```
import numpy as n
import pandas as p
```

SERIES

To convert list,numpy arrayt, dictionary into series

```
l=[1,2,3,4]
array=n.array([10,20,20])
d={'a':500,'b':4500}
labales=['a','b','c' ]
```

```
p.Series(data=l)
```

```
0    1
1    2
2    3
3    4
dtype: int64
```

```
p.Series(data=l,index=labales)
```

```
a    1
b    2
c    3
d    4
dtype: int64
```

numpy to series

```
p.Series(array)
```

```
0    10
1    20
2    20
dtype: int64
```

```
p.Series(array,index=labales)
```

```
a    10
b    20
c    20
dtype: int64
```

Dictionary to series

```
p.Series(d)
```

```
a     500
b    4500
dtype: int64
```

Operations on series

```
series1=p.Series([1,2,3,4],index=['India','America','Germany','China'])
```

```
series2=p.Series([10,20,30,40],index=['India','America','Uganda','China'])
```

On adding 2 series, we get a numerical value(added) for "like" indices and NAN for different indices

```
series1+series2
```

```
America    22.0
China      44.0
Germany     NaN
India      11.0
Uganda      NaN
dtype: float64
```

DATA FRAMES

[ ] ↳ *31 cells hidden*

MISSING VALUES

CREATING A DATA FRAME WITH VALUES INCLUDING 'NaN'

```
df=p.DataFrame({'A':[1,2,n.nan],'B':[5,n.nan,n.nan],'C':[1,2,3]})
```

```
df
```

	A	B	C
0	1.0	5.0	1
1	2.0	NaN	2

Drop all rows with NaN value/missing value

```
df.dropna()
```

	A	B	C
0	1.0	5.0	1

Drop all columns with NaN value/missing value

```
df.dropna(axis=1)
```

	C
0	1
1	2
2	3

Removes rows with 2 NaN values(since thresh=2)

```
df.dropna(thresh=2)
```

	A	B	C
0	1.0	5.0	1
1	2.0	NaN	2

Fill the missing values with "FILL VALUE"

```
df.fillna(value="FILL VALUE")
```

	A	B	C

Replace NaN/missing values with average/mean of coloumns

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

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

OPERATIONS IN PANDAS

```
df=p.DataFrame({'col1':[1,2,3,4],'col2':[444,555,666,777],'col3':['ab','cd','ef','gh']})
df
```

	col1	col2	col3
0	1	444	ab
1	2	555	cd
2	3	666	ef
3	4	777	gh

```
df['col2'].unique()
```

```
array([444, 555, 666, 777])
```

```
df['col1'].nunique()
```

```
4
```

```
df['col2'].value_counts()
```

```
444    1
555    1
666    1
777    1
Name: col2, dtype: int64
```

```
newdf=df[(df['col2']>3)&(df['col2']==555)]
newdf
```

	col1	col2	col3
1	2	555	cd

APPLYING FUNCTIONS

```
def times2(x):
  return x*2
```

```
df['col1'].apply(times2)
```

```
0    2
1    4
2    6
3    8
Name: col1, dtype: int64
```

```
df['col3'].apply(len)
```

```
0    2
1    2
2    2
3    2
Name: col3, dtype: int64
```

```
df['col1'].sum()
```

```
10
```

```
del df.col2
df
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-19-ac0a40d11344> in <module>()
----> 1 del df.col2
      2 df

AttributeError: col2
```

SEARCH STACK OVERFLOW

GET COLOUMN AND INDEX NAME

```
df.columns
```

```
Index(['col1', 'col3'], dtype='object')
```

```
df.index
```

```
RangeIndex(start=0, stop=4, step=1)
```

SORTING & ORDERING DATA FRAME

```
df
```

	col1	col3
0	1	ab
1	2	cd
2	3	ef
3	4	gh

```
df.sort_values(by='col3')
```

	col1	col3
0	1	ab
1	2	cd
2	3	ef
3	4	gh

```
df.sort_values(by='col3',inplace=True)
```

```
df=p.DataFrame({'col1':[1,2,3,4],'col2':[444,555,666,777]})
df
```

	col1	col2
0	1	444
1	2	555
2	3	666
3	4	777

FINDING NULL VALUES / CHECKING FOR NULL VALUES

```
df.isnull()
```

	col1	col2
0	False	False
1	False	False
2	False	False
3	False	False

```
import numpy as n
import pandas as p
```

```
df=p.read_csv('/content/Salaries.csv')
```

```
df.head(3)
```

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	Totall
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.00	400184.25	NaN	567595
1	2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966.02	245131.88	137811.38	NaN	538909

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30362 entries, 0 to 30361
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Id                30362 non-null  int64  
 1   EmployeeName      30361 non-null  object 
 2   JobTitle          30361 non-null  object 
 3   BasePay           30361 non-null  float64
 4   OvertimePay       30361 non-null  float64
 5   OtherPay          30361 non-null  float64
 6   Benefits          0 non-null      float64
 7   TotalPay          30361 non-null  float64
 8   TotalPayBenefits  30361 non-null  float64
 9   Year              30361 non-null  float64
 10  Notes             0 non-null      float64
 11  Agency            30361 non-null  object 
 12  Status            0 non-null      float64
dtypes: float64(9), int64(1), object(3)
memory usage: 3.0+ MB
```

```
df['BasePay'].mean()
```

```
74696.90481835336
```

```
df['OvertimePay'].max()
```

```
245131.88
```

```
df[df['EmployeeName'] == 'JOSEPH DRISCOLL']['JobTitle'  ]
```

```
24    CAPTAIN, FIRE SUPPRESSION
Name: JobTitle, dtype: object
```

```
df['TotalPayBenefits'][df['EmployeeName']=='JOSEPH DRISCOLL']
```

```
24    270324.91
Name: TotalPayBenefits, dtype: float64
```

```
df[df['TotalPay'].max()==df['TotalPay']]
```

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalI
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT	167411.18	0.0	400184.25	NaN	567595

```
df[df['TotalPay'].min()==df['TotalPay']]
```

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay
30360	30361	SU QING CHEN	LIBRARY PAGE	14604.0	0.0	78.22	NaN	14682.22

```
df.groupby('Year').mean()['BasePay']
```

```
Year
2011.0    74696.904818
Name: BasePay, dtype: float64
```

```
df['JobTitle'].nunique()
```

```
1026
```

```
df['JobTitle'].value_counts().head()
```

```
TRANSIT OPERATOR      2063
REGISTERED NURSE      1188
SPECIAL NURSE          834
FIREFIGHTER            786
POLICE OFFICER III     771
Name: JobTitle, dtype: int64
```

```
(df[df['Year']==2013]['JobTitle'].value_counts()==1).sum()
```

```
0
```

```
df['JobTitle'].apply(lambda str:('chief' in str.lower())).sum()
```

```
619
```

```
def find_chief(job_title):
    if 'chief' in job_title.lower().split():
        return True
    else:
        return False

df = p.read_csv('Salaries.csv')

sum(df['JobTitle'].apply(lambda x: find_chief(x)))
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2718: DtypeWarni
  interactivity=interactivity, compiler=compiler, result=result)
469
```

```
import pandas as p
import numpy as n
```

LOADING THE DATASET

```
ecom=p.read_csv('/content/Ecommerce Purchases.csv',encoding= 'unicode_escape')
```

CHECK THE HEAD OF THE DATA FRAME

```
ecom.head(3)
```

	Address	Lot	AM or PM	Browser Info	Company	Credit Card	CC Exp Date	CC Security Code
0	16629 Pace Camp Apt. 448\nAlexisborough, NE 77...	46 in	PM	Opera/9.56.(X11; Linux x86_64; sl-SI) Presto/2...	Martinez-Herman	6011929061123406	02/20	900
1	9374 Jasmine Spurs Suite 508\nSouth John, TN 8...	28 rn	PM	Opera/8.93.(Windows 98; Win 9x 4.90; en-US) Pr...	Fletcher, Richards and Whitaker	3337758169645356	11/18	561
2	Unit 0065 Box 5052\nDPO AP 27450	94 vE	PM	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT ...	Simpson, Williams and Pham	675957666125	08/19	699

NO OF ROWS AND COLUMNS

```
ecom.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Address           10000 non-null  object
 1   Lot               10000 non-null  object
 2   AM or PM          10000 non-null  object
 3   Browser Info      10000 non-null  object
```

```
 4   Company           10000 non-null  object
 5   Credit Card       10000 non-null  int64
 6   CC Exp Date       10000 non-null  object
 7   CC Security Code  10000 non-null  int64
 8   CC Provider       10000 non-null  object
 9   Email             10000 non-null  object
 10  Job               10000 non-null  object
 11  IP Address        10000 non-null  object
 12  Language          10000 non-null  object
 13  Purchase Price    10000 non-null  float64
dtypes: float64(1), int64(2), object(11)
memory usage: 1.1+ MB
```

```
ecom['Purchase Price'].mean()
```

```
50.34730200000025
```

```
ecom['Purchase Price'].max()
```

```
99.99
```

```
ecom['Purchase Price'].min()
```

```
0.0
```

```
ecom[ecom['Language']=='en'].count()
```

```
Address             1098
Lot                 1098
AM or PM            1098
Browser Info        1098
Company             1098
Credit Card         1098
CC Exp Date         1098
CC Security Code    1098
CC Provider         1098
Email               1098
Job                 1098
IP Address          1098
Language            1098
Purchase Price      1098
dtype: int64
```

```
ecom[ecom['Job']=='Lawyer'].count()
```

```
Address             30
Lot                 30
AM or PM            30
Browser Info        30
Company             30
Credit Card         30
```

```
CC Exp Date         30
CC Security Code    30
CC Provider         30
Email               30
Job                 30
IP Address          30
Language            30
Purchase Price      30
dtype: int64
```

```
ecom['AM or PM'].value_counts()
```

```
PM    5068
AM    4932
Name: AM or PM, dtype: int64
```

```
ecom['Job'].value_counts().head()
```

```
Interior and spatial designer        31
Lawyer                               30
Social researcher                    28
Purchasing manager                   27
Research officer, political party    27
Name: Job, dtype: int64
```

```
ecom[ecom['Lot']=='90 WT']['Purchase Price']
```

```
513    75.1
Name: Purchase Price, dtype: float64
```

```
ecom[ecom['Credit Card']==4926535242672853]['Email']
```

```
1234    bondellen@williams-garza.com
Name: Email, dtype: object
```

```
ecom[(ecom['CC Provider']=='American Express')&(ecom['Purchase Price']>95)].count()
```

```
Address             39
Lot                 39
AM or PM            39
Browser Info        39
Company             39
Credit Card         39
CC Exp Date         39
CC Security Code    39
CC Provider         39
Email               39
Job                 39
IP Address          39
Language            39
```

```
Purchase Price      39
dtype: int64
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-73f96bfe4f0b> in <module>()
----> 1 sum(ecom['CC Exp Date'].apply(lambda x: x[3:]) == '25')

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

SEARCH STACK OVERFLOW