CloudCommand

Cyber Security

CloudCommand GmbH chr.schumacher@gmx.tm

CloudCommand

Grundlagen der Programmierung

CloudCommand GmbH chr.schumacher@gmx.tm

Datenbank-zugriffe

Python Befehle zum Datenbankhandling

- Die notwendigen Befehle werden anhand eines Beispielprogramms besprochen. (Datei im Anhang)

Python Befehle zum Datenbankhandling

- Der Zugriff auf eine SQL-Datenbank in python kann z.B. durch import des Moduls **mysql.connector** geschehen:

  import mysql.connector

Python Befehle zum Datenbankhandling

- Im ersten Schritt werden die Verbindungsparameter und die Credentials für die Datenbank übergeben.

```
conn = mysql.connector.connect(
        host='localhost‘,
        user='root', Benutzername
        password=,‘
)
```

Python Befehle zum Datenbankhandling

- Im zweiten Schritt wird der Datenbankcursor initialisiert.

cursor = conn.cursor()

```
6     conn = mysql.connector.connect(
7         host='localhost',
8         user='root',
9         password=''
10    )
11    cursor = conn.cursor()
```

Python Befehle zum Datenbankhandling

- Nun können SQL-Befehle benutzt werden. Das Python Schlüsselwort dazu lautet

  cursor.execute(<SQL-Statement>)

Python Befehle zum Datenbankhandling

- **Beispiel:** Erstellen der Datenbank **kundendatenbank**.

```
# Datenbank 'beispieldatenbank' erstellen, falls sie nicht existiert
try:
    cursor.execute("CREATE DATABASE IF NOT EXISTS kundendatenbank")
    print("Datenbank 'kundendatenbank' erfolgreich erstellt oder bereits vorhanden.")
except mysql.connector.Error as err:
    print(f"Fehler beim Erstellen der Datenbank: {err}")
    exit(1)
```

Python Befehle zum Datenbankhandling

- Die Datenbank wurde erstellt.
- Nächster Schritt: Verbindung zur Datenbank aufnehmen

conn.database = ‚kundendatenbank‘

```
21      # Verbindung zur erstellten Datenbank herstellen
22      conn.database = 'kundendatenbank'
```

Python Befehle zum Datenbankhandling

- Da die Datenbank **kundendatenbank** leer ist, wird im Folgenden eine neue, leere Tabelle darin erstellt

```
24      # Tabelle 'kunden' erstellen
25      cursor.execute('''
26      CREATE TABLE IF NOT EXISTS kunden (
27          id INT AUTO_INCREMENT PRIMARY KEY,
28          vorname VARCHAR(100) NOT NULL,
29          nachname VARCHAR(100) NOT NULL,
30          stadt VARCHAR(100) NOT NULL,
31          email VARCHAR(100) NOT NULL
32      )
33      ''')
```

Python Befehle zum Datenbankhandling

- Die neu erstellte Tabelle ist leer.
- Es gibt mehrere Wege diese Tabelle mit Inhalt zu füllen
  - Über das Web-UI ‚phpMyAdmin‘
  - 3rd Party Tools, z.B. ‚HeidiSQL‘ und Andere
  - Durch unser Python-Programm.

- Zunächst werden vorgefertigte Datensätze in die DB geschrieben.

Python Befehle zum Datenbankhandling

- Die Beispieldaten, werden einer Liste entnommen

```
kunden_daten = [
    ('Anna', 'Schmidt', 'Berlin', 'anna.schmidt@example.com'),
    ('Max', 'Müller', 'Hamburg', 'max.mueller@example.com'),
    ('Julia', 'Meier', 'München', 'julia.meier@example.com'),
    ('Thomas', 'Schneider', 'Köln', 'thomas.schneider@example.com'),
    ('Laura', 'Fischer', 'Frankfurt', 'laura.fischer@example.com'),
    ('Peter', 'Weber', 'Stuttgart', 'peter.weber@example.com'),
    ('Lisa', 'Becker', 'Düsseldorf', 'lisa.becker@example.com'),
    ('Michael', 'Hoffmann', 'Dortmund', 'michael.hoffmann@example.com'),
    ('Katrin', 'Schulz', 'Essen', 'katrin.schulz@example.com'),
    ('Stefan', 'Koch', 'Bremen', 'stefan.koch@example.com')
]
```

Python Befehle zum Datenbankhandling

- … und wie folgt in die Datenbank geschrieben:
  - SQL-Befehl: **Insert into…**
  - Kommando mit **conn.commit()** absenden.

```
cursor.executemany( operation: '''
INSERT INTO kunden (vorname, nachname, stadt, email)
VALUES (%s, %s, %s, %s)
''', kunden_daten)
conn.commit()
print("Beispieldatensätze erfolgreich in die Tabelle 'kunden' eingefügt.")
```

Python Befehle zum Datenbankhandling

Fehlerabfrage:

- Die betrachteten Befehle sind aller Bestandteil des ersten **try**-Blocks.
- Sollte etwas nicht wie gewünscht ausgeführt werden, wird der dazugehörige **except**-Block ausgeführt:

```
except mysql.connector.Error as err:
    if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
        print("Zugriff verweigert: Überprüfen Sie den Benutzernamen oder das Passwort.")
    elif err.errno == errorcode.ER_BAD_DB_ERROR:
        print("Datenbank existiert nicht.")
    else:
        print(err)
```

Python Befehle zum Datenbankhandling

- try/except, letzter Teil: **finally**
  - Zu Anfang wurde versucht eine Verbindung zur DB herzustellen. (erfolgreich oder ggf. nicht)
  - Der abschließende **except**-Block wird diese Verbindung wieder sauber schließen.

Python Befehle zum Datenbankhandling

- Daten durch eine Benutzereingabe in die Datenbank einfügen
- Daten werden mit **INSERT INTO** in Tabellen eingefügt.
- Der Parameter für die **Query** im **Execute**-Kommando lautet demnach:

```
query = '''INSERT INTO Kunden (Vorname, Nachname, stadt, Email)
VALUES (%s, %s, %s, %s)'''
```

Python Befehle zum Datenbankhandling

- Beispielcode für das SQL-Kommando **insert**:

```
vorname = input("Vorname: ")
nachname = input("Nachname: ")
ort = input("Ort: ")
email = input("Email: ")

add_customer_query = '''
INSERT INTO Kunden (Vorname, Nachname, stadt, Email)
VALUES (%s, %s, %s, %s)
'''
cursor.execute(add_customer_query, params: (vorname, nachname, ort, email))
conn.commit()
print("Neuer Kunde erfolgreich hinzugefügt.")
```

CloudCommand

CloudCommand GmbH chr.schumacher@gmx.tm