DAY-3

```
#to add a new row to existing array
import numpy as np
arr = np.array([[1,2,3],[45,4,7],[9,6,10]])
print("\n",arr)
na = np.array([12,23,45])
new_arr = np.r_[arr,[na]]
print("\n",new_arr)
```

```
 [[ 1  2  3]
 [45  4  7]
 [ 9  6 10]]

 [[ 1  2  3]
 [45  4  7]
 [ 9  6 10]
 [12 23 45]]
```

```
#frequency of all elements in series
import pandas as pd
se = pd.Series([1,1,1,2,2,2,2,3,3,3,4,4,5])
np.unique(se, return_counts=True)
```

```
(array([1, 2, 3, 4, 5], dtype=int64), array([3, 4, 3,
2, 1], dtype=int64))
```

```
df = pd.DataFrame(np.random.randn(5,4),
['A','B','C','D','E'],['w','x','y','z'])
print(df)
print("\n",df['w']) #-->to access specific column
print("\n",df.loc['A']) #-->to access specific row
```

```
          w         x         y         z
A -0.508318  0.437463 -1.472084 -0.217303
B  1.371073  0.337058 -1.046453  0.833393
C -1.215821 -1.533711 -1.080541  0.732395
D -0.487381  0.312995  0.615833  2.028830
E -0.036268 -1.747780 -1.628303  0.351286
```

```
A   -0.508318
B    1.371073
C   -1.215821
D   -0.487381
E   -0.036268
Name: w, dtype: float64

 w   -0.508318
x    0.437463
y   -1.472084
z   -0.217303
Name: A, dtype: float64
```

#to access multiple columns

```
df.loc[['A','C']]
```

	w	x	y	z
A	-0.508318	0.437463	-1.472084	-0.217303
C	-1.215821	-1.533711	-1.080541	0.732395

#to access rows based on index position

```
df.iloc[3] #--> 3 is index value
```

```
w   -0.487381
x    0.312995
y    0.615833
z    2.028830
Name: D, dtype: float64
```

#to get specific value of co-ordinate in dataframe

```
df.loc['B','y'] #Syntax --> df.loc[row,column]
```

```
-1.0464531916275492
```

#to get multiple co-ordinates

```
df.loc[['A','B'],['w','z']] #df.loc[[rows],[columns]]
```

	w	z
A	-0.508318	-0.217303
B	1.371073	0.833393

```
#retrieving the data based on condition
print(df[df>0]) #Nan for false
print("\n",df[df['w']>0])
```

```
          w         x         y         z
A       NaN  0.437463       NaN       NaN
B  1.371073  0.337058       NaN  0.833393
C       NaN       NaN       NaN  0.732395
D       NaN  0.312995  0.615833  2.028830
E       NaN       NaN       NaN  0.351286


          w         x         y         z
B  1.371073  0.337058 -1.046453  0.833393
```

```
#to convert dictionary to dataframe
d = {"A": [1,2,np.nan], "B": [5,np.nan,np.nan], "C":
[1,2,3], "D": [np.nan,np.nan,np.nan]}
df1 = pd.DataFrame(d)
df1
```

	A	B	C	D
0	1.0	5.0	1	NaN
1	2.0	NaN	2	NaN
2	NaN	NaN	3	NaN

```
#to drop values with Nan
#dropna(how, thresh, axis, inplace)
print(df1.dropna())
#how="all" or "any"
print("\n",df1.dropna(how="all")) #It will be
checking the rows with all
print("\n",df1.dropna(how="any")) #It will delete the
rows which have single Nan value
```

```
Empty DataFrame
Columns: [A, B, C, D]
Index: []
```

```
     A    B  C   D
0  1.0  5.0  1 NaN
1  2.0  NaN  2 NaN
2  NaN  NaN  3 NaN
```

```
 Empty DataFrame
Columns: [A, B, C, D]
Index: []
```

```
df1.dropna(how="all",axis=1)
```

	A	B	C
0	1.0	5.0	1
1	2.0	NaN	2
2	NaN	NaN	3

```
#to check the sum of Nan values in each column
df1.isna().sum()
```

```
A    1
B    2
C    0
D    3
dtype: int64
```

#thresh

```
print(df1.dropna(thresh=2,axis=1))
```

```
     A  C
0  1.0  1
1  2.0  2
2  NaN  3
```

#fillna()

```
df1.fillna(value=0.0)
```

	A	B	C	D
0	1.0	5.0	1	0.0
1	2.0	0.0	2	0.0
2	0.0	0.0	3	0.0

```
df1['A'].fillna(value=df1['A'].mean())
```

```
0    1.0
1    2.0
2    1.5
Name: A, dtype: float64
```

 # Group by
 # To group based on column and perform aggregate functions

```
data = {'company':
['Google','Google','Meta','Meta','Fb','Fb'],
```

```
        'person':
['sam','mani','sai','prasanth','mohan','mounika'],
        'sales': [300, 500, 100, 200, 600, 20]}
df2 = pd.DataFrame(data)
df2
```

	company	person	sales
0	Google	sam	300
1	Google	mani	500
2	Meta	sai	100
3	Meta	prasanth	200
4	Fb	mohan	600
5	Fb	mounika	20

```
bycomp = df2.groupby('company')
bycomp
bycomp.sum()['sales']
bycomp.value_counts()
bycomp['company'].value_counts()
```

```
company
Fb        2
Google    2
Meta      2
Name: count, dtype: int64
```

```
# To get the maximum salary person from each company
df2.loc[bycomp['sales'].idxmax()][['person','sales']]
```

	person	sales
4	mohan	600
1	mani	500
3	prasanth	200

```
bycomp.describe()
```

	sales							
	count	mean	std	min	25%	50%	75%	max
company								
Fb	2.0	310.0	410.121933	20.0	165.0	310.0	455.0	600.0
Google	2.0	400.0	141.421356	300.0	350.0	400.0	450.0	500.0
Meta	2.0	150.0	70.710678	100.0	125.0	150.0	175.0	200.0

```
# to access a csv_file
dp = pd.read_csv("samplecsv.csv")
dp.head() #to get first 5 rows


# To access a xlsx_file
dp1 = pd.ExcelFile("sample.xlsx")
```

```
# to access a csv_file
dp = pd.read_csv("delimatercsv.csv", sep=';',
names=['sid','ages','places'])
dp.drop(0) #to delete a row
```

	sid	ages	places
1	Alice	25	New York
2	Bob	30	San Francisco
3	Charlie	22	Los Angeles

```
# to access a csv_file
df_titanic = pd.read_csv("titanic_train.csv")
df_titanic.columns
df_titanic.head()
df_titanic['Pclass'].unique()
df_titanic['Pclass'].value_counts()
```

```
Pclass
3    491
1    216
2    184
Name: count, dtype: int64
```

```
df_titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
```

```
 3   Name          891 non-null    object
 4   Sex           891 non-null    object
 5   Age           714 non-null    float64
 6   SibSp         891 non-null    int64
 7   Parch         891 non-null    int64
 8   Ticket        891 non-null    object
 9   Fare          891 non-null    float64
 10  Cabin         204 non-null    object
 11  Embarked      889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
df_titanic.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000

```
#Sum of Nan's
df_titanic.isna().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
df_titanic.isna().sum()*100/len(df_titanic)
```

```
PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            19.865320
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.000000
Cabin          77.104377
Embarked        0.224467
dtype: float64
```

```
#To drop a column
df_titanic.drop(['Cabin'],axis=1)
```

```
#People dead
len(df_titanic)-df_titanic['Survived'].sum()
```

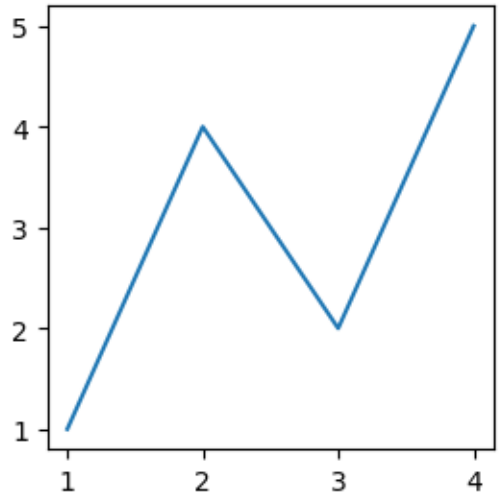
549

#Matplotlib
#It is data visualization library, which was inspired by matlab. It is used for data visualization in form of various plots

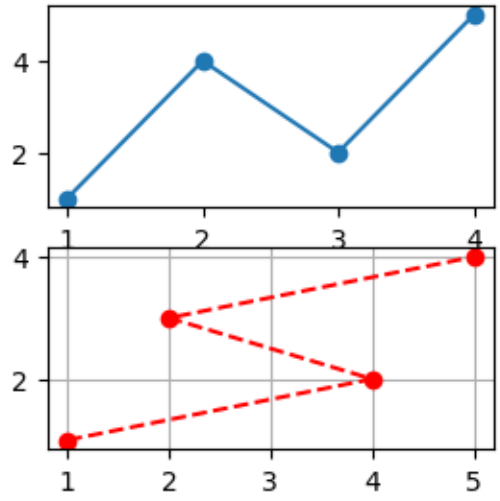
```
import matplotlib.pyplot as plt

x = [1,2,3,4]
y = [1,4,2,5]
plt.figure(figsize=(3,3))
plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x1a0c354b050>]
```

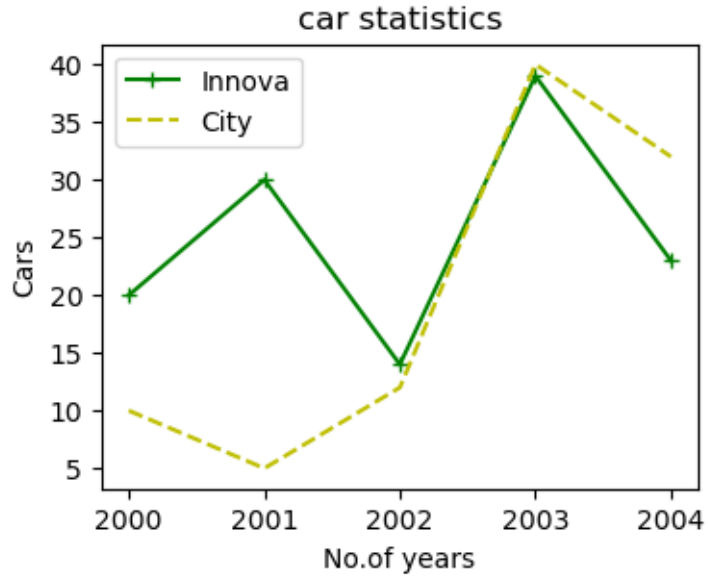



```
plt.figure(figsize=(3,3))
plt.subplot(2,1,1)
plt.plot(x,y,marker='o')
plt.subplot(2,1,2)
plt.plot(y,x,'r',marker='o',linestyle='dashed')
plt.grid()
```

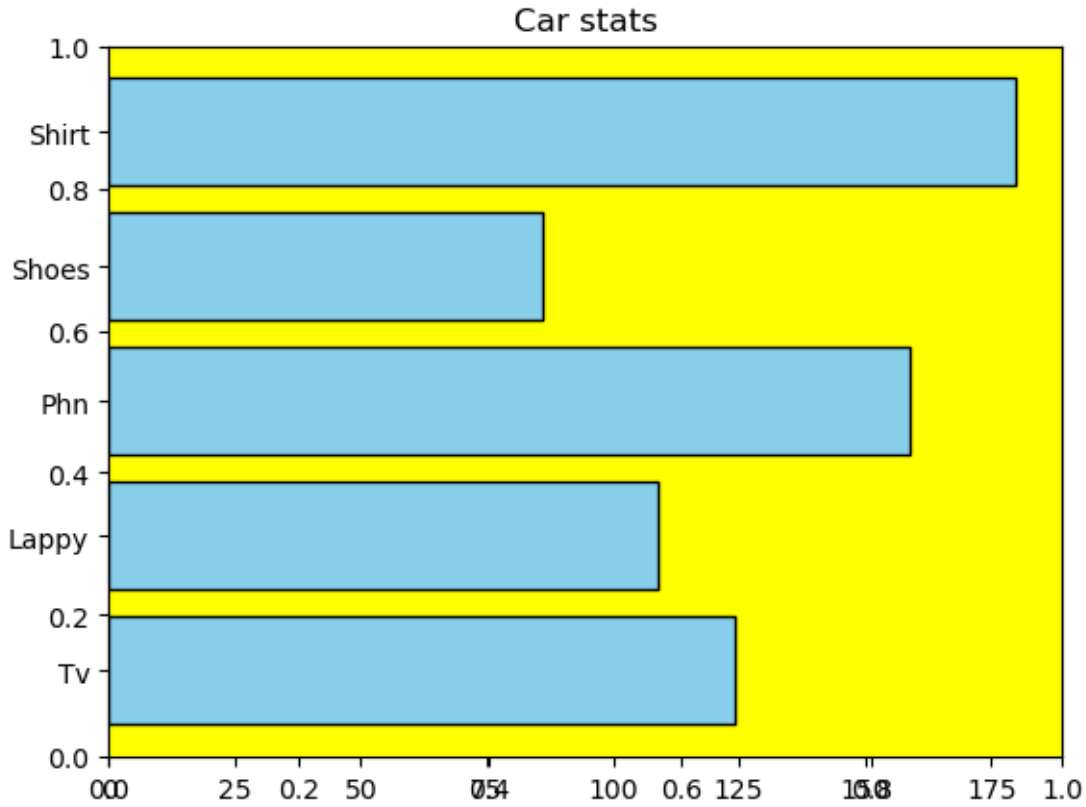



```
import numpy as np
plt.figure(figsize=(4,3))
years = np.arange(2000,2005,dtype=int)
y_innova = np.array([20,30,14,39,23])
honda = np.array([10,5,12,40,32])
years, y_innova
plt.plot(years, y_innova,'-+g')
plt.plot(years, honda,'--y')
plt.title("car statistics")
plt.xlabel("No.of years")
plt.ylabel("Cars")
plt.legend(['Innova','City'])
```

```
<matplotlib.legend.Legend at 0x1a0c9f8ca90>
```



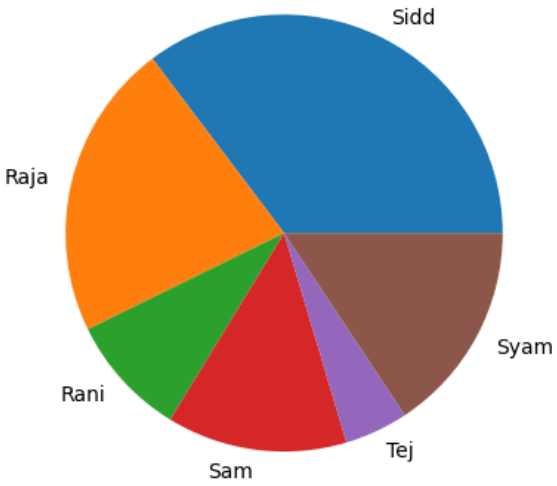
```
#barplot
products = ['Tv','Lappy','Phn','Shoes','Shirt']
sales = np.random.randint(30,200,size=len(products))
plt.title("Car stats")
ax = plt.axes()
plt.barh(products, sales, color="skyblue",
edgecolor="black")
ax.set_facecolor("yellow")
```




```
names = ["Sidd","Raja","Rani","Sam","Tej","Syam"]
scores = [90,56,23,34,12,40]
plt.pie(scores, labels=names)
```

```
([<matplotlib.patches.Wedge at 0x1a0cc245910>,
  <matplotlib.patches.Wedge at 0x1a0cc247850>,
  <matplotlib.patches.Wedge at 0x1a0cc245050>,
  <matplotlib.patches.Wedge at 0x1a0cc310490>,
  <matplotlib.patches.Wedge at 0x1a0cc1e7190>,
  <matplotlib.patches.Wedge at 0x1a0cc316f50>],
 [Text(0.49031216423879204, 0.9846796339924332, 'Sidd
'),
  Text(-1.070001114972945, 0.25514234058002755, 'Raja
'),
```

```
  Text(-0.8129097548643817, -0.7410652673323255, 'Ran
i'),
  Text(-0.14189902764368106, -1.0908091794414723, 'Sa
m'),
  Text(0.4659034895633585, -0.9964607059049975, 'Tej'
),
  Text(0.969113489904746, -0.5204027706350572, 'Syam'
)])
```




```
gender = ['male','female']
plt.figure(facecolor = "#94F008")
male = (df_titanic['Sex']=='male').sum()
female = (df_titanic['Sex']=='female').sum()
plt.bar(gender,[male,female],width=0.3)
```

```
<BarContainer object of 2 artists>
```