Advanced Programming with Python

connecting to databases

Pepe García jgarciah@faculty.ie.edu

SCHOOL OF SCIENCE & TECHNOLOGY

Plan for today

- SQLAlchemy

Plan for today

- SQLAlchemy
- Connecting to Databases

SCHOOL OF SCIENCE & TECHNOLOGY

SQLAlchemy

SQLAlchemy is a Python library to interact with SQL databases. It is included in Anaconda.

SQLAlchemy. Engine

The entry point to SQLAlchemy is the engine. It allows us to specify where is our database, and how do we connect to it.

SCHOOL OF SCIENCE & TECHNOLOGY

SQLAlchemy. Engine

The entry point to SQLAlchemy is the engine. It allows us to specify where is our database, and how do we connect to it.

SQLite

The database we're going to use ourselves is called SQLite. It's a good database for testing purposes, and local storage, since all data is contained on a single file

SCHOOL OF SCIENCE & TECHNOLOGY

SQLAlchemy. Engine

In order to create an engine, we need to import the create_engine function.

```
from sqlalchemy import create_engine
```

SQLAlchemy. Engine

Then, we can specify how do we connect to our DB. In our case we just specify the file, but we would need to specify server, host, port, username, pass in other database engines.

```
from sqlalchemy import create_engine
engine = create_engine("sqlite:///paypalme.db")
```

SCHOOL OF SCIENCE & TECHNOLOGY

SQLAlchemy. Connection

With the engine, we specify how does one connect to the underlying DB, but we don't yet create the connection. In order to create the connection, we need to call engine.connect()

SQLAlchemy. Connection

Something very important is that, as with files, the connection must be always closed. So we can either:

```
from sqlalchemy import create_engine

engine = create_engine("sqlite:///paypalme.db")

connection = engine.connect()

# more interesting stuff

connection.close()
```

SCHOOL OF SCIENCE & TECHNOLOGY

SQLAlchemy. Connection

Or we can use a with block! (this is what we usually do)

```
from sqlalchemy import create_engine

engine = create_engine("sqlite:///paypalme.db")

with engine.connect() as connection:
    # more interesting stuff
    pass

# connection.close() Not needed anymore, `with` closes it aut
```

SCHOOL OF SCIENCE & TECHNOLOGY

SQLAlchemy. Fetching data

Finally, once we have the connection, we can start executing SQL queries!

```
# ...
query = "SELECT * FROM users"
users = connection.execute(query)
for user in users
    print(user[0]) # Results are represented as Python tuples
```

SCHOOL OF SCIENCE & TECHNOLOGY

Example

Let's explore our data!

Exercise

Exercise 1

Let's implement login in paypalme using databases!

Interlude

SQL Injection

When we accept input from the user, we **always** need to *sanitize* that input. Sanitization is the process of making it safe to be passed to the database. see https://xkcd.com/327/

SCHOOL OF SCIENCE & TECHNOLOGY

Interlude. Sanitizing inputs

Luckily, most DB libraries will handle sanitization for us. Sqlalchemy provides the `text` function:

```
from sqlalchemy import text

t = text("SELECT * FROM users")
result = connection.execute(t)
```

SCHOOL OF SCIENCE & TECHNOLOGY

Interlude. Sanitizing inputs

If we wanted to parametrize with some user's input, we can use *bind parameters*:

```
from sqlalchemy import text

t = text("SELECT * FROM users WHERE user_id=:user_id")

# SQLAlchemy will sanitize whatever we pass as bind params, pr
# SQL Injection
result = connection.execute(t, user_id="3")
```

SCHOOL OF SCIENCE & TECHNOLOGY

Exercise

Exercise 2

In the private area, show the transactions for the logged user.

Exercise

Exercise 3

Make it possible to create new transactions! Create a form at the bottom of the private area that receives the data for the transaction and shop.

SCHOOL OF SCIENCE & TECHNOLOGY