Lecture 8:
Data Analysis with Python

Introduction to Python
efl Data Science Courses

Niklas Trimpe

Overview of what you've learned

Data types – symbols, numbers, Booleans

Data structures – tuples, lists, sets, dictionaries

Control structures – case distinction (if, else),
loops (for, while)

Functions

Libraries

- Basic python libraries – os, csv, re
- Data wrangling libraries – pandas, numpy

Data visualization

- Plot types
- Packages for plotting

→ **Fundamental tools for working with data**

Table of Contents

- Recap Intro to Python course
- Pandas cont‘d
- Exploratory Data Analysis (Live Coding Case Study)

Pandas – what you‘ve learned so far

- The structure and key characteristics of a pandas DataFrame
- Reading data to pandas DataFrames
- Handling missing values
- Data manipulation
- Data analysis with pandas

→ Until now, we concentrated on a single DataFrame

→ **In this lecture:** How to deal with multiple DataFrames

Pandas cont‘d

- In most data mining tasks, data is provided/retrieved from multiple data sources, and therefore, multiple files (such as csv, txt etc.)
- To get the data ready for analysis, we need to merge the relevant information from the different data sources
- The pandas package provides various functionalities for easily combining Series and DataFrame objects

- In this lecture, we will cover the most important standard techniques:
  - Concatenate DataFrames along rows and columns
  - Merge DataFrames on specific keys by different join logics

Sources and further reading:
https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html
https://www.datacamp.com/community/tutorials/joining-dataframes-pandas

Pandas – Concatenating DataFrames

- First, we create some dummy DataFrames on which we will perform all operations in this lecture

```
import pandas as pd

#create dummy DataFrames

Ddata1 = {
          'id': ['1', '2', '3', '4', '5'],
          'Feature1': ['A', 'C', 'E', 'G', 'I'],
          'Feature2': ['B', 'D', 'F', 'H', 'J']}

Ddata2 = {
          'id': ['1', '2', '6', '7', '8'],
          'Feature1': ['K', 'M', 'O', 'Q', 'S'],
          'Feature2': ['L', 'N', 'P', 'R', 'T']}

Ddata3 = {
          'id': ['1', '2', '3', '4', '5', '7', '8', '9', '10', '11'],
          'Feature3': [12, 13, 14, 15, 16, 17, 15, 12, 13, 23]}

df1 = pd.DataFrame(Ddata1, columns = ['id', 'Feature1', 'Feature2'])
df2 = pd.DataFrame(Ddata2, columns = ['id', 'Feature1', 'Feature2'])
df3 = pd.DataFrame(Ddata3, columns = ['id', 'Feature3'])
```

df1

	id	Feature1	Feature2
0	1	A	B
1	2	C	D
2	3	E	F
3	4	G	H
4	5	I	J

df2

	id	Feature1	Feature2
0	1	K	L
1	2	M	N
2	6	O	P
3	7	Q	R
4	8	S	T

df3

	id	Feature3
0	1	12
1	2	13
2	3	14
3	4	15
4	5	16
5	7	17
6	8	15
7	9	12
8	10	13
9	11	23

Pandas – Concatenating DataFrames

- To simply **concatenate** DataFrames **along the row**, we use the concat() function in pandas
- Pass the names of the DataFrames in a list [df1, df2] as the argument to the concat() function

```
#concatenate two DataFrames along the row
df_row = pd.concat([df1, df2])
```

df1

	id	Feature1	Feature2
0	1	A	B
1	2	C	D
2	3	E	F
3	4	G	H
4	5	I	J

df2

	id	Feature1	Feature2
0	1	K	L
1	2	M	N
2	6	O	P
3	7	Q	R
4	8	S	T

→ df_row

	id	Feature1	Feature2
0	1	A	B
1	2	C	D
2	3	E	F
3	4	G	H
4	5	I	J
0	1	K	L
1	2	M	N
2	6	O	P
3	7	Q	R
4	8	S	T

Attention!

- Since df1 and df2 have the same indexes, this operation results in a DataFrame with duplicative indexes
- To automatically adjust the indexes after concatenating both DataFrames, set the argument ignore_index as True

Pandas – Concatenating DataFrames

- To simply **concatenate** DataFrames **along the row**, we use the concat() function in pandas
- Pass the names of the DataFrames in a list [df1, df2] as the argument to the concat() function

```
#concatenate two DataFrames along the row, ignore_index = True
df_row = pd.concat([df1, df2], ignore_index = True)
```

df1

	id	Feature1	Feature2
0	1	A	B
1	2	C	D
2	3	E	F
3	4	G	H
4	5	I	J

df2

	id	Feature1	Feature2
0	1	K	L
1	2	M	N
2	6	O	P
3	7	Q	R
4	8	S	T

→ df_row

	id	Feature1	Feature2
0	1	A	B
1	2	C	D
2	3	E	F
3	4	G	H
4	5	I	J
5	1	K	L
6	2	M	N
7	6	O	P
8	7	Q	R
9	8	S	T

Pandas – Concatenating DataFrames

- pandas provides the option to label the DataFrames, after the concatenation, with a key so that you know which data came from which DataFrame.
- Pass the optional argument keys specifying the labels of the DataFrames in a list

```
#concatenate two DataFrames along the row using labes for each DataFrame
df_keys = pd.concat([df1, df2], keys = ['df1', 'df2'])
```

df1

	id	Feature1	Feature2
0	1	A	B
1	2	C	D
2	3	E	F
3	4	G	H
4	5	I	J

df2

	id	Feature1	Feature2
0	1	K	L
1	2	M	N
2	6	O	P
3	7	Q	R
4	8	S	T

→ df_keys

		id	Feature1	Feature2
df1	0	1	A	B
df1	1	2	C	D
df1	2	3	E	F
df1	3	4	G	H
df1	4	5	I	J
df2	0	1	K	L
df2	1	2	M	N
df2	2	6	O	P
df2	3	7	Q	R
df2	4	8	S	T

Pandas – Concatenating DataFrames

- pandas provides the option to label the DataFrames, after the concatenation, with a key so that you know which data came from which DataFrame.
- Pass the optional argument keys specifying the labels of the DataFrames in a list

```
#concatenate two DataFrames along the row using labes for each DataFrame
df_keys = pd.concat([df1, df2], keys = ['df1', 'df2'])
```

- Using the loc method, you can retrieve the data of each DataFrame

df_keys

		id	Feature1	Feature2
df1	0	1	A	B
df1	1	2	C	D
df1	2	3	E	F
df1	3	4	G	H
df1	4	5	I	J
df2	0	1	K	L
df2	1	2	M	N
df2	2	6	O	P
df2	3	7	Q	R
df2	4	8	S	T

→

```
df_keys.loc['df2']
```

	id	Feature1	Feature2
0	1	K	L
1	2	M	N
2	6	O	P
3	7	Q	R
4	8	S	T

Pandas – Concatenating DataFrames

- To **concatenate** DataFrames **along the columns**, we specify the axis parameter as 1
- This method automatically joins both DataFrames by the index

```
#concatenate two DataFrames along the column
df_col = pd.concat([df1, df2], axis = 1)
```

df1

	id	Feature1	Feature2
0	1	A	B
1	2	C	D
2	3	E	F
3	4	G	H
4	5	I	J

df2

	id	Feature1	Feature2
0	1	K	L
1	2	M	N
2	6	O	P
3	7	Q	R
4	8	S	T

→ df_col

	id	Feature1	Feature2	id	Feature1	Feature2
0	1	A	B	1	K	L
1	2	C	D	2	M	N
2	3	E	F	6	O	P
3	4	G	H	7	Q	R
4	5	I	J	8	S	T

Pandas – Merging DataFrames

- Another operation related to DataFrames is the **merging** operation
- Two DataFrames might hold different kinds of information about the same entity and linked by some common feature/column
- pandas has full-featured, high performance in-memory join operations similar to relational databases like SQL
- pandas provides a single function, merge(), as the entry point for all standard database join operations between **DataFrame** or named **Series** objects
- The how argument to merge() specifies how to determine which keys are to be included in the resulting table

Merge method	SQL Join Name	Description
left	LEFT OUTER JOIN	Use keys from left frame only
right	RIGHT OUTER JOIN	Use keys from right frame only
outer	FULL OUTER JOIN	Use union of keys from both frames
inner	INNER JOIN	Use intersection of keys from both frames

Pandas – Merging DataFrames

- Use the merge function and pass the names of the DataFrames as well as the name of the common column as the argument on
- Per default the merge() function will perform an inner join using the intersection of keys from both DataFrames, otherwise the merge method needs to be specified with the how argument

left

	key	A	B
0	K0	A0	B0
1	K2	A2	B2
2	K3	A3	B3

right

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2

```
pd.merge(left, right, on='key')
```

→

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K2	A2	B2	C2	D2

```
pd.merge(left, right, on='key', how = 'outer')
```

→

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K2	A2	B2	C2	D2
2	K3	A3	B3	NaN	NaN
3	K1	NaN	NaN	C1	D1

```
pd.merge(left, right, on='key', how = 'left')
```

→

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K2	A2	B2	C2	D2
2	K3	A3	B3	NaN	NaN

```
pd.merge(left, right, on='key', how = 'right')
```

→

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K2	A2	B2	C2	D2
3	K1	NaN	NaN	C1	D1

Pandas – Merging DataFrames

- Use the merge function and pass the names of the DataFrames as well as the name of the common column as the argument on
- Per default the merge() function will perform an inner join using the intersection of keys from both DataFrames

```
#merge two dataframes based on a common column
df_merge_col = pd.merge(df_row, df3, on='id')
```

df_row

	id	Feature1	Feature2
0	1	A	B
1	2	C	D
2	3	E	F
3	4	G	H
4	5	I	J
5	1	K	L
6	2	M	N
7	6	O	P
8	7	Q	R
9	8	S	T

df3

	id	Feature3
0	1	12
1	2	13
2	3	14
3	4	15
4	5	16
5	7	17
6	8	15
7	9	12
8	10	13
9	11	23

→ df_merge_col

	id	Feature1	Feature2	Feature3
0	1	A	B	12
1	1	K	L	12
2	2	C	D	13
3	2	M	N	13
4	3	E	F	14
5	4	G	H	15
6	5	I	J	16
7	7	Q	R	17
8	8	S	T	15

Pandas – Merging DataFrames

- It might happen that the column on which you want to merge the DataFrames have different names
- For such merges, you will have to specify the arguments left_on as the left DataFrame's column name and right_on as the right DataFrame's column name

```
#merge two dataframes with different names for the common column
df_merge_difkey = pd.merge(left, right, left_on='key', right_on='id')
```

left

	key	A	B
0	K0	A0	B0
1	K2	A2	B2
2	K3	A3	B3

right

	id	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2

→ df_merge_difkey

	key	A	B	id	C	D
0	K0	A0	B0	K0	C0	D0
1	K2	A2	B2	K2	C2	D2

Pandas – Merging DataFrames

- You can also use multiple keys to merge DataFrames
- Just pass a list of keys ['key1', 'key2’] as an argument to on, left_on, or right_on

```
#merge two dataframes with multiple keys
df_merge_multkey = pd.merge(left, right, on=['key1', 'key2'])
```

left

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

right

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

→ df_merge_multkey

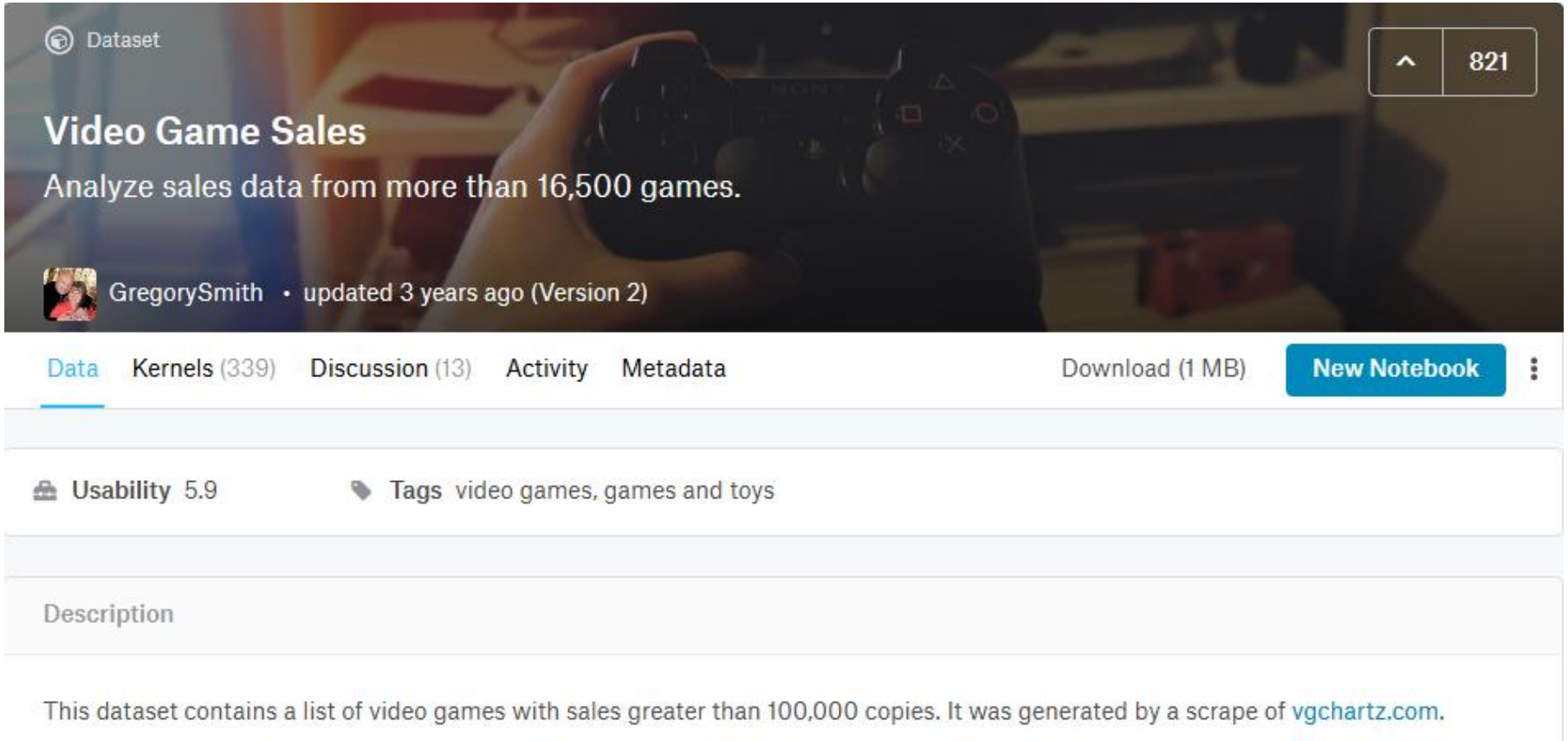
	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2

Table of Contents

- Recap Intro to Python course
- Pandas cont‘d
- Exploratory Data Analysis

Exploratory Data Analysis

- We again use the video game sales data set from kaggle



Exploratory Data Analysis

- We again use the video game sales data set from kaggle
- This data set contains a list of video games with sales greater than 100,000 copies
- The data set contains the following information
  - **Rank** - Ranking of overall sales
  - **Name** - The games name
  - **Platform** - Platform of the games release (i.e. PC,PS4, etc.)
  - **Year** - Year of the game's release
  - **Genre** - Genre of the game
  - **Publisher** - Publisher of the game
  - **NA_Sales** - Sales in North America (in millions)
  - **EU_Sales** - Sales in Europe (in millions)
  - **JP_Sales** - Sales in Japan (in millions)
  - **Other_Sales** - Sales in the rest of the world (in millions)
  - **Global_Sales** - Total worldwide sales

Exploratory Data Analysis

- We import the data and perform the same pre-processing tasks as in lectures 6 to drop rows with missing values

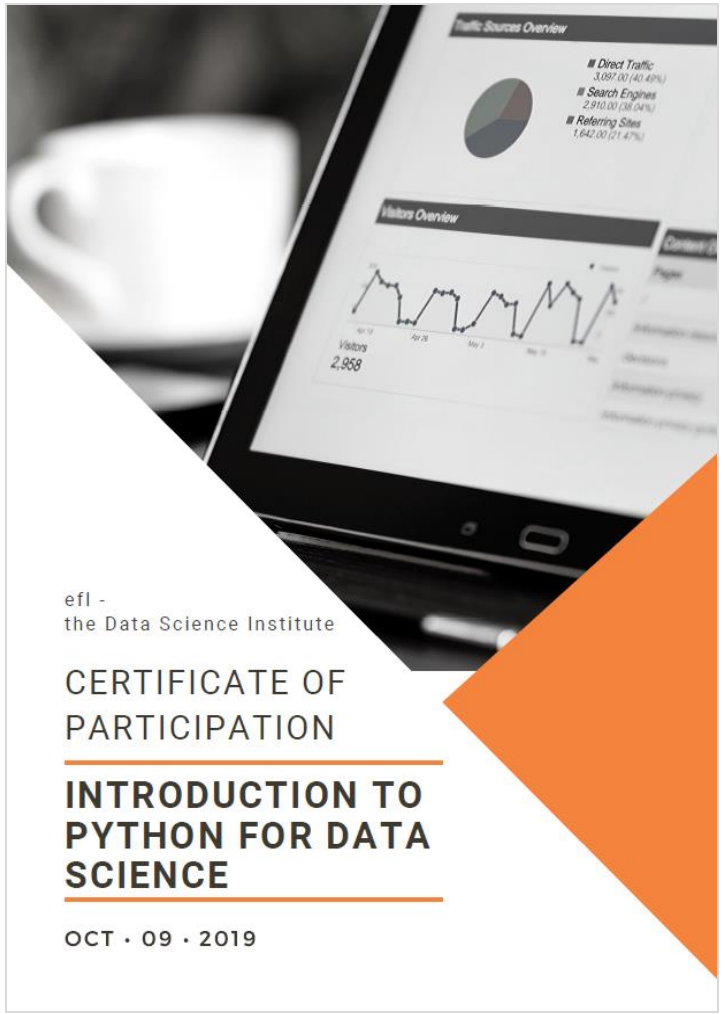
```
# Import libraries
import pandas as pd
import numpy as np

#read data to dataframe
df = pd.read_csv("../vgsales.csv")

################## perform cleaning from the pandas lecture ###################
#get information on dataframe
print(df.info())

#check missing values
df_null_values = df.isnull().sum()
print(df_null_values)

#drop missing values
df = df.dropna(axis=0)
df.info()

#convert year to int
df['Year'] = df['Year'].astype('int64’)
df.info()

#reset the index
df = df.reset_index(drop=True)
```

→ The data is ready for analysis ☺

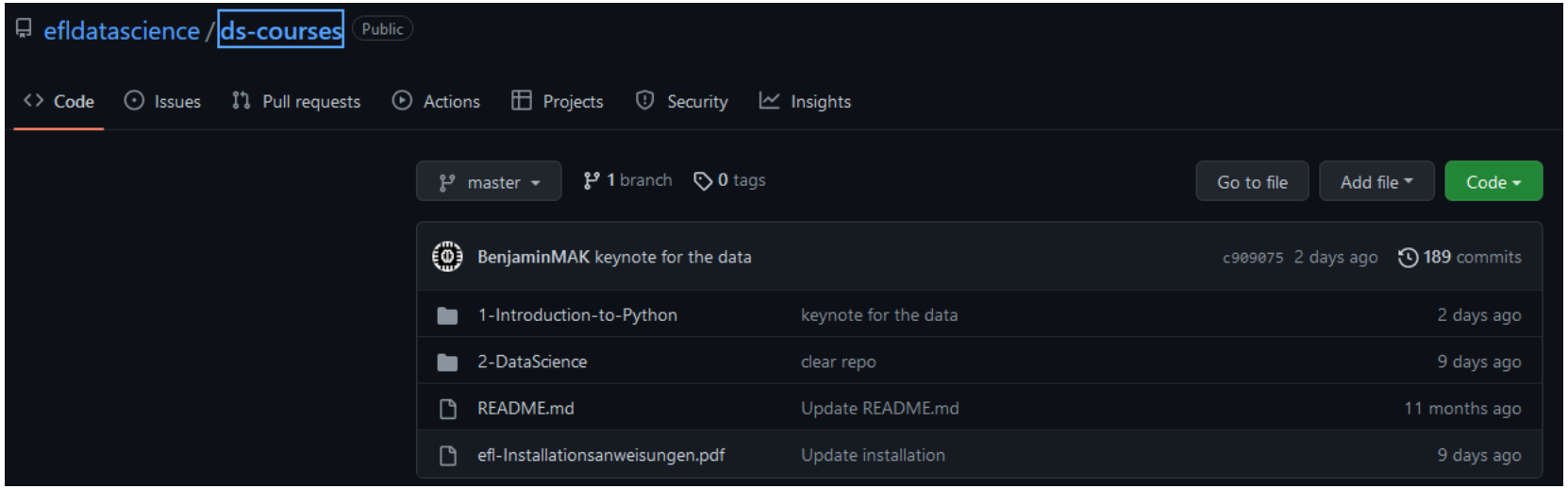
Information for the assignment

Requirements for receiving the course certificate:

- Completion of the tasks
- Description of the results of the tasks and how the tasks were solved
- **Formalities:**
  - Python code with comments and answers to the respective questions
- **Upload assignments:** 19.12.2025
- **Submission deadline:** 16.01.2026
- Submit to dscourses@eflab.de with the subject „Lösungen zum Python-Quiz“



Assignment tasks



The assignment tasks will be available in our course repository from 19th December on:

https://github.com/efl-DataScienceInstitute/ds-courses-students/tree/main