# Merkblatt Sortieren und Gruppieren

## 1 Pandas: Sortieren und Gruppieren

[2]:
```
import numpy as np
import pandas as pd

df = pd.read_csv("../../data/Temperature/GlobalLandTemperaturesByMajorCity.csv.
↪bz2")
df["dt"] = pd.to_datetime(df["dt"])
df["dtYear"] = df["dt"].dt.year

df.head()
```

[2]:

| | dt | AverageTemperature | AverageTemperatureUncertainty | City | Country | Latitude | Longitude | dtYear |
|---|---|---|---|---|---|---|---|---|
| 0 | 1849-01-01 | 26.704 | 1.435 | Abidjan | Côte D'Ivoire | 5.63N | 3.23W | 1849 |
| 1 | 1849-02-01 | 27.434 | 1.362 | Abidjan | Côte D'Ivoire | 5.63N | 3.23W | 1849 |
| 2 | 1849-03-01 | 28.101 | 1.612 | Abidjan | Côte D'Ivoire | 5.63N | 3.23W | 1849 |
| 3 | 1849-04-01 | 26.140 | 1.387 | Abidjan | Côte D'Ivoire | 5.63N | 3.23W | 1849 |
| 4 | 1849-05-01 | 25.427 | 1.200 | Abidjan | Côte D'Ivoire | 5.63N | 3.23W | 1849 |

### 1.1 Sortieren

Nachfolgende Programmzeile sortiert das Dataframe `df` zuerst aufsteigend alphabetisch nach dem Namen des Landes (Spalte: *Country*), danach nach der alphabetischen Reihfolge der Spalte *City*. Zelle mit nicht vorhandenem Wert - NaN - werden zuletzt ausgegeben (hier im Beispiel nicht vorhanden).

**Hinweis:** Der Backslash \ am Ende einer Zeile ermöglicht es uns, die Code-Zeile in der nächsten Zeile fortzusetzen. Dadurch können wir den Code schön formatieren:

[3]:
```
df\
    .sort_values(
```

```
        by = ["Country", "City"],
        ascending=True,
        na_position="last")\
    .head()
```

[3]:

| | dt | AverageTemperature | AverageTemperatureUncertainty | City | Country | Latitude | Longitude | dtYear |
|---|---|---|---|---|---|---|---|---|
| 105635 | 1833-01-01 | 2.290 | 2.487 | Kabul | Afghanistan | 34.56N | 70.05E | 1833 |
| 105636 | 1833-02-01 | 3.319 | 2.325 | Kabul | Afghanistan | 34.56N | 70.05E | 1833 |
| 105637 | 1833-03-01 | 7.444 | 2.733 | Kabul | Afghanistan | 34.56N | 70.05E | 1833 |
| 105638 | 1833-04-01 | 13.576 | 2.266 | Kabul | Afghanistan | 34.56N | 70.05E | 1833 |
| 105639 | 1833-05-01 | 19.321 | 2.362 | Kabul | Afghanistan | 34.56N | 70.05E | 1833 |

Mit der Methode `sort_values()` wird ein Dataframe sortiert. Standardmäßig wird hier dann eine Kopie erstellt, mit der wir weiter arbeiten können. Über den Parameter `inplace` können wir dies verändern - mehr dazu später..

Mit dem Parameter `by` wird angeben welche Spalte sortiert werden soll. Werden wie hier mehrere Spaltenüberschriften angegeben, dann wird erst nach der erste Spaltenüberschrift sortiert. Sollte diese in mehrenen Zeilen (Datensätzen) den gleichen Inhalt aufweisen, so werden diese Zeilen (Datensätze) mit gleichem Inhalt in dieser Spalte nach der der zweiten Spalte sortiert. Es können auch mehr als zwei Spalten angegeben werden.

Hierdurch kann sicher gestellt werden, dass eine Sortierreihenfolge nicht zufällig ist, wenn in der Spalte nach der zunächst sortiert wird mehrere Zeilen mit gleichen Inhalt vorkommen.

Der Parameter `ascending` gibt an, ob aufsteigend oder absteigend sortiert werden soll. Da es auch Werte vom Typ "NaN" geben kann, also inbesondere Zellen mit keiner Eingabe, kann definiert werden, wo diese in der Sortierung erscheinen sollen. Die Wahl besteht zwischen ganz am Anfang der Liste oder ganz am Ende.

Weiterhin gibt es wieder den Boolean-Parameter `inplace`. True bedeutet, dass das Dataframe selber geändert wird, d.h. die Variable, die vor dem `.sort_values()` steht, wird direkt angepasst. Wenn wir den Parameter auf False setzten (oder weglassen), wird ein neues DataFrame erstellt, welches wir entgegen nehmen können.

## 1.2 Gruppieren

[11]:
```
import numpy as np
```

Die nachfolgende Programmzeile gruppiert die Daten nach der Spalte "Country". Dann wird für jedes Land, für das Daten vorhanden sind, der maximale Wert der Spalte `AverageTemperature` ausgegeben.

**Wichtig:** Hier geben wir durch das `.head()` nur die ersten paar Einträge aus.

[5]:
```
df.groupby(by = "Country")\
    .agg(MaxAverageTemperature=("AverageTemperature", np.max))\
    .head()
```

[5]:
```
             MaxAverageTemperature
Country
Afghanistan                 27.583
Angola                      27.207
Australia                   23.013
Bangladesh                  30.669
Brazil                      29.986
```

Der beispielhafte Aufruf von ´groupby´ zeigt, dass er allein nicht ausreicht, ein Ergebnis zu erzeugen. Mit Hilfe von ´agg´ muss noch angegeben werden, wie die neue Spalte mit dem Ergebnis heißen soll, und mit welcher Funktion aus welcher Spalte das Ergebnis erstellt werden soll.

Für diese Rechenoperation benötigen wir dann die max-Funktion von Numpy (`np.max`).

**Wichtig:** Diese wird hier als Funktion übergeben, und dann erst durch Pandas ausgeführt. Daher finden sich hinter dem np.max keine runden Klammern. Um die Funktion direkt aufzurufen, würden wir `np.max([1, 2, 3])` schreiben - das dürfen wir hier aber beim `.agg()` nicht tun, hier muss die Funktion als Funktion übergeben werden!

Für Operationen mit Pandas wird häufig Numpy benötigt.

### 1.3 iloc()

Die obige Ausgabe zum Codebeispiel zum Sortieren der Daten zeigt das Problem oder auch das Feature. Durch Sortieren wird nicht die "Nummer" der Zeile bzw. des Datensatzes im Dataframe geändert. Mit `loc()` werden die gleichen Zeilen ausgegeben wie vor dem Sortieren.

Bei einigen Anwendungen ist es notwendig, Zeilen nach der Reihenfolge in der Sortierung anzusprechen. Hierfür gibt es `iloc()`.

Das nachfolgende Beispiel verdeutlich den Unterschied. Wie oben wird das Dataframe sortiert - diesmal aber `inplace`. Zeile bzw. Datensatz mit der Indexnummer 3 im Dataframe ist eine andere als in der Zeile 3 (also der vierten Zeile) in der sortieren Ausgabe.

[8]:
```
df.sort_values(by = ["Country", "City"], ascending = True, inplace=True,␣
 ↪na_position = "last")
df.head()
```

[8]:
```
                dt  AverageTemperature  AverageTemperatureUncertainty   City  \
105635 1833-01-01               2.290                          2.487  Kabul
105636 1833-02-01               3.319                          2.325  Kabul
105637 1833-03-01               7.444                          2.733  Kabul
105638 1833-04-01              13.576                          2.266  Kabul
105639 1833-05-01              19.321                          2.362  Kabul
```

```
            Country Latitude Longitude  dtYear
105635  Afghanistan   34.56N    70.05E    1833
105636  Afghanistan   34.56N    70.05E    1833
105637  Afghanistan   34.56N    70.05E    1833
105638  Afghanistan   34.56N    70.05E    1833
105639  Afghanistan   34.56N    70.05E    1833
```

[9]:
```
print(df.loc[3])
print("---")
print(df.iloc[3])
```

```
dt                                1849-04-01 00:00:00
AverageTemperature                              26.14
AverageTemperatureUncertainty                   1.387
City                                          Abidjan
Country                                 Côte D'Ivoire
Latitude                                        5.63N
Longitude                                       3.23W
dtYear                                           1849
Name: 3, dtype: object
---
dt                                1833-04-01 00:00:00
AverageTemperature                             13.576
AverageTemperatureUncertainty                   2.266
City                                            Kabul
Country                                   Afghanistan
Latitude                                        34.56N
Longitude                                       70.05E
dtYear                                           1833
Name: 105638, dtype: object
```

[ ]: