InfPALS
Pandas

Pandas is a library providing data structures and data analysis tools. It allows us to load data from different sources, and then allows us to manipulate and search the data.

To get introduced to pandas, we will take you through some examples.

Enter the DataFrame

Before we get started, first we'll introduce you to the **DataFrame**, the core data structure of Pandas. It's effectively a 2-dimensional table, supporting axis names and normal table operations. For example, raw_data_1 in exercise 7 looks like this (note that Pandas automatically indexes each row for us):

	subject_id	first_name	last_name
0	1	Alex	Anderson
1	2	Amy	Ackerman
2	3	Allen	Ali
3	4	Alice	Aoni
4	5	Ayoung	Atiches

Below are some exercises to learn the basics of Pandas.
10 minutes to Pandas (http://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html)is your friend and you will find a lot of help over there to start off. After scanning through it, try to answer the questions on your own (with some help from stack overflow and the API for Pandas dataframe(https://pandas.pydata.org/pandas-docs/stable/reference/frame.html))– what is your motivation for studying Pandas, why we need it, what is its main functionality, how widely it can be used.

1) Download the .csv file at http://www.football-data.co.uk/mmz4281/1718/E0.csv and put it into your working directory. This contains the data we will be working with.

2) Open a new Jupyter notebook and start setting up:

```
import pandas as pd
import numpy as np
import matplotlib as plt
import os
os.listdir()
```

os is a python module that gives us access to some functionality of the operating system. We are just using it to list the files and folders in our working directory. You should see the files you have created today, the checkpoints (saves), and the file E0.csv that you just downloaded.

3) Read E0.csv into a DataFrame.

```
df = pd.read_csv("E0.csv")
```

This creates a dataframe df which contains the data from the file E0.cvs that we just downloaded.

Have a look at the data by simply running:

```
df
```

As you can see, the table contains data of football games in the E0 division. The abbreviations in the headers of the first column that we can see mean:

Div = League Division Date = Match Date (dd/mm/yy) HomeTeam = Home Team AwayTeam = Away Team FTHG= Full Time Home Team Goals FTAG = Full Time Away Team Goals FTR = Full Time Result (H=Home Win, D=Draw, A=Away Win) HTHG = Half Time Home Team Goals HTAG = Half Time Away Team Goals HTR = Half Time Result (H=Home Win, D=Draw, A=Away Win)

Try adding header=None as a parameter when reading in the data (`df =`

`pd.read_csv("E0.csv")`. What happens?

4) Once we have some data, we want to know some information about it. Write appropriate commands to obtain:

a) number of columns

```
df.shape[1]
```

b) number of records

```
df.shape[0]
```

c) first, extract first 10 rows

```
df.head(10)
```

If no parameter is given, the default value is 5.

d) and then, the last 25

```
df.tail(25)
```

Again, if no parameter is given, the default value is 5.

e) to get a description of the dataframe db, try:

```
df.info()
```

This gives some information about the dataframe: 1) the number of entries, 2) information of every column (in a long list), 3) the datatypes in your dataframe, and 4) the memory usage.

5) Use the data to answer some questions.

We can find some useful stats about the columns. Try for example:

```
df.mean()
```

```
df.max()
```

```
df.max()
```

We can also select only the column we're interested in:

```
df2 = df['FTHG']
```

This creates a series of only the numbers of goals scored by the home team in each game. From this, we can use the same handy operations.

a) In E0.csv, what is the most number of goals scored by a home team (FTHG - represents Full Time Home Goals)? And for an away team (FTAG - Full Time Away Goals)?

Another useful function is groupby. This can be used to group the data based on certain attributes. If you try:

```
df.groupby("HomeTeam)
```

The results will not be very helpful. To get some more clarity, try:

```
df.groupby("HomeTeam).groups
```

Combine groupby, selection of column, and an operation on the data to answer the following questions:

b) Which team scored the highest number of Home Shots (HS) in one match? And the away team (AS)? Which team had the most shots across all games?

6) We can also play around with the table (let's try out your googling skills for this one!)

a) Change type of all values in AF (Away Team Fouls Committed) column to float.

b) List all the full-time results (FTR) and sum the total number of distinct results.

c) When encountering larger datasets, it's helpful to reduce the space it takes up in memory by dropping unnecessary columns. Try *dropping* the 'Div', 'Date' and 'Referee' columns (note that most DataFrame functions are not inplace, hence you'll need to assign the result to itself for it to stay that way).

7) Creating new dataframes from dictionaries in Jupyter is simple.

Here's a couple different ones to try out:

```
raw_data_1 = pd.DataFrame({ 'subject_id': ['1', '2', '3', '4', '5'], 'first_name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'], 'last_name': ['Anderson', 'Ackerman', 'Ali', 'Aoni', 'Atiches']})
```

```
raw_data_2 =  pd.DataFrame({ 'subject_id': ['4', '5', '6', '7', '8'], 'first_name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'], 'last_name': ['Bonder', 'Black', 'Balwner', 'Brice', 'Btisan']})
```

```
raw_data_3 =  pd.DataFrame({ 'subject_id': ['1', '2', '3', '4', '5', '7', '8', '9', '10', '11'], 'test_id': [51, 15, 15, 61, 16, 14, 15, 1, 61, 16]})
```

a) Try making the following dataframe yourself

	Name	age	mat_number
0	Al	18	s1111111
1	Boris	21	s2222222
2	Carina	34	s3333333
3	Donna	19	s4444444

8) You can merge dataframes in different ways
(hint : refer to the documentation user guide for Merge, join, and concatenate (https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html) - check the examples!).

a) Join the first two dataframes along rows and assign it to allDataOne.

b) Join the first two dataframes along column and assign it to allDataTwo.

c) Merge allDataOne and third data along the subject_id.

d) Merge only the data that has the same 'subject_id' on both raw_data_1 and raw_data_2.
Using the how attribute (check the user guide!) try to get the following two tables:

i. Merge and display only the people with the same subject id.

	first_name_x	last_name_x	subject_id	first_name_y	last_name_y
0	Alice	Aoni	4	Billy	Bonder
1	Ayoung	Atiches	5	Brian	Black

ii. Merge on subject id and display all people.

	first_name_x	last_name_x	subject_id	first_name_y	last_name_y
0	Alex	Anderson	1	NaN	NaN
1	Amy	Ackerman	2	NaN	NaN
2	Allen	Ali	3	NaN	NaN
3	Alice	Aoni	4	Billy	Bonder
4	Ayoung	Atiches	5	Brian	Black
5	NaN	NaN	6	Bran	Balwner
6	NaN	NaN	7	Bryce	Brice
7	NaN	NaN	8	Betty	Btisan