

## 1. Python Grundlagen

### 1.1. Lesen

Lesen Sie die Kapitel 4 bis 7 aus dem Buch <https://doi.org/10.1007/978-3-658-26133-7> (PDF aus dem Netz der HOST). NICHT aber Kapitel 5.4 Mengen, 7.3 Globale und lokale Variablen, 7.6 Dekorateure.

Am besten nicht nur Lesen, sondern gleich ausprobieren!



### 1.2. Selber programmieren

#### a) Raw string

Geben Sie mit `print(r'...')` folgenden Text aus:

```
Newline\n, Backslash\""
```

#### b) Relation

Definieren Sie eine Variable (als Dictionary von Arrays), welche folgende Relation abbildet:

year	temp
2000	12
2001	13
2002	20
2003	19
2004	16
2005	15
2006	15.2
2007	16
2008	30
2009	20
2010	10

#### c) Selektion

Geben Sie die Temperatur von 2005 aus. Hinweis: `index()` findet einen Wert in einer Liste.

#### d) Funktion

Definieren Sie eine Funktion, die für ein gegebenes Jahr die Temperatur zurückliefert.

## Übungen für Python Pandas Kapitel 1

Schreiben Sie ein `assert`, um die Funktion zu testen.

### e) Schleife und Ausgabe

Schreiben Sie eine `for`-Schleife, die für die Jahre 2005 bis 2010 folgende Ausgabe erzeugt (verwenden Sie `print(f'...')`):

```
Im Jahr 2005 war es 15 °C warm.  
Im Jahr 2006 war es 15.2 °C warm.  
Im Jahr 2007 war es 16 °C warm.  
Im Jahr 2008 war es 30 °C warm.  
Im Jahr 2009 war es 20 °C warm.
```

### f) Aggregation

Berechnen Sie die Durchschnittstemperatur über alle Jahre der Relation.

### g) Datentyp

Was ist der Datentyp der Spalte „temp“?

Moral der Übung:

- Wir werden recht wenig Standard-Python benötigen. Wenn man hiermit zurechtkommt, reicht es für einen Start.
- Man kann alles, was wir mit Pandas machen werden, auch mit Standard-Python hinbekommen. Es wird aber nicht so einfach gehen.

## 2. DataFrame Konstruktor

### 2.1.

- a) Erzeugen Sie ein DataFrame für die folgende Relation „wetter“

Die Reihenfolge in der Spalte „Stunde“ ist absichtlich unsortiert. Bitte so übernehmen.






Stunde	Temperatur	Taupunkt	Luftdruck	Regen	Sonne	Wind max	Windrichtung
6	-0,3	-2	1.011	0	0	12	250
7	0,6	-1	1.010	0	0	16	263
8	1,3	-0	1.010	0	0	22	270
9	1,9	0	1.010	0	17	22	272
10	2,3	1	1.009	0	28	33	269
11	2,4	1	1.008	0	27	35	264
12	3,1	1	1.007	0	38	35	266
13	3,4	2	1.006	0	33	60	264
14	3,5	2	1.004	0	13	44	256
15	3,4	2	1.003	0	1	48	252
16	3,5	2	1.001	1	0	56	251
17	3,6	2	1.000	1	0	53	252
18	4,0	3	999	1	0	50	257
19	4,6	3	997	2	0	52	258
20	4,8	3	997	3	0	54	266
21	4,7	3	996	4	0	53	274
22	3,3	1	995	4	1	95	282
23	2,8	-0	994	5	0	68	287
1	-0,6	-2	1.012	0	0	13	261
2	-0,8	-2	1.012	0	0	11	248
3	-0,7	-2	1.012	0	0	9	244
4	-0,4	-2	1.011	0	0	10	270
5	-0,3	-2	1.011	0	0	12	277

- b) Prüfen/korrigieren Sie die Datentypen. Der Taupunkt soll denselben Datentyp wie die Temperatur haben.
- c) Wie viele Zeilen/Spalten hat der DataFrame „wetter“?
- d) Für ein DataFrame `df` kann man mit `df.T` die Tabelle transponieren. Was passiert dann mit den dtypes und warum? Wie sieht der Index dann aus?

### 2.2. Webseite lesen

Lesen Sie die erste Tabelle der Webseite

[https://en.wikipedia.org/wiki/List\\_of\\_states\\_and\\_territories\\_of\\_the\\_United\\_States](https://en.wikipedia.org/wiki/List_of_states_and_territories_of_the_United_States)

States of the United States of America												
Flag, name and postal abbreviation <sup>[13]</sup>	Cities			Ratification or admission <sup>[C]</sup>	Population <sup>[15]</sup>	Total area <sup>[16]</sup>		Land area <sup>[16]</sup>		Water area <sup>[16]</sup>		Number of Reps.
	Capital	Largest <sup>[17]</sup>				mi²	km²	mi²	km²	mi²	km²	
 Alabama	AL	Montgomery	Birmingham	Dec 14, 1819	5,024,279	52,420	135,767	50,645	131,171	1,775	4,597	7
 Alaska	AK	Juneau	Anchorage	Jan 3, 1959	733,391	665,384	1,723,337	570,641	1,477,953	94,743	245,384	1
 Arizona	AZ	Phoenix		Feb 14, 1912	7,151,502	113,990	295,234	113,594	294,207	396	1,026	9
 Arkansas	AR	Little Rock		Jun 15, 1836	3,011,524	53,179	137,732	52,035	134,771	1,143	2,961	4
 California	CA	Sacramento	Los Angeles	Sep 9, 1850	39,538,223	163,695	423,967	155,779	403,466	7,916	20,501	53
Colorado	CO	Denver		Aug 1, 1876	5,773,714	104,094	269,601	103,642	268,431	452	1,170	7

## Übungen für Python Pandas Kapitel 1

Passen Sie Ihr DataFrame so an, dass es so für die weitere Analyse aussieht (Spalten entfernen, umbenennen, es interessieren nur km und nicht Meilen).

```
: df.head()
```

	state	abbrev	capital	largest_pop_city	established	population	total area	land area	water area	#reps
0	Alabama	AL	Montgomery	Birmingham	Dec 14, 1819	5024279	135767	131171	4597	7
1	Alaska	AK	Juneau	Anchorage	Jan 3, 1959	733391	1723337	1477953	245384	1
2	Arizona	AZ	Phoenix	Phoenix	Feb 14, 1912	7151502	295234	294207	1026	9
3	Arkansas	AR	Little Rock	Little Rock	Jun 15, 1836	3011524	137732	134771	2961	4
4	California	CA	Sacramento	Los Angeles	Sep 9, 1850	39538223	423967	403466	20501	53

## 3. Projektion

### 3.1. Schema

Wir arbeiten mit dem DataFrame „wetter“ aus Aufgabe 2 weiter.

- Benenne das Attribut „Wind max“ um in „Windspitze“.
- Benenne in einem Schritt alle Attribute um in (a, b, c, d, ...). Könnte das Kommando in einer Anweisungskette verwendet werden?
- Wir arbeiten mit „wetter“ ohne Anwendung von b) weiter. Geben Sie die Daten ohne die Wind-Attribute und in einer anderen Reihenfolge der Attribute aus.

### 3.2. Erweiterte Projektion

- Ergänzen Sie „wetter“ um ein Attribut „spread“ berechnet als Temperatur – Taupunkt.
- Löschen Sie das Attribut „Taupunkt“.
- Löschen Sie alle Attribute außer Spread, Stunde und Taupunkt.
- Betrachten Sie folgende Anweisung

```
df["mortality"] = df["death"] / df["cases"] * 100
```

df

	year	state	cases	death	mortality
0	2000	MV	0	0	NaN
1	2001	MV	1	0	0.000000
2	2002	Saxen	55	7	12.727273
3	2000	Bayern	33	4	12.121212

Drücken Sie dies äquivalent mit SQL aus (bitte auch das Schema vorher und nachher beachten; was passiert bei Division durch 0?).

- Rechnen Sie die Temperatur in Grad Fahrenheit um.
- Kette

Wir starten wieder mit dem ursprünglichen „wetter“.

Schreiben Sie ein Kommando (mit mehreren verketteten Methoden), welches

- den Datentyp des Taupunktes verändert wie bei Aufgabe 2.1 b)
- das Attribut „Wind max“ umbenennt
- das Attribut „spread“ berechnet
- die Attribute „Regen“ und „Sonne“ entfernt

## 4. Selektion

Wir starten wieder mit dem ursprünglichen „wetter“.

Schreiben Sie folgendes SQL äquivalent in Pandas

- 4.1. `SELECT * FROM "wetter" WHERE "Temperatur" < 0;`
- 4.2. Definieren Sie den Index mit dem Schema ("Stunde").
- 4.3. `SELECT * FROM "wetter" WHERE "Stunde" IN (5, 1, 3);`
- 4.4. `SELECT * FROM "wetter" WHERE "Stunde" BETWEEN 3 and 6;`
- 4.5. Der Index sei immer noch ("Stunde").  
`SELECT * FROM "wetter" WHERE "Stunde" <= 20 AND "Regen" > 0;`
- 4.6. `SELECT "Luftdruck", "Temperatur"`  
`FROM "wetter"`  
`WHERE "Wind max" > 70 OR "Taupunkt" > 2;`
- 4.7. Erzeugen Sie ein neues Attribut "Text" mit den Werten 'sonne' wenn die Sonne scheint (Wert in der Spalte > 0), 'regen' wenn es regnet und 'april' wenn beides der Fall ist. Ansonsten soll der Wert pd.NA sein.
- 4.8. `SELECT * FROM "wetter" WHERE "Text" IN ('april', 'regen');`

## 5. pd.NA

### 5.1. Welche pandas Datentypen unterstützen pd.NA Werte?

Probieren Sie es aus, indem eine Series oder DataFrame für den Datentyp bauen und dann mit None, np.NAN und pd.NA arbeiten. Welche dieser Werte nimmt der Datentyp auf? Verifizieren Sie ggf. dass wirklich pd.NA als Wert im Feld steht (und nicht nur, dass „NA“ beim print() angezeigt wird).

## 6. Gruppieren und Aggregieren

Wir verwenden das Titanic Dataset

url = <https://raw.githubusercontent.com/pandas-dev/pandas/master/doc/data/titanic.csv>

### 6.1. Einfache Gruppierungen

Geben Sie jeweils auch ein passendes SQL-SELECT an.

- a) Wie viele Passagiere haben keine Information zu ihrer Kabine?
- b) Haben nur Passagiere der dritten Klasse keine Kabine oder haben wir eher ein Datenqualitätsproblem?
- c) Welche unterschiedlichen Kabinen gab es? Erstellen Sie eine sortierte Liste.
- d) Wie viele unterschiedliche Kabinen gab es?
- e) Gibt es einen Zusammenhang zwischen der Klasse und der Kabinennummer?
- f) Wie viele Passagiere waren im Schnitt und wie viele maximal in einer Kabine?
- g) In wie viele Kabinen waren Minderjährige gebucht?

### 6.2. Minimum

Das Minimum ist nicht eindeutig:  $\min\{1, 3, 6, 3, 8, 5, 3, 3\} = 3$  (der Wert 3 wird mehrfach angenommen).

- a) Finden Sie den kleinsten und größten Fahrpreis je Klasse.
- b) Oft ist man nicht nur am Wert, sondern auch an den zugehörigen Daten interessiert, die zu diesem Wert gehören. Finden Sie mit `idxmin()` die Namen der Passagiere mit den extremen Fahrpreisen aus a) heraus.
- c) Falls das Minimum / Maximum nicht eindeutig ist, welchen Wert liefert dann `idxmin()`?
- d) Wir würden gerne alle Passagiere mit extremen Fahrpreisen gemäß a) kennen.

In SQL ist das auch etwas umständlich. Hier könnte man für das Minimum so vorgehen:

```
SELECT Pclass, Name, Fare
FROM titanic t
WHERE Fare = (SELECT MIN(Fare) FROM titanic m WHERE m.Pclass =
t.Pclass)
ORDER BY Pclass;
```

Wie lösen Sie das in pandas (Tipp: `nlargest()` oder `rank()`)?

### 6.3. Quantile

Berücksichtigt `quantile()` Null-Werte? Testen Sie.

### 6.4. Aggregation und Sortierung

Für welche der vorgestellten Aggregations-Funktionen beeinflusst die Sortierung der Daten im DataFrame das Ergebnis?



### 6.5. Duplikate

- a) Mit dem Attribut `.is_unique` kann man herausfinden, ob eine Series nur eindeutige Werte enthält.  
Überprüfen Sie, ob („Name“) ein Schlüssel ist.
- b) Definieren Sie „Name“ als Index und stellen Sie per `assert` sicher, dass er eindeutig ist.
- c) Überprüfen Sie, ob („Ticket“, „Cabin“) ein Schlüssel ist.
- d) Mit `unique()` könnte man sich eine Liste erzeugen, die keine Duplikate mehr enthält. Oft möchte man jedoch wissen, wo die Duplikate herkommen. In SQL würde man es so machen (dann muss man aber immer noch ein `SELECT` herumbauen, um die zugehörigen Passagiere herauszufinden):

```
SELECT Ticket, Cabin, COUNT(*)  
FROM titanic  
GROUP BY Ticket, Cabin  
HAVING COUNT(*) > 1
```

Ermitteln Sie die Passagiere, für die es mindestens 4 (damit das Ergebnis kleiner wird) Duplikate für („Ticket“, „Cabin“) erzeugen.

- e) Ermitteln Sie die Duplikate von d) über `value_counts()`.

### 6.6. Wie wird man ein SQL-HAVING in pandas abbilden?

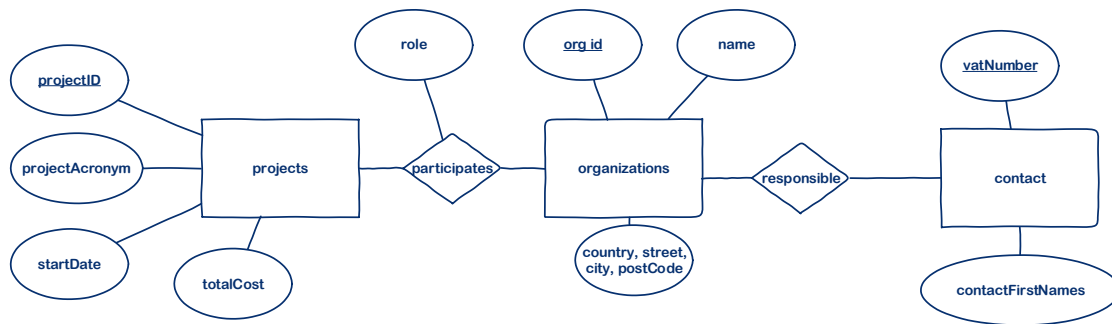
## 7. Join

### 7.1. H2020 Projekte

Grundlagen sind die „H2020 Organizations“ und „H2020 Projects“ auf der Webseite.

<https://data.europa.eu/euodp/de/data/dataset/cordisH2020projects>

Ein ERM zur Erläuterung (stimmt nicht ganz: eine organization kann demselben project mehrfach zugeordnet sein mit je unterschiedlicher role):



a) Laden Sie die entsprechenden CSV Dateien

Beachten Sie, dass die Behörde europäisch ist und setzen Sie daher beim Laden der CSV Datei das CSV-Trennzeichen und Dezimaltrennzeichen entsprechend.

b) Wie viele Projekte und Organisationen im Programm H2020 gibt es?

In den beiden Dateien steht jeweils die „id“ für den Schlüssel der Organisation (org id) bzw. des Projektes (projectID). Man könnte die „id“ sicherheitshalber gleich umbenennen.

c) Joinen Sie die beiden Dateien über die Projekt ID.

d) Welche Organisation hat das größte Budget (Summe aller Projekte) und wie hoch ist es?

e) Wie viele Zeilen wurden nicht erfolgreich gejoint?

## 8. RDMBS

- INSERT nachbauen
- UPDATE nachbauen  
Halbiere den Titanic-Fare für Frauen.
- DELETE nachbauen  
df.drop(index='value', inplace=True) über den Index
- UNION, INTERSECT, MINUS nachbauen
- Subqueries nachbauen