# Introduction to SQLAlchemy

Data Boot Camp

# Class Objectives

By the end of today's class you will be able to:

- Connect to a SQL database with SQLAlchemy.
- Perform a SQL query with SQLAlchemy.
- Create Python classes and objects.
- Use a Python class to model a SQL table.

## Let’s Do Some Research!

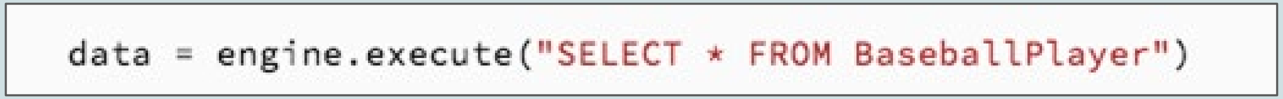
- Within your group, take a few minutes to research the answers to the following:
  - What is an ORM?
  - What are some of the benefits to using an ORM?
  - What are some of the disadvantages of using an ORM?

## SQLAlchemy is a Python library that works across a variety of SQL dialects.

Write the query once, run it anywhere!

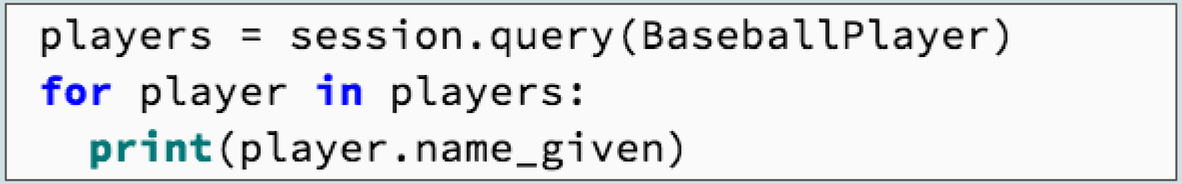
## SQLAlchemy ORM Is Flexible

It’s possible to query a database using more SQL...



```
data = engine.execute("SELECT * FROM BaseballPlayer")
```

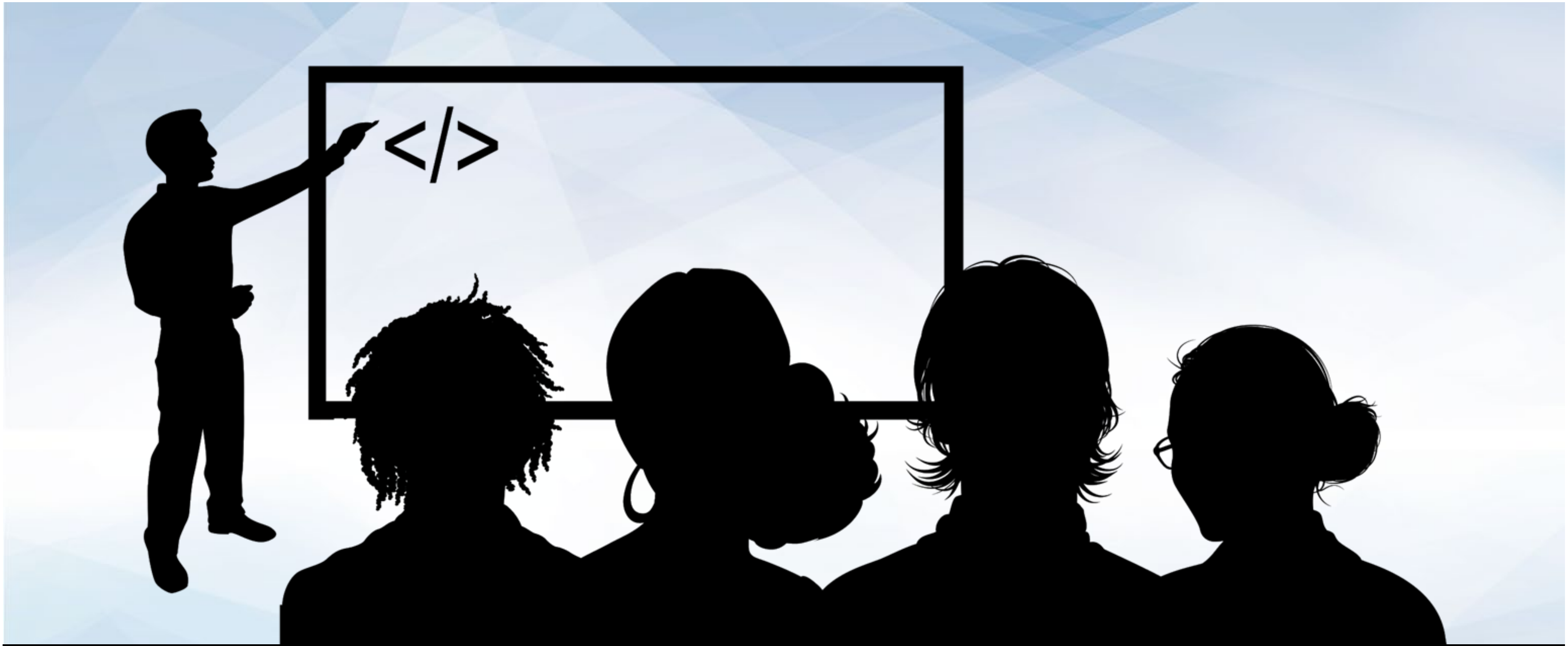
...or more Python!



```
players = session.query(BaseballPlayer)
for player in players:
  print(player.name_given)
```

# Looking at SQLAlchemy Documentation

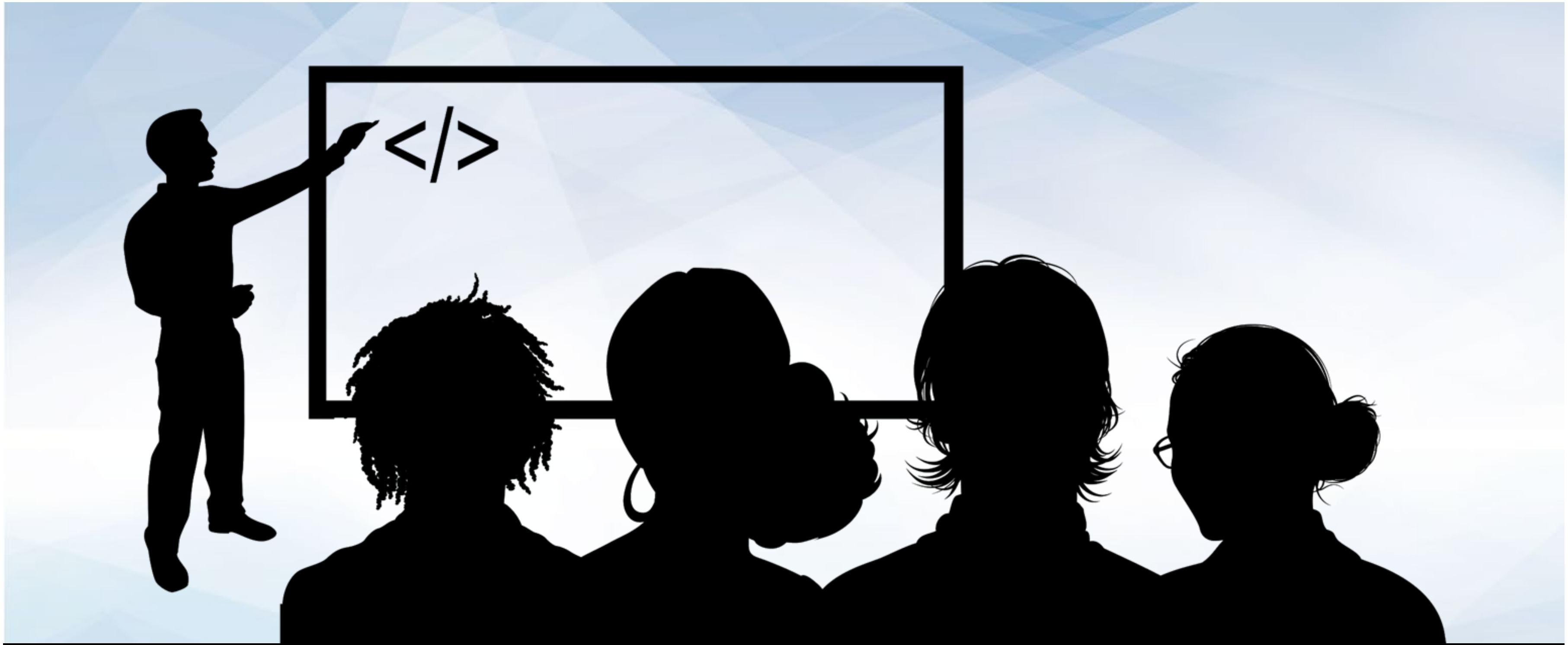
Let’s take a moment to look at the SQLAlchemy documentation!



## Instructor Demonstration

Building a SQLAlchemy Connection

Today we will only be working with one SQL dialect - SQLite!



## Instructor Demonstration

### SQLAlchemy and Pandas

One of the most impressive aspects of SQLAlchemy...

...is how it integrates with **Pandas!**

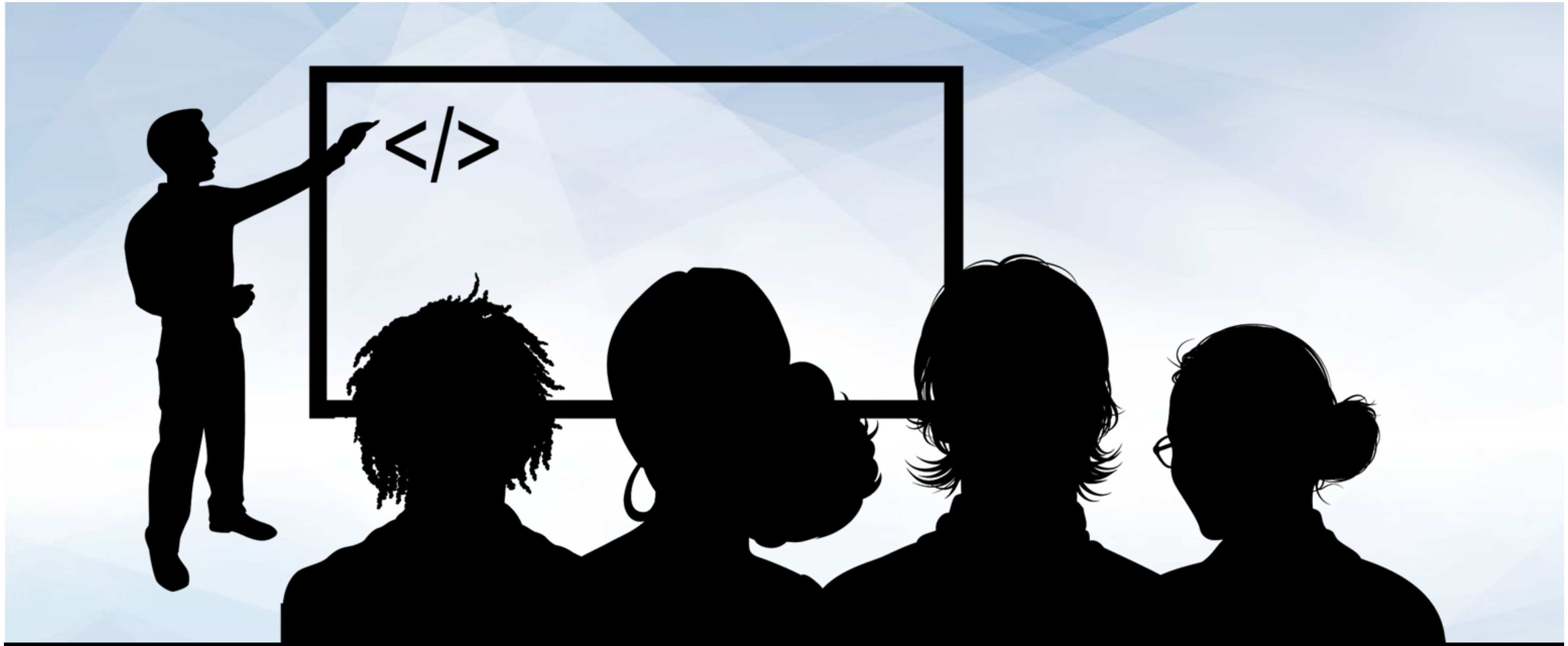
## Pandas integrates with SQLAlchemy

- Once we connect to our SQL database using SQLAlchemy

- We can query directly using pandas

```
# Create Engine
engine = create_engine(f"sqlite:///{database_path}")
conn = engine.connect()
```

```
# Query All Records in the the Database
data = pd.read_sql("SELECT * FROM Census_Data", conn)
```



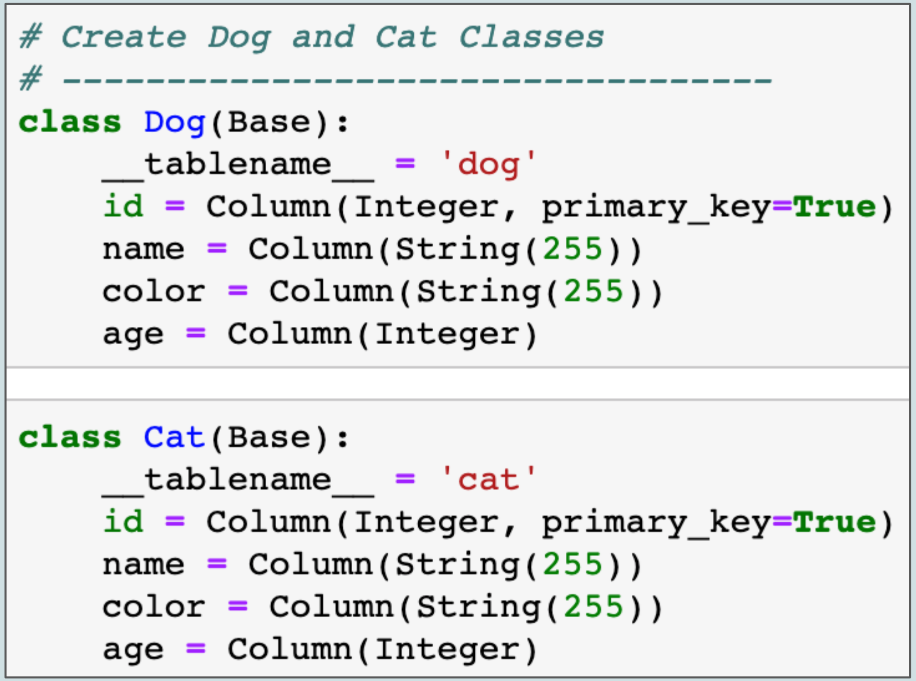
## Instructor Demonstration

Preview SQLAlchemy with Classes

# SQLAlchemy with Classes

- SQLAlchemy is not just for making SQL queries in Python
  - It can also **update** a SQL database using Python classes

- Python classes are traditionally used to bundle data and functions together
  - In SQLAlchemy they are used to define structures



```
# Create Dog and Cat Classes
# ----------------------------------
class Dog(Base):
    __tablename__ = 'dog'
    id = Column(Integer, primary_key=True)
    name = Column(String(255))
    color = Column(String(255))
    age = Column(Integer)

class Cat(Base):
    __tablename__ = 'cat'
    id = Column(Integer, primary_key=True)
    name = Column(String(255))
    color = Column(String(255))
    age = Column(Integer)
```

## Time For a Crash Course in Programming!

- Object oriented programming
  - Style of coding based around “objects”
- Objects may contain:
  - Attributes (data)
  - Methods (functions)
- Python is an object oriented programming language
  - Classes are used to interact and create objects
  - Makes code more reproducible/ adaptable

# Adding Methods to Python Classes is Easy as 1, 2, 3!

1. Define the function using def

1. Provide a name and list of parameters

1. Use class.method() to run the method in your script!

```python
# Define the Expert class
class Expert():

    # A required function to initialize the class object
    def __init__(self, name):
        self.name = name

    # A method that takes another object as its argument
    def boast(self, obj):

        # Print out Expert object's name
        print("Hi. My name is", self.name)

        # Print out the name of the Film class object
        print("I know a lot about", obj.name)
        print("It is", obj.length, "minutes long")
        print("It was released in", obj.release_year)
        print("It is in", obj.language)
```