**11-14 notes**

Introduction to SQLalchemy

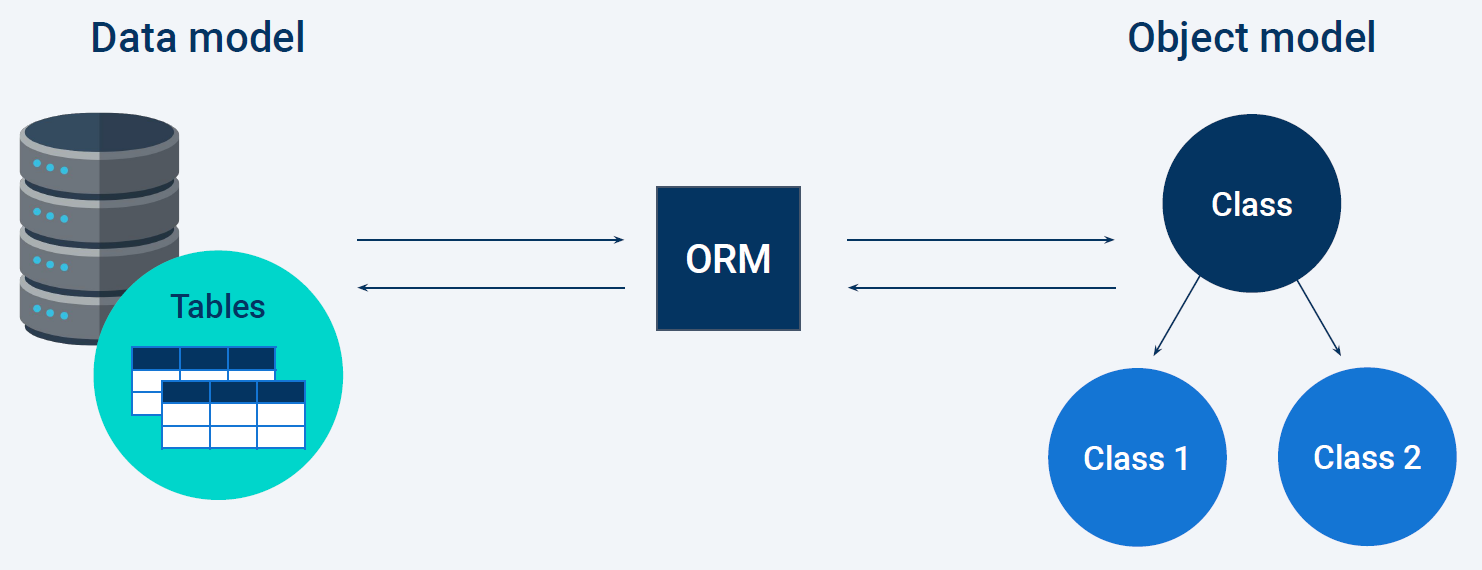
Goals:

By the end of this lesson, you will be able to:

* Connect to a SQL database by using SQLAlchemy.
* Perform basic SQL queries by using engine.execute().
* Create Python classes and objects.
* Perform create, read, update, delete (CRUD) operations on data in a SQL database by using the SQLAlchemy object-relational mapper (ORM).

Object Relational Mapper (ORM)-

Tables are called classes



Traits are inherited from the classes upstream, like a in a phylogeny tree.

Advantages of using an ORM:

* The ability to work across different SQL dialects by using the same basic Python query.
* The ability to create command line interfaces that allow users to construct SQL queries without needing to know the language.

Disadvantages:

* ORMs are like a new dialect of a language, so you have to learn how to use them.
* They may reduce control or ability to optimize a query.

**SQL Alchemy** is a Python library designed to work with SQL databases.

SQL example:

data = engine.execute(text("SELECT \* FROM BaseballPlayers"))

select \* from data

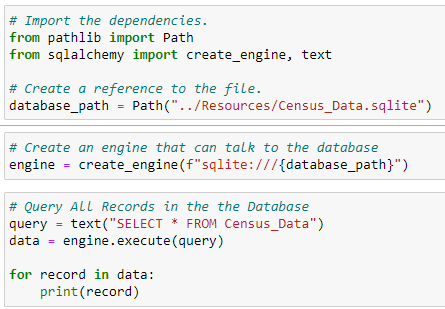
Python example:

players = session.query(BaseballPlayer)

for player in players:

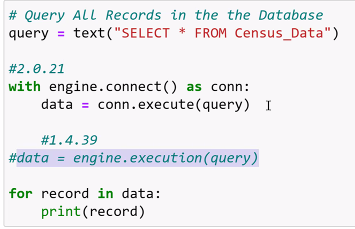
print(player.name\_given)

SQLite is a SQL dialect that shares much of the same syntax as PostgreSQL, but it is entirely serverless.



.sqlite files have all that is needed to read a database. The database, reading engine and all the tables are all stored within the SQLite file.

Querying all records in the database in code above is in version 1.4.

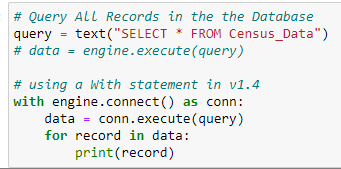


^ In v2.0.21, you have to use the **with** command above.

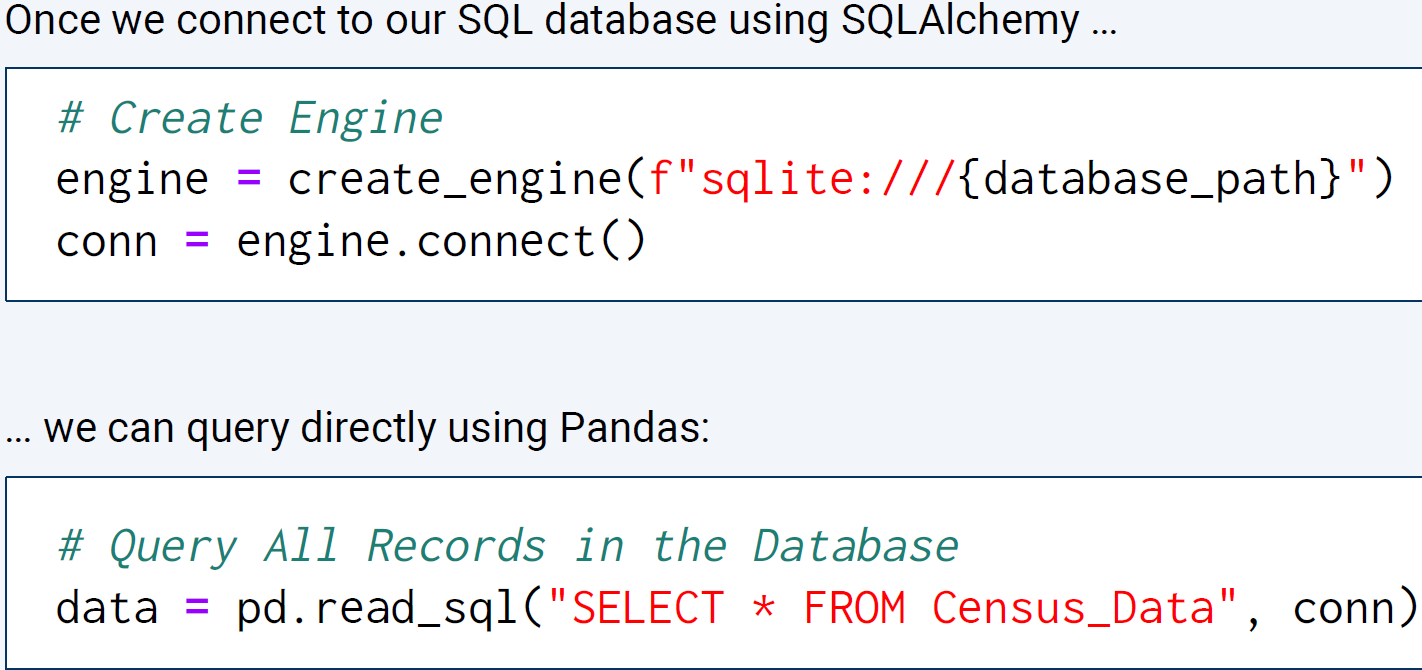
* With command is compatible in 1.4 as well
* With statement creates the connection (conn), which you can also use elsewhere later.

^ the highlighted data line will NOT work in 2.0, so you might as well use the With statement.

* Tab the For loop in. There was an issue where it wouldn’t work if not tabbed in in 1.4 that was fixed in v2.0 (see below)

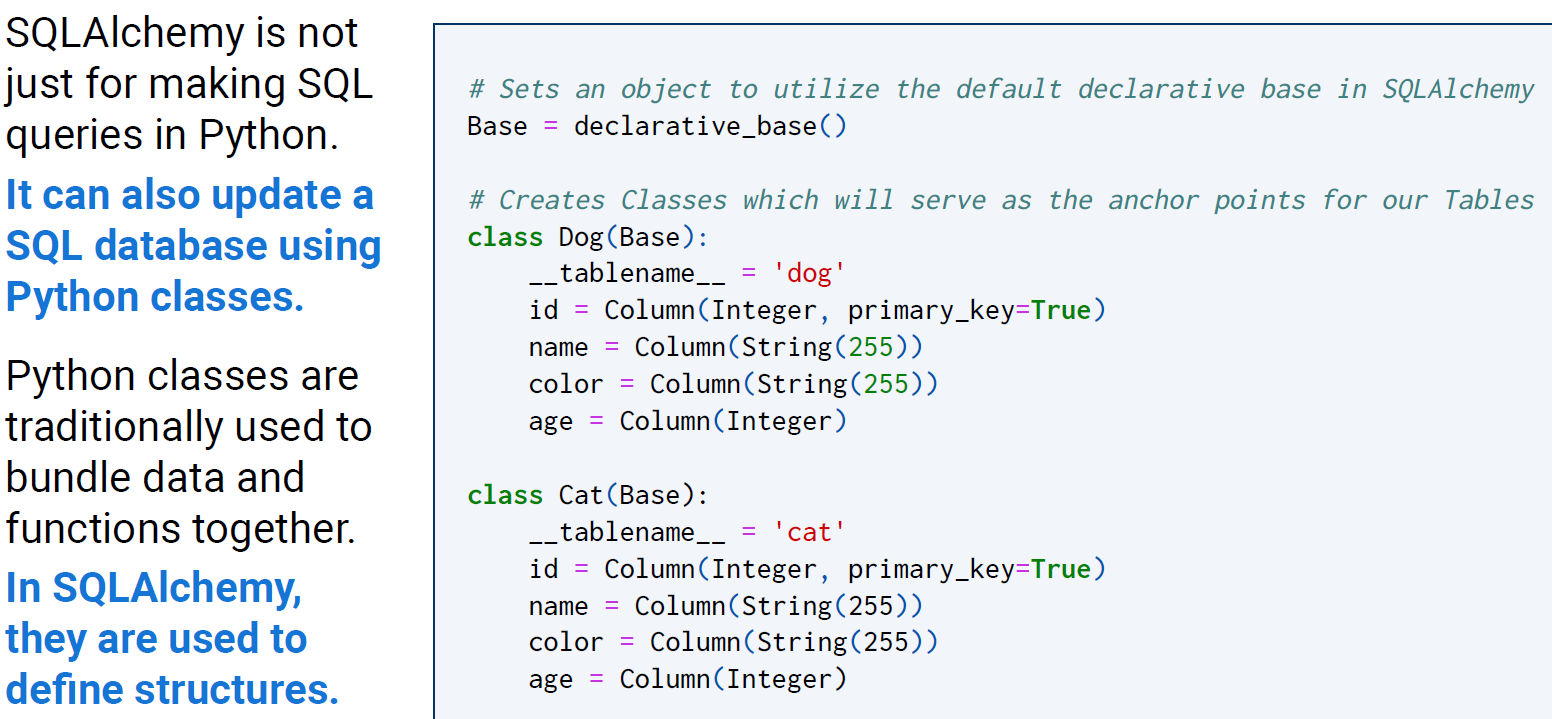


SQL Alchemy and pandas:



**A deeper dive into object oriented programming:**

How to add data to a SQLite db using python:



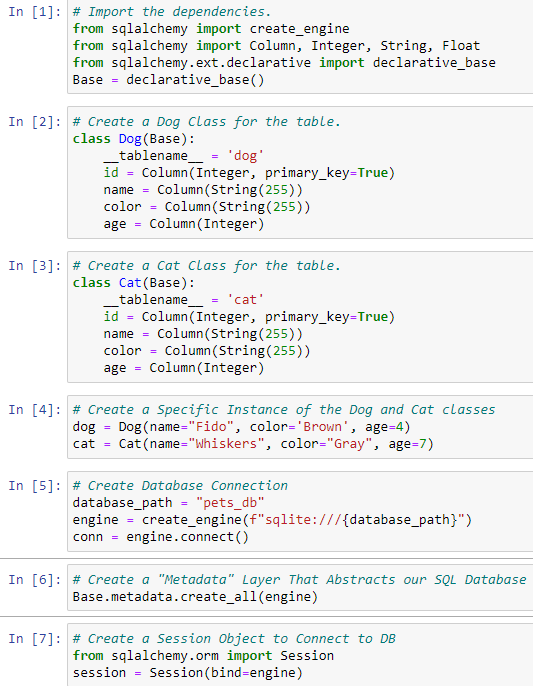
***Classes are essentially blueprints for Python objects; they allow developers to create organized variables with keys, values, and methods on the fly.***

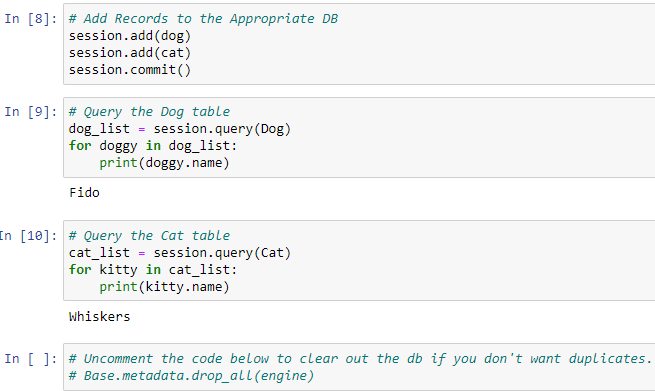
Review 7:40-7:45pm in the recording

^ example above- create a class called dog, a table, and the columns.

* You can use the class to make another table
* Each class is an object within python.

^ Must use full words, not Int and Str. The declarative\_base in SQL Alchemy uses the full words.





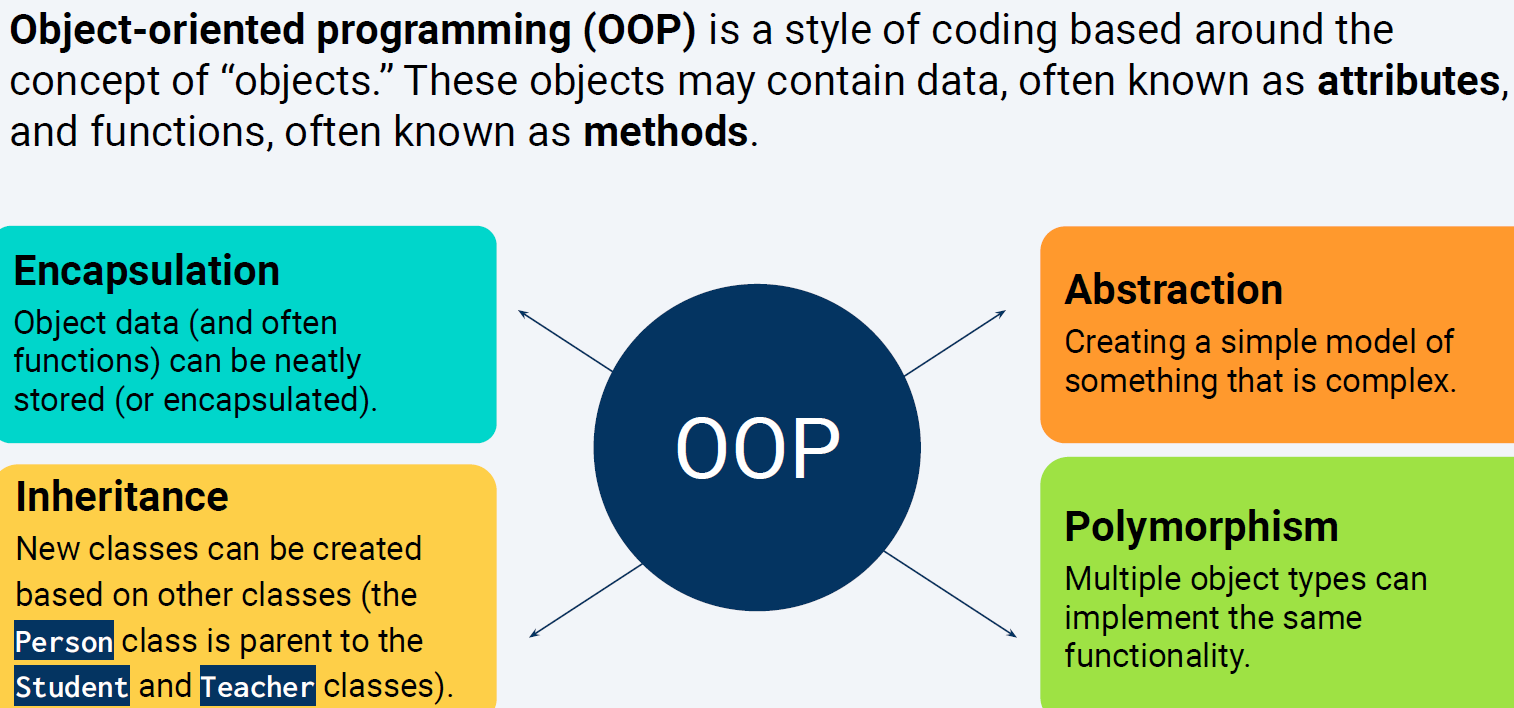
^ In [6]- the command to create all engines that we’ll use.

^ In [7] makes a session that you use to add the records you made to the database in the sqlite file.

^ In [8] actually adds the records to the database in the sqlite file.

^ In 9 and 10 are just querying the database to make sure that the records are in there.

**Object Oriented Programming:**



Review 8:12-8:22pm in recording

^ Example of Abstraction is an ERD

^ Youtube videos available explaining OOP in further detail.

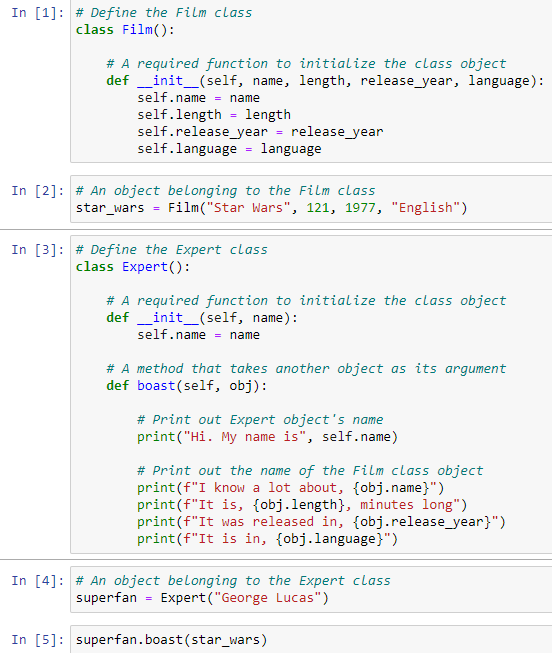


^ init is short for initialize (aka- start an instance of)

^ self means make a copy in the local sqlite file.

^ in [3]- if you were to change dog.name to mydog.name, it would return lassy instead of Fido

^ every initialization will make it’s own copy of the class.



^ Classes can have methods. boast is a method within the Expert class.

* This is similar to using \_\_init\_\_

^ In [4] uses the class, Expert to pump out the text, using the object named star\_wars to populate the obj.name, obj.length, obj.release\_year and obj.language

