Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

FS24 CAS PML - Python

Niklaus Johner

niklausbernhard.johner@bfh.ch

FS24 CAS PML - Python

19. Pandas

Pandas Beschreibung

- Pandas ist das Python Modul für Datenanalyse und Manipulation
  - Datensätze vereinen (merge)
  - Daten gruppieren (d.h. in Gruppen aufsplitten)
  - Operationen auf Gruppen ausführen
  - Label-based indexing und slicing
- Stellt zwei Klassen zur Verfügung:
  - 1-Dimensionale Daten (*Series*)
  - 2-Dimensionale Daten mit Labels (*DataFrame*)

DataFrame

- Der *DataFrame* ist ähnlich mit *data.frame* in R
- 2-Dimensionale Daten mit Labels
  - Für Spalten
  - Für Zeilen
- Ist ein wrapper um *numpy.ndarray*
  - Operationen wie mit numpy arrays (element-wise mit broadcasting)
  - Man kann Masken verwenden für die Indizierung

DataFrame Initialisierung

- Der *DataFrame* wird einfach mit der Klasse instanziert

```
import pandas
col_names = ["A", "B", "C"]
row_names = ["first", "second", "third"]
data = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
data_frame = pandas.DataFrame(data, row_names, col_names)
```

- Der *DataFrame* sieht dann so aus:

```
In [14]: print(data_frame)
         A  B  C
first    1  2  3
second   4  5  6
third    7  8  9
```

DataFrame index-based indexing mit *iloc*

- Das *iloc* Attribut erlaubt index basiertes slicing von Kolonne und Zeile
- Wie in numpy

```
In [4]: data_frame.iloc[:2, 0:2]
Out[4]:
          A  B
first     1  2
second    4  5
```

DataFrame label-based indexing mit *loc*

- Das *loc* Attribut erlaubt Label basiertes slicing von Kolonne und Zeile
- Immer inklusiv
- Nur Label basiert

```
In [34]: data_frame.loc["first":"second", "B":"C"]
Out[34]:
         B  C
first    2  3
second   5  6
```

DataFrame indexing und Slicing

- Spalte kann mit dem Label geholt werden (ähnlich wie dictionary)

```
In [24]: data_frame["A"]
Out[24]:
first     1
second    4
third     7
Name: A, dtype: int64
```

- Ist auch als Attribut verfügbar

```
In [27]: data_frame.A
Out[27]:
first     1
second    4
third     7
Name: A, dtype: int64
```

DataFrame Indexing und Slicing

- Zeilen können mit Label-Slicing geholt werden

```
In [25]: data_frame["first":"second"]
Out[25]:
         A  B  C
first    1  2  3
second   4  5  6
```

- Oder mit normalem Slicing

```
In [26]: data_frame[1:2]
Out[26]:
         A  B  C
second   4  5  6
```

Achtung: Label-Slicing beinhaltet auch das end Label

DataFrame *apply*

- Die *apply* Methode erlaubt eine Funktion auf jedem Element oder *series* von einem DataFrame aufzurufen.

```
In [44]: import numpy as np
    ...: data_frame.apply(np.sqrt)
Out[44]:
                A         B         C
first    1.000000  1.414214  1.732051
second   2.000000  2.236068  2.449490
third    2.645751  2.828427  3.000000
```

Auf jedem Element

DataFrame *apply*

- Die *apply* Methode erlaubt eine Funktion auf jedem Element oder *series* von einem DataFrame aufzurufen.

```
In [45]: data_frame.apply(min)
Out[45]:
A    1
B    2
C    3
dtype: int64
```

Auf jeder Spalte (*serie*): Data Aggregation

DataFrame *groupby*

- Mit der *groupby* Methode werden Daten nach einer Spalte gruppiert
- Man kann dann auf dem zurückgegebenen Objekt Funktionen pro Gruppe anwenden

```
In [74]: col_names = ["A", "B"]
    ...: data = [[1, 2], [1, 4], [2, 4], [1, 2]]
    ...: data_frame = pandas.DataFrame(data, columns=col_names)

In [75]: grouped = data_frame.groupby("A")
    ...: grouped.mean()
Out[75]:
          B
A
1  2.666667
2  4.000000
```

DataFrame *merge*

- Mit der *merge* Methode werden Daten von zwei DataFrames vereinigt.

```
In [17]: books = pandas.DataFrame({
    ...:     "title": ["It", "Dune", "Carrie"],
    ...:     "author_id": [1, 2, 1]})
    ...: authors = pandas.DataFrame({
    ...:     "author_id": [1, 2],
    ...:     "name": ["King", "Herbert"]})
    ...: books.merge(authors, on="author_id")
Out[17]:
    title  author_id     name
0      It          1     King
1    Dune          2  Herbert
2  Carrie          1     King
```