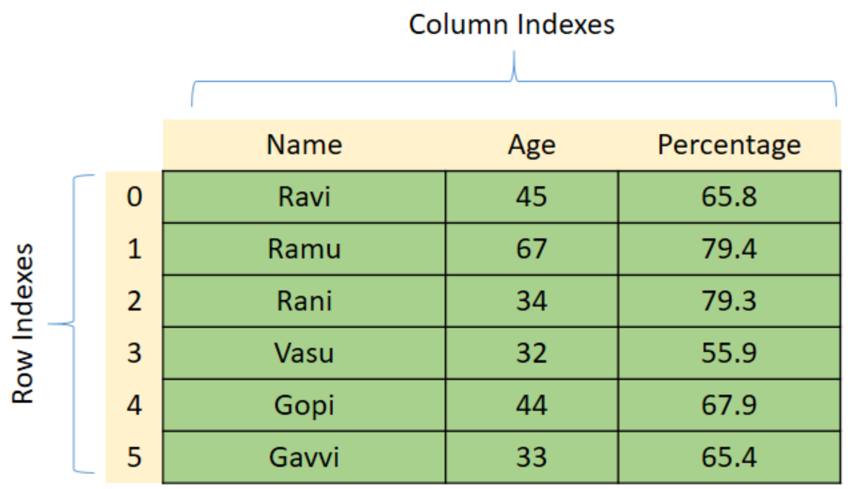
pandas.DataFrame

- **Two dimensional data structure**
- **Used to store tabular data --> rows and column with row index and column index**
- **Every column can have its own type of data.**
- **We can think of it like a spreadsheet or database table.**

¶



	Name	Age	Percentage
0	Ravi	45	65.8
1	Ramu	67	79.4
2	Rani	34	79.3
3	Vasu	32	55.9
4	Gopi	44	67.9
5	Gavvi	33	65.4

In [1]:
```
# Import pandas
import pandas as pd
```

DataFrame Creation

Creation of Empty DataFrame

In [2]:
```
dfempty = pd.DataFrame()
print(dfempty)
```

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

Creation of DataFrame from dictionary of lists

Here each item(key:value pair) wiil be a column in the dataframe

In [3]:
```
dict1 = {"Name":["Ramu","Rani","Raju"], "Age":[35, 45, 65], "Percentage":[67.5,89.6,98.7]}

df1 = pd.DataFrame(dict1)

print(df1)
```

```
   Name  Age  Percentage
0  Ramu   35        67.5
1  Rani   45        89.6
2  Raju   65        98.7
```

Creation of DataFrame from list of dictionaries

Each dictionary in the list will become a row in the dataframe

In [4]:
```
list1 = [{"Name":"Ramu", "Age":35, "Percentage":67.5}, {"Name":"Rani", "Age":45, "Percentage":89.6}, {"Name":"Raju", "Age":65, "P

df2 = pd.DataFrame(list1)

print(df2)
```

```
   Name  Age  Percentage
0  Ramu   35        67.5
1  Rani   45        89.6
2  Raju   65        98.7
```

Creation of a DataFrame from list of lists

Each inner list will be a row in the dataframe

We can assign row indices and column names while creating a dataframe with index and columns attributes

In [2]:
```
list2 = [[10, 100, 1000], [20, 200, 2000]]

df2 = pd.DataFrame(list2)

print(df2)
```

```
    0    1     2
0  10  100  1000
1  20  200  2000
```

In [3]:
```

df22 = pd.DataFrame(list2, index = ['a','b'], columns = ["Tens", "Hundrads", "Thousands"])

print(df22)
```

```
   Tens  Hundrads  Thousands
a    10       100       1000
b    20       200       2000
```

In [4]:
```
# Change the labels for rows and columns
df2.index = ["r1", "r2"]
df2.columns = ["10s", "100s", "1000s"]

print(df2)
```

```
    10s  100s  1000s
r1   10   100   1000
r2   20   200   2000
```

Working with CSV/Excel files

- CSV --> Comma Separated Values

Importing CSV file to a DataFrame

- **pandas.read_csv(path)**: Function to create a DataFrame from a csv file.

In [11]:
```
df3 = pd.read_csv(r"c:\Users\admin\Desktop\Employee1.csv")
print(df3)
```

```
      ID      Name Department  Salary
0  ID001    Kishor        ECE    5000
1  ID002     Bhanu        CSE    4000
2  ID003  Srikanth         IT    4500
3  ID004    Harish         ME    2000
```

- **pandas.read_csv(path, names = [list_of_column_names])** : names keyword argument can be used when file is not having header.

In [13]:
```
df4 = pd.read_csv(r"c:\Users\admin\Desktop\Employee2.csv", names = ["ID", "Name", "Department", "Salary"])
print(df4)
```

```
      ID      Name Department  Salary
0  ID001    Kishor        ECE    5000
1  ID002     Bhanu        CSE    4000
2  ID003  Srikanth         IT    4500
3  ID004    Harish         ME   42000
```

Exporting DataFrame to a CSV file

- **pandas.DataFrame.to_csv(path)**: Copies the contents with row index to a csv file specified by path.

```
In [14]: df4.to_csv(r"c:\Users\admin\Desktop\Employee3.csv")
```

- **pandas.DataFrame.to_csv(path, index = False)**: Copies the contents without row index to a csv file specified by path.

```
In [15]: df4.to_csv(r"c:\Users\admin\Desktop\Employee4.csv", index = False)
```

Working with Excel

- **pandas.read_excel(path)**: Function to create a DataFrame from an excel file.
- **pandas.to_excel(path)**: Function to save a DataFrame to a excel file.

```
In [ ]:
```