Data Science 1

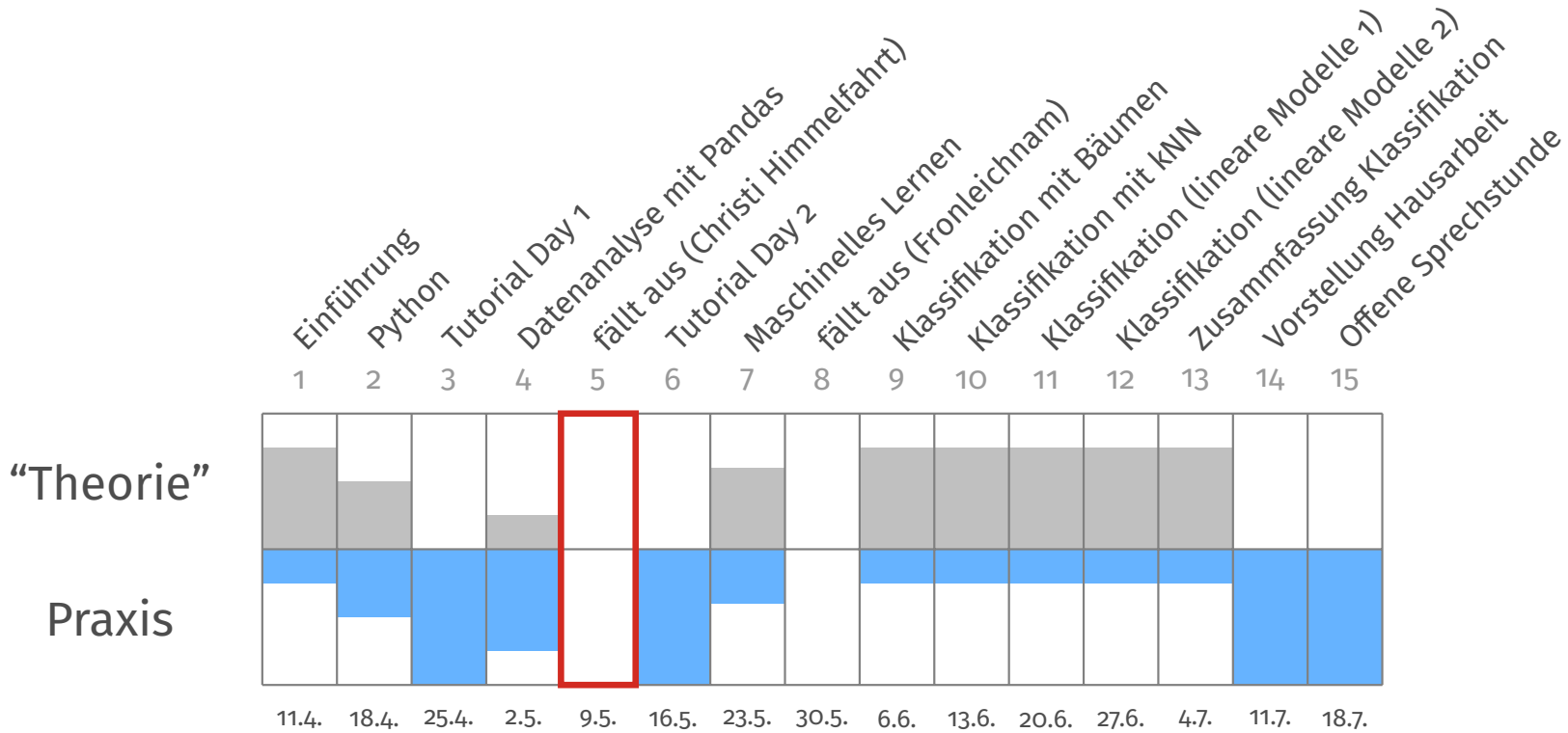
Tutorial Day 2

Prof. Dr. Christian Bockermann

Hochschule Bochum

Sommersemester 2024

Wo sind wir heute?



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

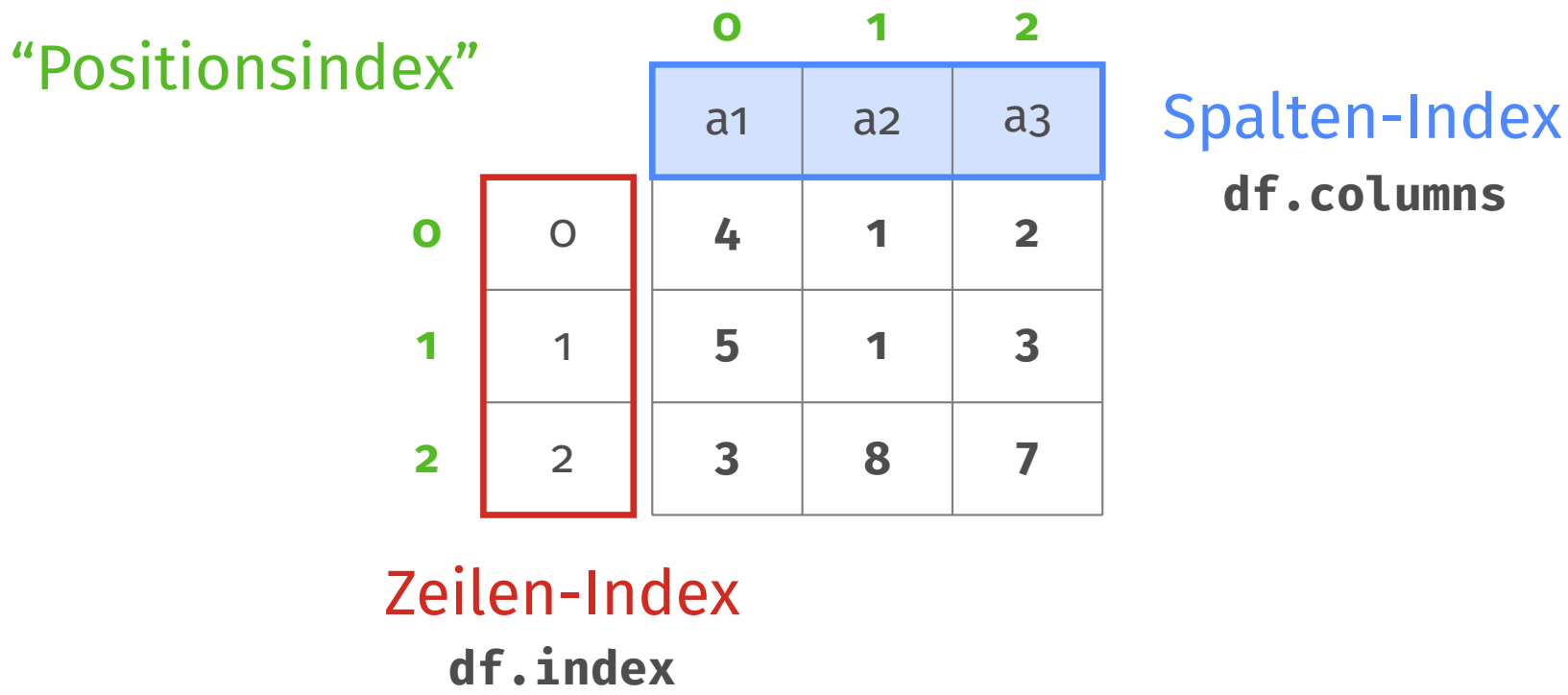
`df.columns`

Zeilen-Index

`df.index`

Pandas: DataFrame

Ein DataFrame `df` ist eine Tabellenstruktur:



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]
```

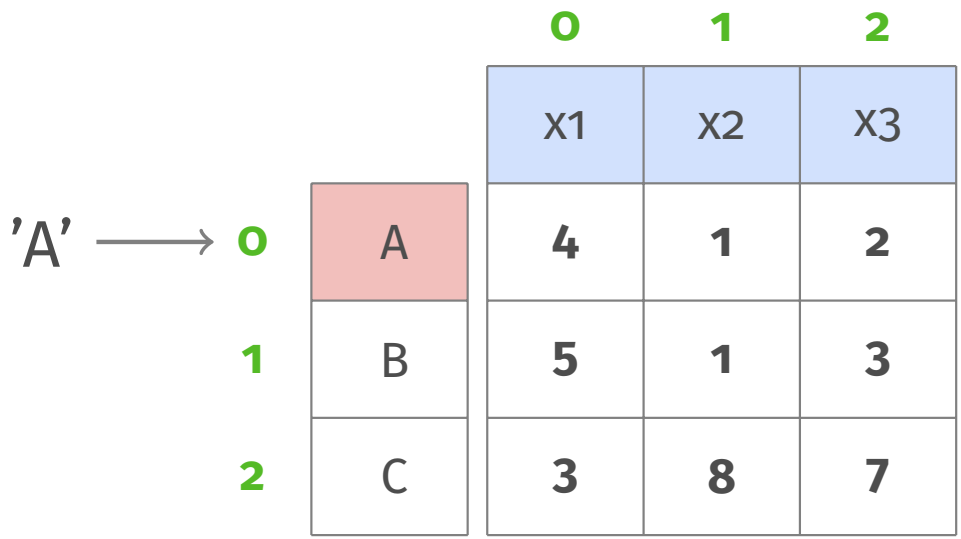
Was passiert bei `.loc[..]`?

```
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

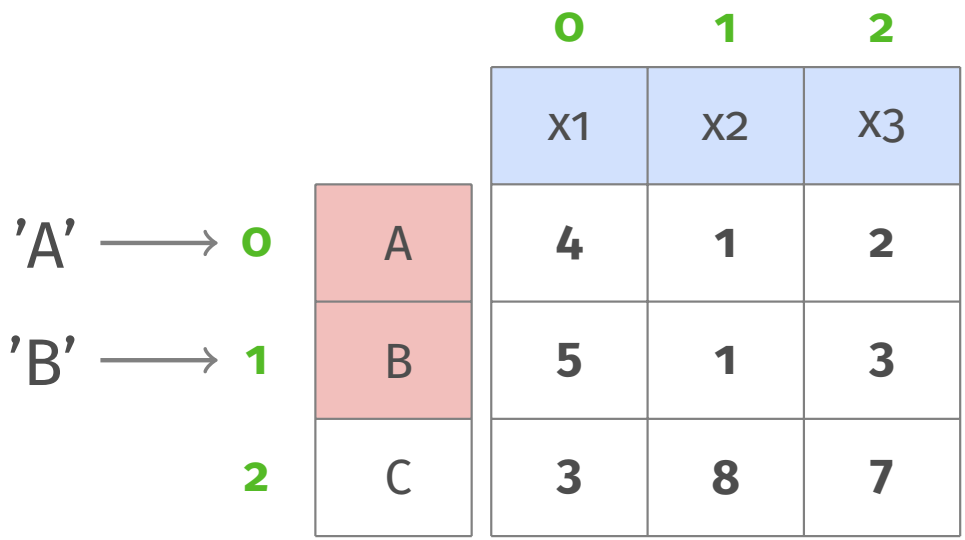
Was passiert bei `.loc[..]`?

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



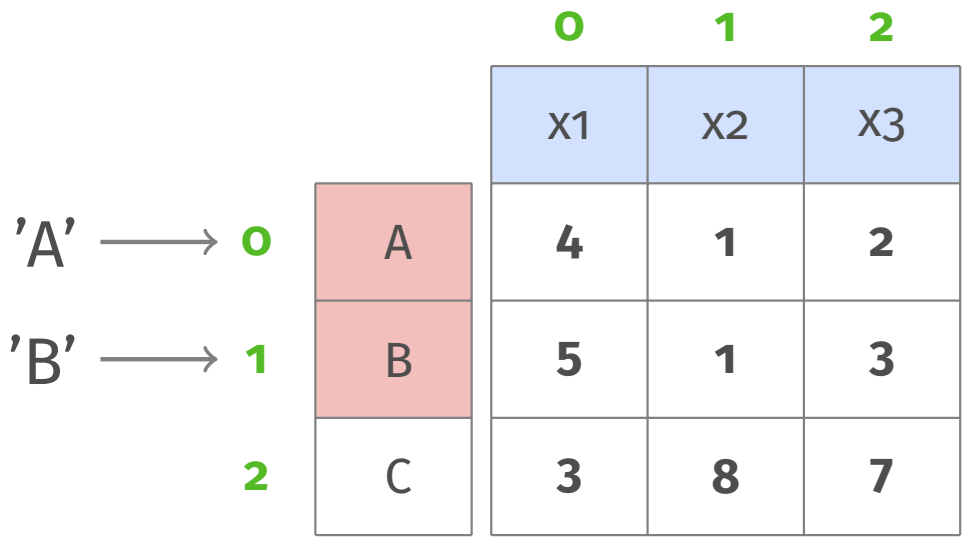
Was passiert bei `.loc[..]`?

```
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.

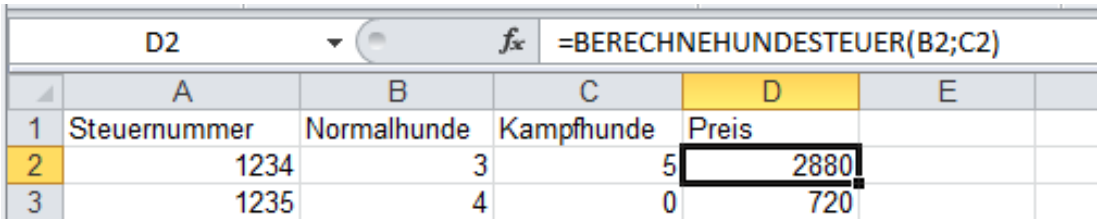
Für Normalhunde wird ein Staffelsteuersatz verwendet:

- 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

Aufgaben: Pandas Series

Pandas Series

Die folgenden S-Aufgaben, beziehen sich auf Pandas `Series`.

Es geht um

- das Erstellen von Series Objekten,
- den Zugriff auf Elemente von Series Objekten,
- die Berechnung einfacher Werten.

Aufgabe S1

Betrachten Sie die Folgen

$$4, 8, 7 \quad \text{und} \quad 1, -3, -2, 5$$

1. Definieren Sie eine Variable `series1` mit der ersten Folge 4, 8, 7.
2. Definieren Sie eine Variable `series2` mit der zweiten Folge.

Aufgabe S2 - Zugriff auf Elemente

Sie haben in Aufgabe S1 die Series `series1` definiert

1. Definieren Sie eine Variable a, die das zweite Element der Series enthält.
2. Definieren Sie eine Variable b, die das letzte Element der Series enthält
   (Erinnern Sie sich an *Slicing* und negative Indizes?).

Aufgabe S3 - Operationen mit Series

Sie haben in Aufgabe S1 die Series `series1` und `series2` definiert.

1. Addieren Sie die beiden Series Elemente und weisen Sie das Ergebnis der Variablen `series3` zu.

Benutzen Sie danach den Aufruf

```
print(series3)
```

um sich das Ergenis anzuschauen.

Aufgabe S4 - Filtern

Über den Zugriffsoperator `s[...]` lassen sich Series Werte filtern:

1. Schreiben Sie eine Funktion `groessere0`, die eine Series übergeben bekommt und eine Series mit den Werten zurückliefert, die größer oder gleich 0 sind.

Aufgabe S5 - Funktionen auf `Series` Objekten

1. Schreiben Sie eine Funktion `smax`, die den maximalen Wert einer Series zurückgibt.
2. Schreiben Sie eine Funktion `smin`, die den minimalen Wert zurückgibt.

Aufgabe S6 - Funktionen auf `Series` Objekten

Die Min-Max Normalisierung ist dadurch definiert, dass jeder Wert v durch

$$v' = \frac{v - min}{max - min}$$

ersetzt wird.

1. Schreiben Sie eine Funktion `snorm(series)`, die aus einer Series eine normalisierte Series nach der Min-Max Normalisierung berechnet.

Aufgaben: Pandas DataFrame

Die folgenden Aufgaben beziehen sich auf `DataFrame`

1. Erstellen von DataFrames
2. Selektieren von Zeilen/Spalten
3. Eigene Funktionen auf DataFrame Objekten

Aufgabe D1

Erstellen Sie den folgenden `DataFrame` und weisen Sie ihn der Variablen **`df1`** zu:

A	B
1	**2**
3	**4**

Aufgabe D2

Erstellen Sie ein `Series` Objekt mit den Werten 5 und 6 und fügen Sie es als neue Spalte “C” in den DataFrame **`df1`** ein. Der DataFrame sollte dann folgendermaßen aussehen:

A	B	C
1	**2**	**5**
3	**4**	**6**

Aufgabe D3

Erstellen Sie im `DataFrame` **`df1`** die Spalte “Z”, die die Summe der Spalten “A”, “B” und “C” enthält:

A	B	C	Z
1	**2**	**5**	**8**
3	**4**	**6**	**13**

Aufgabe D4

Teilen Sie im DataFrame **`df1`** alle Spalten ausser der Spalte “Z” durch die Werte der “Z”-Spalte.

Das Ergebnis sollte folgendermaßen aussehen:

A	B	C	Z
0.125	**0.250**	**0.625**	**8**
0.231	**0.308**	**0.462**	**13**

Selektieren von Zellen/Bereichen

Die Funktion `tutorial.toy_data()` liefert das folgende `DataFrame` Objekt zurück:

a1	a2	a3	a4
4	1	2	9
5	1	3	6
3	8	7	4

1. Definieren Sie die Variable `df_rot`, die den rot markierten Teil des DataFrames enthält.
2. Definieren Sie die Variable `df_blau`, für den blau markierten Teil.

Selektieren von Zellen/Bereichen

1. Definieren Sie einen quadratischen DataFrame `qf`, indem Sie aus dem `toy_data()` DataFrame die ersten $n = 3$ Zeilen/Spalten selektieren.
2. Schreiben Sie eine Funktion `quadrat(df)`, die aus einem DataFrame den größten quadratischen Block extrahiert (von oben links beginnend).
3. Schreiben Sie eine Funktion `diag(df)`, die für einen DataFrame die Liste der Elemente auf der Hauptdiagonalen zurückliefert.

Aufgabe D7*

Mit .apply(f, axis=1) wird eine Funktion auf alle Zeilen angewandt:

```
# f gibt fuer jede Zeile 1 zurueck:
def f(row):
    # row ist ein Series Objekt!
  return 1

# f auf alle Zeilen anwenden
series = df.apply(f, axis=1)
```

Aufgabe D7*

Betrachten Sie die folgende Tabelle:

Länge (cm)	**Breite** (cm)	**Höhe** (cm)	**Entfernung** (km)	**Dienstleister**
100	100	40	20	WPS
110	100	100	60	UPS
100	210	100	70	WPS
50	100	70	89	DHL

(Aufgabenblatt 4 aus Wirtschaftsinformatik 1)

Der Datensatz (ohne die Dienstleister-Spalte) ist als CSV-Datei im `data` Verzeichnis enthalten.

Aufgabe D7*

1. Laden Sie den Versand-Datensatz in einen DataFrame.
2. Schreiben Sie eine Funktion, die den Dienstleister pro Zeile berechnet.
3. Berechnen Sie die Spalte Dienstleister mit ihrer Funktion.

Regeln für die Dienstleister-Funktion:

- Paketvolumen < $1m^3$, dann Dienstleister DHL
- $1m^3 \leq$ Paketvolumen $< 2m^3$, dann Dienstleister UPS
- Paketvolumen $\geq 2m^3$, dann Dienstleister WPS
- Wenn Entfernung $< 50km$, dann auf jeden Fall WPS