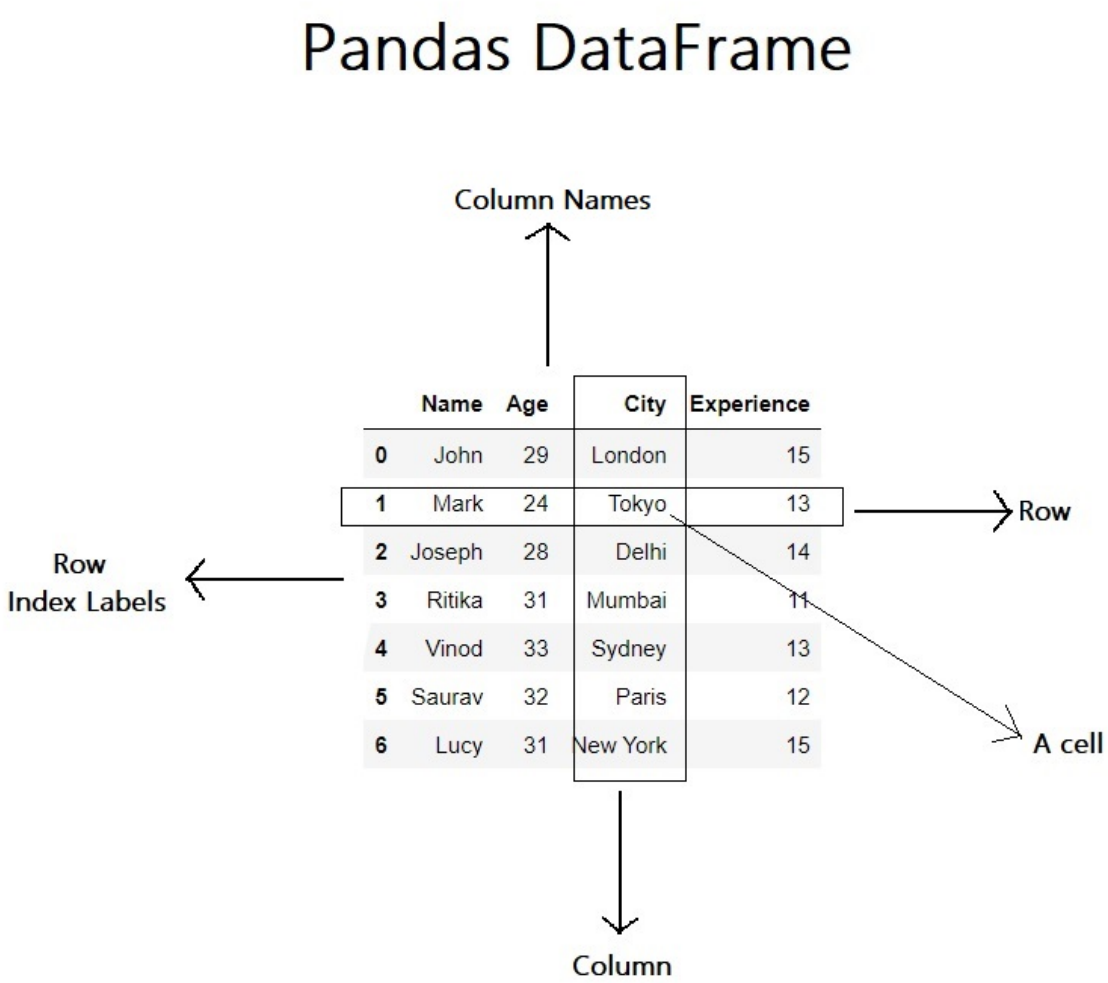
Pandas Dataframe

- A DataFrame is a two-dimensional data structure with heterogeneous data.
- A DataFrame is made up of a collection of Series.
- Pandas DataFrame consists of three principal components, the data, rows, and columns.

Figure 1:

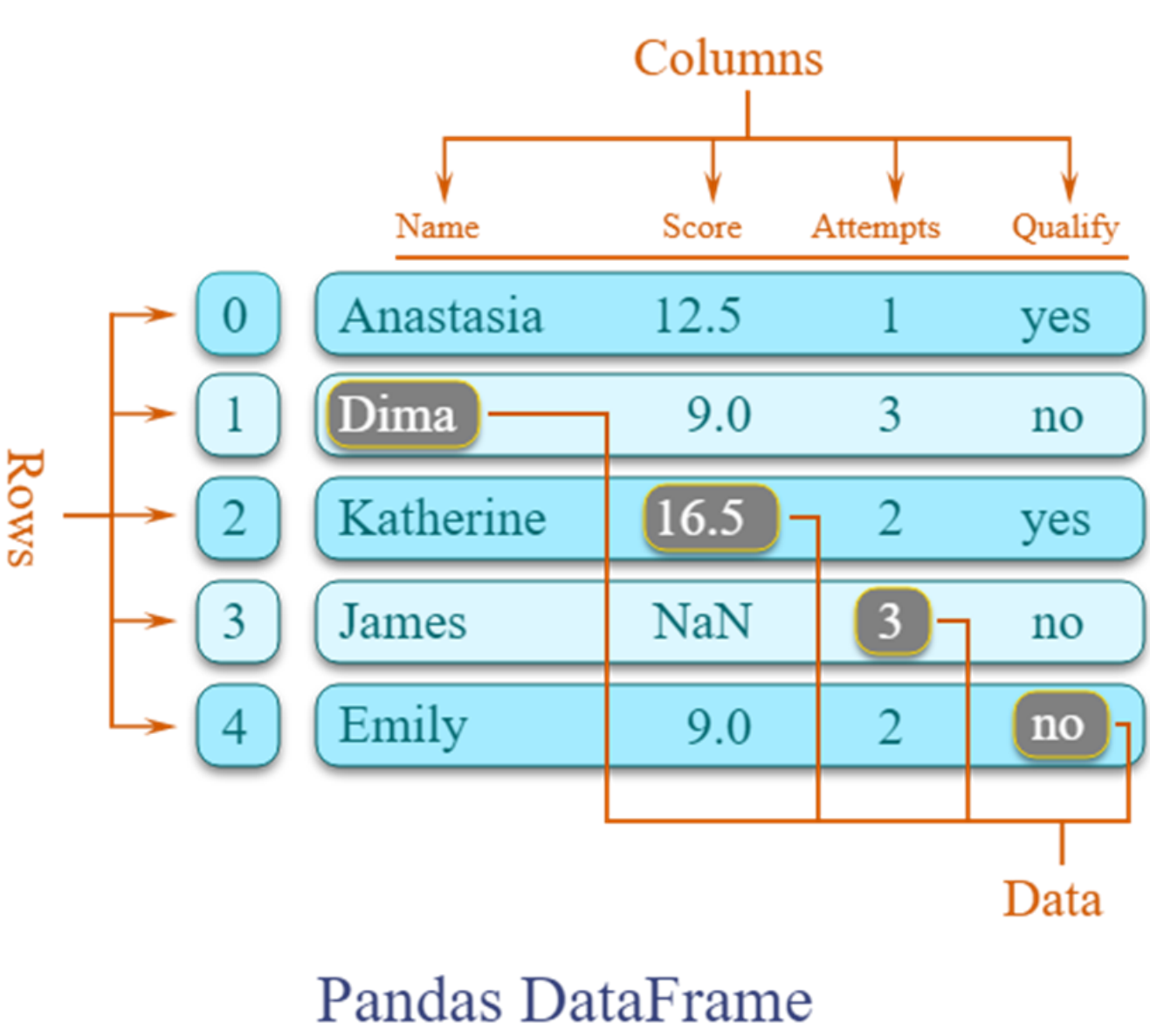


	Name	Age	City	Experience
0	John	29	London	15
1	Mark	24	Tokyo	13
2	Joseph	28	Delhi	14
3	Ritika	31	Mumbai	11
4	Vinod	33	Sydney	13
5	Saurav	32	Paris	12
6	Lucy	31	New York	15

Each row as has an index label associated with it and each column has a column name associated with it.

We can select and process individual rows, columns or cells in DataFrame.

Figure 2:



Pandas DataFrame

How to Create DataFrame?

- The Pandas module provides a function Dataframe(), which accepts a data as the argument and returns a DataFrame object containing the given elements.

Syntax:

> **pandas.DataFrame(data, index, columns, dtype)**

Parameters:

> **data:** data takes various forms like ndarray, series, map, lists, dict, constants and also another DataFrame.

> **columns:** For column labels

> **index:** For the row labels, the Index to be used for frame (optional)

> **dtype:** Data type of each column (optional)

> **name:** str (optional)

- In DataFrame() the data can be:
    1. A Python list, tuple, dictionary
        - List of array, list, tuple, dictionary, Series
        - Dictionary of array, lists, tuple, Dictionaries

2. A 2D NumPy array
3. A Series
4. CSV Files
5. Another DataFrame

Note:

- *These are called Methods of creating DataFrames.*

Note: In the real world, a Pandas DataFrame will be created by loading the datasets from existing storage, storage can be SQL Database, CSV file, and Excel file.

Creating an empty DataFrame

```
#import the pandas library and aliasing as pd
import pandas as pd

d = pd.DataFrame()

print(d)
```

```
Empty DataFrame
Columns: []
Index: []
```

Different Methods of Creating DataFrame:

Method 1: Creating DataFrame using a single tuple/list

```
# import pandas as pd
import pandas as pd

# Creating a DataFrame using Tuple
df = pd.DataFrame(('Python', 'For', 'Data', 'Science', 'AI', 'ML', 'DL')
 )
print(df)
```

```
           0
0   Python
1      For
2     Data
3  Science
4       AI
5       ML
6       DL
```

```
# import pandas as pd
import pandas as pd

# Creating a DataFrame using List
df = pd.DataFrame(['Python', 'For', 'Data', 'Science', 'AI', 'ML', 'DL'])
print(df)
```

```
           0
0   Python
1      For
2     Data
3  Science
4       AI
```

```
5       ML
```

Method 2: Creating DataFrame using a list of list.

```
import pandas as pd

#Creating DataFrame using a list of list
df = pd.DataFrame([['Alex',10],['Bob',12],['Clarke',13]])

print(df)
```

```
          0   1
0    Alex  10
1     Bob  12
2  Clarke  13
```

Method 3: Creating DataFrame using a list of dictionary.

```
import pandas as pd

#Creating DataFrame using a list of dictionary
df = pd.DataFrame([{'Alex':10, 'Bob':20, 'Clarke':30},{'Bob':12},{'Clarke':13}])

print(df)
```

```
   Alex   Bob  Clarke
0  10.0  20.0    30.0
1   NaN  12.0     NaN
2   NaN   NaN    13.0
```

Method 4: Creating a Dataframe using dictionary of lists

```
import pandas as pd

#Creating a Dataframe using dictionary of lists
df = pd.DataFrame({'Name':['Tom', 'nick', 'krish', 'jack'], 'Age':[20, 21, 19, 18]})

# Print the output.
print(df)
```

```
    Name  Age
0    Tom   20
1   nick   21
2  krish   19
3   jack   18
```

```
import pandas as pd
# Create a dictionary of lists
employees = { 'Name': ['John', 'Mark', 'Joseph', 'Ritika', 'Vinod', 'Saurav', 'Lucy'],
              'Age': [29, 24, 28, 31, 33, 32, 31],
              'City': ['London', 'Tokyo', 'Delhi', 'Mumbai', 'Sydney', 'Paris', 'New York'],
              'Experience': [15, 13, 14, 11, 13, 12, 15]}

# Create a Pandas DataFrame from Dictionaries
df = pd.DataFrame(employees)

# Display the DataFrame
print(df)
```

```
     Name  Age      City  Experience
0    John   29    London          15
1    Mark   24     Tokyo          13
2  Joseph   28     Delhi          14
```

```
3  Ritika   31    Mumbai          11
4   Vinod   33    Sydney          13
5  Saurav   32     Paris          12
6    Lucy   31  New York          15
```

Method 5: Creating DataFrame using a NumPy Array.

Import the Pandas and Numpy modules.

Create a Numpy array.

Create list of index values and column values for the DataFrame.

Create the DataFrame.

Display the DataFrame.

```
import numpy as np
# creating the Numpy array

df = pd.DataFrame({
'name':['Jane','John','Ashley','Mike','Emily','Jack','Catlin'],
'ctg':['A','A','C','B','B','C','B'],
'val':np.random.random(7).round(2),
'val2':np.random.randint(1,10, size=7),
'val3':np.array(['Aditya', 20,'a',30,'Emily',40,50])})

print(df)
```

```
     name ctg   val  val2    val3
0    Jane   A  0.25     6  Aditya
1    John   A  0.08     7      20
2  Ashley   C  0.92     3       a
3    Mike   B  0.76     4      30
4   Emily   B  0.83     6   Emily
5    Jack   C  0.97     3      40
6  Catlin   B  0.34     5      50
```

```
# importing the modules
import pandas as pd
import numpy as np

# creating the Numpy array
df = pd.DataFrame({
'name':['Jane','John','Ashley','Mike','Emily','Jack','Catlin'],
'ctg':['A','A','C','B','B','C','B'],
'val':np.random.random(7).round(2),
'val2':np.random.randint(1,10, size=7)})

print(df)
```

```
     name ctg   val  val2
0    Jane   A  0.62     8
1    John   A  0.63     4
2  Ashley   C  0.80     5
3    Mike   B  0.00     6
4   Emily   B  0.98     5
5    Jack   C  0.38     9
6  Catlin   B  0.56     2
```

Method 6: Creating DataFrame using a Series.

Creating Dataframe from multiple Series

```
#Importing Pandas library
import pandas as pd

#Creating a DataFrame using Series
a = pd.DataFrame(pd.Series(['Jit', 'Purn', 'Arp', 'Jot']))

#Printing elements of Dataframe
print(a)
```

```
      0
0   Jit
1  Purn
2   Arp
3   Jot
```

Method 7: Creating Dataframe from .csv files

Pandas module provides a function read_csv() to create a DataFrame from .csv files.

It takes the csv file path or name as argument and imports the content of a csv file into a Dataframe object.

```
import pandas as pd

df = pd.read_csv ('/content/drive/MyDrive/Automobile.csv')

print(df)
```

```
      2           audi        sedan   176.6    ohc       24     Yes
      3           audi        sedan   176.6   None       18     Yes
      4           audi        sedan   177.3    ohc       19     Yes
      5           audi        wagon   192.7    ohc       19     Yes
      6            bmw        sedan   176.8    ohc       23     Yes
      7            bmw        sedan   176.8    ohc       23     Yes
      8            bmw        sedan   176.8    ohc       21     Yes
      9            bmw        sedan   189.0    ohc       16     Yes
      10           bmw        sedan   193.8    ohc       16     Yes
      11           bmw        sedan   197.0    ohc       15     Yes
      12     chevrolet    hatchback   141.1    ohc       47      No
      13     chevrolet    hatchback   155.9    ohc       38      No
      14     chevrolet        sedan   158.8    ohc       38      No
      15         dodge    hatchback   157.3    ohc       31      No
      16         dodge    hatchback   157.3    ohc       31      No
      17         honda        wagon   157.1    ohc       30      No
      18         honda        sedan   175.4    ohc       24      No
      19         honda        sedan   169.1    ohc       25      No
      20         isuzu        sedan   170.7    ohc       24      No
      21         isuzu        sedan   155.9    ohc       38      No
      22         isuzu        sedan   155.9    ohc       38      No
      23        jaguar        sedan   199.6   dohc       15     Yes
      24        jaguar        sedan   199.6   dohc       15     Yes
      25        jaguar        sedan   191.7   ohcv       13     Yes
      26         mazda    hatchback   159.1    ohc       30     Yes
      27         mazda    hatchback   159.1    ohc       31     Yes
      28         mazda    hatchback   159.1    ohc       31     Yes
      29         mazda    hatchback   169.0  rotor       17     Yes
      30         mazda        sedan   175.0    ohc       31     Yes
      31  mercedes-benz        sedan   190.9    ohc       22     Yes
      32  mercedes-benz        wagon   190.9    ohc       22     Yes
      33  mercedes-benz        sedan   208.1   ohcv       14     Yes
      34  mercedes-benz      hardtop   199.2   ohcv       14     Yes
      35     mitsubishi    hatchback   157.3    ohc       37     Yes
      36     mitsubishi    hatchback   157.3    ohc       31     Yes
      37     mitsubishi        sedan   172.4    ohc       25     Yes
      38     mitsubishi        sedan   172.4    ohc       25     Yes
      39         nissan        sedan   165.3    ohc       45      No
      40         nissan        sedan   165.3    ohc       31      No
      41         nissan        sedan   165.3    ohc       31      No
      42         nissan        wagon   170.2    ohc       31      No
      43         nissan        sedan   184.6   ohcv       19      No
      44        porsche      hardtop   168.9   ohcf       17     Yes
      45        porsche  convertible   168.9   ohcf       17     Yes
```

```
46        porsche    hatchback   175.7  dohcv       17     Yes
47         toyota    hatchback   158.7    ohc       35      No
48         toyota    hatchback   158.7    ohc       31      No
49         toyota    hatchback   158.7    ohc       31      No
50         toyota        wagon   169.7    ohc       31      No
51         toyota        wagon   169.7    ohc       27      No
52         toyota        wagon   169.7    ohc       27      No
53         toyota        wagon   187.8   dohc       19      No
54     volkswagen        sedan   171.7    ohc       37     Yes
55     volkswagen        sedan   171.7    ohc       27     Yes
56     volkswagen        sedan   171.7    ohc       37     Yes
57     volkswagen        sedan   171.7    ohc       26     Yes
58          volvo        sedan   188.8    ohc       23      No
59          volvo        wagon   188.8    ohc       23      No
```

```
import pandas as pd
df = pd.read_csv ('/content/drive/MyDrive/nba.csv')
print(df)
```

```
                Name            Team  Number Position   Age Height  Weight  \
0    Avery Bradley  Boston Celtics     0.0       PG  25.0    6-2   180.0
1      Jae Crowder  Boston Celtics    99.0       SF  25.0    6-6   235.0
2     John Holland  Boston Celtics    30.0       SG  27.0    6-5   205.0
3      R.J. Hunter  Boston Celtics    28.0       SG  22.0    6-5   185.0
4    Jonas Jerebko  Boston Celtics     8.0       PF  29.0   6-10   231.0
..             ...             ...     ...      ...   ...    ...     ...
453   Shelvin Mack       Utah Jazz     8.0       PG  26.0    6-3   203.0
454      Raul Neto       Utah Jazz    25.0       PG  24.0    6-1   179.0
455   Tibor Pleiss       Utah Jazz    21.0        C  26.0    7-3   256.0
456    Jeff Withey       Utah Jazz    24.0        C  26.0    7-0   231.0
457            NaN             NaN     NaN      NaN   NaN    NaN     NaN

                College     Salary
0                 Texas  7730337.0
1             Marquette  6796117.0
2    Boston University        NaN
3         Georgia State  1148640.0
4                   NaN  5000000.0
..                  ...        ...
453             Butler  2433333.0
454                NaN   900000.0
455                NaN  2900000.0
456             Kansas   947276.0
457                NaN        NaN

[458 rows x 9 columns]
```

Additional Points:

Creating DataFrame using another DataFrame.

Create New DataFrame Using Multiple Columns from Old DataFrame

Syntax:

> new_data_frame = old_data_frame[['col1','col2']].copy()

```
new = result[['Student']].copy()

print(new)
```

```
     Student
0  Jitender
```

```
1   Purnima
2     Arpit
3     Jyoti
```

Data alignment and arithmetic

Data alignment between DataFrame objects automatically align on both the columns and the index (row labels). Again, the resulting object will have the union of the column and row labels.

```
import numpy as np
import pandas as pd

df = pd.DataFrame(np.random.randn(8, 4), columns=['P', 'Q', 'R', 'S'])

df2 = pd.DataFrame(np.random.randn(9, 3), columns=['P', 'Q', 'R'])
```

```
df + df2
```

	P	Q	R	S
0	1.204556	-0.922001	-0.607577	NaN
1	1.606281	-1.363708	-0.765690	NaN
2	-0.898783	-1.542265	-0.776271	NaN
3	0.798002	1.668014	2.529941	NaN
4	0.860325	-3.006297	2.014316	NaN
5	-1.659224	-0.210117	3.188818	NaN
6	-0.635862	-0.523173	-1.195987	NaN
7	0.537766	0.243515	0.101564	NaN
8	NaN	NaN	NaN	NaN

```
df * 4 + 2
```

	P	Q	R	S
0	0.459102	4.128742	-1.173893	4.102802
1	1.584004	0.064187	4.365017	9.455785
2	-1.580716	0.896264	4.894933	5.029261
3	-1.503916	4.551874	12.651901	-2.351067
4	2.475362	-5.701144	3.996747	2.608698
5	0.206815	-0.050143	11.086625	-4.427300
6	-3.705365	12.012445	-1.613883	-1.301659
7	6.086888	5.602417	-1.667183	4.939936

```
1 / df
```

	P	Q	R	S
0	-2.595888	1.879044	-1.260282	1.902223
1	-9.615468	-2.066316	1.691320	0.536496
2	-1.117095	-3.624054	1.381724	1.320454
3	-1.141580	1.567475	0.375520	-0.919315

```
print(df.round(2))

df1= df **2
print(df1)
print(df1.round(2))
```

```
      P     Q     R     S
0 -0.39  0.53 -0.79  0.53
1 -0.10 -0.48  0.59  1.86
2 -0.90 -0.28  0.72  0.76
3 -0.88  0.64  2.66 -1.09
4  0.12 -1.93  0.50  0.15
5 -0.45 -0.51  2.27 -1.61
6 -1.43  2.50 -0.90 -0.83
7  1.02  0.90 -0.92  0.73
          P         Q         R         S
0  0.148398  0.283221  0.629600  0.276361
1  0.010816  0.234211  0.349582  3.474296
2  0.801345  0.076140  0.523790  0.573526
3  0.767339  0.407004  7.091437  1.183237
4  0.014123  3.706726  0.249188  0.023157
5  0.200970  0.262693  5.160422  2.581886
6  2.034449  6.265565  0.816259  0.681310
7  1.043916  0.811088  0.840515  0.540201
      P     Q     R     S
0  0.15  0.28  0.63  0.28
1  0.01  0.23  0.35  3.47
2  0.80  0.08  0.52  0.57
3  0.77  0.41  7.09  1.18
4  0.01  3.71  0.25  0.02
5  0.20  0.26  5.16  2.58
6  2.03  6.27  0.82  0.68
7  1.04  0.81  0.84  0.54
```

Boolean operators work as well:

```
df1 = pd.DataFrame({'x': [1, 0, 1], 'y': [0, 1, 1]}, dtype=bool)

df2 = pd.DataFrame({'x': [0, 1, 1], 'y': [1, 1, 0]}, dtype=bool)
```

```
df1 & df2
```

	x	y
0	False	False
1	False	True
2	True	False

```
df1 | df2
```

	x	y
0	True	True

```
df1 ^ df2
```

	x	y
0	True	True
1	True	False
2	False	True

```
-df1
```

	x	y
0	False	True
1	True	False
2	False	False