DataCraft

# 5
# Datum & Zeitstempel

# klassisches Python: Date

DataCraft

Umgang mit Datum und Zeitstempel im klassisches Python (nicht pandas) mit dem Package datetime

- Das Modul datetime bietet mehrere Klassen: date, time, datetime, timedelta

- **date** ist für Datumsangaben zuständig

```
from datetime import date
d = date.today()
print(d)
print(f'Tag: {d.day}, Monat: {d.month}, Jahr: {d.year}')
```

- Formatierung als String mittels **strftime**

```
d.strftime("%d.%m.%Y")
```

- String zu Datum mittels **datetime.strptime**

```
from datetime import datetime
datetime.strptime("02.05.2022", "%d.%m.%Y").date()
```

# Klassisches Python: Datetime

DataCraft

Codes zur Datumsformatierung

| Code | Bedeutung | Beispiel |
|---|---|---|
| %a | Wochentagskürzel | Mon |
| %A | Wochentag | Monday |
| %b | Monatskürzel | Dec |
| %B | Monatsname | December |
| %m | zweistelliger Monat | 12 |
| %d | zweistelliger Tag | 03 |
| %y | zweistelliges Jahr | 22 |
| %Y | vierstelliges Jahr | 2022 |

Wochentag und Monat hängen von der lokalen Einstellung ab:

```
import locale
locale.setlocale(
  locale.LC_TIME,
  "de_DE"
)
```

**Achtung:** Es wird davon abgeraten, locale zu ändern. Besser Package babel verwenden

Pandas bietet einen Datentyp datetime64 an

- Datumsspalten aus csv-Dateien werden normalerweise als Object/String eingelesen

- Konvertiertung mit `pd.to_datetime(Serie, format)`, wobei format das Format angibt, z.B. deutsch DD.MM.YYYY enspricht `format="%d.%m.%Y"`

```
taxis["pickup"] = pd.to_datetime(taxis["pickup"])
```

- Zugriff auf einzelne Komponenten mittels .dt.xxx (wie bei strings mit .str), z.B. day, month, year, dayofweek, week, dayofyear, quarter, hour, minute, second

```
taxis["pickup"].dt.day
taxis["pickup"].dt.month
taxis["pickup"].dt.dayofweek   # Montag=0, Sonntag=6
```

- Einige wenige Komponenten werden über Funktionen aufgerufen

```
taxis["pickup"].dt.day_name()
taxis["pickup"].dt.month_name()
```

- siehe https://pandas.pydata.org/docs/reference/api/pandas.Timestamp.html

# Filterung nach Datum

DataCraft

- Filterung über Bedingung (Vergleich mit Datumsstring):

```
taxis[taxis["pickup"]>="2019-03-30"]
# Mit zwei Bedingungen bekommen wir einen Zeitabschnitt
taxis[(taxis["pickup"]>="2019-03-30") & \
        (taxis["pickup"]<"2019-03-31")]
taxis[taxis["pickup"]>="2019-03-30 10:28"]
```

- Oder per between() oder isin()

```
taxis[taxis["pickup"].between("2019-03-30","2019-03-31")]
taxis[taxis["pickup"].dt.day.between(20,21)]
taxis[taxis["pickup"].dt.day.isin([20,21])]
```

# Umwandlung von datetime in date

DataCraft

- Umwandlung von datetime in date mit dt.date

```
taxis["pickup_date"] = taxis["pickup"].dt.date
```

- Alternativ setzen der Uhrzeit auf 00:00 Uhr mittels *normalize()*. Dann bleibt der Datentyp datetime

```
taxis["pickup"].dt.normalize()
```

- Filterung auf ein Datum

```
taxis[(taxis["pickup"]>="2019-03-30") & \
        (taxis["pickup"]<"2019-03-31")]
# oder
taxis[taxis["pickup"].dt.normalize() == "2019-03-01"]
# oder
taxis[taxis["pickup_date"] == datetime.date(2019, 3, 23)]
```

# Zeitdifferenzen

DataCraft

- Werden zwei Zeitstempel voneinander abgezogen, ergibt sich ein Timedelta-Objekt, z.B. `31 days 00:14:42`, welches aus mehreren Komponenten besteht

```
delta = taxis["pickup"].max() - taxis["pickup"].min()
```

- Zugriff auf die einzelnen Komponenten

```
delta.days                  # ganze Tage
delta.seconds               # Sekunden ohne Tage
delta.total_seconds()       # Alle Sekunden (inkl. Tage)
delta.components            # Alle Komponenten
delta.components.minutes # Zugriff auf die Komponenten
```

- siehe https://pandas.pydata.org/docs/reference/api/pandas.Timedelta.html