```
#Numpy -
import numpy as np
```

```
#array(),arange(),reshape(),sum(),mean(),max(),min(),std(),argmax(),argmin()
#shape,ndim,size
```

```
a = np.array([4,6,8,9,3])
print(a)
print(a.shape)
print(a.ndim)
print(a.size)
```

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

```
a = np.array([[4,6,8,9,3],[11,12,13,56,78],[45,67,34,12,56]])
print(a)
print(a.shape)
print(a.ndim)
print(a.size)
```

```
[[ 4  6  8  9  3]
 [11 12 13 56 78]
 [45 67 34 12 56]]
(3, 5)
2
15
```

```
b = np.arange(0,36).reshape(3,4,3)
b
```

```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8],
        [ 9, 10, 11]],

       [[12, 13, 14],
        [15, 16, 17],
        [18, 19, 20],
        [21, 22, 23]],

       [[24, 25, 26],
        [27, 28, 29],
        [30, 31, 32],
        [33, 34, 35]]])
```

```
x1 = np.random.randint(2,20,15)
x2 = np.random.randint(3,30,15)
```

```
x3 = np.random.randint(7,25,15)

X = np.array([x1,x2,x3])
X
```

```
np.transpose(X)
```

```
X.reshape(15,3)
```

```
k = np.random.randn(3)
k
```

```
l = np.random.rand(3)
l
```

```
a
```

```
#row slicing , column slicing
a[0:2 ,:]
```

```
array([[ 4,  6,  8,  9,  3],
       [11, 12, 13, 56, 78]])
```

```
a[:,2:4]
```

```
array([[ 8,  9],
       [13, 56],
       [34, 12]])
```

```
b
```

```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8],
        [ 9, 10, 11]],

       [[12, 13, 14],
        [15, 16, 17],
        [18, 19, 20],
        [21, 22, 23]],

       [[24, 25, 26],
        [27, 28, 29],
        [30, 31, 32],
        [33, 34, 35]]])
```

```
print(np.sum(a))
print(np.max(a))
print(np.min(a))
print(np.std(a))
print(np.argmax(a))
print(np.argmin(a))
print(np.mean(a))
```

```
414
78
3
25.011464038183238
9
4
27.6
```

```
a
```

```
array([[ 4,  6,  8,  9,  3],
       [11, 12, 13, 56, 78],
       [45, 67, 34, 12, 56]])
```

```
#Dot product
#Combing two different matrices
```

```
a
```

```
array([[ 4,  6,  8,  9,  3],
       [11, 12, 13, 56, 78],
       [45, 67, 34, 12, 56]])
```

```
b = np.arange(0,5).reshape(5,1)
b
```

```
array([[0],
       [1],
       [2],
       [3],
       [4]])
```

```
np.dot(a,b)
np.matmul(a,b)
```

```
#Stacking
a
```

```
c = np.arange(20).reshape(4,5)
c
```

```
z = np.vstack((a,c))
```

```
d = np.arange(3).reshape(3,1)
d
```

```
np.hstack((a,d))
```

```
z
```

```
z[:,-2:]
```

```
print(np.zeros([3,3]))
print(np.ones([3,3]))
print(np.eye(3))
```

```
#Pandas - Access, Analyse, preprocess over data
import pandas as pd
```

```
#DataFrame
p = pd.DataFrame(z,columns=['a','b','c','d','e'])
p
```

```
p[['a','b','c']]
```

```
#loc
#iloc
p.loc[0:4,'c':'e']
```

```
p.iloc[0:4,2:4]
```

```
chips = pd.read_table('http://bit.ly/chiporders')
```

```
chips
```

| | order_id | quantity | item_name | choic |
|---|---|---|---|---|
| **0** | 1 | 1 | Chips and Fresh Tomato Salsa | |
| **1** | 1 | 1 | Izze | |
| **2** | 1 | 1 | Nantucket Nectar | |
| **3** | 1 | 1 | Chips and Tomatillo-Green Chili Salsa | |
| **4** | 2 | 2 | Chicken Bowl | [Tomatillo-Red Chili Salsa (Hot), |
| **...** | ... | ... | ... | |
| **4617** | 1833 | 1 | Steak Burrito | [Fresh Tomato Salsa, [Rice, Black |
| **4618** | 1833 | 1 | Steak Burrito | [Fresh Tomato Salsa, [Rice, Sour Cı |
| **4619** | 1834 | 1 | Chicken Salad Bowl | [Fresh Tomato Salsa, [Fajita Veg |
| **4620** | 1834 | 1 | Chicken Salad Bowl | [Fresh Tomato Salsa, [Fajita Veg |
| **4621** | 1834 | 1 | Chicken Salad Bowl | [Fresh Tomato Salsa, [Fajita Veg |

4622 rows × 5 columns

```
#Extract all the data with Chicken is present as their item name
#Calculate mean price of all the item prices
#Find out if there is any missing value present
```

```
#1. Find out the data types in dataframe
```

```
#2. Find out if there is any missing value
```

```
chips.dtypes
chips.shape
```

```
(4622, 5)
```

```
chips.isnull().sum()
```

```
order_id                0
quantity                0
item_name               0
choice_description    1246
item_price              0
dtype: int64
```

```
chips['choice_description']
chips_new = chips.drop(['choice_description'],axis=1)
```

```
chips.head(5)
```

| | order_id | quantity | item_name | choice_descr |
|---|---|---|---|---|
| **0** | 1 | 1 | Chips and Fresh Tomato Salsa | |
| **1** | 1 | 1 | Izze | [Cler |
| **2** | 1 | 1 | Nantucket Nectar | |
| **3** | 1 | 1 | Chips and Tomatillo-Green Chili Salsa | |
| **4** | 2 | 2 | Chicken Bowl | [Tomatillo-Red Chili Salsa (Hot), [Black I |

```
chips_new
```

```
chips_new2 = chips.dropna()
```

```
chips.isnull().sum()
```

```
print(chips_new.isnull().sum())
print(chips_new.shape)
```

```
print(chips_new2.isnull().sum())
print(chips_new2.shape)
```

```
chips['choice_description'].fillna('abcde',inplace=True)
```

```
chips
```

```
chips['item_price'].sum()
```

```
chips['item_price'].str.replace('$','').astype(float).mean()
```

```
#.str.replace() - it is used to replace certain value in a pandas series
#.str.contains() - to verify certain string in a pandas series
#.astype()
#.dtypes
```

```
#Data filtering
chips['item_name'].str.contains('Chicken').sum()
```

```
1560
```

```
chips[chips['item_name'].str.contains('Chicken')]
```

| | order_id | quantity | item_name | choice_description |
|---|---|---|---|---|
| **4** | 2 | 2 | Chicken Bowl | [Tomatillo-Red Chili Salsa (Hot), [Black Beans... |
| **5** | 3 | 1 | Chicken Bowl | [Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou... |
| **11** | 6 | 1 | Chicken Crispy Tacos | [Roasted Chili Corn Salsa, [Fajita Vegetables,... |
| **12** | 6 | 1 | Chicken Soft Tacos | [Roasted Chili Corn Salsa, [Rice, Black Beans,... |
| **13** | 7 | 1 | Chicken Bowl | [Fresh Tomato Salsa, [Fajita Vegetables, Rice,... |
| **...** | ... | ... | ... | ... |
| **4604** | 1828 | 1 | Chicken Bowl | [Fresh Tomato Salsa, [Rice, Black Beans, Chees... |
| **4615** | 1832 | 1 | Chicken Soft Tacos | [Fresh Tomato Salsa, [Rice, Cheese, Sour Cream]] |
| **4619** | 1834 | 1 | Chicken Salad Bowl | [Fresh Tomato Salsa, [Fajita Vegetables, Pinto... |
| **4620** | 1834 | 1 | Chicken Salad Bowl | [Fresh Tomato Salsa, [Fajita Vegetables, Lettu... |
| **4621** | 1834 | 1 | Chicken Salad Bowl | [Fresh Tomato Salsa, [Fajita Vegetables, Pinto... |

1560 rows × 5 columns

```
chips[chips['quantity'] == 2]
```

| | order_id | quantity | item_name | choice_description | item_price |
|---|---|---|---|---|---|
| **4** | 2 | 2 | Chicken Bowl | [Tomatillo-Red Chili Salsa (Hot), [Black Beans... | $16.98 |
| **18** | 9 | 2 | Canned Soda | [Sprite] | $2.18 |
| **51** | 23 | 2 | Canned Soda | [Mountain Dew] | $2.18 |
| **135** | 60 | 2 | Chicken Salad Bowl | [Tomatillo Green Chili Salsa, [Sour Cream, Che... | $22.50 |
| **148** | 67 | 2 | Steak Burrito | [Tomatillo-Red Chili Salsa (Hot), [Rice, Chees... | $17.98 |
| **...** | ... | ... | ... | ... | ... |
| **4435** | 1767 | 2 | Chicken Bowl | [Fresh Tomato Salsa, [Rice, Pinto Beans, Chees... | $17.50 |
| **4499** | 1789 | 2 | Canned Soft Drink | [Coke] | $2.50 |
| **4560** | 1812 | 2 | Canned Soft Drink | [Coke] | $2.50 |

```
import pandas as pd
ipl = pd.read_csv(r'matches.csv')
```

```
#ipl = pd.read_csv(r'D:\myfolder\matches.csv')
```

```
ipl.head(5)
```

```
ipl['city'].value_counts()
```

```
Mumbai            85
Bangalore         66
Kolkata           61
Delhi             60
Hyderabad         49
Chennai           48
Chandigarh        46
Jaipur            33
Pune              32
Durban            15
Ahmedabad         12
Centurion         12
Visakhapatnam     11
Rajkot            10
Dharamsala         9
Johannesburg       8
Cape Town          7
Port Elizabeth     7
Abu Dhabi          7
Cuttack            7
Ranchi             7
Sharjah            6
Raipur             6
Indore             5
Kochi              5
Kanpur             4
East London        3
Nagpur             3
Kimberley          3
Bloemfontein       2
Name: city, dtype: int64
```

```
ipl['city'].unique()
```

```
array(['Hyderabad', 'Pune', 'Rajkot', 'Indore', 'Bangalore', 'Mumbai',
       'Kolkata', 'Delhi', 'Chandigarh', 'Kanpur', 'Jaipur', 'Chennai',
       'Cape Town', 'Port Elizabeth', 'Durban', 'Centurion',
       'East London', 'Johannesburg', 'Kimberley', 'Bloemfontein',
       'Ahmedabad', 'Cuttack', 'Nagpur', 'Dharamsala', 'Kochi',
       'Visakhapatnam', 'Raipur', 'Ranchi', 'Abu Dhabi', 'Sharjah', nan],
      dtype=object)
```

```
len(ipl['city'].unique())
```

```
31
```

```
len(ipl)
```

```
636
```

```
ipl['win_by_runs'].idxmax()
```

```
43
```

```
ipl.iloc[43]['winner']
```

```
'Mumbai Indians'
```

```
ipl.iloc[43]
```

```
id                               44
season                         2017
city                          Delhi
date                     2017-05-06
team1                Mumbai Indians
team2              Delhi Daredevils
toss_winner        Delhi Daredevils
toss_decision                 field
result                       normal
dl_applied                        0
winner               Mumbai Indians
win_by_runs                     146
win_by_wickets                    0
player_of_match         LMP Simmons
venue              Feroz Shah Kotla
umpire1                 Nitin Menon
umpire2                   CK Nandan
umpire3                         NaN
Name: 43, dtype: object
```

```
ipl['win_by_wickets'].max()
```

```
10
```

```
ipl[ipl['win_by_wickets'] == 10]['winner']
```

```
2            Kolkata Knight Riders
34                 Kings XI Punjab
71                 Deccan Chargers
119               Delhi Daredevils
183    Royal Challengers Bangalore
298               Rajasthan Royals
376                 Mumbai Indians
390            Chennai Super Kings
542    Royal Challengers Bangalore
590            Sunrisers Hyderabad
Name: winner, dtype: object
```

```
ipl['toss_decision']=='bat'
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-24-78908464042a> in <module>()
----> 1 data['toss_decision']=='bat'

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

SEARCH STACK OVERFLOW