HAB 619 Introduction to Scientific Computing in Sports Science

11

SERDAR ARITAN

serdar.aritan@hacettepe.edu.tr

Biyomekanik Araştırma Grubu
www.biomech.hacettepe.edu.tr
Spor Bilimleri Fakültesi
www.sbt.hacettepe.edu.tr
Hacettepe Universitesi, Ankara, Türkiye
www.hacettepe.edu.tr

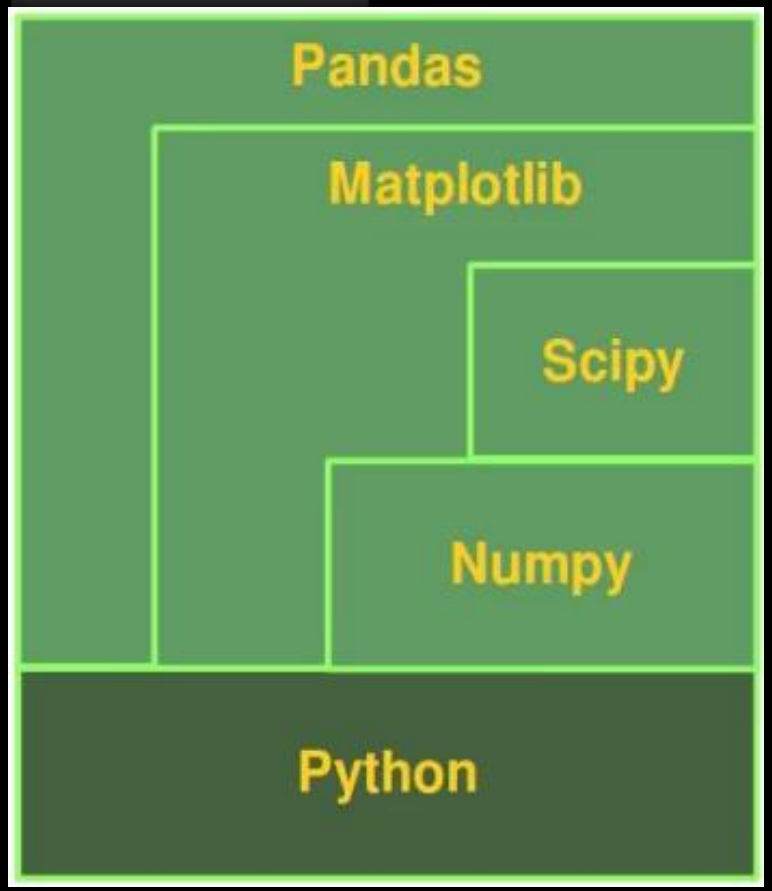
De Motu Animalium G.Borelli (1680)

What is Python Pandas?

Pandas is used for data manipulation, analysis and cleaning. Python pandas is well suited for different kinds of data, such as:

- Tabular data with heterogeneously-typed columns
- Ordered and unordered time series data
- Arbitrary matrix data with row & column labels
- Unlabelled data
- Any other form of observational or statistical data sets

What is Python Pandas?



Numpy is a module which provides the basic data structures, implementing multi-dimensional arrays and matrices.

SciPy is based on top of Numpy, i.e. it uses the data structures provided by NumPy. It extends the capabilities of NumPy with further useful functions for minimization, regression, Fourier-transformation and many others.

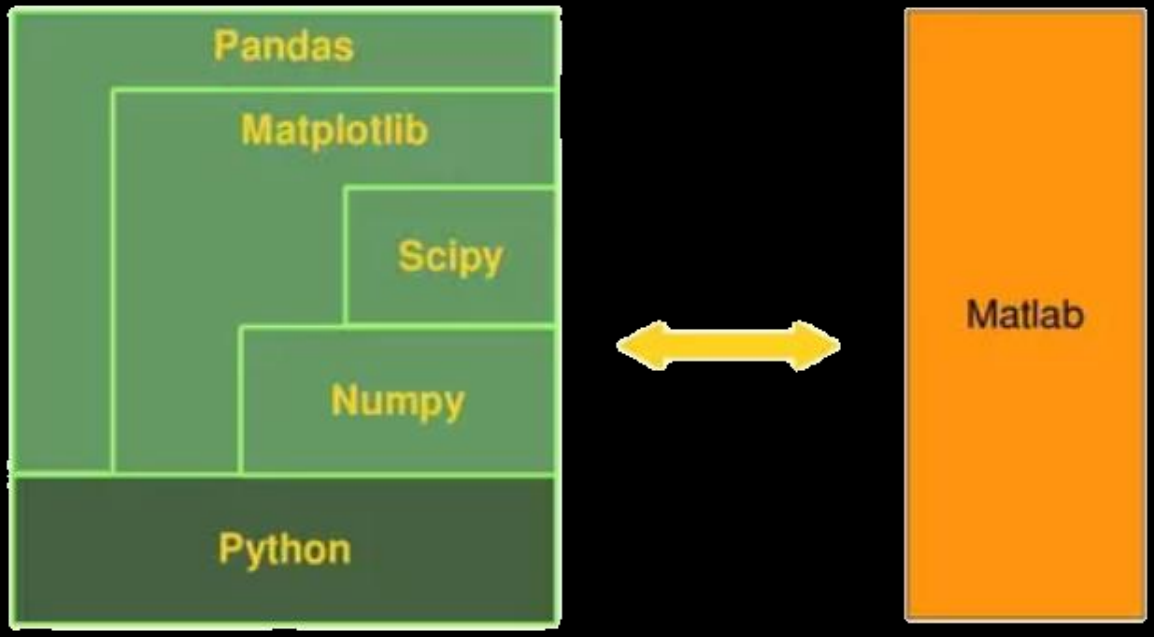
Matplotlib is a plotting library for the Python programming language and the numerically oriented modules like NumPy and SciPy

Pandas is using all of the previously mentioned modules. **The special focus of Pandas consists in offering data structures and operations for manipulating numerical tables and time series**. The name is derived from the term "**panel data**". Pandas is well suited for working with tabular data as it is known from spread sheet programming like Excel.

What is Python Pandas?

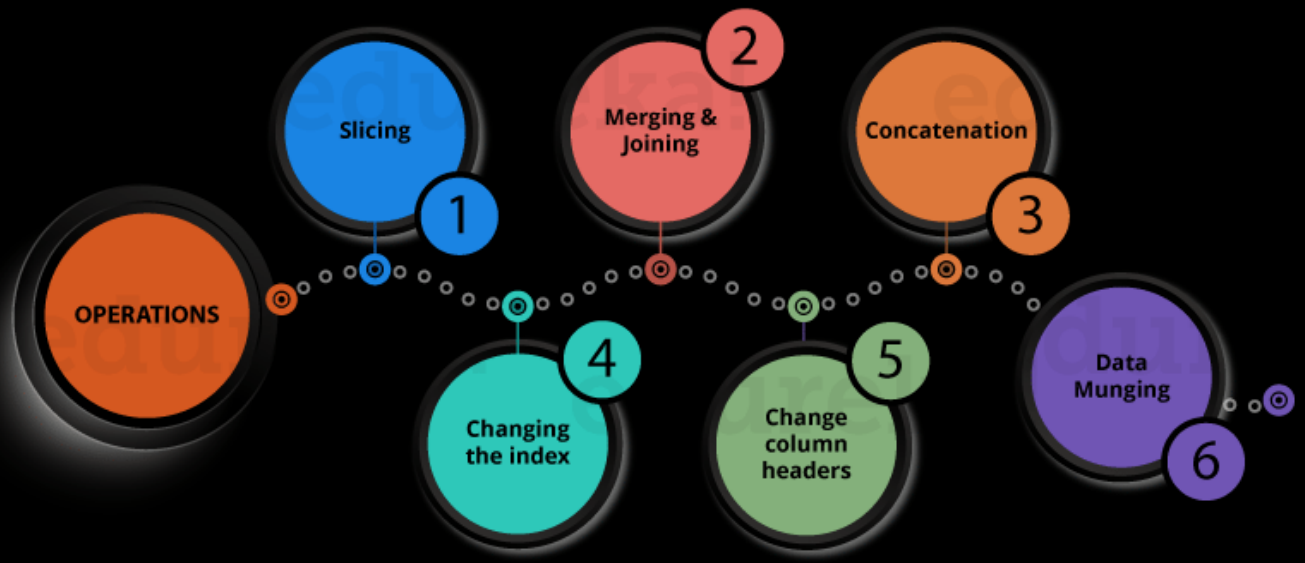
Even though MATLAB has a huge number of additional toolboxes available, Python has the advantage that it is a more modern and complete programming language. Python is continually becoming more powerful by a rapidly growing number of specialized modules.

Python in combination with Numpy, Scipy, Matplotlib and Pandas can be used as a complete replacement for MATLAB.



What is Python Pandas?

Using Python pandas, you can perform a lot of operations with series, data frames, missing data, group by etc. Some of the common operations for data manipulation are listed below:



What is Python Pandas?

A Series is a **one-dimensional array-like object** containing a sequence of values (of similar types to NumPy types) of the same type and an associated array of data labels, called its index.

```
import pandas as pd

obj = pd.Series([4, 7, -5, 3])

>>> obj

0    4
1    7
2   -5
3    3
dtype: int64
```

What is Python Pandas?

```
>>> Obj.array
>>> <PandasArray>
[4, 7, -5, 3]
Length: 4, dtype: int64

>>> Obj.index
>>> RangeIndex(start=0, stop=4, step=1)
>>> obj2 = pd.Series([4, 7, -5, 3], index=["d","b","a", "c"])
    d    4
    b    7
    a   -5
    c    3
    dtype: int64

>>> obj2["d"] = 6
```

What is Python Pandas?

```
>>> obj2[ obj2 > 0 ]
    d     6
    b     7
    c     3
    dtype: int64

>>> obj2 * 2
    d    12
    b    14
    a   -10
    c     6
    dtype: int64
```

What is Python Pandas?

```
>>> import numpy as np
>>> np.exp(obj2)
    d     403.428793
    b    1096.633158
    a       0.006738
    c      20.085537
    dtype: int64

>>> "b" in obj2
    True

>>> "e" in obj2
    False
```

What is Python Pandas?

```
>>> sdata = {"Ohio": 35000, "Texas": 71000, "Oregon":
16000, "Utah": 5000}

>>>  obj3 = pd.Series(sdata)

>>> obj3
    Ohio      35000
    Texas     71000
    Oregon    16000
    Utah       5000
    dtype: int64
>>> obj3.to_dict()
 {'Ohio': 35000, 'Texas': 71000, 'Oregon': 16000, 'Utah':
5000}
```

What is Python Pandas?

A DataFrame represents a rectangular table of data and contains an ordered, named collection of columns, each of which can be a different value type (numeric, string, Boolean, etc.). The DataFrame has both a row and column index; it can be thought of as a dictionary of Series all sharing the same index.

Slicing the Data Frame

In order to perform slicing on data, you need a data frame.

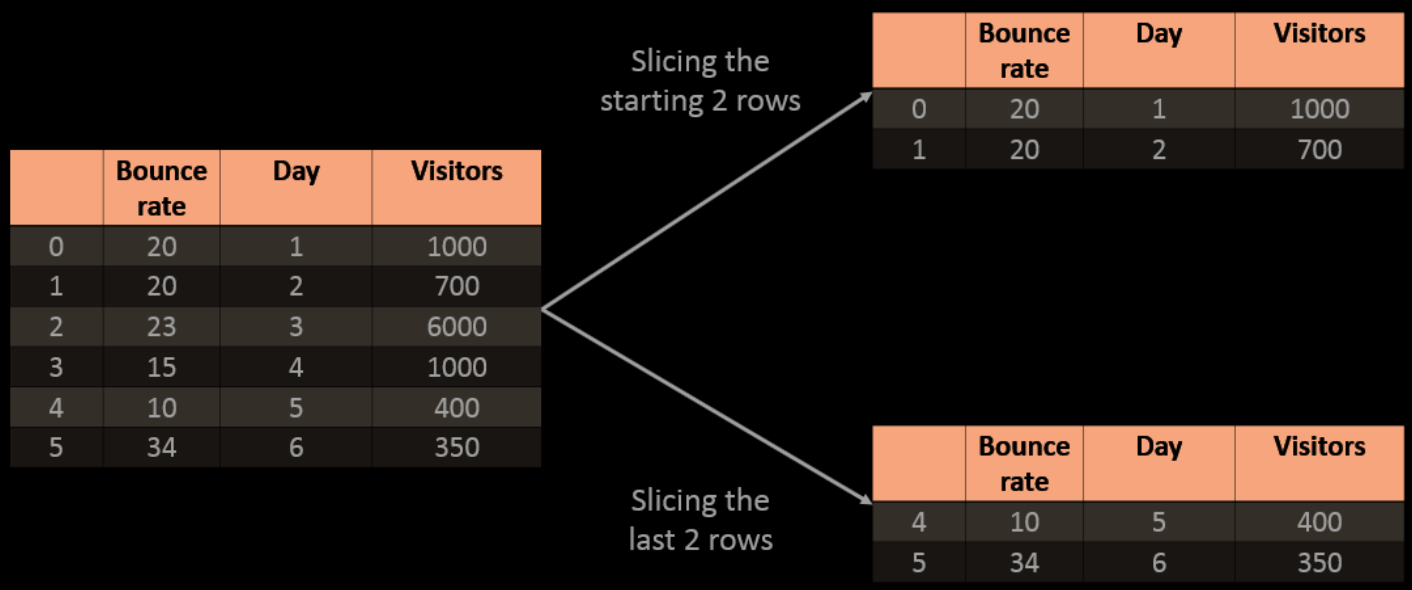
```
import pandas as pd

XYZ = {'Day':[1,2,3,4,5,6], "Visitors":[1000,
700,6000,1000,400,350], "Bounce_Rate":[20,20,
23,15,10,34]}

df= pd.DataFrame(XYZ)

print(df)
```

```
   Day  Visitors  Bounce_Rate
0    1      1000           20
1    2       700           20
2    3      6000           23
3    4      1000           15
4    5       400           10
5    6       350           34
```

Slicing the Data Frame

slice a particular column from this data frame.

```
print(df.head(2))
Day   Visitors   Bounce_Rate
0     1        1000              20
1     2         700              20
```



	Bounce rate	Day	Visitors
0	20	1	1000
1	20	2	700
2	23	3	6000
3	15	4	1000
4	10	5	400
5	34	6	350

	Bounce rate	Day	Visitors
0	20	1	1000
1	20	2	700

	Bounce rate	Day	Visitors
4	10	5	400
5	34	6	350

Slicing the Data Frame

slice a particular column from this data frame.

```
print(df.tail(2))
Day  Visitors  Bounce_Rate
4    5         400              10
5    6         350              34
```

	Bounce rate	Day	Visitors
0	20	1	1000
1	20	2	700
2	23	3	6000
3	15	4	1000
4	10	5	400
5	34	6	350

Slicing the starting 2 rows

	Bounce rate	Day	Visitors
0	20	1	1000
1	20	2	700

Slicing the last 2 rows

	Bounce rate	Day	Visitors
4	10	5	400
5	34	6	350

Merging & Joining

merge two data frames to form a single data frame.

```
import pandas as pd

df1= pd.DataFrame({
"HPI":[80,90,70,60],"Int_Rate":[2,1,2,3],"IND_GDP":[50,4
5,45,67]}, index=[2001,2002,2003,2004])

df2 = pd.DataFrame({
"HPI":[80,90,70,60],"Int_Rate":[2,1,2,3],"IND_GDP":[50,4
5,45,67]}, index=[2005,2006,2007,2008])

merged= pd.merge(df1,df2)
print(merged)

merged= pd.merge(df1,df2,on ="HPI")
print(merged)
```

Merging & Joining

merge two data frames to form a single data frame

	HPI	Int_Rate	IND_GDP
0	80	2	50
1	90	1	45
2	70	2	45
3	60	3	67

	HPI	Int_Rate_x	IND_GDP_x	Int_Rate_y	IND_GDP_y
0	80	2	50	2	50
1	90	1	45	1	45
2	70	2	45	2	45
3	60	3	67	3	67

Merging & Joining

Combine two differently indexed dataframes into a single result dataframe

```
import pandas as pd

df1 = pd.DataFrame({
"Int_Rate":[2,1,2,3], "IND_GDP":[50,45,45,67]},
index=[2001, 2002,2003,2004])

df2 = pd.DataFrame({
"Low_Tier_HPI":[50,45,67,34],"Unemployment":[1,3,5,6]},
index=[2001, 2003,2004,2004])

joined= df1.join(df2)

print(joined)
```

Merging & Joining

Combine two differently indexed dataframes into a single result dataframe

	Int_Rate	IND_GDP	Low_Tier_HPI	Unemployment
2001	2	50	50.0	1.0
2002	1	45	NaN	NaN
2003	2	45	45.0	3.0
2004	3	67	67.0	5.0
2004	3	67	34.0	6.0

Concatenation

Concatenation basically glues the dataframes together

```
import pandas as pd

df1= pd.DataFrame({
"HPI":[80,90,70,60],"Int_Rate":[2,1,2,3],"IND_GDP":[50,4
5,45,67]}, index=[2001,2002,2003,2004])

df2 = pd.DataFrame({
"HPI":[80,90,70,60],"Int_Rate":[2,1,2,3],"IND_GDP":[50,4
5,45,67]}, index=[2005,2006,2007,2008])

concat= pd.concat([df1,df2])
print(concat)
```

Concatenation

Concatenation basically glues the dataframes together

	HPI	Int_Rate	IND_GDP
2001	80	2	50
2002	90	1	45
2003	70	2	45
2004	60	3	67
2005	80	2	50
2006	90	1	45
2007	70	2	45
2008	60	3	67

Concatenation

Concatenation basically glues the dataframes together

```
Concat_1= pd.concat([df1,df2],axis=1)
print(concat_1)
```

	HPI	Int_Rate	IND_GDP	HPI	Int_Rate	IND_GDP
2001	80.0	2.0	50.0	NaN	NaN	NaN
2002	90.0	1.0	45.0	NaN	NaN	NaN
2003	70.0	2.0	45.0	NaN	NaN	NaN
2004	60.0	3.0	67.0	NaN	NaN	NaN
2005	NaN	NaN	NaN	80.0	2.0	50.0
2006	NaN	NaN	NaN	90.0	1.0	45.0
2007	NaN	NaN	NaN	70.0	2.0	45.0
2008	NaN	NaN	NaN	60.0	3.0	67.0

Concatenation

how to change the index values in a dataframe

```
import pandas as pd

df= pd.DataFrame({"Day":[1,2,3,4], "Visitors":[200,
100,230,300], "Bounce_Rate":[20,45,60,10]})

print(df)
```

```
   Day  Visitors  Bounce_Rate
0    1       200           20
1    2       100           45
2    3       230           60
3    4       300           10
```

Concatenation

how to change the index values in a dataframe

```
import pandas as pd

df= pd.DataFrame({"Day":[1,2,3,4], "Visitors":[200,
100,230,300], "Bounce_Rate":[20,45,60,10]})

df.set_index("Day", inplace= True)

print(df)
```

```
     Visitors  Bounce_Rate
Day
1         200           20
2         100           45
3         230           60
4         300           10
```

Concatenation

how to change the index values in a dataframe

```
import pandas as pd

df= pd.DataFrame({"Day":[1,2,3,4], "Visitors":[200,
100,230,300], "Bounce_Rate":[20,45,60,10]})

df.set_index("Bounce_Rate", inplace= True)

print(df)
```

```
                Day  Visitors
Bounce_Rate
20                1       200
45                2       100
60                3       230
10                4       300
```

Concatenation

how to change the index values in a dataframe

```
import pandas as pd

df = pd.DataFrame({"Day":[1,2,3,4], "Visitors":[200,
100,230,300], "Bounce_Rate":[20,45,60,10]})
print(df)

df = df.rename(columns={"Visitors":"Users"})
print(df)
```

```
   Day  Visitors  Bounce_Rate
0    1       200           20
1    2       100           45
2    3       230           60
3    4       300           10
   Day  Users  Bounce_Rate
0    1    200           20
1    2    100           45
2    3    230           60
3    4    300           10
```