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

In [2]:
```
s = pd.Series([100, 200, 300, 400, 500], index=['a','b','c','d','e'])

print("Pandas Series:")
print(s)
print("\nAccess single value:", s['c'])
```

```
Pandas Series:
a    100
b    200
c    300
d    400
e    500
dtype: int64

Access single value: 300
```

In [3]:
```
type(s)
```

Out[3]:
```
pandas.core.series.Series
```

In [4]:
```
#perform arithmatic operation
print("Original:\n", s)
print("\nMultiplied by 2:\n", s * 2)
```

```
Original:
 a    100
b    200
c    300
d    400
e    500
dtype: int64

Multiplied by 2:
 a     200
b     400
c     600
d     800
e    1000
dtype: int64
```

In [5]:
```
# Creating a DataFrame
data = {
    'Name': ['Maruti', 'Balasaheb', 'Sunil'],
    'Age': [31, 30, 33],
    'Marks': [32, 34, 12]
}

df = pd.DataFrame(data)

print("Pandas DataFrame:")
print(df)
print("\nAccess single column (as Series):")
print(df['Marks'])
```

```
Pandas DataFrame:
        Name  Age  Marks
0     Maruti   31     32
1  Balasaheb   30     34
2      Sunil   33     12

Access single column (as Series):
0    32
1    34
2    12
Name: Marks, dtype: int64
```

In [6]:
```
type(df)
```

Out[6]:
```
pandas.core.frame.DataFrame
```

In [7]:
```
print("\n",df['Name'])             # select single column
print("\n",df[['Name','Marks']])   # select multiple columns
print("\n",df.iloc[0])             # first row by index
print("\n",df.iloc[1:4])           # rows by index range
print("\n",df.loc[df['Marks']>30]) # filter condition
```

```
 0       Maruti
1    Balasaheb
2        Sunil
Name: Name, dtype: object

          Name  Marks
0     Maruti     32
1  Balasaheb     34
2      Sunil     12

 Name      Maruti
Age           31
Marks         32
Name: 0, dtype: object

          Name  Age  Marks
1  Balasaheb   30     34
2      Sunil   33     12

          Name  Age  Marks
0     Maruti   31     32
1  Balasaheb   30     34
```

In [8]:
```
data=pd.read_csv("sc_data.csv")         # load CSV file
```

In [9]:
```
print(data.head()     )    # first 5 rows
print("\n",data.tail())    # last 5 rows
print("\n",data.shape )    # (rows, columns)
print("\n",data.info())    # summary
```

```
   rank      name     ticker  market_cap_04_08_2025  price_04_08_2025  \
0     1    NVIDIA       NVDA          4236614041600          173.7200   
1     2  Broadcom       AVGO          1357609697280          288.6400   
2     3      TSMC        TSM          1219921444864          235.2100   
3     4   Samsung  005930.KS           326234945517           49.5673   
4     5       AMD        AMD           278314942464          171.6510   

         country  
0  United States  
1  United States  
2         Taiwan  
3    South Korea  
4  United States  

     rank                name    ticker  market_cap_04_08_2025  \
147   148  GCT Semiconductor      GCTS               74175432   
148   149      Amtech Systems      ASYS               65271840   
149   150          Pixelworks      PXLW               56552416   
150   151          Mobix Labs      MOBX               43760624   
151   152              Kalray  ALKAL.PA                8735389   

     price_04_08_2025        country  
147          1.330000  United States  
148          4.560000  United States  
149         10.780000  United States  
150          0.818900  United States  
151          0.714697         France  

 (152, 6)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 152 entries, 0 to 151
Data columns (total 6 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   rank                   152 non-null    int64  
 1   name                   152 non-null    object 
 2   ticker                 152 non-null    object 
 3   market_cap_04_08_2025  152 non-null    int64  
 4   price_04_08_2025       152 non-null    float64
 5   country                152 non-null    object 
dtypes: float64(1), int64(2), object(3)
memory usage: 7.3+ KB

 None
```

In [10]:
```
print(data.dtypes          )    # print("\n",data types of e)ach column
print("\n",data.describe())    # summary statistics (mean, std, min, max, etc.)
print("\n",data.columns    )    # list of column names
print("\n",data.index      )    # row index
```

```
rank                         int64
name                        object
ticker                      object
market_cap_04_08_2025        int64
price_04_08_2025           float64
country                     object
dtype: object

              rank  market_cap_04_08_2025  price_04_08_2025
count   152.000000           1.520000e+02        152.000000
mean     76.500000           6.777295e+10         73.558129
std      44.022721           3.735016e+11        134.513950
min       1.000000           8.735389e+06          0.128263
25%      38.750000           1.006935e+09          8.015935
50%      76.500000           4.950831e+09         21.431100
75%     114.250000           1.688528e+10         79.056950
max     152.000000           4.236614e+12        886.640000

 Index(['rank', 'name', 'ticker', 'market_cap_04_08_2025', 'price_04_08_2025',
       'country'],
      dtype='object')

 RangeIndex(start=0, stop=152, step=1)
```

In [11]:

```
print(data.describe())                                    # summary statistics (m
print("\nMean:",data['price_04_08_2025'].mean())          # average price_04_08_2
print("\nMedian:",data['price_04_08_2025'].median())      # median
print("\nMax:",data['price_04_08_2025'].max())            # maximum value
print("\nMin:",data['price_04_08_2025'].min())            # minimum value
print("\nSum:",data['price_04_08_2025'].sum())            # sum of column
print("\nFreq:",data['price_04_08_2025'].value_counts()) # frequency count
```

```
             rank  market_cap_04_08_2025  price_04_08_2025
count  152.000000           1.520000e+02        152.000000
mean    76.500000           6.777295e+10         73.558129
std     44.022721           3.735016e+11        134.513950
min      1.000000           8.735389e+06          0.128263
25%     38.750000           1.006935e+09          8.015935
50%     76.500000           4.950831e+09         21.431100
75%    114.250000           1.688528e+10         79.056950
max    152.000000           4.236614e+12        886.640000

Mean: 73.55812860526318

Median: 21.4311

Max: 886.64

Min: 0.128263

Sum: 11180.835548000003

Freq: price_04_08_2025
173.720000    1
288.640000    1
235.210000    1
49.567300     1
171.651000    1
             ..
1.330000      1
4.560000      1
10.780000     1
0.818900      1
0.714697      1
Name: count, Length: 152, dtype: int64
```