DataCraft

12
SQL und Python

Mit Python lässt sich problemlos auf Datenbanken zugreifen.

- SQLite mit dem Package **sqlite3** *(für den Aufbau der Verbindung. Funktion sqlite3.connect(dateipfad))*

- **pandas** bietet zum Lesen die Funktionen *pd.read_sql(query, con)* bzw. und zum Schreiben die Funktion *df.to_sql(tabellenname, con)*.

Zugriff auf SQLite in Python

DataCraft

```
import sqlite3

# Öffnen oder Erstellen
con = sqlite3.connect("test_db.sqlite")

# Schreiben von Daten
cur = con.cursor()
cur.execute("CREATE TABLE Population(id INTEGER PRIMARY KEY,
        country TEXT, population INT)")
cur.execute("INSERT INTO Population VALUES(1,'Germany',81197537)")
cur.execute("INSERT INTO Population VALUES(2,'France', 66415161)")
cur.execute("INSERT INTO Population VALUES(3,'Spain', 46439864)")
cur.execute("INSERT INTO Population VALUES(4,'Italy', 60795612)")
cur.execute("INSERT INTO Population VALUES(5,'Spain', 46439864)")
cur.execute("commit")
cur.close()

# Ausführen einer SELECT-Abfrage
query = "SELECT country FROM Population WHERE population > 50000000;"
df = pd.read_sql(query, con)
```

Für jede Datenbank muss zuerst eine Verbindung aufgebaut werden. Dafür gibt es verschiedene Möglichkeiten (direkt, ODBC, JDBC, …). **SQLAlchemy** wird von pandas bevorzugt, das wiederum **psycopg2** benutzt.

Der Verbindungsstring gibt Datenbankart, Adresse, Datenbankname, Benutzername und Passwort an

```
'postgresql://user:password@adresse:port/dbname')
```

z.B.

```
'postgresql://python_rw:geheim@localhost:5432/postgres'
```

Achtung: Hier stehen die Verbindungsdaten im Klartext. Aufpassen z.B. mit github. Es gibt Möglichkeiten, dieses aus einer config-Datei, Keylocker, … auszulesen

DataCraft

In SQLAlchemy wird eine Verbindung als Engine bezeichnet. Nach dem Aufbau der Verbindung kann pandas zum Ausführen von SQL-Statements oder zum Einlesen gesamter Tabellen benutzt werden

```
import sqlalchemy

# Aufbau des  Verbindungsstrings
con = sqlalchemy.create_engine(
        'postgresql://python_rw:python_rw@localhost:5432/postgres')

query = "select * from formel1.circuits"
circuits = pd.read_sql(query, con)

# alternativ über read_sql_table
circuits = pd.read_sql_table("circuits", con=con, schema="formel1")
```

Sensible Daten wie Passwörter sollten nicht direkt im Python-Skript stehen. Die Gefahr besteht, das diese öffentlich werden (z.B. über Github).

Möglichkeit 1: config.ini-Datei

```
import configparser

config = configparser.ConfigParser()
config.read("config.ini")
config["local_postgres"]["DB_USER"]
```

config.ini

```
[local_postgres]
DB_USER = python_rw
DB_PW = python_rw
DB_ADDRESS = localhost
DB_PORT = 5432
DB_NAME = postgres
```

Ausschluß von Dateien/Verzeichnissen in Git durch Erstellen einer Datei *.gitignore* und Auflisten der Dateien/Verzeichnisse/Pattern

DataCraft

Möglichkeit 2: Umgebungsvariablen

Kommandozeile als Administrator ausführen

```
setx DB_USER python_rw /M
```

(oder Einstellungen -> erweiterte Systemeinstellungen -> Umgebungsvariablen -> Systemvariablen)

VSCode muss neu gestartet werden

```
import os
DB_USER = os.getenv("DB_USER')
```