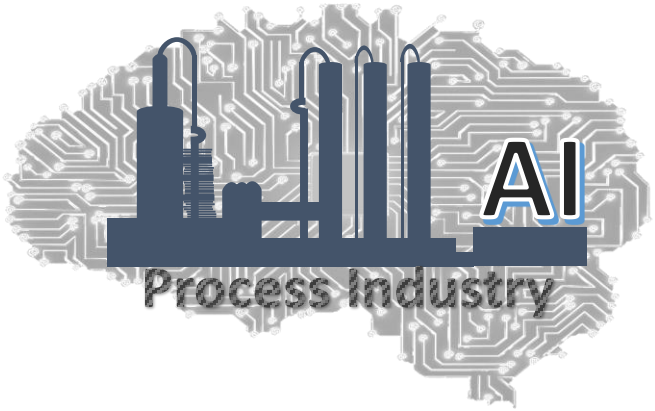
Statistical Techniques for Monitoring Industrial Processes



Lecture : Scientific Computing Package: Pandas

Module : Python Installation and Basics

Course TOC

- ❑ Introduction to Statistical Process Monitoring (SPM)
- ❑ Python Installation and basics (optional)
  - ➢ Development environment, Scientific computing packages
- ❑ Univariate SPM & Control Charts
  - ➢ Shewhart Charts
  - ➢ CUSUM Charts
  - ➢ EWMA Charts
- ❑ Multivariate SPM
  - ➢ Fault detection using Principal Component Analysis (PCA)
  - ➢ Fault detection using Partial Least Squares (PLS) regression
  - ➢ Fault diagnosis using PCA/PLS contribution charts
  - ➢ Strategies for handling nonlinear, dynamic, multimode systems
- ❑ Deploying SPM solutions

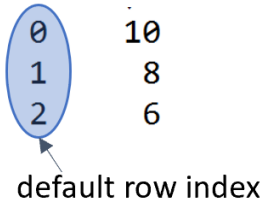
Pandas for Advanced Tabular Data Analysis

Built on top on NumPy, Pandas provides a rich set of features for data manipulation and analysis

- *label-based data slicing*
- *SQL-like data filtering, grouping, merging*
- *time-series functionalities (such as rolling-window average)*

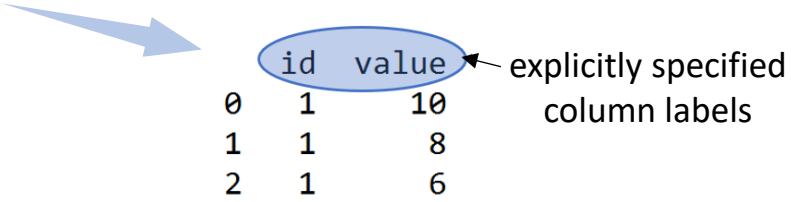
Series: 1D data structure

```
# create a Series

import pandas as pd
data = [10,8,6]
s = pd.Series(data)
```



Dataframe: 2D data structure

```
# create a Dataframe

import pandas as pd
data = [[1,10],[1,8],[1,6]]
df = pd.DataFrame(data, columns=['id', 'value'])
```



Pandas Data Structure ⇌ NumPy Data Structure

1D NumPy array to Series

```
s = pd.Series(np.array([10,8,6]))
```

2D NumPy array to Dataframe

```
df = pd.DataFrame(np.array([[1,10],[1,8],[1,6]]))
```

Series to NumPy array

```
arr_1D = s.values
```

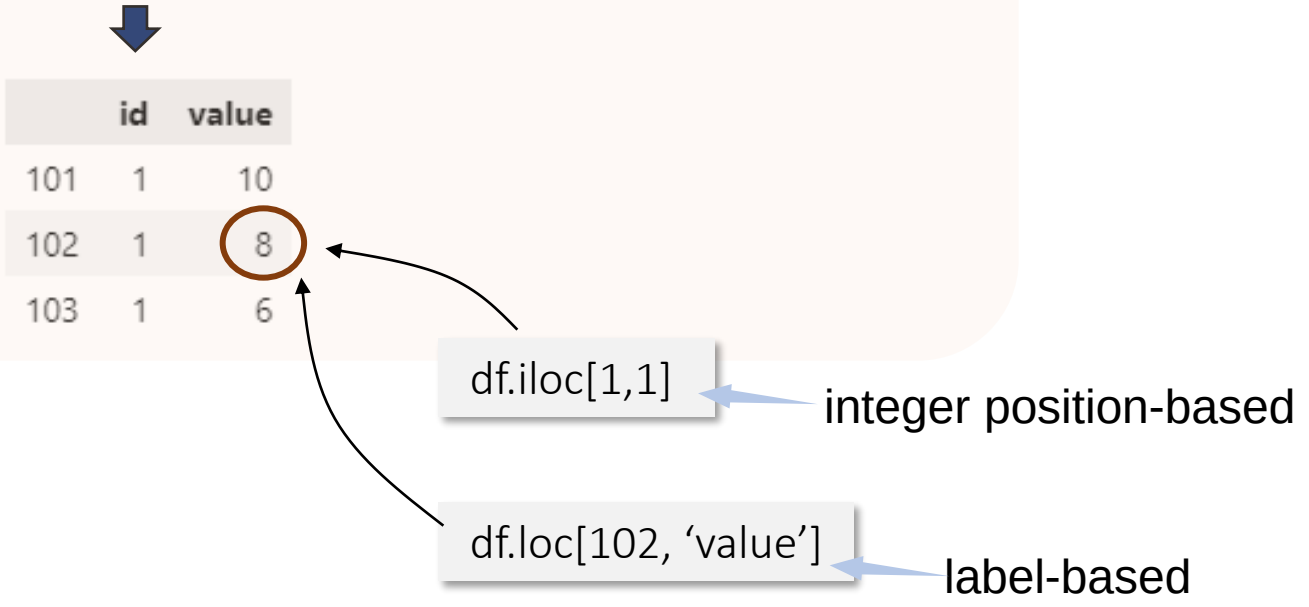
Dataframe to NumPy array

```
arr_2D = df.values
```

Data Access

- Pandas allows accessing rows and columns using integer positions (as we saw for NumPy arrays) and labels!

```
df = pd.DataFrame(np.array([[1,10],[1,8],[1,6]]),
                  columns=['id', 'value'], index=[101, 102, 103])
```



Data Slicing

- Works like that for NumPy arrays when slicing rows and columns by integer positions

df

	var1	var2	var3	var4
0	1	10	3	4
1	1	8	2	6
2	1	6	0	0
3	0	0	3	1

- Select *var2* and *var3* data from rows 1 to 3

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

or

```
df.loc[1:, ['var2', 'var3']]
```

Data Manipulation

Pandas is a powerful Python library for data manipulation and analysis

df

	trial #	var1	var2	var3	var4
0	1	1	10	3	4
1	1	1	8	2	6
2	1	1	6	0	0
3	2	0	0	3	1
4	2	3	3	3	0
5	2	2	6	1	0

Filtering

Keep data only from *trial # 2*

```
df_filtered = df[df['trial #']==2]
```

	trial #	var1	var2	var3	var4
3	2	0	0	3	1
4	2	3	3	3	0
5	2	2	6	1	0

Aggregation

Find mean of each variable across all trials

```
meanValues = df.iloc[:,1:].mean()
```

```
var1    1.333333
var2    5.500000
var3    2.000000
var4    1.833333
```

Grouping

Find trial-wise mean of each variable

```
meanValues_byTrial = df.groupby('trial #').mean()
```

trial #	var1	var2	var3	var4
1	1.000000	8.0	1.666667	3.333333
2	1.666667	3.0	2.333333	0.333333

File I/O and Data Summary

Pandas can read and write data from various sources like CSV, Excel, SQL databases, etc.

	A	B	C	D
1	Sr. No.	Category	Value	
2	1	low	0.1	
3	2	medium	0.6	
4	3	medium	0.6	
5	4	low	0.15	
6	5	high	0.9	
7		excelDataFile.xlsx		

```
excelData = pd.read_excel('excelDataFile.xlsx')
excelData.head() # quick look at data
```

	Sr. No.	Category	Value
0	1	low	0.10
1	2	medium	0.60
2	3	medium	0.60
3	4	low	0.15
4	5	high	0.90

```
excelData = pd.read_excel('excelDataFile.xlsx')
excelData.describe(include='all') # dataset's descriptive statistics
```

	Sr. No.	Category	Value
count	5.000000	5	5.000000
unique	NaN	3	NaN
top	NaN	low	NaN
freq	NaN	2	NaN
mean	3.000000	NaN	0.470000
std	1.581139	NaN	0.338378
min	1.000000	NaN	0.100000
25%	2.000000	NaN	0.150000
50%	3.000000	NaN	0.600000
75%	4.000000	NaN	0.600000
max	5.000000	NaN	0.900000

Statistical Techniques for Monitoring Industrial Processes

Next Lecture : Other Useful Packages for Data Analytics

Module : Python Installation and Basics