INTRODUCTION TO PANDAS

AGENDA

- What is Pandas
- Difference between Series and DataFrame in pandas
- Difference between numpy array and pandas dataframe
- Learning by doing
- Cheat sheet
- Exercises

WHAT IS PANDAS?

- Python library to analyse data
- Used to:
  - explore
  - clean
  - process data
- Data table = DataFrame in Pandas
- Compatible with many file formats
  - ✓ csv
  - ✓ excel
  - ✓ sql
  - ✓ json

Useful links:

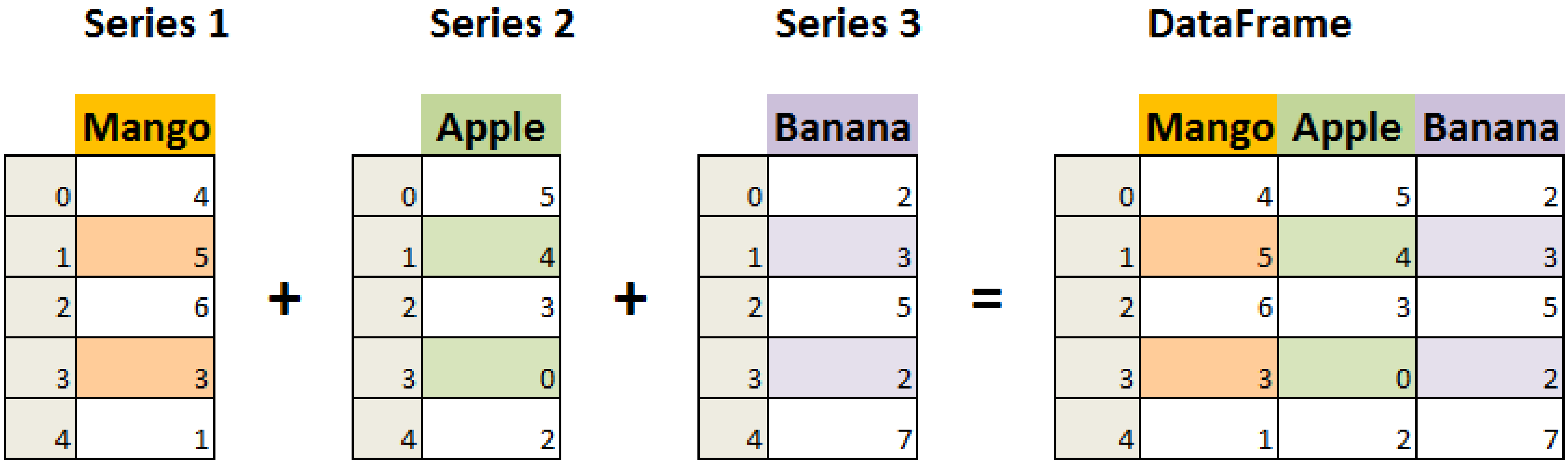
- ✓ https://pandas.pydata.org/docs/getting_started/index.html#getting-s
- ✓ https://pandas.pydata.org/docs/user_guide/index.html#user-guide

SERIES VS. DATAFRAME

Series	DataFrame
1-Dimensional array with any type of data	2-Dimensional tabular structure
Equivalent to a column in table	Equivalent to a table
Homogenous data type	Heterogenous data types
Immutable size	Mutable size

- By default: values labelled with index number
  - Possible to change the label
  - Label used to access specific values

https://www.naukri.com/learning/articles/series-vs-dataframe-in-pandas/



Series 1

	Mango
0	4
1	5
2	6
3	3
4	1

+

Series 2

	Apple
0	5
1	4
2	3
3	0
4	2

+

Series 3

	Banana
0	2
1	3
2	5
3	2
4	7

=

DataFrame

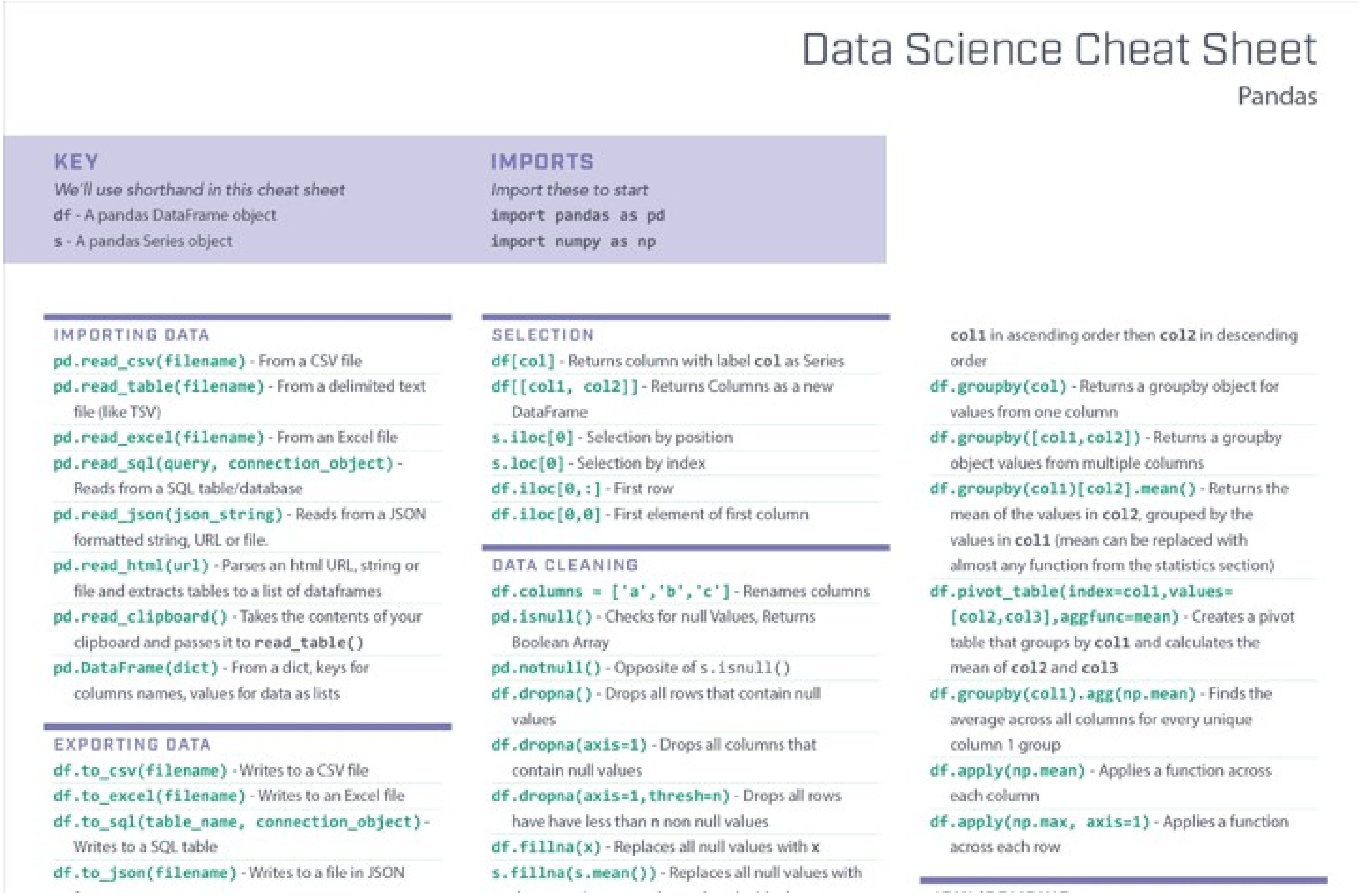
	Mango	Apple	Banana
0	4	5	2
1	5	4	3
2	6	3	5
3	3	0	2
4	1	2	7

SERIES VS. DATAFRAME

Characteristics	NumPy Array	Pandas DataFrame
Homogeneity	Only homogeneous elements (elements of same data type)	Heterogeneous elements
Mutability	Mutable	Mutable
Access	Using integer positions	Using both integer position & index
Flexibility	No flexibility to deal with dynamic data sequence & mixed data types	Have that flexibility
Dimension	Multi-dimensional	Two-dimensional
Used to	Perform mathematical (	Pre-process & handle data

https://www.askpython.com/python/pandas-dataframe-vs-numpy-arrays

CODING TOGETHER



Data Science Cheat Sheet

Pandas

KEY

We'll use shorthand in this cheat sheet

`df` - A pandas DataFrame object

`s` - A pandas Series object

IMPORTS

Import these to start

```
import pandas as pd
import numpy as np
```

IMPORTING DATA

`pd.read_csv(filename)` - From a CSV file

`pd.read_table(filename)` - From a delimited text file (like TSV)

`pd.read_excel(filename)` - From an Excel file

`pd.read_sql(query, connection_object)` - Reads from a SQL table/database

`pd.read_json(json_string)` - Reads from a JSON formatted string, URL or file.

`pd.read_html(url)` - Parses an html URL, string or file and extracts tables to a list of dataframes

`pd.read_clipboard()` - Takes the contents of your clipboard and passes it to `read_table()`

`pd.DataFrame(dict)` - From a dict, keys for columns names, values for data as lists

EXPORTING DATA

`df.to_csv(filename)` - Writes to a CSV file

`df.to_excel(filename)` - Writes to an Excel file

`df.to_sql(table_name, connection_object)` - Writes to a SQL table

`df.to_json(filename)` - Writes to a file in JSON

SELECTION

`df[col]` - Returns column with label `col` as Series

`df[[col1, col2]]` - Returns Columns as a new DataFrame

`s.iloc[0]` - Selection by position

`s.loc[0]` - Selection by index

`df.iloc[0,:]` - First row

`df.iloc[0,0]` - First element of first column

DATA CLEANING

`df.columns = ['a','b','c']` - Renames columns

`pd.isnull()` - Checks for null Values, Returns Boolean Array

`pd.notnull()` - Opposite of `s.isnull()`

`df.dropna()` - Drops all rows that contain null values

`df.dropna(axis=1)` - Drops all columns that contain null values

`df.dropna(axis=1,thresh=n)` - Drops all rows have have less than `n` non null values

`df.fillna(x)` - Replaces all null values with `x`

`s.fillna(s.mean())` - Replaces all null values with

`col1` in ascending order then `col2` in descending order

`df.groupby(col)` - Returns a groupby object for values from one column

`df.groupby([col1,col2])` - Returns a groupby object values from multiple columns

`df.groupby(col1)[col2].mean()` - Returns the mean of the values in `col2`, grouped by the values in `col1` (mean can be replaced with almost any function from the statistics section)

`df.pivot_table(index=col1,values=[col2,col3],aggfunc=mean)` - Creates a pivot table that groups by `col1` and calculates the mean of `col2` and `col3`

`df.groupby(col1).agg(np.mean)` - Finds the average across all columns for every unique column 1 group

`df.apply(np.mean)` - Applies a function across each column

`df.apply(np.max, axis=1)` - Applies a function across each row

DO IT YOURSELF

- ➢ Exercise 1

- ➢ Exercise 2

AARHUS
BSS

DEPARTMENT OF BUSINESS
DEVELOPMENT
AND TECHNOLOGY

AARHUS UNIVERSITY