Data Science 1

Tutorial Day 2

Prof. Dr. Christian Bockermann

Hochschule Bochum

Sommersemester 2022

Agenda

1 Wiederholung: Pandas

2 Von Excel (VBA) zu Pandas

Wiederholung: Pandas

Pandas: DataFrame

Ein DataFrame `df` ist eine Tabellenstruktur:

	a1	a2	a3
0	**4**	**1**	**2**
1	**5**	**1**	**3**
2	**3**	**8**	**7**

Pandas: DataFrame

Ein DataFrame `df` ist eine Tabellenstruktur:

	a1	a2	a3
0	**4**	**1**	**2**
1	**5**	**1**	**3**
2	**3**	**8**	**7**

Spalten-Index

`df.columns`

Zeilen-Index

`df.index`

Pandas: DataFrame

Ein DataFrame `df` ist eine Tabellenstruktur:

“Positionsindex”

		0	1	2
		a1	a2	a3
0	0	4	1	2
1	1	5	1	3
2	2	3	8	7

Spalten-Index

`df.columns`

Zeilen-Index

`df.index`

Ein DataFrame df mit anderen Indizes

```
df.index = ['A', 'B', 'C']
df.columns = ['x1','x2','x3']
```

“Positionsindex”

		0	1	2
		x1	x2	x3
0	A	4	1	2
1	B	5	1	3
2	C	3	8	7

Spalten-Index
df.columns

Zeilen-Index
df.index

Einzelne Zeilen/Spalten sind `Series` Objekte

Zugriff auf `df` über verschiedene Elemente:

```
df[0:2]     # Zeilen mit Slicing
df[['a1','a2']]  # Spalten durch Namensliste

# Zugriff mit Positionsindex:
df.iloc[zeilen,spalten]

# Zugriff mit Zeilen/Spalten-Index:
df.loc[zeilen,spalten]
```

Mit **df.iloc[...]** wird nach Position selektiert

```
df.iloc[ ZEILEN,SPALTEN ]
```

ZEILEN bzw. SPALTEN sind Zahlen, Listen von Zahlen, Slices

```
# Zelle in erster Zeile, dritter Spalte:
df.iloc[0,2]

#  Die erste Zeile (als Series Objekt!)
df.iloc[0,:]

# Die erste Spalte:
df.iloc[:,0]
```

Selektieren mit `.iloc[..]`

Beim Slicing a:b gehört a dazu, b nicht mehr:

```
# die Zeilen 0 und 1:
df.iloc[0:2,:]

# die Spalten 0 und 1:
df.iloc[:,0:2]
```

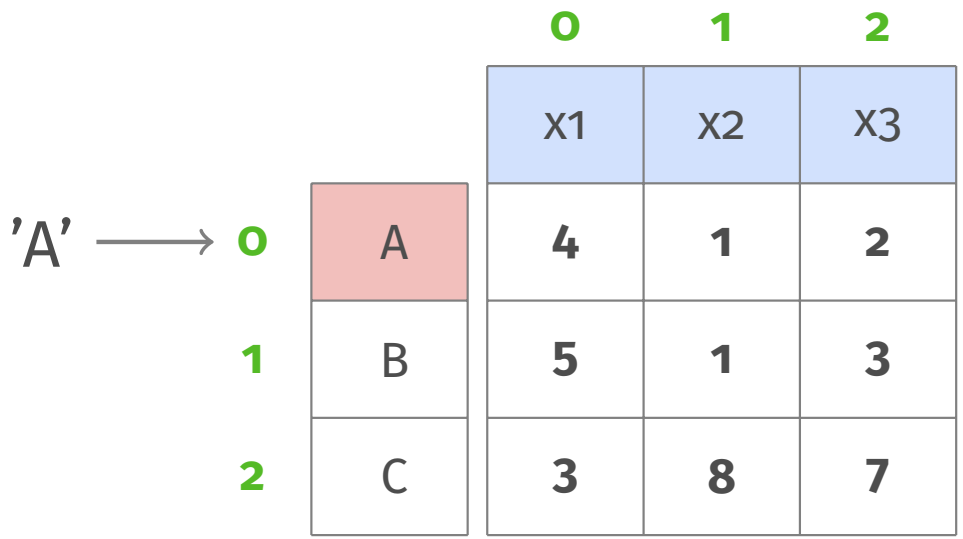
Was passiert bei `.loc[..]`?

```
df.loc['A':'B']
```

		0	1	2
		x1	x2	x3
0	A	4	1	2
1	B	5	1	3
2	C	3	8	7

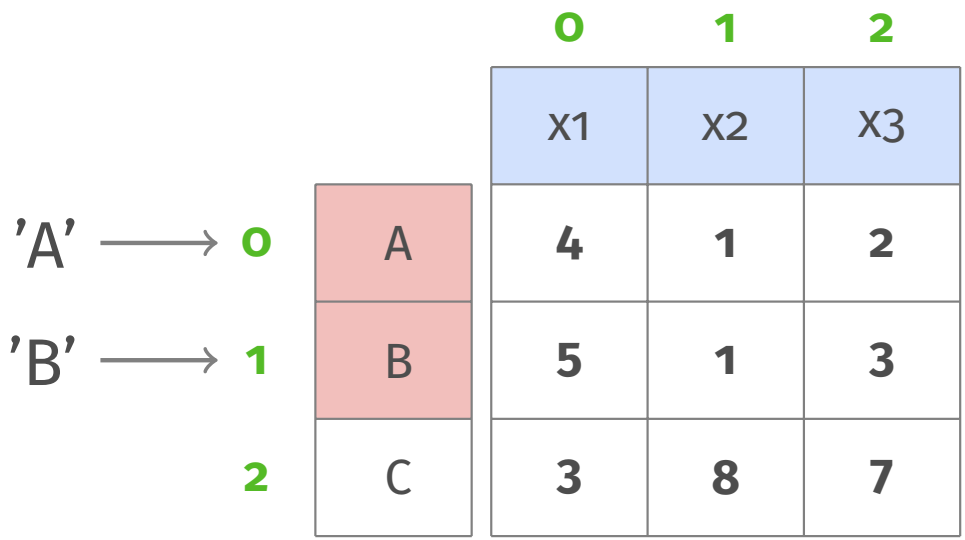
Was passiert bei `.loc[..]`?

```
df.loc['A':'B']
```



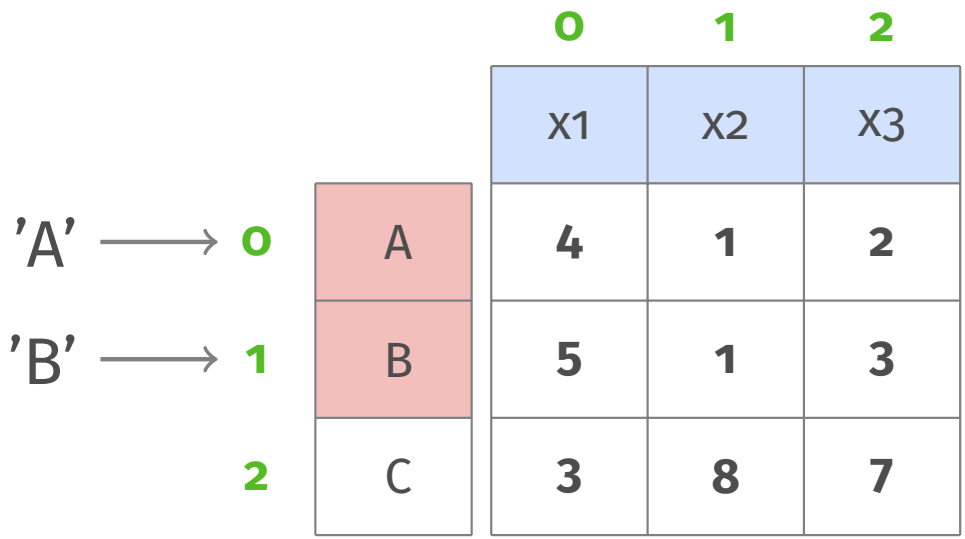
Was passiert bei `.loc[..]`?

```
df.loc['A':'B']
```



Was passiert bei `.loc[..]`?

```
df.loc['A':'B']
```



Es werden beide Zeilen (A und B) selektiert!

Wirtschaftsinformatik 1, Aufgabenblatt 4

Aufgabenblatt 4: Klausurbeispiele für benutzerdefinierte Funktionen

Aufgabe 4_1
Sie arbeiten beim Steueramt der Stadt Bochum und sollen ein VBA-Programm schreiben, mit dem die Hundesteuer berechnet werden kann.

Bei der Berechnung der Steuern wird zwischen Normalhunden und Kampfhunden unterschieden.

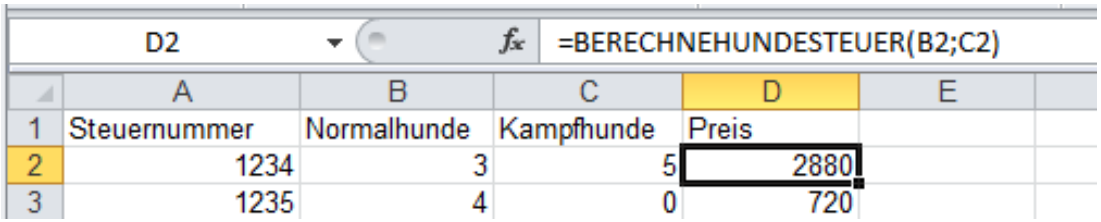
Für Normalhunde wird ein Staffelsteuersatz verwendet:

- Bei einem Hund im Haushalt kostet der Hund 120 € pro Jahr
- Bei zwei oder drei Hunden im Haushalt kostet jeder Hund 150 € pro Jahr
- Bei vier und mehr Hunden im Haushalt kostet jeder Hund 180 € pro Jahr

Kampfhunde werden grundsätzlich ebenfalls wie Normalhunde behandelt (das heißt die Anzahl von Kampfhunden und Normalhunden wird addiert und danach nach obiger Staffelung der Steuersatz berechnet), es gilt jedoch folgende Sonderregelung:

- Übersteigt die Anzahl der Kampfhunde die Anzahle der Normalhunde, so wird der zu entrichtende Steuersatz verdoppelt.

Die Steuersätze werden auf Konstanten abgelegt. Unten dargestellte Benutzer-Schnittstelle soll realisiert werden:



Wie lösen Informatiker Probleme?

Wie lösen Informatiker Probleme?

1. In kleine Probleme zerteilen
2. Kleine Probleme lösen
3. Lösungen zusammensetzen

Wie lösen Informatiker Probleme?

1. In kleine Probleme zerteilen
2. Kleine Probleme lösen
3. Lösungen zusammensetzen

Funktionen eignen sich gut, um Teilprobleme zu lösen!

Hundesteuern, Wirtschaftsinformatik 1, Aufgabenblatt 4

- Wie werden die Steuern grundsätzlich berechnet?
- Welche Eingabewerte braucht man?
- Welche Ausnahmen gibt es? Was ändert sich dann?
- Weitere Sonderregeln?

Hundesteuern, Wirtschaftsinformatik 1, Aufgabenblatt 4

- Wie werden die Steuern grundsätzlich berechnet?
- Welche Eingabewerte braucht man?
- Welche Ausnahmen gibt es? Was ändert sich dann?
- Weitere Sonderregeln?

Aufgabe:

- Schreiben Sie die Funktion `hundesteuer(..)`

Demo:

Entwicklung einer Lösung für die Hundesteuer-Aufgabe

- Wieso sind Teilprobleme gut?
- Wie finde ich ggf. Programmierfehler?

Von Excel (VBA) zu Pandas

Daten zur Aufgabe

Excel-Datei unter

`Kurse/DataScience1/data/hundesteuer.xls`

In Python mit Pandas:

```
import pandas as pd

df = pd.read_excel("Kurse/../hundesteuer.xls")
```

Excel Daten

Steuernummer	Normalhunde	Kampfhunde	Preis
1234	3	5	2880
1235	4	0	720
1236	1	0	120

Excel Daten

	0	1	2	3
	Steuernummer	Normalhunde	Kampfhunde	Preis
0	1234	3	5	2880
1	1235	4	0	720
2	1236	1	0	120

Excel Daten

	0	1	2	3
	Steuernummer	Normalhunde	Kampfhunde	Preis
0	1234	3	5	2880
1	1235	4	0	720
2	1236	1	0	120

```
df.iloc[ .. ]
```

Excel Daten

	0	1	2	3
	Steuernummer	Normalhunde	Kampfhunde	Preis
0	1234	3	5	2880
1	1235	4	0	720
2	1236	1	0	120

```
df.iloc[ 0, 2 ]
```

Zeilenweiser Zugriff

```
df = pd.read_excel("../hundesteuer.xls")

for i in range(len(df)):
  normalhunde = df.iloc[i, 1]
  kampfhunde = df.iloc[i, 2]

  print("zeile:   " + i)
  print("  normal: " + normalhunde)
  print("  kampf:  " + kampfhunde)
```

Zeile in DataFrame = Series

```
zeile = df.iloc[0,:]

#zeile ist dann ein Series Objekt:
#
# Steuernummer    1234
# Normalhunde        3
# Kampfhunde         5
# Preis           2880
# Steuern           42
#Name: A, dtype: int64
```

Zeile in DataFrame

```
zeile = df.iloc[0,:]

normal = zeile['Normalhunde']
kampf = zeile['Kampfhunde']
```

Zeile in DataFrame

```
zeile = df.iloc[0,:]

normal = zeile['Normalhunde']
kampf = zeile['Kampfhunde']
```

Hundesteuern aus Zeile (Series) berechnen:

```
def berechneHundesteuer(zeile):
  normal = zeile['Normalhunde']
  kampf = zeile['Kampfhunde']
  return hundesteuer(normal, kampf)
```

Funktion für jede Zeile aufrufen: `df.apply`

```
# rufe berechneHundesteuer fuer jede Zeile auf
df.apply(berechneHundesteuer, axis=1)
```

Funktion für jede Zeile aufrufen: `df.apply`

```
# rufe berechneHundesteuer fuer jede Zeile auf
df.apply(berechneHundesteuer, axis=1)
```

- Ergebnis ist Series Objekt
- Series enthält Ergebnis der Funktion für jede Zeile