Merkblatt Pandas

1 Pandas: Zugriff auf Dataframes

Das Modul Pandas ermöglicht es, CSV-Dateien als Dataframes in der Umgebung von Python zu verwenden, also insbesondere Werte auslesen und auswerten zu können. Dieses Dataframes werden als Quasi-Standard unter dem Variablennamen `df` gepeichert.

1.1 Pandas importieren

Um CSV.Dateien zu importieren, muss zunächst das Pandas-Modul mit folgender Codezeile importieren und in `pd`umbenannt werden. Die Bezeichnung `pd` ist wie `df` eine Konvention unter Programmierern; es könnte auch ein anderer Name für diese Variable gewählt werden.

```
import pandas as pd
```

Jetzt kann zum Beispiel CSV-Datei in ein Dataframe umgewandelt werden, indem die CSV-Datei (hier `CSV-Datei.csv`) importiert und als Dataframe (hier `df`) abgespeichert wird.

```
df = pd.read_csv("CSV-Datei.csv")
```

Vergleiche auch Merkblatt zu CSV-Dateien.

Pands enthält auch Methoden zum Importieren weiterer Dateiformate. Die Dokumentation und Google bieten Informationen hierzu.

1.2 Zugriff auf Spalten

Um auf Spalten zuzugreifen, setzen wird nach dem `df` in eckigen Klammern der Spaltentitel eingefügt. So wandelt man die Spalte in eine sogenannte Pandas-series um

```
df["Spaltentitel"]
```

Um Spalten aus dem Dataframe zu entfernen, benutzen wird der `.drop`-Befehl benutzt - gefolgt von der Spalte, die entfernen werden soll (hier `Unnütz`). Mit dem Parameter `axis` auf 1, um eine Zeile zu entfernen, und dem Parameter `inplace` auf `True`, damit das Entfernen direkt im Dataframe erfolgt, sieht das so aus:

```
df.drop(["Unnütz"], axis = 1, inplace = True)
```

Um eine neue Spalte anzulegen geben, wird der neue Spaltentitel(hier `Neu`) mit einem Wert gefüllt.

```
df["Neu"] = "Spaltenwert"
```

1.3 Zugriff auf Zeilen

Mit Hilfe des `.loc` Befehls gefolgt von der Zeilennummer in einer eckigen Klammer kann auf Zeilen zugegriffen werden:

```
df.loc[0]
```

Um auf eine Spalte innerhalb einer Zeile zugreifen zu können, wird hinter die Zeilennummer noch der Spaltenname geschrieben.

```
df.loc[0]["Spaltenname"]
```

Zeilen können auch in ein Dictionary umgewandelt werden, indem der Ausdruck in eine runde Klammer gesetzt und davor der Befehl `dict` angegeben wird.

```
dict(df.loc[0])
```

1.4 Daten ändern mit loc()

Um eine ganze Spalte zu verändern, wird erst der Spaltenname und dann der neuee Wert angegeben. Dieser Wert wird dann in dem bestehenden Dataframe abgespeichert.

```
df["Stückzahl"] = 100
```

Einen Wert innerhalb einer Spalte kann man ändern mit dem `.loc` Befehl. Als erstes wird gewählt, aus welcher Spalte geändert werden soll (hier Bundesland), danach folgt der Wert, der ersetzt werden soll (hier Nordrhein Westfalen), Zum Schluss wird der neue Wert (hier NRW) angegeben.

```
df.loc[df["Bundesland"] == "Nordrhein Westfalen", "Bundesland"] = "NRW"
```

1.5 Zeilen und Spalten auswerten

Folgende Befehle sind hilfreich, um Spalten auszuwerten.

1.5.1 min()

Mit dem `min()` Befehl wird der niedrigste Wert einer Spalte ausgegeben.

```
df["Temperatur"].min()
```

1.5.2 max()

mit dem `max()` Befehl wird der größte Wert einer Spalte ausgegeben..

```
df["Temperatur"].max()
```

1.5.3 mean()

mit dem `mean()` Befehl wird der Mittelwert einer Spalte ausgegeben

```
df["Temperatur"].mean()
```

1.5.4 sum()

mit dem `sum()` Befehl werden die Summe einer Spalte ausgegeben.

```
df["Temperatur"].sum()
```

[ ]: