# Built first Algorithm

Ethans
Learn from experts

**Know Euclidean Distance**

In mathematics the **Euclidean distance** or **Euclidean metric** is the "ordinary" (i.e. straight-line) distance between two points in Euclidean space. With this distance, Euclidean space becomes a metric space. The associated norm is called the **Euclidean norm.** From Wikipedia

**One dimension** [ edit ]

In one dimension, the distance between two points on the real line is the absolute value of their numerical difference. Thus if $x$ and $y$ are two points on the real line, then the distance between them is given by:

$$\sqrt{(x-y)^2} = |x-y|.$$

In one dimension, there is a single homogeneous, translation-invariant metric (in other words, a distance that is induced by a norm), up to a scale factor of length, which is the Euclidean distance. In higher dimensions there are other possible norms.

**Two dimensions** [ edit ]

In the Euclidean plane, if $\mathbf{p} = (p_1, p_2)$ and $\mathbf{q} = (q_1, q_2)$ then the distance is given by

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \sqrt{(q_1-p_1)^2 + (q_2-p_2)^2}.$$

This is equivalent to the Pythagorean theorem.

Alternatively, it follows from (2) that if the polar coordinates of the point $\mathbf{p}$ are $(r_1, \theta_1)$ and those of $\mathbf{q}$ are $(r_2, \theta_2)$, then the distance between the points is

$$\sqrt{r_1^2 + r_2^2 - 2r_1r_2\cos(\theta_1 - \theta_2)}.$$

**Three dimensions** [ edit ]

In three-dimensional Euclidean space, the distance is

$$d(p,q) = \sqrt{(p_1-q_1)^2 + (p_2-q_2)^2 + (p_3-q_3)^2}.$$





# Problem statement - 1

Ethans
Learn from experts

Find out the nearest point between several defined co ordinates?

Problem Statement can be related with Google map, to find out the shortest distance between two destination?

## Solution

Ethans
Learn from experts

```
# Shortest distance between two points
co_ordinates = {1:(3,10), 2:(1,9), 3:(10,20),
                4:(2,7), 5:(11,30), 6:(14,45)}

check_ordinates = (4,22)
out_check = {}
for key, value in co_ordinates.items():
    distance = 0
    for i in range(len(value)):
        distance = distance + \
                   (co_ordinates[key][i] - check_ordinates[i]) ** 2

    out_check[key] = distance ** 0.5

sorted_distance = sorted(out_check.items(), key=lambda x: x[1])
print '%r is near to co_ordinates %r ' \
      %(check_ordinates, co_ordinates[sorted_distance[0][0]])
```





## Data Visualization

Ethans
Learn from experts

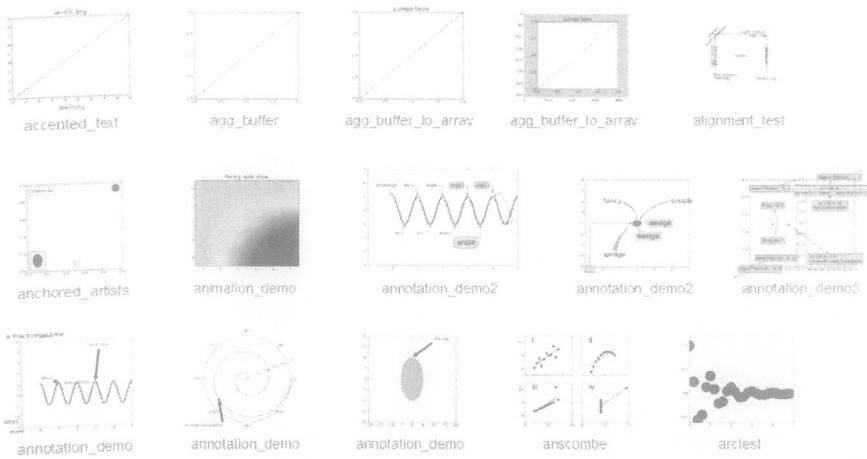
- Matplotlib is a Python package for 2D-graphics.
- It provides a way to visualize data from Python and provide quality figures in many formats
- For simple plotting the pyplot interface provides a MATLAB-like interface, particularly when combined with Python. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

To install matplot lib run:
*pip install matplotlib*

Matplot contains more than 100 figures and all of them can be accessed from:
http://matplotlib.org/gallery.html

## Sample Figures

Ethans
Learn from experts







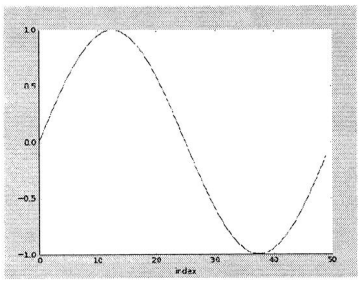

## Plot Graph

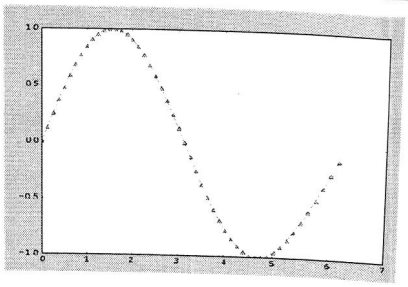
Ethans
Learn from experts

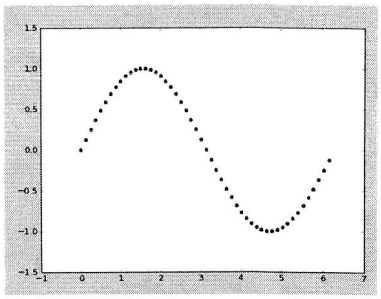
```
# Line Graph
import matplotlib.pyplot as plt
from numpy import *
from pylab import *
x = arange(50)*2*pi/50
y = sin(x)
plt.plot(y)
plt.xlabel('index')
plt.show()

# Line formating
plt.plot(x, sin(x), 'y-^')
plt.show()

# Scatter Graph
plt.scatter(x, y)
plt.show()
```

## Plot Graph

Ethans
Learn from experts







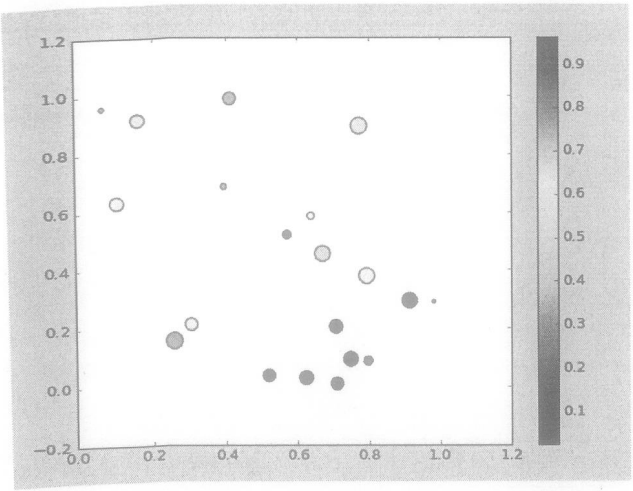

## Other Graphs

Ethans
Learn From experts

```
# Color bar graph
from scipy import *
x = rand(20)
y = rand(20)
size = rand(20)*200
color= rand(20)
plt.scatter(x, y, size , color)
plt.colorbar()
plt.show()

# Bar graph
plt.bar(x, y, width = x[1]-x[0])
plt.show()

# histogram
plt.hist(randn(20))
plt.show()
```

## Color Graph

Ethans
Learn from experts





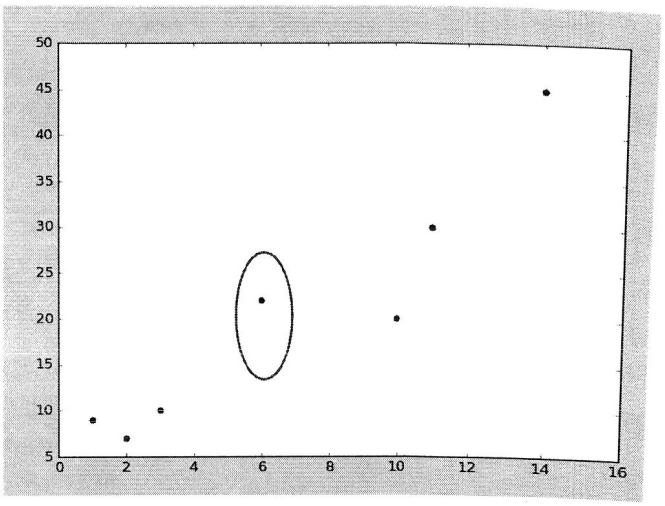



## Problem statement – 1 with Visuals

Ethans
Learn from experts

```
import matplotlib.pyplot as plt
from numpy import *
from pylab import *

xco_ordinates = array([3,1,10,2,11,14])
yco_ordinates = array([10,9,20,7,30,45])

# Scatter Graph
plt.scatter(xco_ordinates, yco_ordinates)
plt.show()

# After adding the points
xco_ordinates = array([3,1,10,2,11,14, 6])
yco_ordinates = array([10,9,20,7,30,45, 22])

# Scatter Graph
plt.scatter(xco_ordinates, yco_ordinates)
plt.show()
```

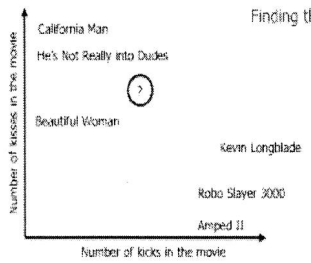
## Solution








## Understanding KNN Algorithim




| Movie Title | Distance to Movie "?" |
|---|---|
| California Man | 20.5 |
| He's Not really into Dudes | 18.7 |
| Beautiful Woman | 19.2 |
| Kevin Longblade | 115.3 |
| Robo Slayer | 117.4 |
| Amped II | 118.9 |

| Movie Title | # of kicks | # of kisses | Type of Movie |
|---|---|---|---|
| California Man | 3 | 104 | Romance |
| He's Not really into Dudes | 2 | 100 | Romance |
| Beautiful Woman | 1 | 81 | Romance |
| Kevin Longblade | 101 | 10 | Action |
| Robo Slayer | 99 | 5 | Action |
| Amped II | 98 | 2 | Action |

## Creating Training Set

Ethans
Learn from experts

```
import numpy as np
from operator import itemgetter

# Training Dataset
def create_Dataset(dataset, label):
    group = np.array(dataset)
    labels = label
    return group, labels
```





## Creating classifier

Ethans
Learn from experts

```
# Build a classifier
def classify(inX, dataset, labels, k):
    datasetSize = dataset.shape[0]
    # Taken x2-x1 and y2-y1 and so onnn
    diffMat = np.tile(inX, (datasetSize, 1)) - dataset
    # Taken the sqaure of elements
    sqdiffMat = diffMat ** 2
    # Add x + y
    sqDistances = sqdiffMat.sum(axis=1)
    # Add square root of the result
    distance = sqDistances ** 0.5
    # Sort them based on indexes and get the all indexes
    sortedDist = distance.argsort()
    classCount = {}
    for i in range(k):
        voteLabel = labels[sortedDist[i]]
        classCount[voteLabel] = classCount.get(voteLabel, 0) + 1

    sortedCount = sorted(classCount.iteritems(), key=itemgetter(1), reverse=Tru
    return sortedCount[0][0]
```

## Call KNN

Ethans Learn From experts

```
import KNN_algo as KNN

predefined_dataset = [[3, 104],[2,100],[1,81],[101, 10],[99, 5],[98, 2]]
label = ['Romance', 'Romance', 'Romance', 'Action', 'Action', 'Action']

group, labels = KNN.create_Dataset(predefined_dataset, label)

set1 = [18, 90]
#set2 = [2, 199]

print 'Set1 is near point:  ', KNN.classify(set1, group, labels, 3)
#print 'Set2 is near point:  ', KNN.classify(set2, group, labels, 3)
```





## Scikit-learn

Ethans Learn From experts

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under BSD and distribute under many Linux distributions, encouraging academic and commercial use.

The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

**NumPy**: Base n-dimensional array package
**SciPy**: Fundamental library for scientific computing
**Matplotlib**: Comprehensive 2D/3D plotting
**IPython**: Enhanced interactive console
**Sympy**: Symbolic mathematics
**Pandas**: Data structures and analysis

## What are the features?

Ethans
Learn from experts

The library is focused on modelling data but not focused on loading, manipulating and summarizing data. For these features, we either use NumPy and Pandas.

Some popular groups of models provided by scikit-learn include:

**Supervised Models**: a vast array not limited to generalized linear models, discriminate analysis, naive bayes, lazy methods, neural networks, support vector machines and decision trees.
**Clustering**: for grouping unlabeled data such as KMeans.
**Cross Validation**: for estimating the performance of supervised models on unseen data.
**Datasets**: for test datasets and for generating datasets.
**Dimensionality Reduction**: for reducing the number of attributes in data for summarization, visualization and feature selection such as Principal component analysis.
**Ensemble methods**: for combining the predictions of multiple supervised models.
**Feature extraction**: for defining attributes in image and text data.
**Feature selection**: for identifying meaningful attributes from which to create supervised models.
**Manifold Learning**: For summarizing and depicting complex multi-dimensional data.





## Install scikit-learn

Ethans
Learn from experts

pip install scikit-learn

**import it using**

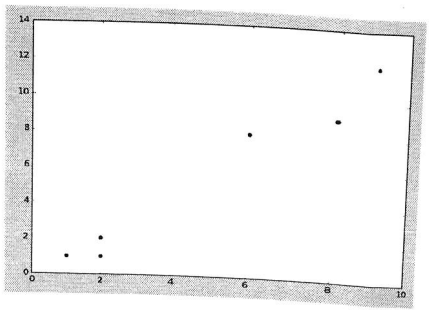
import sklearn

## Linear SVC

Ethans
Learn from experts

The Support Vector Machine, created by Vladimir Vapnik in the 60s, but pretty much overlooked until the 90s is still one of most popular machine learning classifiers.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm

x = [2, 6, 2, 8, 1, 9]
y = [1, 8, 2, 9, 1, 12]

plt.scatter(x, y)
plt.show()
```

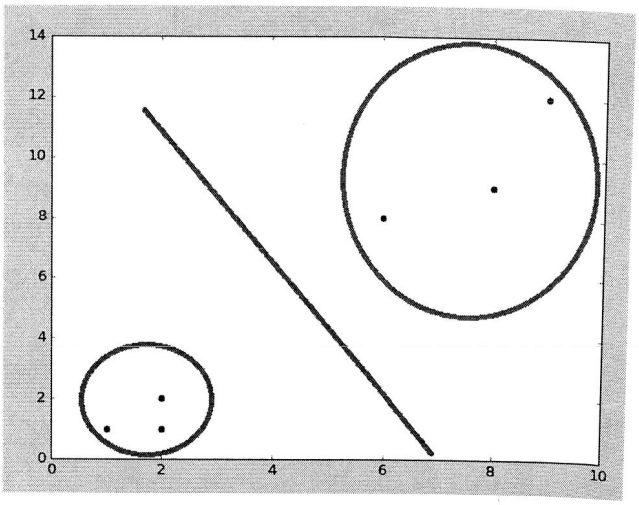



Slide 299

www.ethans.co.in

## How it works?

Ethans
Learn from experts



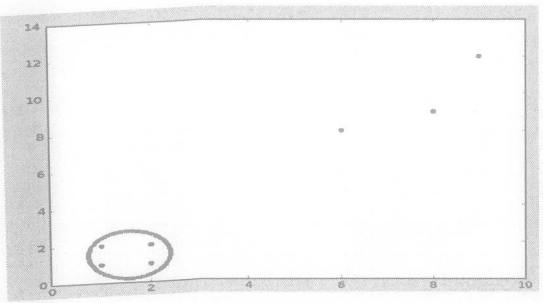

Slide 300

www.ethans.co.in

## Classifier

Ethans
Learn from experts

```
# Classifier starts here
from sklearn import svm
coordinates = np.array([[2,1],[6,8],[2,2],[8,9],[1,1],[9,12]])
labels = ['Action','Romance','Action','Romance','Action','Romance']
clf = svm.SVC(kernel='linear')
clf.fit(coordinates,labels)
print(clf.predict([1,2]))
```




Slide 301

www.ethans.co.in

## Clustering

Ethans
Learn from experts

The difference between supervised and unsupervised machine learning is whether or not we have provided the machine with labeled data.

Unsupervised machine learning is where we does not provide the machine with labeled data, and the machine is expected to derive structure from the data all on its own.

Slide 302

www.ethans.co.in

## Kmeans

Ethans
Learn From experts

```
from sklearn.cluster import KMeans
coordinates = np.array([[2,1],[6,8],[2,2],[8,9],[1,1],[9,12]])
kmeans = KMeans(n_clusters=2)
kmeans.fit(coordinates)

centroids = kmeans.cluster_centers_
labels = kmeans.labels_

print 'Printing all centroids: ', centroids
print 'Printing all labels: ', labels
```





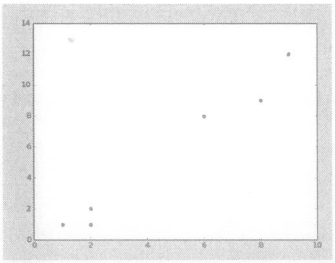
## Kmeans

Ethans
Learn From experts

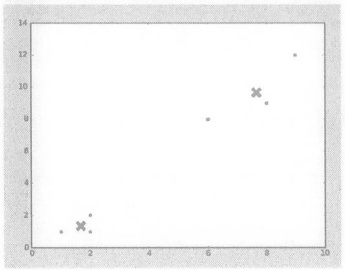
```
from sklearn.cluster import KMeans
coordinates = np.array([[2,1],[6,8],[2,2],[8,9],[1,1],[9,12]])
kmeans = KMeans(n_clusters=2)
kmeans.fit(coordinates)

centroids = kmeans.cluster_centers_
labels = kmeans.labels_

print 'Printing all centroids: ', centroids
print 'Printing all labels: ', labels
```

## Kmeans - Graph

Ethans
Learn from experts



Before Clustering them



After Clustering





## Objective – Module 12

Ethans
Learn from experts

**Python Pandas**

- What is Pandas
- Creating Series
- Creating Data Frames,
- Grouping, Sorting
- Plotting Data
- Data analysis with data set
- Practical use cases using data analysis.

## What is Pandas?

Ethans Learn from experts

- pandas is a Python package for providing fast, flexible, and expressive data structures.
- Data structures designed in pandas to make working with 'relational' or 'labeled' data
- Pandas is the most powerful and flexible open source data analysis / manipulation tool available in python.

- pandas is built on top of NumPy module and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.
- It handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.
- Source - http://pandas.pydata.org/pandas-docs/stable/

Slide 307

www.ethans.co.in

## Data Structure

Ethans Learn From experts

- The two primary data structures of pandas are built on top of NumPy they are Series (1-dimensional) and DataFrame (2-dimensional).

**These DS will help to do:**

- Easy handling of missing data and adding/removing of columns
- Automatic data alignment and powerful, flexible group by functionality to perform operations on data sets, for both aggregating and transforming data
- Intelligent label-based slicing, fancy indexing, and sub setting of large data sets
- Easy merging and joining data sets
- Flexible reshaping and pivoting of data sets
- Robust IO tools for loading data from flat files.

Slide 308

www.ethans.co.in

## Series

Ethans
Learn from experts

- Series is a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.).
- The axis labels are collectively referred to as the **index**

**# Create pandas Series**

```
>>> import pandas as pd
>>> S = pd.Series(data, index=index)
```

Data in above can be: List, Tuple, dict, NumPy array or any scalar value
Index – Should contains the label. **index** must be the same length as **data** .

```
>>> S = pd.Series('Ethans', index = ('Name',))
>>> S
Name     Ethans
```





## Series – With other objects

Ethans
Learn from experts

```
>>> import pandas as pd
>>> S = pd.Series([1,2,3])
>>> S
0    1
1    2
2    3
dtype: int64
>>> S = pd.Series((1,2,3))
>>> S
0    1
1    2
2    3
dtype: int32
>>> S = pd.Series({1:2, 3:4})
>>> S
1    2
3    4
dtype: int64
```

## Series – With numpy and LoL

Ethans
Learn from experts

```
>>> S = pd.Series(np.array([1,2,3]), index = (1,2,3))
>>> S
1    1
2    2
3    3
dtype: int32
>>>
>>> S = pd.Series(np.array([1,2,3]))
>>> S
0    1
1    2
2    3
dtype: int32
```

```
>>> employees = pd.Series([['ethans', 'Bob', 'Aaron'], [35, 40, 29]])
>>> employees
0    [ethans, Bob, Aaron]
1            [35, 40, 29]
dtype: object
```





## Create Series: All example

Ethans
Learn From experts

```
# Creating a series in Pandas
import pandas as pd
s1 = pd.Series([1,2,3]) # from list
s1 = pd.Series((1,2,3)) # from tuple
s1 = pd.Series(list('pandas')) # from string
s1 = pd.Series((1,2,3), index=(1,2,3)) # from tuple, with indexes
s1 = pd.Series({1:2,3:4}) # from dictionary
s1 = pd.Series({1:2,3:4}, index= (1,2,3,4,5)) # from dictionary

import numpy as np
s1 = pd.Series(np.array([1,2,3]), index=(1,2,3)) # from np, with indexes
s1 = pd.Series([['Jatin','Aakash','Rahul'], [35, 28, 31]],
               index=('names', 'age')) # from list of list
```

## Series – Get Index value

Ethans Learn from experts

```
>>> d = {'Pune': 900, 'Mumbai': 1300, 'Delhi': 900, 'Bangalore': 1100,
'Goa': 450,  'Daman': None}

>>> cities = pd.Series(d)
>>> cities
Bangalore     1100.0
Daman            NaN
Delhi          900.0
Goa            450.0
Mumbai        1300.0
Pune           900.0
dtype: float64
```

```
>>> cities['Pune']
900.0
>>> cities[['Pune', 'Mumbai', 'Goa']]
Pune          900.0
Mumbai       1300.0
Goa           450.0
dtype: float64
```

Slide 313

www.ethans.co.in

## Series – Call basic functions

Ethans Learn from experts

```
>>> cities.isnull()
Bangalore     False
Daman          True
Delhi         False
Goa           False
Mumbai        False
Pune          False
dtype: bool
>>> cities.notnull()
Bangalore      True
Daman         False
Delhi          True
Goa            True
Mumbai         True
Pune           True
dtype: bool
>>> cities[cities.isnull()]
Daman   NaN
```

```
>>> cities < 1000
Bangalore    False
Daman        False
Delhi         True
Goa           True
Mumbai       False
Pune          True
dtype: bool
>>> cities[cities < 1000]
Delhi    900.0
Goa      450.0
Pune     900.0
dtype: float64
>>> cities[cities == 900]
Delhi    900.0
Pune     900.0
dtype: float64
```

Slide 314

www.ethans.co.in

## Series – Call basic functions

Ethans
Learn From experts

```
>>> cities * 2
Bangalore     2200.0
Daman            NaN
Delhi         1800.0
Goa            900.0
Mumbai        2600.0
Pune          1800.0
dtype: float64
>>> np.square(cities)
Bangalore     1210000.0
Daman               NaN
Delhi          810000.0
Goa            202500.0
Mumbai        1690000.0
Pune           810000.0
dtype: float64
```

```
>>> cities['Pune'] = 800
>>> cities[cities > 1000] = 800
>>> cities
Bangalore     800.0
Daman           NaN
Delhi         900.0
Goa           450.0
Mumbai        800.0
Pune          800.0
dtype: float64
>>> 'Chennai' in cities
False
>>> 'Daman' in cities
True
```





## Data Frame

Ethans
Learn From experts

- Data Frame is two-dimensional labeled array capable of holding any data type. DataFrame is a tabular data structure comprised of rows and columns, like a spreadsheet, database table, or R's data.frame object.

**# Create pandas Data Frame**

```
>>> import pandas as pd
>>> df = pd.DataFrame({'names':['Ethan', 'Bob', 'Aaron'],
                       'Age':[30, 35, 40],
                       'City':['Pune', 'New York', 'Texas']})
>>>
>>> df
   Age      City  names
0   30      Pune  Ethan
1   35  New York    Bob
2   40     Texas  Aaron
```

## Data Frame – with Series

Ethans
Learn from experts

```
>>> d = {'one' : pd.Series([1., 2., 3.], index=['a', 'b', 'c']),
    'two' : pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd'])}

>>>
>>> df1 = pd.DataFrame(d)
>>> df1
   one  two
a  1.0  1.0
b  2.0  2.0
c  3.0  3.0
d  NaN  4.0
```

```
>>> df1 = pd.DataFrame(d, index = ['d', 'c', 'a'])
>>> df1
   one  two
d  NaN  4.0
c  3.0  3.0
a  1.0  1.0
```

```
>>> df1 = pd.DataFrame(d, index = ['d', 'c', 'a'],
    columns = ['one', 'newOne'])
>>> df1
   one newOne
d  NaN    NaN
c  3.0    NaN
a  1.0    NaN
```





## Data Frame – with LoD

Ethans
Learn from experts

```
>>> d = [{'a': 1, 'b': 2}, {'a': 3, 'b': 4, 'c': 5}]
>>> df = pd.DataFrame(d)
>>> df
   a  b    c
0  1  2  NaN
1  3  4  5.0
>>> pd.DataFrame(d, index=['first', 'second'])
        a  b    c
first   1  2  NaN
second  3  4  5.0
>>> pd.DataFrame(d, columns=['a', 'b'])
   a  b
0  1  2
1  3  4
```

## selection, addition and deletion

Ethans Learn from experts

```
>>> D = {'India':[1, .4,'Delhi'], 'China': [1.5, .2, 'Beijing']}
>>> df = pd.DataFrame(D, index = ['Population', 'PD', 'Capital'])
>>> df
            China  India
Population    1.5      1
PD            0.2    0.4
Capital   Beijing  Delhi
>>> df['India']
Population      1
PD            0.4
Capital     Delhi
Name: India, dtype: object
>>>
>>> df['India']['PD']
0.4
>>> df['Brazil'] = [.5, .4, 'Brasilia']
>>> df
            China  India     Brazil
Population    1.5      1        0.5
PD            0.2    0.4        0.4
Capital   Beijing  Delhi   Brasilia
```

```
>>> del df['China']
>>> df
            India     Brazil
Population      1        0.5
PD            0.4        0.4
Capital     Delhi   Brasilia
```

Slide 319

www.ethans.co.in

## Functions

Ethans Learn from experts

```
>>> data = {'Country':['India', 'China', 'Brazil'],
        'Year':[2013, 2014, 2015],
        'Population':[1,1.5, .5]}
>>>
>>> df = pd.DataFrame(data)
>>> df
  Country  Population  Year
0   India         1.0  2013
1   China         1.5  2014
2  Brazil         0.5  2015
```

```
>>> df.describe()
       Population     Year
count        3.00      3.0
mean         1.00   2014.0
std          0.50      1.0
min          0.50   2013.0
25%          0.75   2013.5
50%          1.00   2014.0
75%          1.25   2014.5
max          1.50   2015.0
```

```
>>> df.sum()
Country          IndiaChinaBrazil
Population                    3.0
Year                         6042
dtype: object
>>> df.mean()
Population       1.0
Year          2014.0
dtype: float64
>>> df.max()
Country         India
Population        1.5
Year             2015
```

```
>>> df.head(1)
  Country  Population  Year
0   India         1.0  2013
>>> df.tail(1)
  Country  Population  Year
2  Brazil         0.5  2015
```

Slide 320

www.ethans.co.in