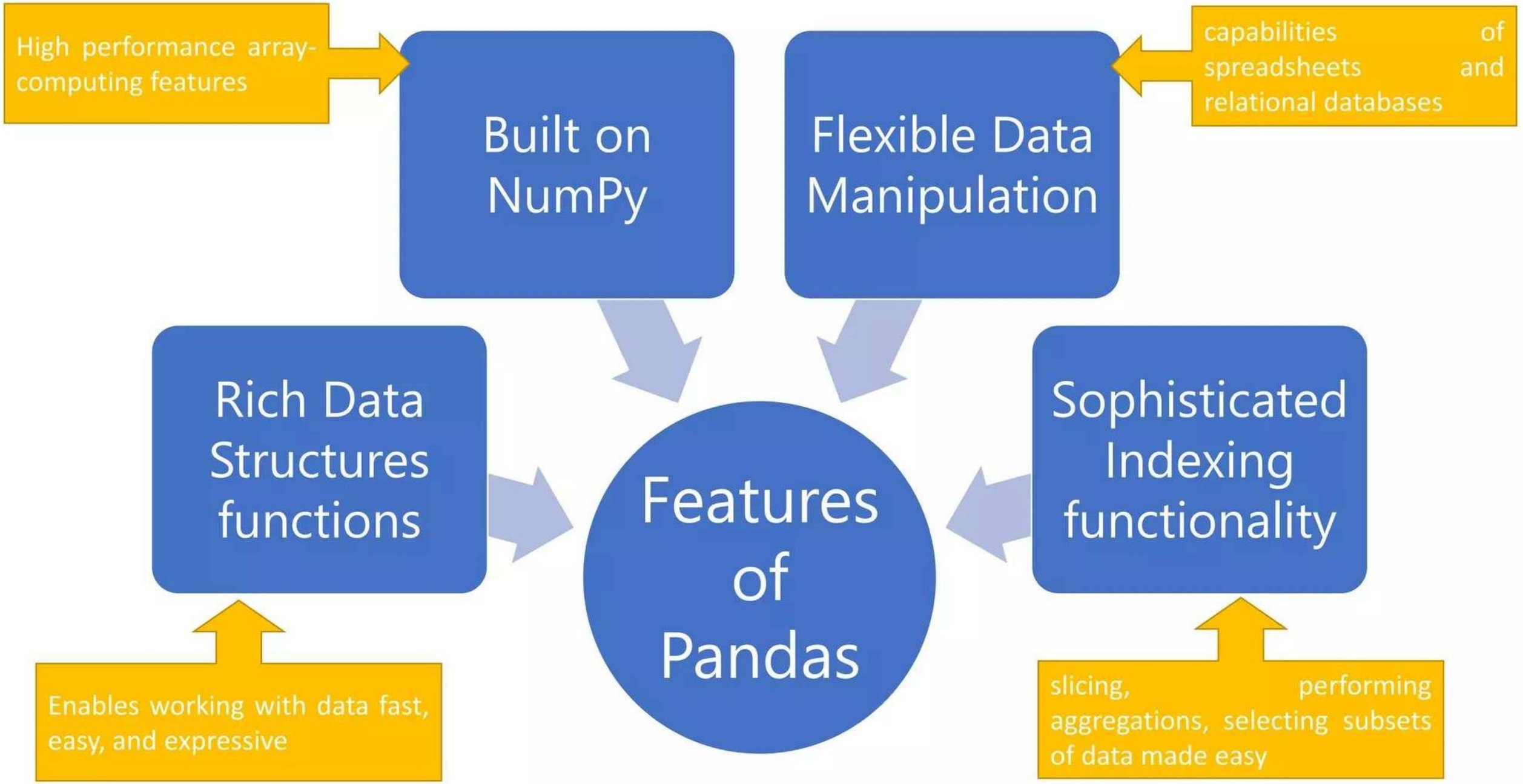
Data Analysis using Python Pandas

What is Python Pandas?

Powerful and productive Python data analysis and management library

Pandas


High performance array-computing features
Built on NumPy
Flexible Data Manipulation
capabilities of spreadsheets and relational databases
Rich Data Structures functions
Features of Pandas
Sophisticated Indexing functionality
Enables working with data fast, easy, and expressive
slicing, performing aggregations, selecting subsets of data made easy

Working with Pandas

Installing Pandas:

Installing Pandas is very similar to installing NumPy. To install Pandas from command line, we need to type in:

```
pip install pandas
```

To work with Pandas library :

```
import pandas as pd
```

Pandas Data Structures: Series vs DataFrame

- A series can be seen as a one dimensional array with index values whereas a DataFrame is a wo dimensional data structure having rows and columns

Series

	apples
0	3
1	2
2	0
3	1

Series

	oranges
0	0
1	3
2	7
3	2

DataFrame

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

Creating a Series

- A series can be created from scalar values, dictionaries, NumPy arrays.

```
import pandas as pd
series1 = pd.Series([10,20,30])
print(series1)
```

giving an alias name to pandas

List

Series object

Output

```
0    10
1    20
2    30
dtype: int64
```

index

Accessing series

- We can access series elements through positional index or a label index

```
import pandas as pd
sr = pd.Series([15,20,40])
print(sr[2])
```

Positional index

Output

```
40
```

```
srCaps = pd.Series(['NewDelhi', 'WashingtonDC',
'London', 'Paris'], index=['India', 'USA', 'UK',
'France’])
print(srCaps['India’])
```

Label index

Output

```
'NewDelhi'
```

We can also use negative indexing and slicing to access series elements.

```
import pandas as pd
srCaps = pd.Series(['NewDelhi', 'WashingtonDC', 'London',
'Paris'],index=['India', 'USA', 'UK', 'France'])
print(srCaps[-1])
print(srCaps[1:3])
```

Negative index

Slicing

Output

```
Paris
USA    WashingtonDC
UK           London
dtype: object
```

Series functions

Some useful series functions or methods are: head(), tail() and count()

head(): Returns the first n members of the series. Default value is 5

tail(): Returns the last n members of the series. Default value is 5

count(): Returns the total number of non NaN members of the series.

Creating DataFrames

- Data Frames can be created from any of the basic Data Structures like, lists, dictionaries, dictionaries of lists, arrays lists of dictionaries and also from series.

An empty DataFrame can be created as follows:

```
import pandas as pd
dFrameEmt = pd.DataFrame()
print(dFrameEmt)
```

Output

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

Creating a DataFrame from a Dictionary of Series

```
import pandas as pd
ResultSheet={'Arnab': pd.Series([90, 91, 97],
index=['Maths','Science','Hindi']),
'Ramit': pd.Series([92, 81, 96], index=['Maths','Science','Hindi']),
'Samridhi': pd.Series([89, 91, 88], index=['Maths','Science','Hindi']),
'Riya': pd.Series([81, 71, 67],  index=['Maths','Science','Hindi']),
'Mallika': pd.Series([94, 95, 99], index=['Maths','Science','Hindi'])}
ResultDF = pd.DataFrame(ResultSheet)
print(ResultDF)
```

Output

	Arnab	Ramit	Samridhi	Riya	Mallika
Maths	90	92	89	81	94
Science	91	81	91	71	95
Hindi	97	96	88	67	99

Accessing and indexing DataFrames

- Data elements in a DataFrame can be accessed using indexing. There are two ways of indexing Dataframes : Label based indexing and Boolean Indexing.

Label Based Indexing

Example: To print the marks of science for all the students

```
print(ResultDF.loc['Science'])
```

Output

```
Arnab      91
Ramit      81
Samridhi   91
Riya       71
Mallika    95
Name: Science, dtype: int64
```

Label Based Indexing

Example: To print the marks of a student in all the subjects

```
print(ResultDF['Riya'])
```

Output

```
Maths      81
Science    71
Hindi      67
Name: Riya, dtype: int64
```

Label Based Indexing

Example: To print the marks of all students in particular subjects

```
print(ResultDF.loc[['Maths','Hindi']])
```

Output

```
        Arnab   Ramit   Samridhi   Riya   Mallika
Maths      90      92         89     81        94
Hindi      97      96         88     67        99
```

Boolean Indexing

Example: To display whether a student has scored more than 90 in Science

```
print(ResultDF.loc['Science']>90)
```

Output

```
Arnab        True
Ramit       False
Samridhi     True
Riya        False
Mallika      True
Name: Science, dtype: bool
```

Slicing DataFrames

Example: To display the rows Math till Science

```
print(ResultDF.loc['Maths':'Science'])
```

Output

```
         Arnab  Ramit  Samridhi  Riya  Mallika
Maths       90     92        89    81       94
Science     91     81        91    71       95
```

Descriptive statistics

- The main utility of Pandas DataFrames is Data Analysis.
- Descriptive Statistics means applying some functions to analyse data.

Calculating max(),min()

- `print(df.max())` will print maximum value in each column
- `print(df.max(axis=1))` will print maximum value in each column

-
```
dfUT2=df[df.UT==2]
print(dfUT2.max())
```
will print maximum value for ut 2

-
```
dfMishti = df.loc[df.Name == 'Mishti’]
print(dfMishti)
```
- `print(dfMishti[['Maths','Science','S.St','Hindi','Eng']].min()))`

Calculating sum(), mean(), median(), mode()

- print(df.sum())
- print(df['Maths'].sum())
- print(dfRaman[['Maths','Science','S.St','Hindi','Eng']].sum())
- print(df.count())