```
import numpy as np
import pandas as pd
```

Series is 1D and DataFrames are 2D objects

- But why?
- And what exactly is index?

```
# can we have multiple index? Let's try
index_val = [('cse',2019),('cse',2020),('cse',2021),('cse',2022),('ece',2019),('ece',2020),('ece',2021),('ec
a = pd.Series([1,2,3,4,5,6,7,8],index=index_val)
a
```

```
# The problem?
a['cse']
```

```
# The solution -> multiindex series(also known as Hierarchical Indexing)
# multiple index levels within a single index
```

```
# how to create multiindex object
# 1. pd.MultiIndex.from_tuples()
index_val = [('cse',2019),('cse',2020),('cse',2021),('cse',2022),('ece',2019),('ece',2020),('ece',2021),('ec
multiindex = pd.MultiIndex.from_tuples(index_val)
multiindex.levels[1]
# 2. pd.MultiIndex.from_product()
pd.MultiIndex.from_product([['cse','ece'],[2019,2020,2021,2022]])
```

```
# level inside multiindex object
```

```
# creating a series with multiindex object
s = pd.Series([1,2,3,4,5,6,7,8],index=multiindex)
s
```

```
# how to fetch items from such a series
s['cse']
```

```
# a logical question to ask
```

```
# unstack
temp = s.unstack()
temp
```

```
# stack
temp.stack()
```

```
# Then what was the point of multiindex series?
# multiindex dataframe
```

```
branch_df1 = pd.DataFrame(
    [
        [1,2],
        [3,4],
        [5,6],
        [7,8],
        [9,10],
        [11,12],
        [13,14],
        [15,16],
    ],
    index = multiindex,
    columns = ['avg_package','students']
)

branch_df1
```

```
branch_df1['students']
```

```
# Are columns really different from index?
```

```
# multiindex df from columns perspective
branch_df2 = pd.DataFrame(
    [
        [1,2,0,0],
        [3,4,0,0],
        [5,6,0,0],
        [7,8,0,0],
    ],
    index = [2019,2020,2021,2022],
    columns = pd.MultiIndex.from_product([['delhi','mumbai'],['avg_package','students']])
)

branch_df2
```

```
branch_df2.loc[2019]
```

```
# Multiindex df in terms of both cols and index

branch_df3 = pd.DataFrame(
    [
        [1,2,0,0],
        [3,4,0,0],
        [5,6,0,0],
        [7,8,0,0],
        [9,10,0,0],
        [11,12,0,0],
        [13,14,0,0],
        [15,16,0,0],
    ],
    index = multiindex,
    columns = pd.MultiIndex.from_product([['delhi','mumbai'],['avg_package','students']])
)
```

```
branch_df3
```

Stacking and Unstacking

```
branch_df3.stack().stack()
```

Working with multiindex dataframes

```
# head and tail
branch_df3.head()
# shape
branch_df3.shape
# info
branch_df3.info()
# duplicated -> isnull
branch_df3.duplicated()
branch_df3.isnull()
```

```
# Extracting rows single
branch_df3.loc[('cse',2022)]
```

```
# multiple
branch_df3.loc[('cse',2019):('ece',2020):2]
```

```
# using iloc
branch_df3.iloc[0:5:2]
```

```
# Extracting cols
branch_df3['delhi']['students']
```

```
branch_df3.iloc[:,1:3]
```

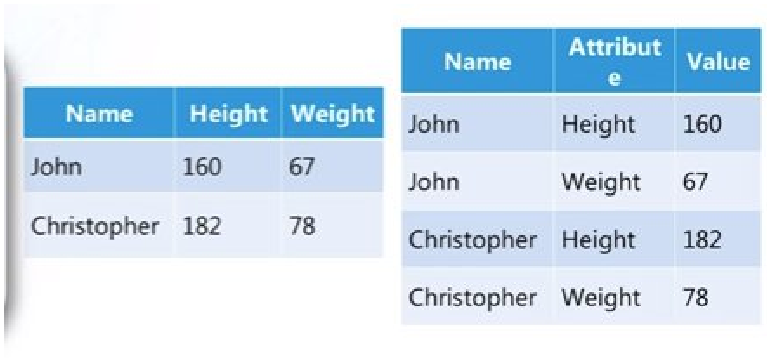
```
# Extracting both
branch_df3.iloc[[0,4],[1,2]]
```

```
# sort index
# both -> descending -> diff order
# based on one level
branch_df3.sort_index(ascending=False)
branch_df3.sort_index(ascending=[False,True])
branch_df3.sort_index(level=0,ascending=[False])
```

```
# multiindex dataframe(col) -> transpose
branch_df3.transpose()
```

```
# swaplevel
branch_df3.swaplevel(axis=1)
```

Long Vs Wide Data



Name	Height	Weight
John	160	67
Christopher	182	78

Name	Attribute	Value
John	Height	160
John	Weight	67
Christopher	Height	182
Christopher	Weight	78

Wide format is where we have a single row for every data point with multiple columns to hold the values of various attributes.

Long format is where, for each data point we have as many rows as the number of attributes and each row contains the value of a particular attribute for a given data point.

```
# melt -> simple example branch
# wide to long
pd.DataFrame({'cse':[120]}).melt()
```

```
# melt -> branch with year
pd.DataFrame({'cse':[120],'ece':[100],'mech':[50]}).melt(var_name='branch',value_name='num_students')
```

```
pd.DataFrame(
    {
        'branch':['cse','ece','mech'],
        '2020':[100,150,60],
        '2021':[120,130,80],
        '2022':[150,140,70]
    }
).melt(id_vars=['branch'],var_name='year',value_name='students')
```

```
# melt -> real world example
death = pd.read_csv('/content/time_series_covid19_deaths_global.csv')
confirm = pd.read_csv('/content/time_series_covid19_confirmed_global.csv')
```

```
death.head()
```

```
confirm.head()
```

```
death = death.melt(id_vars=['Province/State','Country/Region','Lat','Long'],var_name='date',value_name='num_
confirm = confirm.melt(id_vars=['Province/State','Country/Region','Lat','Long'],var_name='date',value_name='
```

```
death.head()
```

```
confirm.merge(death,on=['Province/State','Country/Region','Lat','Long','date'])[['Country/Region','date','nu
```

Pivot Table

The pivot table takes simple column-wise data as input, and groups the entries into a two-dimensional table that provides a multidimensional summarization of the data.

```
import numpy as np
import pandas as pd
import seaborn as sns
```

```
df = sns.load_dataset('tips')
df.head()
```

```
df.groupby('sex')[['total_bill']].mean()
```

```
df.groupby(['sex','smoker'])[['total_bill']].mean().unstack()
```

```
df.pivot_table(index='sex',columns='smoker',values='total_bill')
```

```
# aggfunc
df.pivot_table(index='sex',columns='smoker',values='total_bill',aggfunc='std')
```

```
# all cols together
df.pivot_table(index='sex',columns='smoker')['size']
```

```
# multidimensional
df.pivot_table(index=['sex','smoker'],columns=['day','time'],aggfunc={'size':'mean','tip':'max','total_bill'
```

```
# margins
df.pivot_table(index='sex',columns='smoker',values='total_bill',aggfunc='sum',margins=True)
```

```
# plotting graphs
df = pd.read_csv('/content/expense_data.csv')
```

```
df.head()
```

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

```
df.info()
```

```
df['Date'] = pd.to_datetime(df['Date'])
```

```
df.info()
```

```
df['month'] = df['Date'].dt.month_name()
```

```
df.head()
```

```
df.pivot_table(index='month',columns='Category',values='INR',aggfunc='sum',fill_value=0).plot()
```

```
df.pivot_table(index='month',columns='Income/Expense',values='INR',aggfunc='sum',fill_value=0).plot()
```

```
df.pivot_table(index='month',columns='Account',values='INR',aggfunc='sum',fill_value=0).plot()
```