SQL ALCHEMY

sqlalchemy היא ספרייה שעוטפת מסד-נתונים, ומאפשרת לגשת אליו באופן מונחה עצמים. המונח המקצועי הוא: (Object Relational Mapper) ORM- מיפוי בין עצמים (Object) לבין מסד-נתונים יחסי (Relational).

אחד היתרונות שלה, שהיא מאפשרת שימוש בכמה סוגים שונים של מסדי נתונים מבלי לשנות את קוד המקור יותר מידי, כך שאם למשל נרצה להשתמש ב-sqllite כדי לבחון את המערכת, ובמסד נתונים אחר בשביל הפרודוקציה, בסה"כ צריך רק לתת כתובת uri שונה ל-sqlalchemy שעליה היא תבצע את יצירת השרת, כל שאר הקוד יישאר אותו הדבר.

התקנה:

```
pip install sqlalchemy
```

עכשיו צריך להגדיר את המסד נתונים בו משתמשים. המסד מוגדר על-ידי URL. בינתיים נשתמש ב-sqlite בתור המסד, לכן בתחילת ה-uri נכתוב 'sqlite:///' השלושה קווים מסמנים path יחסי של המסד נתונים מהקובץ הנוכחי (relative path) , ואם המסד נתונים לא מוגדר בכלל הוא יוגדר באותה תיקייה של הסקריפט; אחר כך נוסיף את שם המסד נתונים, נניח קוראים my_database.db :

```
import sqlalchemy
engine = sqlalchemy.create_engine("sqlite:///my_database1.db") # sqlite and not sqlite3
```

מה שמעולה ב-sqlalchemy שאנחנו יכולים לייצג את טבלאות המסד נתונים שלנו כמחלקות מונחות עצמים שיורשות מאובייקט שנקרא Base . כל עמודה מוגדרת כמחלקה שנקראת Column וחייבת להכיל את טיפוס הנתונים שאמור לייצג אותה, למשל אם העמודה מייצגת גיל נצטרך להגדיר אותה כאינטג'ר(db.Integer), ושמות כמחרוזת (db.String() ואפשר להגדיר גם גודל מקסימלי, למשל 20 תווים: (db.String(20) וכו'. פרמטרים נוספים שניתן להגדיר הם אם העמודה היא primary key כלומר כזאת שניתן להשתמש בה כדי להסיק על שאר הערכים מטבלאות חיצוניות, למשל תעודת זהות היא primary-key כי היא מספר ייחודי לכל בן אדם ,ואם יש לי אותו אני אז יכול לקבל ספציפית את הערכים של האדם הזה, לעומת שם שיכול להיות שיש שני אנשים עם אותו שם, ואז אני אקבל את שניהם ולאו דווקא את הבן אדם הספציפי שרציתי לקבל; עוד משהו שניתן להגדיר זה האם העמודה ייחודית, כלומר כזאת שאין לה חזרות, האם היא nullable כלומר ניתן לתת לך ערך null, וכו'. כך נגדיר את הטבלה user שהגדרנו קודם, הפעם באופן מונחה-עצמים:

```
from sqlalchemy.orm import declarative_base
Base = declarative_base()

class User(Base):
    __tablename__ = 'user'
    id = sqlalchemy.Column(sqlalchemy.Integer , primary_key= True)
    name = sqlalchemy.Column(sqlalchemy.String(20), unique=True, default='Anonymous')
    phone = sqlalchemy.Column(sqlalchemy.Integer , unique = True)
    email = sqlalchemy.Column(sqlalchemy.String(20) , unique=True, nullable = False)
    password = sqlalchemy.Column(sqlalchemy.String(20), nullable= False)
    def __repr__(self):
        return f'User({self.id!r}, {self.name!r}, {self.phone!r}, {self.email!r})'
Base.metadata.create_all(engine)  # Create the database and all tables.
```

גם כשמכניסים נתונים לטבלה, אנחנו למעשה מכניסים לשם אובייקטים. ראשית אנחנו צריכים לפתוח "סשן" (session), ואז ליצור עצמים ולהוסיף אותם. ובסוף לא לשכוח commit כמו קודם:

```
DBSession = sqlalchemy.orm.sessionmaker(bind=engine)
session = DBSession()

user1 = User(name=name1, phone=phone1,  email=email1, password=password1)
session.add(user1)
print('First user inserted')

session.add(User(name=name2, phone=phone2,  email=email2, password=password2))
print('Second user inserted')

session.commit()
```

ועכשיו הנתונים אמורים להיות במסד-הנתונים.
כדי לבדוק אם הנתונים באמת נמצאים במסד נוכל להשתמש בשאילתה פשוטה, למשל כדי לראות את כל הנתונים של הטבלה User נשתמש ב:

```
session.query(User).all()
```

יש כמה שאילתות שניתן לבצע על המסד ולא ניכנס לפרטים, כרגע מה שבעיקר חשוב לנו הוא לדעת איך לבקש משתמש ספציפי לפי איזשהו ערך של הטבלה, למשל לבקש את המשתמש שקוראים לו Tom Pythonivitch.
בשביל זה נשתמש ב-filter.query() ובשאילתא נכניס לפי איזה פרמטר נרצה להשתמש. נוכל להשתמש ב-all() בשביל לקבל את כל המשתמשים שעונים על השאילתא וב-first() בשביל לקבל את הראשון וכו':

```
session.query(User).filter(User.name=="Tom Pythonovitz").all()
```

```
>>> User.query.filter_by(username = 'Tom Pythonovitch').all()
[User('Tom Pythonovitch' ,'Tom@mail.com',5555555,'default.jpg')]
```

נשמור את המשתמש במשתנה כדי לגשת לשדות ספציפיים שלו, למשל phone:

```
>>> user_1 = User.query.filter_by(username = 'Tom Pythonovitch').first()
>>> user_1.phone
3366858
>>> user_1.id
1
```

כדי למחוק את המסד נתונים נשתמש בפונקציה drop_all(), ואז כדי ליצור את המסד נתונים שוב נשתמש פעם נוספת בפונקציה create_all():

```
>>> Base.metadata.drop_all(engine)
>>> Base.metadata.create_all(engine)
```