SQLite, SQLAlchemy

SQLite, sqlite3

SQlite:

- Biblioteka napisana w języku C, która implementuje silnik bazy danych SQL,
- Nie wymaga dodatkowej konfiguracji,
- 'Bezserwerowa' - nie jest potrzebny osobny proces do zarządzania bazą - bazy jest przechowywana w jednym pliku na dysku.

Moduł sqlite3:

- Impementacja interfejsu do bazy Sqlite w języku Python,
- Interfejs jest zgodny z DB-API 2.0 (specyfikacja PEP249).

sqlite3

Przykład 1 - połączenie z bazą:

```
import sqlite3

connection = sqlite3.connect('example.db')

connection.close()
```

sqlite3

Przykład 2 - utworzenie tabeli:

```
import sqlite3

sql_test_table = '''CREATE TABLE IF NOT EXISTS test (
            id integer,
            name text,
            value real
            )'''

connection = sqlite3.connect('example.db')

cursor = connection.cursor()
cursor.execute(sql_test_table)

connection.commit()

connection.close()
```

sqlite3

Przykład 2a – pobranie informacji o utworzonej tabeli:

```
import sqlite3

connection = sqlite3.connect('example.db')

cursor=connection.cursor()

res = cursor.execute("SELECT name FROM sqlite_schema")
print(res.fetchall())

connection.close()
```

output:
[('test',)]

sqlite3

Przykład 3 - dodawanie rekordów do tabeli:

```
import sqlite3

connection = sqlite3.connect('example.db')

cursor = connection.cursor()

cursor.execute('''CREATE TABLE IF NOT EXISTS
                    test (id integer, name text, value real)''')

cursor.execute('''INSERT INTO test VALUES
                  (1, 'TEST', 1.3)''')

for row in cursor.execute('SELECT * FROM test'):
   print(row)

connection.commit()
connection.close()
```

SQLAlchemy

SQLAlchemy:

- Udostępnia narzędzia/komponenty, które pozwalają na interakcję z bazami na dowolnym poziomie,
- Stanowi spójną i wyczerpującą nakładkę na DBAPI Pythona, zachowując przy tym możliwość korzystania ze specyficznych cech poszczególnych baz danych,
- Składa się z dwóch części: Core (Engine, Dialects, SQL Expression Language, Schema/Types) i ORM (Object Relational Mapper – enable construction of class mapped to relational database tables).

SQLAlchemy

Przykład 4:

```
from sqlalchemy import create_engine

#engine
engine = create_engine ('sqlite:///example.db', echo = True)

from sqlalchemy.ext.declarative import declarative_base

#metaklasa
Base = declarative_base()

from sqlalchemy import Column, Integer, String
from sqlalchemy.types import FLOAT

#table
class Test(Base):
    __tablename__ = 'test'
    id = Column(Integer, primary_key = True)
    name = Column(String(100))
    value = Column(FLOAT)
```

SQLAlchemy

Przykład 4 - cd:

```
#create
Base.metadata.create_all(engine)

#data record
new_test_record = Test (
id = 10,
name = 'ORM',
value = 6.8
)

#session
from sqlalchemy.orm import sessionmaker

Session = sessionmaker(bind = engine)
session = Session()
session.add(new_test_record)
session.commit()

 for test in session.query(Test).all():
    print(test.id, test.name, test.value)
```

Zadanie 1

Stwórz nową bazę i tabelę przechowującą dane o książkach. Wprowadź do niej przykładowe dane.

Zadanie 2

Zmodyfikuj zadanie 1 tak, aby wyświetlić rekordy tabeli książek, dodaj nowy rekord , a następnie anuluj tę operację (anulowanie transakcji dodawania rekordu). Przetestuj przeszukiwanie tabeli.

Zadanie 3

Stwórz relacyjny model bazy danych biblioteki (np. tabele książki, czytelnicy, wypożyczenia). Wprowadź do niej przykładowe dane.

Zadanie 4

Napisz program do obsługi biblioteki, który obsługuje bazę (zadanie 3) z wykorzystaniem SQLAlchemy.