320000e+07	1139.000000	2012.000000
75%	75.250000	5.710000e+07	4986.000000	2014.000000
max	100.000000	2.130000e+08	209351.000000	2018.000000

In [11]:
```
#get the data of the first 4 rows(slicing rows using bracket operators)
df[0:4]
```

Out[11]:

	rank	youtuber	subscribers	video views	video count	category	started
0	1	T-Series	213000000	188,073,919,029	16708.0	Music	2006
1	2	YouTube Movies	150000000	167,122,746,349	NaN	Film & Animation	2015
2	3	Cocomelon - Nursery Rhymes	133000000	126,822,520,940	751.0	Education	2006
3	4	SET India	131000000	101,541,977,714	78334.0	Shows	2006

In [16]:
```
#Indexing columns using bracket operators.
df[['youtuber']]
```

Out[16]:

	youtuber
0	T-Series
1	YouTube Movies
2	Cocomelon - Nursery Rhymes
3	SET India
4	Music
...	...
95	Markiplier
96	Like Nastya ESP
97	Ryan's World
98	ABP News
99	Desi Music Factory

100 rows × 1 columns

In [23]:
```
#filtering rows
condition= df[df['started']>2007]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3621, in Index.get_loc(self, key, method, tolerance)
   3620 try:
-> 3621     return self._engine.get_loc(casted_key)
   3622 except KeyError as err:

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:136, in pandas._libs.index.IndexEngine.get_loc()

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:163, in pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5198, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5206, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'started'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Input In [23], in <cell line: 2>()
      1 #filtering rows
----> 2 condition= df[df['started']>2007]

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:3505, in DataFrame.__getitem__(self, key)
   3503 if self.columns.nlevels > 1:
   3504     return self._getitem_multilevel(key)
-> 3505 indexer = self.columns.get_loc(key)
   3506 if is_integer(indexer):
   3507     indexer = [indexer]

File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3623, in Index.get_loc(self, key, method, tolerance)
   3621     return self._engine.get_loc(casted_key)
   3622 except KeyError as err:
-> 3623     raise KeyError(key) from err
   3624 except TypeError:
   3625     # If we have a listlike key, _check_indexing_error will raise
   3626     #  InvalidIndexError. Otherwise we fall through and re-raise
   3627     #  the TypeError.
   3628     self._check_indexing_error(key)

KeyError: 'started'
```

In [24]:
```
df
```

Out[24]:

	rank	youtuber	subscribers	video views	video count	category	started
0	1	T-Series	213000000	188,073,919,029	16708.0	Music	2006
1	2	YouTube Movies	150000000	167,122,746,349	NaN	Film & Animation	2015
2	3	Cocomelon - Nursery Rhymes	133000000	126,822,520,940	751.0	Education	2006
3	4	SET India	131000000	101,541,977,714	78334.0	Shows	2006
4	5	Music	116000000	78,437,871,689	NaN	Music	2013
...	...	...	...	...	...	...	...
95	96	Markiplier	32600000	18,011,837,263	5129.0	Gaming	2012
96	97	Like Nastya ESP	32600000	15,144,858,210	584.0	Entertainment	2017
97	98	Ryan's World	32400000	51,312,603,726	2155.0	Entertainment	2015
98	99	ABP News	32300000	9,850,740,503	209351.0	People & Blogs	2012
99	100	Desi Music Factory	32200000	9,115,577,588	122.0	Music	2014

100 rows × 7 columns

In [25]:
```
df[df['started']>2005]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3621, in Index.get_loc(self, key, method, tolerance)
   3620 try:
-> 3621     return self._engine.get_loc(casted_key)
   3622 except KeyError as err:

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:136, in pandas._libs.index.IndexEngine.get_loc()

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:163, in pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5198, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5206, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'started'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Input In [25], in <cell line: 1>()
----> 1 df[df['started']>2005]

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:3505, in DataFrame.__getitem__(self, key)
   3503 if self.columns.nlevels > 1:
   3504     return self._getitem_multilevel(key)
-> 3505 indexer = self.columns.get_loc(key)
   3506 if is_integer(indexer):
   3507     indexer = [indexer]

File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3623, in Index.get_loc(self, key, method, tolerance)
   3621     return self._engine.get_loc(casted_key)
   3622 except KeyError as err:
-> 3623     raise KeyError(key) from err
   3624 except TypeError:
   3625     # If we have a listlike key, _check_indexing_error will raise
   3626     #  InvalidIndexError. Otherwise we fall through and re-raise
   3627     #  the TypeError.
   3628     self._check_indexing_error(key)

KeyError: 'started'
```

In [26]:
```
df
```

Out[26]:

	rank	youtuber	subscribers	video views	video count	category	started
0	1	T-Series	213000000	188,073,919,029	16708.0	Music	2006
1	2	YouTube Movies	150000000	167,122,746,349	NaN	Film & Animation	2015
2	3	Cocomelon - Nursery Rhymes	133000000	126,822,520,940	751.0	Education	2006
3	4	SET India	131000000	101,541,977,714	78334.0	Shows	2006
4	5	Music	116000000	78,437,871,689	NaN	Music	2013
...	...	...	...	...	...	...	...
95	96	Markiplier	32600000	18,011,837,263	5129.0	Gaming	2012
96	97	Like Nastya ESP	32600000	15,144,858,210	584.0	Entertainment	2017
97	98	Ryan's World	32400000	51,312,603,726	2155.0	Entertainment	2015
98	99	ABP News	32300000	9,850,740,503	209351.0	People & Blogs	2012
99	100	Desi Music Factory	32200000	9,115,577,588	122.0	Music	2014

100 rows × 7 columns

In [27]:
```
df.loc[0,'youtuber']
```

Out[27]:
```
'T-Series '
```

In [28]:
```
df.loc[[0],['youtuber']]
```

Out[28]:

	youtuber
0	T-Series

In [32]:
```
#slice first 5 rows and all columns
df.loc[0:5,'youtuber':]
```

Out[32]:

	youtuber	subscribers	video views	video count	category	started
0	T-Series	213000000	188,073,919,029	16708.0	Music	2006
1	YouTube Movies	150000000	167,122,746,349	NaN	Film & Animation	2015
2	Cocomelon - Nursery Rhymes	133000000	126,822,520,940	751.0	Education	2006
3	SET India	131000000	101,541,977,714	78334.0	Shows	2006
4	Music	116000000	78,437,871,689	NaN	Music	2013
5	PewDiePie	111000000	28,260,779,633	4472.0	Gaming	2010

In [34]:
```
#filtering data using iloc
df.iloc[3,5]
```

Out[34]:
```
'Shows '
```

In [35]:
```
df.iloc[0:5,0:3]
```

Out[35]:

	rank	youtuber	subscribers
0	1	T-Series	213000000
1	2	YouTube Movies	150000000
2	3	Cocomelon - Nursery Rhymes	133000000
3	4	SET India	131000000
4	5	Music	116000000

In [40]:
```
#Adding and deleting rows and columns
df.loc[7]=[8,'Aman',783678908,120,89939.0,'Music',2022]
```

In [41]:
```
df.head(8)
```

Out[41]:

	rank	youtuber	subscribers	video views	video count	category	started
0	1	T-Series	213000000	188,073,919,029	16708.0	Music	2006
1	2	YouTube Movies	150000000	167,122,746,349	NaN	Film & Animation	2015
2	3	Cocomelon - Nursery Rhymes	133000000	126,822,520,940	751.0	Education	2006
3	4	SET India	131000000	101,541,977,714	78334.0	Shows	2006
4	5	Music	116000000	78,437,871,689	NaN	Music	2013
5	6	PewDiePie	111000000	28,260,779,633	4472.0	Gaming	2010
6	7	MrBeast	93900000	15,417,304,461	721.0	Entertainment	2012
7	8	Aman	783678908	120	89939.0	Music	2022

In [42]:
```
df.drop(7,axis=0,inplace=True)
df
```

Out[42]:

	rank	youtuber	subscribers	video views	video count	category	started
0	1	T-Series	213000000	188,073,919,029	16708.0	Music	2006
1	2	YouTube Movies	150000000	167,122,746,349	NaN	Film & Animation	2015
2	3	Cocomelon - Nursery Rhymes	133000000	126,822,520,940	751.0	Education	2006
3	4	SET India	131000000	101,541,977,714	78334.0	Shows	2006
4	5	Music	116000000	78,437,871,689	NaN	Music	2013
...	...	...	...	...	...	...	...
95	96	Markiplier	32600000	18,011,837,263	5129.0	Gaming	2012
96	97	Like Nastya ESP	32600000	15,144,858,210	584.0	Entertainment	2017
97	98	Ryan's World	32400000	51,312,603,726	2155.0	Entertainment	2015
98	99	ABP News	32300000	9,850,740,503	209351.0	People & Blogs	2012
99	100	Desi Music Factory	32200000	9,115,577,588	122.0	Music	2014

99 rows × 7 columns

In [43]:
```
df.head(8)
```

Out[43]:

	rank	youtuber	subscribers	video views	video count	category	started
0	1	T-Series	213000000	188,073,919,029	16708.0	Music	2006
1	2	YouTube Movies	150000000	167,122,746,349	NaN	Film & Animation	2015
2	3	Cocomelon - Nursery Rhymes	133000000	126,822,520,940	751.0	Education	2006
3	4	SET India	131000000	101,541,977,714	78334.0	Shows	2006
4	5	Music	116000000	78,437,871,689	NaN	Music	2013
5	6	PewDiePie	111000000	28,260,779,633	4472.0	Gaming	2010
6	7	MrBeast	93900000	15,417,304,461	721.0	Entertainment	2012
8	9	Gaming	92100000	71,692,471,446	NaN	Gaming	2013

In [45]:
```
df=df[0:5]
df
```

Out[45]:

	rank	youtuber	subscribers	video views	video count	category	started
0	1	T-Series	213000000	188,073,919,029	16708.0	Music	2006
1	2	YouTube Movies	150000000	167,122,746,349	NaN	Film & Animation	2015
2	3	Cocomelon - Nursery Rhymes	133000000	126,822,520,940	751.0	Education	2006
3	4	SET India	131000000	101,541,977,714	78334.0	Shows	2006
4	5	Music	116000000	78,437,871,689	NaN	Music	2013

In [48]:
```
df['My_column'] =["A","B","C","D","E"]
df
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_12124\3131067220.py:1: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  df['My_column'] =["A","B","C","D","E"]
```

Out[48]:

	rank	youtuber	subscribers	video views	video count	category	started	My_column
0	1	T-Series	213000000	188,073,919,029	16708.0	Music	2006	A
1	2	YouTube Movies	150000000	167,122,746,349	NaN	Film & Animation	2015	B
2	3	Cocomelon - Nursery Rhymes	133000000	126,822,520,940	751.0	Education	2006	C
3	4	SET India	131000000	101,541,977,714	78334.0	Shows	2006	D
4	5	Music	116000000	78,437,871,689	NaN	Music	2013	E

In [51]:
```
df.head()
```

Out[51]:

	rank	youtuber	subscribers	video views	video count	category	started	My_column
0	1	T-Series	213000000	188,073,919,029	16708.0	Music	2006	A
1	2	YouTube Movies	150000000	167,122,746,349	NaN	Film & Animation	2015	B
2	3	Cocomelon - Nursery Rhymes	133000000	126,822,520,940	751.0	Education	2006	C
3	4	SET India	131000000	101,541,977,714	78334.0	Shows	2006	D
4	5	Music	116000000	78,437,871,689	NaN	Music	2013	E

In [52]:
```
df.drop('My_column', axis=1, inplace=True)
df
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_12124\4161074454.py:1: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  df.drop('My_column', axis=1, inplace=True)
```

Out[52]:

	rank	youtuber	subscribers	video views	video count	category	started
0	1	T-Series	213000000	188,073,919,029	16708.0	Music	2006
1	2	YouTube Movies	150000000	167,122,746,349	NaN	Film & Animation	2015
2	3	Cocomelon - Nursery Rhymes	133000000	126,822,520,940	751.0	Education	2006
3	4	SET India	131000000	101,541,977,714	78334.0	Shows	2006
4	5	Music	116000000	78,437,871,689	NaN	Music	2013

In [54]:
```
#sorting values
df.sort_values(by='youtuber')
```

Out[54]:

	rank	youtuber	subscribers	video views	video count	category	started
2	3	Cocomelon - Nursery Rhymes	133000000	126,822,520,940	751.0	Education	2006
4	5	Music	116000000	78,437,871,689	NaN	Music	2013
3	4	SET India	131000000	101,541,977,714	78334.0	Shows	2006
0	1	T-Series	213000000	188,073,919,029	16708.0	Music	2006
1	2	YouTube Movies	150000000	167,122,746,349	NaN	Film & Animation	2015

In [55]:
```
df.to_csv('myfile.csv',index_label= False)
```

In [56]:
```
newdf=pd.read_csv('myfile.csv')
newdf
```

Out[56]:

	rank	youtuber	subscribers	video views	video count	category	started
0	1	T-Series	213000000	188,073,919,029	16708.0	Music	2006
1	2	YouTube Movies	150000000	167,122,746,349	NaN	Film & Animation	2015
2	3	Cocomelon - Nursery Rhymes	133000000	126,822,520,940	751.0	Education	2006
3	4	SET India	131000000	101,541,977,714	78334.0	Shows	2006
4	5	Music	116000000	78,437,871,689	NaN	Music	2013

In [63]:
```
#Concatenating DataFrames
df1=df[0:2]
df1
```

Out[63]:

	rank	youtuber	subscribers	video views	video count	category	started
0	1	T-Series	213000000	188,073,919,029	16708.0	Music	2006
1	2	YouTube Movies	150000000	167,122,746,349	NaN	Film & Animation	2015

In [64]:
```
df2=df[3:5]
df2
```

Out[64]:

	rank	youtuber	subscribers	video views	video count	category	started
3	4	SET India	131000000	101,541,977,714	78334.0	Shows	2006
4	5	Music	116000000	78,437,871,689	NaN	Music	2013

In [65]:
```
pd.concat([df1,df2],axis=1)
```

Out[65]:

	rank	youtuber	subscribers	video views	video count	category	started	rank	youtuber	sub
0	1	T-Series	213000000.0	188,073,919,029	16708.0	Music	2006.0	NaN	NaN	
1	2	YouTube Movies	150000000.0	167,122,746,349	NaN	Film & Animation	2015.0	NaN	NaN	
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4	SET India	131C
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5	Music	116C

In [66]:
```
data={'Gender':['female','male','female','male'],'Score':[85,88,90,95]}
df = pd.DataFrame(data)
df
```

Out[66]:

	Gender	Score
0	female	85
1	male	88
2	female	90
3	male	95

In [68]:
```
df.groupby(df['Gender']).mean()
```

Out[68]:

Gender	Score
female	87.5
male	91.5

In [70]:
```
#Median based anomaly detection
import pandas as pd
import numpy as np
x = pd.Series([2.1,2.2,4.5,2.2,2.4])
median = np.median(x)
threshold = 2
outliers = []
for item in x:
    if abs(item-median)>threshold:
        outliers.append(item)
print(outliers)
```

```
[4.5]
```

In [73]:
```
from numpy import NaN

data1={'Name':["Edward","Edison","James","Aman"],'Age':[28,27,NaN,30]}
da=pd.DataFrame(data1)
da
```

Out[73]:

	Name	Age
0	Edward	28.0
1	Edison	27.0
2	James	NaN
3	Aman	30.0

In [74]:
```
da.isnull()
```

Out[74]:

	Name	Age
0	False	False
1	False	False
2	False	True
3	False	False

In [75]:
```
da.isnull().sum()
```

Out[75]:
```
Name    0
Age     1
dtype: int64
```

In [76]:
```
#Replace the missing value with the mean value of Age
da.fillna(da.mean(),inplace=True)
da
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_12124\3870954151.py:2: FutureWarnin
g: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=
None') is deprecated; in a future version this will raise TypeError.  Select
only valid columns before calling the reduction.
  da.fillna(da.mean(),inplace=True)
```

Out[76]:

	Name	Age
0	Edward	28.000000
1	Edison	27.000000
2	James	28.333333
3	Aman	30.000000

In [77]:
```
da.fillna(da.mode())
da
```

Out[77]:

	Name	Age
0	Edward	28.000000
1	Edison	27.000000
2	James	28.333333
3	Aman	30.000000

In [78]:
```
#Regular Expressions
import re
```

In [79]:
```
txt ="Python is my programming language. I love Python"
x = re.findall("Python",txt)
x
```

Out[79]: ['Python', 'Python']

In [80]:
```
len(x)
```

Out[80]: 2

In [84]:
```
txt= "Python was released in 1991"
number= re.findall('\d',txt)
number
```

Out[84]: ['1', '9', '9', '1']

In [86]:
```
txt= "Python was released in 1991"
number= re.findall('\d+',txt)
number
```

Out[86]: ['1991']

In [87]:
```
import pandas as pd
import re
```

In [88]:
```
txt= "Hello World"
match_object = re.search('World',txt)
match_object
```

Out[88]: <re.Match object; span=(6, 11), match='World'>

In [89]:
```
match_object.span()
```

Out[89]: (6, 11)

In [90]:
```
import pandas as pd
import re
```

In [91]:
```
txt= "C is my favourite programming language"
re.sub(pattern="C", repl="Python",string=txt)
```

Out[91]:
```
'Python is my favourite programming language'
```

In [92]:
```
df1 = {'Age':[28,27,30,36,27],'Salary':[10000,15000,11000,13000,14000]}
df=pd.DataFrame(df1)
df
```

Out[92]:

	Age	Salary
0	28	10000
1	27	15000
2	30	11000
3	36	13000
4	27	14000

In [93]:
```
#Formula for Feature Scaling
df = (df - df.min()) / (df.max() -df.min())
```

In [94]:
```
df
```

Out[94]:

	Age	Salary
0	0.111111	0.0
1	0.000000	1.0
2	0.333333	0.2
3	1.000000	0.6
4	0.000000	0.8

In [95]:
```
df = (df - df.mean()) / df.std()
df
```

Out[95]:

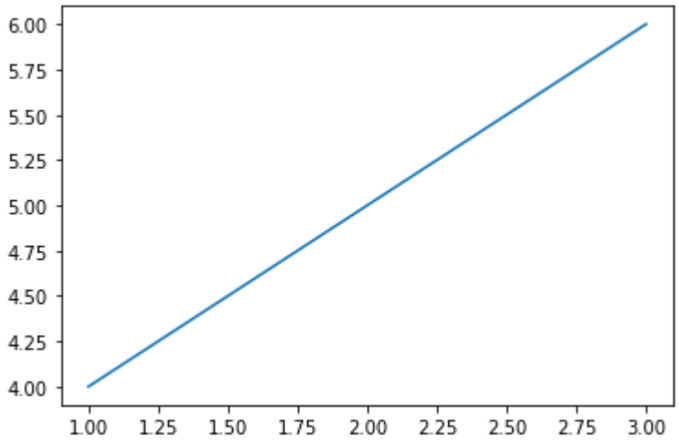
	Age	Salary
0	-0.423109	-1.253831
1	-0.687552	1.157383
2	0.105777	-0.771589
3	1.692435	0.192897
4	-0.687552	0.675140

In [96]:
```
df.std()
```

Out[96]:
```
Age       1.0
Salary    1.0
dtype: float64
```

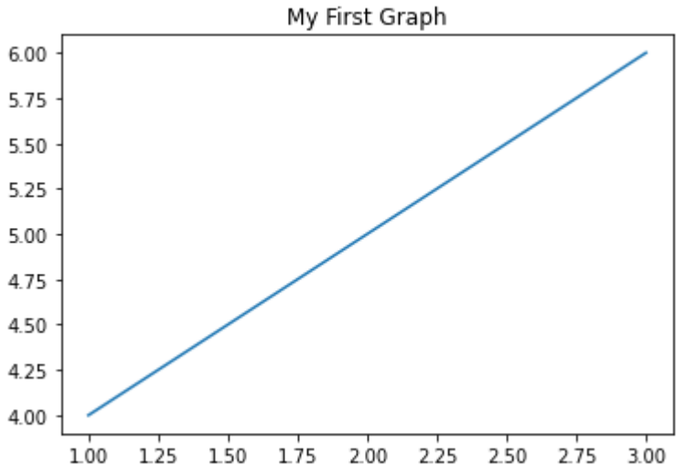
In [97]:
```
#Plotting Line Plots using Matplotlib
import matplotlib.pyplot as plt
x_axis=[1,2,3]
y_axis=[4,5,6]

plt.plot(x_axis,y_axis)
```

Out[97]:
```
[<matplotlib.lines.Line2D at 0x1873b623b20>]
```



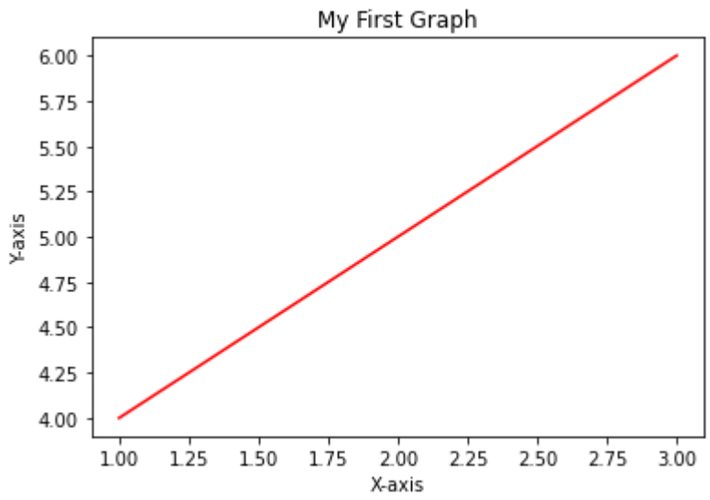
In [98]:
```
x_axis=[1,2,3]
y_axis=[4,5,6]

plt.title("My First Graph")
plt.plot(x_axis,y_axis)
```

Out[98]:
```
[<matplotlib.lines.Line2D at 0x1873d73e3b0>]
```



In [99]:
```
x_axis=[1,2,3]
y_axis=[4,5,6]

plt.title("My First Graph")
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.plot(x_axis,y_axis,'r')
```

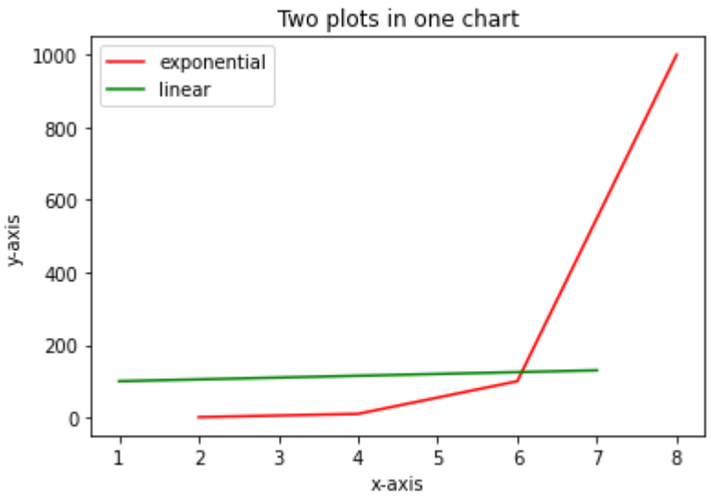
Out[99]: [<matplotlib.lines.Line2D at 0x1873d7af970>]



In [100]:
```
x1_axis=[2,4,6,8]
y1_axis=[1,10,100,1000]

x2_axis=[1,3,5,7]
y2_axis=[100,110,120,130]
```

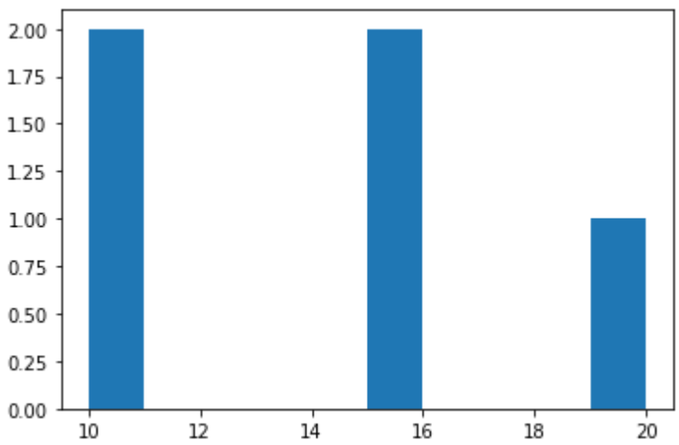
In [102]:
```
plt.title('Two plots in one chart')
plt.xlabel('x-axis')
plt.ylabel('y-axis')

plt.plot(x1_axis,y1_axis,'r')
plt.plot(x2_axis,y2_axis,'g')

plt.legend(['exponential','linear'])
```

Out[102]:
```
<matplotlib.legend.Legend at 0x1873d847790>
```



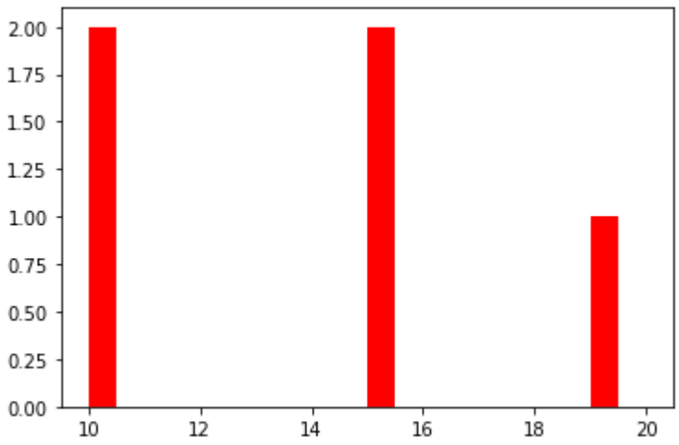
In [103]:
```
#Plotting histograms
values=[10,15,20,10,15]
plt.hist(values)
```

Out[103]:
```
(array([2., 0., 0., 0., 0., 2., 0., 0., 0., 1.]),
 array([10., 11., 12., 13., 14., 15., 16., 17., 18., 19., 20.]),
 <BarContainer object of 10 artists>)
```



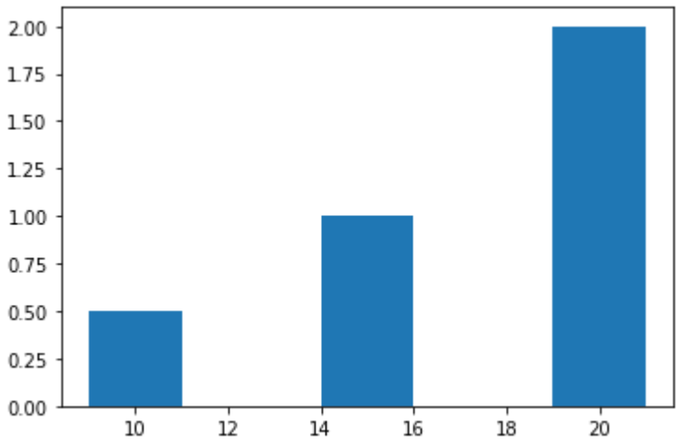
In [104]:
```
values=[10,15,20,10,15]
plt.hist(values,color="red",width=0.5)
```

Out[104]:
```
(array([2., 0., 0., 0., 0., 2., 0., 0., 0., 1.]),
 array([10., 11., 12., 13., 14., 15., 16., 17., 18., 19., 20.]),
 <BarContainer object of 10 artists>)
```



In [105]:
```
#Plotting Bar Charts
values=[10,15,20]
plt.bar(values,[0.5,1,2],width=2)
```

Out[105]:
```
<BarContainer object of 3 artists>
```

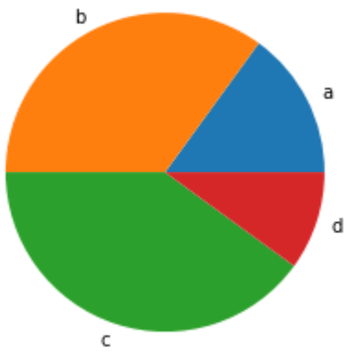


In [106]:
```
#Plotting Pie Charts
values=[15,35,40,10]
plt.pie(values)
```

Out[106]:
```
([<matplotlib.patches.Wedge at 0x1873ea4a3b0>,
  <matplotlib.patches.Wedge at 0x1873ea4a890>,
  <matplotlib.patches.Wedge at 0x1873ea4ad70>,
  <matplotlib.patches.Wedge at 0x1873ea4b250>],
 [Text(0.9801071672559598, 0.4993895680663527, ''),
  Text(-0.4993895680663527, 0.9801071672559598, ''),
  Text(-0.33991867422268845, -1.0461621742897658, ''),
  Text(1.0461621822461364, -0.3399186497354948, '')])
```

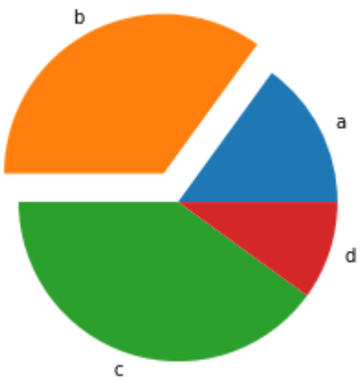
In [107]:
```
values=[15,35,40,10]
plt.pie(values,labels=['a','b','c','d'])
```

Out[107]:
```
([<matplotlib.patches.Wedge at 0x1873eaa9e70>,
  <matplotlib.patches.Wedge at 0x1873eaaa350>,
  <matplotlib.patches.Wedge at 0x1873eaaa7d0>,
  <matplotlib.patches.Wedge at 0x1873eaaacb0>],
 [Text(0.9801071672559598, 0.4993895680663527, 'a'),
  Text(-0.4993895680663527, 0.9801071672559598, 'b'),
  Text(-0.33991867422268845, -1.0461621742897658, 'c'),
  Text(1.0461621822461364, -0.3399186497354948, 'd')])
```



In [108]:
```
values=[15,35,40,10]
plt.pie(values,labels=['a','b','c','d'],explode=[0,0.2,0,0])
```

Out[108]:
```
([<matplotlib.patches.Wedge at 0x1873eb053f0>,
  <matplotlib.patches.Wedge at 0x1873eb058d0>,
  <matplotlib.patches.Wedge at 0x1873eb05db0>,
  <matplotlib.patches.Wedge at 0x1873eb06290>],
 [Text(0.9801071672559598, 0.4993895680663527, 'a'),
  Text(-0.5901876713511441, 1.158308470393407, 'b'),
  Text(-0.33991867422268845, -1.0461621742897658, 'c'),
  Text(1.0461621822461364, -0.3399186497354948, 'd')])
```

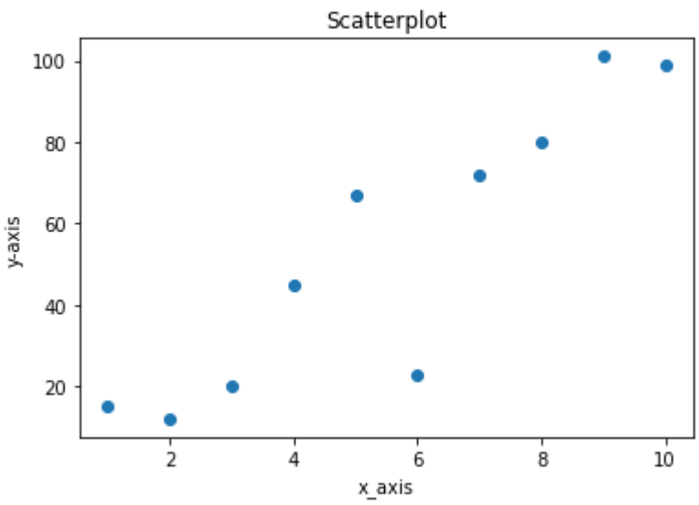


In [109]:
```
values=[15,35,40,10]
plt.pie(values,labels=['a','b','c','d'],colors=['r','g','b','y'])
```

Out[109]:
```
([<matplotlib.patches.Wedge at 0x1873eb58a00>,
  <matplotlib.patches.Wedge at 0x1873eb58ee0>,
  <matplotlib.patches.Wedge at 0x1873eb593c0>,
  <matplotlib.patches.Wedge at 0x1873eb598a0>],
 [Text(0.9801071672559598, 0.4993895680663527, 'a'),
  Text(-0.4993895680663527, 0.9801071672559598, 'b'),
  Text(-0.33991867422268845, -1.0461621742897658, 'c'),
  Text(1.0461621822461364, -0.3399186497354948, 'd')])
```

In [110]:
```
#Plotting Scatterplot
x_axis=[1,2,3,4,5,6,7,8,9,10]
y_axis=[15,12,20,45,67,23,72,80,101,99]

plt.title("Scatterplot")
plt.xlabel('x_axis')
plt.ylabel('y-axis')

plt.scatter(x_axis,y_axis,cmap='Accent')
```

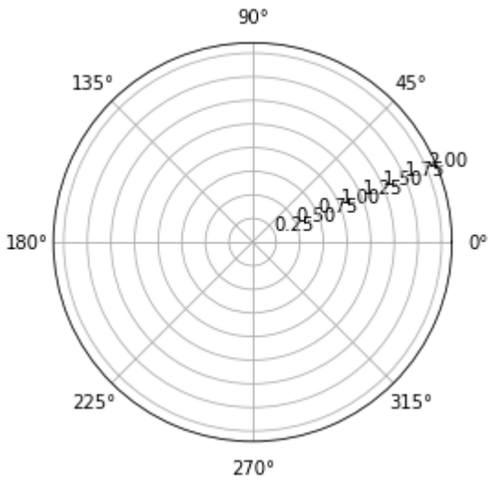
Out[110]:
```
<matplotlib.collections.PathCollection at 0x1873ebb04f0>
```



In [111]:
```
from matplotlib import cm
dir(cm)
```

Out[111]:
```
['Accent',
 'Accent_r',
 'Blues',
 'Blues_r',
 'BrBG',
 'BrBG_r',
 'BuGn',
 'BuGn_r',
 'BuPu',
 'BuPu_r',
 'CMRmap',
 'CMRmap_r',
 'ColormapRegistry',
 'Dark2',
 'Dark2_r',
 'GnBu',
 'GnBu_r',
 'Greens',
 'Greens_r',
```

In [113]:
```
#Plotting polar plot
import numpy as np
import matplotlib.pyplot as plt

theta= np.arange(0,(2*np.pi),0.01)     #generating an array of evenly spaced
r=2

for radian in theta:
    plt.polar(radian,r)
```



In [115]:
```
from sklearn.datasets import load_iris
iris = load_iris()
iris
```

Out[115]:
```
{'data': array([[5.1, 3.5, 1.4, 0.2],
        [4.9, 3. , 1.4, 0.2],
        [4.7, 3.2, 1.3, 0.2],
        [4.6, 3.1, 1.5, 0.2],
        [5. , 3.6, 1.4, 0.2],
        [5.4, 3.9, 1.7, 0.4],
        [4.6, 3.4, 1.4, 0.3],
        [5. , 3.4, 1.5, 0.2],
        [4.4, 2.9, 1.4, 0.2],
        [4.9, 3.1, 1.5, 0.1],
        [5.4, 3.7, 1.5, 0.2],
        [4.8, 3.4, 1.6, 0.2],
        [4.8, 3. , 1.4, 0.1],
        [4.3, 3. , 1.1, 0.1],
        [5.8, 4. , 1.2, 0.2],
        [5.7, 4.4, 1.5, 0.4],
        [5.4, 3.9, 1.3, 0.4],
        [5.1, 3.5, 1.4, 0.3],
        [5.7, 3.8, 1.7, 0.3],
```

In [116]:
```
df=pd.DataFrame(iris.data)
print(df.head())
```

```
     0    1    2    3
0  5.1  3.5  1.4  0.2
1  4.9  3.0  1.4  0.2
2  4.7  3.2  1.3  0.2
3  4.6  3.1  1.5  0.2
4  5.0  3.6  1.4  0.2
```

In [117]:
```
df.columns=iris.feature_names
df.head()
```

Out[117]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [121]:
```
df['variety']=iris.target
df.head()
```

Out[121]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	variety
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [132]:
```
import pandas as pd
import seaborn as sns
```

In [130]:
```
df=pd.read_csv("titanic.csv")
```

In [131]:
```
df
```

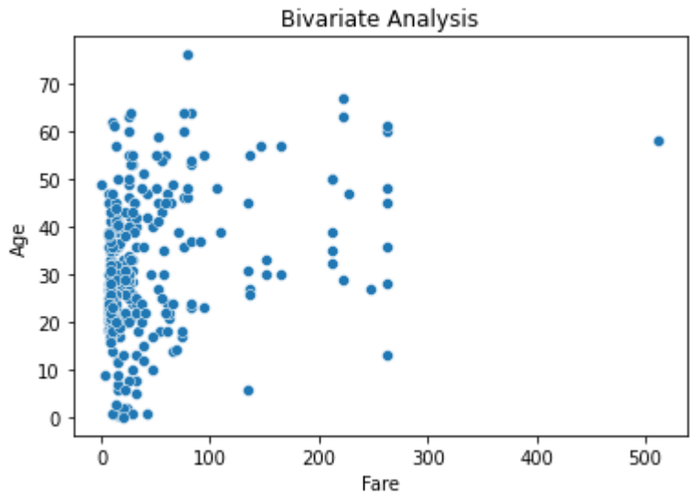
Out[131]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875
...	...	...	...	...	...	...	...	...	...	...
413	1305	0	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500
414	1306	1	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000
415	1307	0	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500
416	1308	0	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500
417	1309	0	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583

418 rows × 12 columns

In [135]:
```
sns.scatterplot(x=df['Fare'],y=df['Age'])
plt.title('Bivariate Analysis')
```

Out[135]:
```
Text(0.5, 1.0, 'Bivariate Analysis')
```



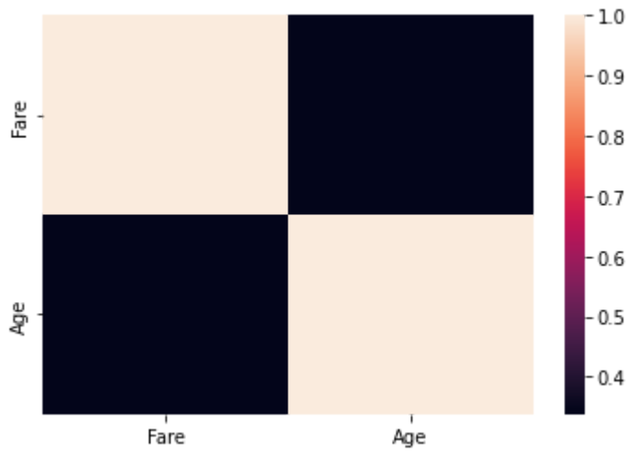
In [136]:
```
df[['Fare','Age']].corr()
```

Out[136]:

	Fare	Age
Fare	1.000000	0.337932
Age	0.337932	1.000000

In [138]:
```
sns.heatmap(df[['Fare','Age']].corr())
```

Out[138]: <AxesSubplot:>



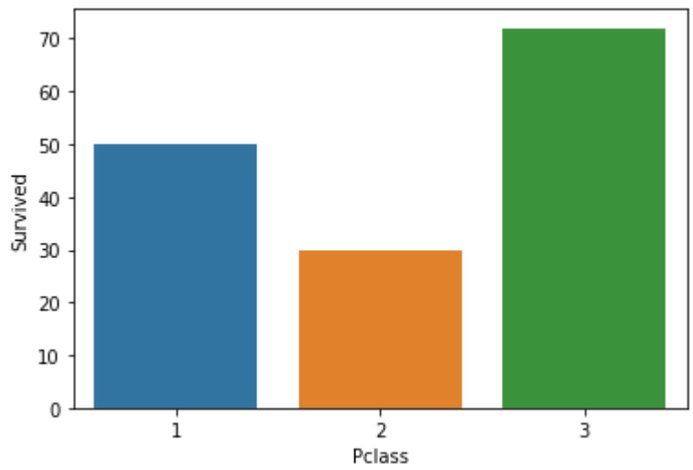
In [139]:
```
survived_ratio= df[['Pclass','Survived']].groupby('Pclass').sum()
survived_ratio
```

Out[139]:

	Survived
Pclass	
1	50
2	30
3	72

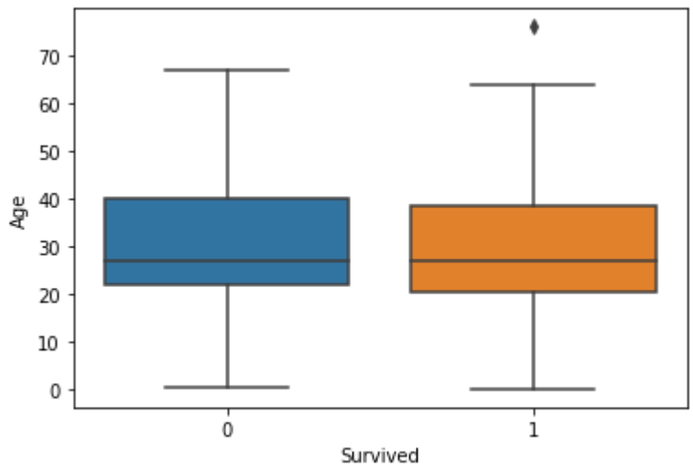
In [141]:
```
sns.barplot(x=survived_ratio.index,y=survived_ratio['Survived'])
```

Out[141]: <AxesSubplot:xlabel='Pclass', ylabel='Survived'>



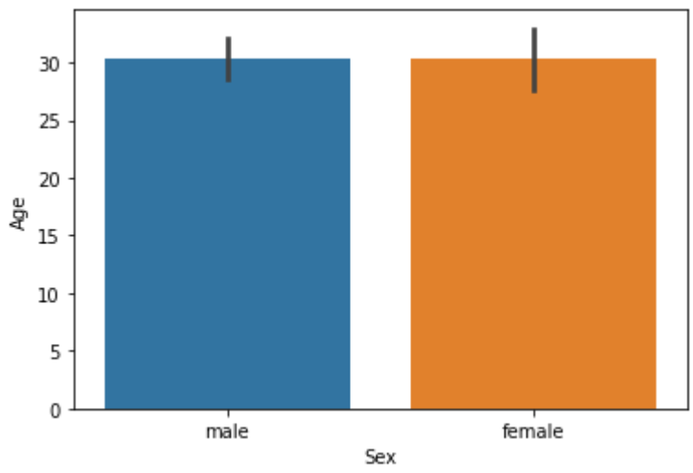
In [143]:
```
sns.boxplot(x=df['Survived'],y=df['Age'])
```

Out[143]: <AxesSubplot:xlabel='Survived', ylabel='Age'>



In [144]:
```
sns.barplot(x=df['Sex'],y=df['Age'])
```

Out[144]: <AxesSubplot:xlabel='Sex', ylabel='Age'>



In [145]:
```
df[['Sex']]
```

Out[145]:

	Sex
0	male
1	female
2	male
3	male
4	female
...	...
413	male
414	female
415	male
416	male
417	male

418 rows × 1 columns

In [146]:
```
df['Sex'].replace({'male':1,'female':0},inplace=True)
df.head()
```

Out[146]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	E
0	892	0	3	Kelly, Mr. James	1	34.5	0	0	330911	7.8292	NaN	
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	0	47.0	1	0	363272	7.0000	NaN	
2	894	0	2	Myles, Mr. Thomas Francis	1	62.0	0	0	240276	9.6875	NaN	
3	895	0	3	Wirz, Mr. Albert	1	27.0	0	0	315154	8.6625	NaN	
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	0	22.0	1	1	3101298	12.2875	NaN	

In [147]:
```
pip install yfinance
```

```
Collecting yfinance
  Downloading yfinance-0.1.74-py2.py3-none-any.whl (27 kB)
Requirement already satisfied: numpy>=1.15 in c:\users\hp\anaconda3\lib\site
-packages (from yfinance) (1.22.4)
Collecting requests>=2.26
  Downloading requests-2.28.1-py3-none-any.whl (62 kB)
Collecting lxml>=4.5.1
  Downloading lxml-4.9.1-cp310-cp310-win_amd64.whl (3.6 MB)
Collecting multitasking>=0.0.7
  Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Requirement already satisfied: pandas>=0.24.0 in c:\users\hp\anaconda3\lib\s
ite-packages (from yfinance) (1.4.2)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\hp\anacond
a3\lib\site-packages (from pandas>=0.24.0->yfinance) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\hp\anaconda3\lib\sit
e-packages (from pandas>=0.24.0->yfinance) (2022.1)
Requirement already satisfied: six>=1.5 in c:\users\hp\anaconda3\lib\site-pa
ckages (from python-dateutil>=2.8.1->pandas>=0.24.0->yfinance) (1.16.0)
Collecting idna<4,>=2.5
  Downloading idna-3.3-py3-none-any.whl (61 kB)
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.12-py2.py3-none-any.whl (140 kB)
Collecting charset-normalizer<3,>=2
  Downloading charset_normalizer-2.1.1-py3-none-any.whl (39 kB)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\hp\anaconda3\l
ib\site-packages (from requests>=2.26->yfinance) (2022.5.18.1)
Installing collected packages: urllib3, idna, charset-normalizer, requests,
multitasking, lxml, yfinance
Successfully installed charset-normalizer-2.1.1 idna-3.3 lxml-4.9.1 multitas
king-0.0.11 requests-2.28.1 urllib3-1.26.12 yfinance-0.1.74
Note: you may need to restart the kernel to use updated packages.
```

In [148]:
```
import yfinance as yf
df=pd.read_csv("AAPL.csv")
df.head()
```

Out[148]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	1980-12-12	0.128348	0.128906	0.128348	0.128348	0.100323	469033600
1	1980-12-15	0.122210	0.122210	0.121652	0.121652	0.095089	175884800
2	1980-12-16	0.113281	0.113281	0.112723	0.112723	0.088110	105728000
3	1980-12-17	0.115513	0.116071	0.115513	0.115513	0.090291	86441600
4	1980-12-18	0.118862	0.119420	0.118862	0.118862	0.092908	73449600

In [152]:
```
type(df.index[0])
```

Out[152]:
```
pandas._libs.tslibs.timestamps.Timestamp
```

In [ ]:
```

```