0x0F. Python - Object-relational mapping

**PythonOOPSQLMySQLORMSQLAlchemy**

* By: Guillaume
* Weight: 1
* Project over - took place from Nov 16, 2023 5:00 AM to Nov 20, 2023 5:00 AM
* An auto review will be launched at the deadline

In a nutshell…

* **Auto QA review:** 107.25/165 mandatory & 20.8/32 optional
* **Altogether:**  **107.25%**
  + Mandatory: 65.0%
  + Optional: 65.0%
  + Calculation:  65.0% + (65.0% \* 65.0%)  == **107.25%**

Before you start…

**Please make sure your MySQL server is in 8.0** -> [How to install MySQL 8.0 in Ubuntu 20.04](https://intranet.alxswe.com/rltoken/paGukker_0KoG3D9FqymNQ)

Background Context

In this project, you will link two amazing worlds: Databases and Python!

In the first part, you will use the module MySQLdb to connect to a MySQL database and execute your SQL queries.

In the second part, you will use the module SQLAlchemy (don’t ask me how to pronounce it…) an Object Relational Mapper (ORM).

The biggest difference is: no more SQL queries! Indeed, the purpose of an ORM is to abstract the storage to the usage. With an ORM, your biggest concern will be “What can I do with my objects” and not “How this object is stored? where? when?”. You won’t write any SQL queries only Python code. Last thing, your code won’t be “storage type” dependent. You will be able to change your storage easily without re-writing your entire project.

Without ORM:

conn = MySQLdb.connect(host="localhost", port=3306, user="root", passwd="root", db="my\_db", charset="utf8")

cur = conn.cursor()

cur.execute("SELECT \* FROM states ORDER BY id ASC") # HERE I have to know SQL to grab all states in my database

query\_rows = cur.fetchall()

for row in query\_rows:

print(row)

cur.close()

conn.close()

With an ORM:

engine = create\_engine('mysql+mysqldb://{}:{}@localhost/{}'.format("root", "root", "my\_db"), pool\_pre\_ping=True)

Base.metadata.create\_all(engine)

session = Session(engine)

for state in session.query(State).order\_by(State.id).all(): # HERE: no SQL query, only objects!

print("{}: {}".format(state.id, state.name))

session.close()

Do you see the difference? Cool, right?

The biggest difficulty with ORM is: The syntax!

Indeed, all of them have the same type of syntax, but not always. Please read tutorials and don’t read the entire documentation before starting, just jump on it if you don’t get something.

Resources

**Read or watch**:

* [Object-relational mappers](https://intranet.alxswe.com/rltoken/a8DUOWhXpNX3TEwgyT-U8A)
* [mysqlclient/MySQLdb documentation](https://intranet.alxswe.com/rltoken/JtFaKjnqxudr6Hi05Us1Lw) (*please don’t pay attention to \_mysql*)
* [MySQLdb tutorial](https://intranet.alxswe.com/rltoken/TdUSYFNGbXJG1WjCEoq5FA)
* [SQLAlchemy tutorial](https://intranet.alxswe.com/rltoken/YyL5hsscviNH04XGW-XpfA)
* [SQLAlchemy](https://intranet.alxswe.com/rltoken/j9azWF2Db_2rNolTxOF3SA)
* [mysqlclient/MySQLdb](https://intranet.alxswe.com/rltoken/0zLhY9KqKjn-zmdb7X598Q)
* [Introduction to SQLAlchemy](https://intranet.alxswe.com/rltoken/pw50Bl1Bj84wksxm018dwA)
* [Flask SQLAlchemy](https://intranet.alxswe.com/rltoken/B-xIdMtGvpus8vHxAIRrPg)
* [10 common stumbling blocks for SQLAlchemy newbies](https://intranet.alxswe.com/rltoken/deIzPMrfK8Ixqm-AboFHWg)
* [Python SQLAlchemy Cheatsheet](https://intranet.alxswe.com/rltoken/dZfUNK3lJicGMK5PU0bE7Q)
* [SQLAlchemy ORM Tutorial for Python Developers](https://intranet.alxswe.com/rltoken/hNxBKC8lHge5XjsRO8ksHQ) (***Warning:****This tutorial is with PostgreSQL, but the concept of SQLAlchemy is the same with MySQL*)
* [SQLAlchemy Tutorial](https://intranet.alxswe.com/rltoken/5G_R2NmQRFqiZb84qxYERQ)
* [Python Virtual Environments: A primer](https://intranet.alxswe.com/rltoken/OXle6kXpmD88D0WbgbTWqg)

Learning Objectives

At the end of this project, you are expected to be able to [explain to anyone](https://intranet.alxswe.com/rltoken/vPPdh3HKg3t23YFxUqHpFg), **without the help of Google**:

General

* Why Python programming is awesome
* How to connect to a MySQL database from a Python script
* How to SELECT rows in a MySQL table from a Python script
* How to INSERT rows in a MySQL table from a Python script
* What ORM means
* How to map a Python Class to a MySQL table
* How to create a Python Virtual Environment

Copyright - Plagiarism

* You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
* You will not be able to meet the objectives of this or any following project by copying and pasting someone else’s work.
* You are not allowed to publish any content of this project.
* Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

* Allowed editors: vi, vim, emacs
* All your files will be interpreted/compiled on Ubuntu 20.04 LTS using python3 (version 3.8.5)
* Your files will be executed with MySQLdb version 2.0.x
* Your files will be executed with SQLAlchemy version 1.4.x
* All your files should end with a new line
* The first line of all your files should be exactly #!/usr/bin/python3
* A README.md file, at the root of the folder of the project, is mandatory
* Your code should use the pycodestyle (version 2.8.\*)
* All your files must be executable
* The length of your files will be tested using wc
* All your modules should have a documentation (python3 -c 'print(\_\_import\_\_("my\_module").\_\_doc\_\_)')
* All your classes should have a documentation (python3 -c 'print(\_\_import\_\_("my\_module").MyClass.\_\_doc\_\_)')
* All your functions (inside and outside a class) should have a documentation (python3 -c 'print(\_\_import\_\_("my\_module").my\_function.\_\_doc\_\_)' and python3 -c 'print(\_\_import\_\_("my\_module").MyClass.my\_function.\_\_doc\_\_)')
* A documentation is not a simple word, it’s a real sentence explaining what’s the purpose of the module, class or method (the length of it will be verified)
* You are not allowed to use execute with sqlalchemy

More Info

Install and activate venv

To create a Python Virtual Environment, allowing you to install specific dependencies for this python project, we will install venv:

$ sudo apt-get install python3.8-venv

$ python3 -m venv venv

$ source venv/bin/activate

Install MySQLdb module version 2.0.x

For installing MySQLdb, you need to have MySQL installed: [How to install MySQL 8.0 in Ubuntu 20.04](https://intranet.alxswe.com/rltoken/paGukker_0KoG3D9FqymNQ)

$ sudo apt-get install python3-dev

$ sudo apt-get install libmysqlclient-dev

$ sudo apt-get install zlib1g-dev

$ sudo pip3 install mysqlclient

...

$ python3

>>> import MySQLdb

>>> MySQLdb.version\_info

(2, 0, 3, 'final', 0)

Install SQLAlchemy module version 1.4.x

$ sudo pip3 install SQLAlchemy

...

$ python3

>>> import sqlalchemy

>>> sqlalchemy.\_\_version\_\_

'1.4.22'

Also, you can have this warning message:

/usr/local/lib/python3.4/dist-packages/sqlalchemy/engine/default.py:552: Warning: (1681, "'@@SESSION.GTID\_EXECUTED' is deprecated and will be re

moved in a future release.")

cursor.execute(statement, parameters)

You can ignore it.

Tasks

0. Get all states

**mandatory**

Score: 65.0% (*Checks completed: 100.0%*)

Write a script that lists all states from the database hbtn\_0e\_0\_usa:

* Your script should take 3 arguments: mysql username, mysql password and database name (no argument validation needed)
* You must use the module MySQLdb (import MySQLdb)
* Your script should connect to a MySQL server running on localhost at port 3306
* Results must be sorted in ascending order by states.id
* Results must be displayed as they are in the example below
* Your code should not be executed when imported

guillaume@ubuntu:~/0x0F$ cat 0-select\_states.sql

-- Create states table in hbtn\_0e\_0\_usa with some data

CREATE DATABASE IF NOT EXISTS hbtn\_0e\_0\_usa;

USE hbtn\_0e\_0\_usa;

CREATE TABLE IF NOT EXISTS states (

id INT NOT NULL AUTO\_INCREMENT,

name VARCHAR(256) NOT NULL,

PRIMARY KEY (id)

);

INSERT INTO states (name) VALUES ("California"), ("Arizona"), ("Texas"), ("New York"), ("Nevada");

guillaume@ubuntu:~/0x0F$ cat 0-select\_states.sql | mysql -uroot -p

Enter password:

guillaume@ubuntu:~/0x0F$ ./0-select\_states.py root root hbtn\_0e\_0\_usa

(1, 'California')

(2, 'Arizona')

(3, 'Texas')

(4, 'New York')

(5, 'Nevada')

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: 0-select\_states.py

 Done! Help Check your code Get a sandbox QA Review

1. Filter states

**mandatory**

Score: 65.0% (*Checks completed: 100.0%*)

Write a script that lists all states with a name starting with N (upper N) from the database hbtn\_0e\_0\_usa:

* Your script should take 3 arguments: mysql username, mysql password and database name (no argument validation needed)
* You must use the module MySQLdb (import MySQLdb)
* Your script should connect to a MySQL server running on localhost at port 3306
* Results must be sorted in ascending order by states.id
* Results must be displayed as they are in the example below
* Your code should not be executed when imported

guillaume@ubuntu:~/0x0F$ cat 0-select\_states.sql

-- Create states table in hbtn\_0e\_0\_usa with some data

CREATE DATABASE IF NOT EXISTS hbtn\_0e\_0\_usa;

USE hbtn\_0e\_0\_usa;

CREATE TABLE IF NOT EXISTS states (

id INT NOT NULL AUTO\_INCREMENT,

name VARCHAR(256) NOT NULL,

PRIMARY KEY (id)

);

INSERT INTO states (name) VALUES ("California"), ("Arizona"), ("Texas"), ("New York"), ("Nevada");

guillaume@ubuntu:~/0x0F$ cat 0-select\_states.sql | mysql -uroot -p

Enter password:

guillaume@ubuntu:~/0x0F$ ./1-filter\_states.py root root hbtn\_0e\_0\_usa

(4, 'New York')

(5, 'Nevada')

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: 1-filter\_states.py

 Done! Help Check your code Get a sandbox QA Review

2. Filter states by user input

**mandatory**

Score: 65.0% (*Checks completed: 100.0%*)

Write a script that takes in an argument and displays all values in the states table of hbtn\_0e\_0\_usa where name matches the argument.

* Your script should take 4 arguments: mysql username, mysql password, database name and state name searched (no argument validation needed)
* You must use the module MySQLdb (import MySQLdb)
* Your script should connect to a MySQL server running on localhost at port 3306
* You must use format to create the SQL query with the user input
* Results must be sorted in ascending order by states.id
* Results must be displayed as they are in the example below
* Your code should not be executed when imported

guillaume@ubuntu:~/0x0F$ cat 0-select\_states.sql

-- Create states table in hbtn\_0e\_0\_usa with some data

CREATE DATABASE IF NOT EXISTS hbtn\_0e\_0\_usa;

USE hbtn\_0e\_0\_usa;

CREATE TABLE IF NOT EXISTS states (

id INT NOT NULL AUTO\_INCREMENT,

name VARCHAR(256) NOT NULL,

PRIMARY KEY (id)

);

INSERT INTO states (name) VALUES ("California"), ("Arizona"), ("Texas"), ("New York"), ("Nevada");

guillaume@ubuntu:~/0x0F$ cat 0-select\_states.sql | mysql -uroot -p

Enter password:

guillaume@ubuntu:~/0x0F$ ./2-my\_filter\_states.py root root hbtn\_0e\_0\_usa 'Arizona'

(2, 'Arizona')

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: 2-my\_filter\_states.py

 Done! Help Check your code Get a sandbox QA Review

3. SQL Injection...

**mandatory**

Score: 65.0% (*Checks completed: 100.0%*)

Wait, do you remember the previous task? Did you test "Arizona'; TRUNCATE TABLE states ; SELECT \* FROM states WHERE name = '" as an input?

guillaume@ubuntu:~/0x0F$ ./2-my\_filter\_states.py root root hbtn\_0e\_0\_usa "Arizona'; TRUNCATE TABLE states ; SELECT \* FROM states WHERE name = '"

(2, 'Arizona')

guillaume@ubuntu:~/0x0F$ ./0-select\_states.py root root hbtn\_0e\_0\_usa

guillaume@ubuntu:~/0x0F$

What? Empty?

Yes, it’s an [SQL injection](https://intranet.alxswe.com/rltoken/qzLjdkHPTue2U1isMj5fJA) to delete all records of a table…

Once again, write a script that takes in arguments and displays all values in the states table of hbtn\_0e\_0\_usa where name matches the argument. But this time, write one that is safe from MySQL injections!

* Your script should take 4 arguments: mysql username, mysql password, database name and state name searched (safe from MySQL injection)
* You must use the module MySQLdb (import MySQLdb)
* Your script should connect to a MySQL server running on localhost at port 3306
* Results must be sorted in ascending order by states.id
* Results must be displayed as they are in the example below
* Your code should not be executed when imported

guillaume@ubuntu:~/0x0F$ cat 0-select\_states.sql

-- Create states table in hbtn\_0e\_0\_usa with some data

CREATE DATABASE IF NOT EXISTS hbtn\_0e\_0\_usa;

USE hbtn\_0e\_0\_usa;

CREATE TABLE IF NOT EXISTS states (

id INT NOT NULL AUTO\_INCREMENT,

name VARCHAR(256) NOT NULL,

PRIMARY KEY (id)

);

INSERT INTO states (name) VALUES ("California"), ("Arizona"), ("Texas"), ("New York"), ("Nevada");

guillaume@ubuntu:~/0x0F$ cat 0-select\_states.sql | mysql -uroot -p

Enter password:

guillaume@ubuntu:~/0x0F$ ./3-my\_safe\_filter\_states.py root root hbtn\_0e\_0\_usa 'Arizona'

(2, 'Arizona')

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: 3-my\_safe\_filter\_states.py

 Done! Help Check your code Get a sandbox QA Review

4. Cities by states

**mandatory**

Score: 65.0% (*Checks completed: 100.0%*)

Write a script that lists all cities from the database hbtn\_0e\_4\_usa

* Your script should take 3 arguments: mysql username, mysql password and database name
* You must use the module MySQLdb (import MySQLdb)
* Your script should connect to a MySQL server running on localhost at port 3306
* Results must be sorted in ascending order by cities.id
* You can use only execute() once
* Results must be displayed as they are in the example below
* Your code should not be executed when imported

guillaume@ubuntu:~/0x0F$ cat 4-cities\_by\_state.sql

-- Create states table in hbtn\_0e\_4\_usa with some data

CREATE DATABASE IF NOT EXISTS hbtn\_0e\_4\_usa;

USE hbtn\_0e\_4\_usa;

CREATE TABLE IF NOT EXISTS states (

id INT NOT NULL AUTO\_INCREMENT,

name VARCHAR(256) NOT NULL,

PRIMARY KEY (id)

);

INSERT INTO states (name) VALUES ("California"), ("Arizona"), ("Texas"), ("New York"), ("Nevada");

CREATE TABLE IF NOT EXISTS cities (

id INT NOT NULL AUTO\_INCREMENT,

state\_id INT NOT NULL,

name VARCHAR(256) NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY(state\_id) REFERENCES states(id)

);

INSERT INTO cities (state\_id, name) VALUES (1, "San Francisco"), (1, "San Jose"), (1, "Los Angeles"), (1, "Fremont"), (1, "Livermore");

INSERT INTO cities (state\_id, name) VALUES (2, "Page"), (2, "Phoenix");

INSERT INTO cities (state\_id, name) VALUES (3, "Dallas"), (3, "Houston"), (3, "Austin");

INSERT INTO cities (state\_id, name) VALUES (4, "New York");

INSERT INTO cities (state\_id, name) VALUES (5, "Las Vegas"), (5, "Reno"), (5, "Henderson"), (5, "Carson City");

guillaume@ubuntu:~/0x0F$ cat 4-cities\_by\_state.sql | mysql -uroot -p

Enter password:

guillaume@ubuntu:~/0x0F$ ./4-cities\_by\_state.py root root hbtn\_0e\_4\_usa

(1, 'San Francisco', 'California')

(2, 'San Jose', 'California')

(3, 'Los Angeles', 'California')

(4, 'Fremont', 'California')

(5, 'Livermore', 'California')

(6, 'Page', 'Arizona')

(7, 'Phoenix', 'Arizona')

(8, 'Dallas', 'Texas')

(9, 'Houston', 'Texas')

(10, 'Austin', 'Texas')

(11, 'New York', 'New York')

(12, 'Las Vegas', 'Nevada')

(13, 'Reno', 'Nevada')

(14, 'Henderson', 'Nevada')

(15, 'Carson City', 'Nevada')

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: 4-cities\_by\_state.py

 Done! Help Check your code Get a sandbox QA Review

5. All cities by state

**mandatory**

Score: 65.0% (*Checks completed: 100.0%*)

Write a script that takes in the name of a state as an argument and lists all cities of that state, using the database hbtn\_0e\_4\_usa

* Your script should take 4 arguments: mysql username, mysql password, database name and state name (SQL injection free!)
* You must use the module MySQLdb (import MySQLdb)
* Your script should connect to a MySQL server running on localhost at port 3306
* Results must be sorted in ascending order by cities.id
* You can use only execute() once
* The results must be displayed as they are in the example below
* Your code should not be executed when imported

guillaume@ubuntu:~/0x0F$ cat 4-cities\_by\_state.sql

-- Create states table in hbtn\_0e\_4\_usa with some data

CREATE DATABASE IF NOT EXISTS hbtn\_0e\_4\_usa;

USE hbtn\_0e\_4\_usa;

CREATE TABLE IF NOT EXISTS states (

id INT NOT NULL AUTO\_INCREMENT,

name VARCHAR(256) NOT NULL,

PRIMARY KEY (id)

);

INSERT INTO states (name) VALUES ("California"), ("Arizona"), ("Texas"), ("New York"), ("Nevada");

CREATE TABLE IF NOT EXISTS cities (

id INT NOT NULL AUTO\_INCREMENT,

state\_id INT NOT NULL,

name VARCHAR(256) NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY(state\_id) REFERENCES states(id)

);

INSERT INTO cities (state\_id, name) VALUES (1, "San Francisco"), (1, "San Jose"), (1, "Los Angeles"), (1, "Fremont"), (1, "Livermore");

INSERT INTO cities (state\_id, name) VALUES (2, "Page"), (2, "Phoenix");

INSERT INTO cities (state\_id, name) VALUES (3, "Dallas"), (3, "Houston"), (3, "Austin");

INSERT INTO cities (state\_id, name) VALUES (4, "New York");

INSERT INTO cities (state\_id, name) VALUES (5, "Las Vegas"), (5, "Reno"), (5, "Henderson"), (5, "Carson City");

guillaume@ubuntu:~/0x0F$ ./5-filter\_cities.py root root hbtn\_0e\_4\_usa Texas

guillaume@ubuntu:~/0x0F$ cat 4-cities\_by\_state.sql | mysql -uroot -p

Enter password:

guillaume@ubuntu:~/0x0F$ ./5-filter\_cities.py root root hbtn\_0e\_4\_usa Texas

Dallas, Houston, Austin

guillaume@ubuntu:~/0x0F$ ./5-filter\_cities.py root root hbtn\_0e\_4\_usa Hawaii

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: 5-filter\_cities.py

 Done! Help Check your code Get a sandbox QA Review

6. First state model

**mandatory**

Score: 65.0% (*Checks completed: 100.0%*)

Write a python file that contains the class definition of a State and an instance Base = declarative\_base():

* State class:
  + inherits from Base [Tips](https://intranet.alxswe.com/rltoken/SFKIwNZ3IG6_4TL6dEsIuA)
  + links to the MySQL table states
  + class attribute id that represents a column of an auto-generated, unique integer, can’t be null and is a primary key
  + class attribute name that represents a column of a string with maximum 128 characters and can’t be null
* You must use the module SQLAlchemy
* Your script should connect to a MySQL server running on localhost at port 3306
* **WARNING:** all classes who inherit from Base **must** be imported before calling Base.metadata.create\_all(engine)

guillaume@ubuntu:~/0x0F$ cat 6-model\_state.sql

-- Create database hbtn\_0e\_6\_usa

CREATE DATABASE IF NOT EXISTS hbtn\_0e\_6\_usa;

USE hbtn\_0e\_6\_usa;

SHOW CREATE TABLE states;

guillaume@ubuntu:~/0x0F$ cat 6-model\_state.sql | mysql -uroot -p

Enter password:

ERROR 1146 (42S02) at line 4: Table 'hbtn\_0e\_6\_usa.states' doesn't exist

guillaume@ubuntu:~/0x0F$ cat 6-model\_state.py

#!/usr/bin/python3

"""Start link class to table in database

"""

import sys

from model\_state import Base, State

from sqlalchemy import (create\_engine)

if \_\_name\_\_ == "\_\_main\_\_":

engine = create\_engine('mysql+mysqldb://{}:{}@localhost/{}'.format(sys.argv[1], sys.argv[2], sys.argv[3]), pool\_pre\_ping=True)

Base.metadata.create\_all(engine)

guillaume@ubuntu:~/0x0F$ ./6-model\_state.py root root hbtn\_0e\_6\_usa

guillaume@ubuntu:~/0x0F$ cat 6-model\_state.sql | mysql -uroot -p

Enter password:

Table Create Table

states CREATE TABLE `states` (\n `id` int(11) NOT NULL AUTO\_INCREMENT,\n `name` varchar(128) NOT NULL,\n PRIMARY KEY (`id`)\n) ENGINE=InnoDB DEFAULT CHARSET=latin1

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: model\_state.py

 Done! Help Check your code Get a sandbox QA Review

7. All states via SQLAlchemy

**mandatory**

Score: 65.0% (*Checks completed: 100.0%*)

Write a script that lists all State objects from the database hbtn\_0e\_6\_usa

* Your script should take 3 arguments: mysql username, mysql password and database name
* You must use the module SQLAlchemy
* You must import State and Base from model\_state - from model\_state import Base, State
* Your script should connect to a MySQL server running on localhost at port 3306
* Results must be sorted in ascending order by states.id
* The results must be displayed as they are in the example below
* Your code should not be executed when imported

guillaume@ubuntu:~/0x0F$ cat 7-model\_state\_fetch\_all.sql

-- Insert states

INSERT INTO states (name) VALUES ("California"), ("Arizona"), ("Texas"), ("New York"), ("Nevada");

guillaume@ubuntu:~/0x0F$ cat 7-model\_state\_fetch\_all.sql | mysql -uroot -p hbtn\_0e\_6\_usa

Enter password:

guillaume@ubuntu:~/0x0F$ ./7-model\_state\_fetch\_all.py root root hbtn\_0e\_6\_usa

1: California

2: Arizona

3: Texas

4: New York

5: Nevada

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: 7-model\_state\_fetch\_all.py

 Done! Help Check your code Get a sandbox QA Review

8. First state

**mandatory**

Score: 65.0% (*Checks completed: 100.0%*)

Write a script that prints the first State object from the database hbtn\_0e\_6\_usa

* Your script should take 3 arguments: mysql username, mysql password and database name
* You must use the module SQLAlchemy
* You must import State and Base from model\_state - from model\_state import Base, State
* Your script should connect to a MySQL server running on localhost at port 3306
* The state you display must be the first in states.id
* You are not allowed to fetch all states from the database before displaying the result
* The results must be displayed as they are in the example below
* If the table states is empty, print Nothing followed by a new line
* Your code should not be executed when imported

guillaume@ubuntu:~/0x0F$ ./8-model\_state\_fetch\_first.py root root hbtn\_0e\_6\_usa

1: California

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: 8-model\_state\_fetch\_first.py

 Done! Help Check your code Get a sandbox QA Review

9. Contains `a`

**mandatory**

Score: 65.0% (*Checks completed: 100.0%*)

Write a script that lists all State objects that contain the letter a from the database hbtn\_0e\_6\_usa

* Your script should take 3 arguments: mysql username, mysql password and database name
* You must use the module SQLAlchemy
* You must import State and Base from model\_state - from model\_state import Base, State
* Your script should connect to a MySQL server running on localhost at port 3306
* Results must be sorted in ascending order by states.id
* The results must be displayed as they are in the example below
* Your code should not be executed when imported

guillaume@ubuntu:~/0x0F$ ./9-model\_state\_filter\_a.py root root hbtn\_0e\_6\_usa

1: California

2: Arizona

3: Texas

5: Nevada

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: 9-model\_state\_filter\_a.py

 Done! Help Check your code Get a sandbox QA Review

10. Get a state

**mandatory**

Score: 65.0% (*Checks completed: 100.0%*)

Write a script that prints the State object with the name passed as argument from the database hbtn\_0e\_6\_usa

* Your script should take 4 arguments: mysql username, mysql password, database name and state name to search (SQL injection free)
* You must use the module SQLAlchemy
* You must import State and Base from model\_state - from model\_state import Base, State
* Your script should connect to a MySQL server running on localhost at port 3306
* You can assume you have one record with the state name to search
* Results must display the states.id
* If no state has the name you searched for, display Not found
* Your code should not be executed when imported

guillaume@ubuntu:~/0x0F$ ./10-model\_state\_my\_get.py root root hbtn\_0e\_6\_usa Texas

3

guillaume@ubuntu:~/0x0F$ ./10-model\_state\_my\_get.py root root hbtn\_0e\_6\_usa Illinois

Not found

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: 10-model\_state\_my\_get.py

 Done! Help Check your code Get a sandbox QA Review

11. Add a new state

**mandatory**

Score: 65.0% (*Checks completed: 100.0%*)

Write a script that adds the State object “Louisiana” to the database hbtn\_0e\_6\_usa

* Your script should take 3 arguments: mysql username, mysql password and database name
* You must use the module SQLAlchemy
* You must import State and Base from model\_state - from model\_state import Base, State
* Your script should connect to a MySQL server running on localhost at port 3306
* Print the new states.id after creation
* Your code should not be executed when imported

guillaume@ubuntu:~/0x0F$ ./11-model\_state\_insert.py root root hbtn\_0e\_6\_usa

6

guillaume@ubuntu:~/0x0F$ ./7-model\_state\_fetch\_all.py root root hbtn\_0e\_6\_usa

1: California

2: Arizona

3: Texas

4: New York

5: Nevada

6: Louisiana

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: 11-model\_state\_insert.py

 Done! Help Check your code Get a sandbox QA Review

12. Update a state

**mandatory**

Score: 65.0% (*Checks completed: 100.0%*)

Write a script that changes the name of a State object from the database hbtn\_0e\_6\_usa

* Your script should take 3 arguments: mysql username, mysql password and database name
* You must use the module SQLAlchemy
* You must import State and Base from model\_state - from model\_state import Base, State
* Your script should connect to a MySQL server running on localhost at port 3306
* Change the name of the State where id = 2 to New Mexico
* Your code should not be executed when imported

guillaume@ubuntu:~/0x0F$ ./12-model\_state\_update\_id\_2.py root root hbtn\_0e\_6\_usa

guillaume@ubuntu:~/0x0F$ ./7-model\_state\_fetch\_all.py root root hbtn\_0e\_6\_usa

1: California

2: New Mexico

3: Texas

4: New York

5: Nevada

6: Louisiana

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: 12-model\_state\_update\_id\_2.py

 Done! Help Check your code Get a sandbox QA Review

13. Delete states

**mandatory**

Score: 65.0% (*Checks completed: 100.0%*)

Write a script that deletes all State objects with a name containing the letter a from the database hbtn\_0e\_6\_usa

* Your script should take 3 arguments: mysql username, mysql password and database name
* You must use the module SQLAlchemy
* You must import State and Base from model\_state - from model\_state import Base, State
* Your script should connect to a MySQL server running on localhost at port 3306
* Your code should not be executed when imported

guillaume@ubuntu:~/0x0F$ ./13-model\_state\_delete\_a.py root root hbtn\_0e\_6\_usa

guillaume@ubuntu:~/0x0F$ ./7-model\_state\_fetch\_all.py root root hbtn\_0e\_6\_usa

2: New Mexico

4: New York

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: 13-model\_state\_delete\_a.py

 Done! Help Check your code Get a sandbox QA Review

14. Cities in state

**mandatory**

Score: 65.0% (*Checks completed: 100.0%*)

Write a Python file similar to model\_state.py named model\_city.py that contains the class definition of a City.

* City class:
  + inherits from Base (imported from model\_state)
  + links to the MySQL table cities
  + class attribute id that represents a column of an auto-generated, unique integer, can’t be null and is a primary key
  + class attribute name that represents a column of a string of 128 characters and can’t be null
  + class attribute state\_id that represents a column of an integer, can’t be null and is a foreign key to states.id
* You must use the module SQLAlchemy

Next, write a script 14-model\_city\_fetch\_by\_state.py that prints all City objects from the database hbtn\_0e\_14\_usa:

* Your script should take 3 arguments: mysql username, mysql password and database name
* You must use the module SQLAlchemy
* You must import State and Base from model\_state - from model\_state import Base, State
* Your script should connect to a MySQL server running on localhost at port 3306
* Results must be sorted in ascending order by cities.id
* Results must be display as they are in the example below (<state name>: (<city id>) <city name>)
* Your code should not be executed when imported

guillaume@ubuntu:~/0x0F$ cat 14-model\_city\_fetch\_by\_state.sql

-- Create database hbtn\_0e\_14\_usa, tables states and cities + some data

CREATE DATABASE IF NOT EXISTS hbtn\_0e\_14\_usa;

USE hbtn\_0e\_14\_usa;

CREATE TABLE IF NOT EXISTS states (

id INT NOT NULL AUTO\_INCREMENT,

name VARCHAR(256) NOT NULL,

PRIMARY KEY (id)

);

INSERT INTO states (name) VALUES ("California"), ("Arizona"), ("Texas"), ("New York"), ("Nevada");

CREATE TABLE IF NOT EXISTS cities (

id INT NOT NULL AUTO\_INCREMENT,

state\_id INT NOT NULL,

name VARCHAR(256) NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY(state\_id) REFERENCES states(id)

);

INSERT INTO cities (state\_id, name) VALUES (1, "San Francisco"), (1, "San Jose"), (1, "Los Angeles"), (1, "Fremont"), (1, "Livermore");

INSERT INTO cities (state\_id, name) VALUES (2, "Page"), (2, "Phoenix");

INSERT INTO cities (state\_id, name) VALUES (3, "Dallas"), (3, "Houston"), (3, "Austin");

INSERT INTO cities (state\_id, name) VALUES (4, "New York");

INSERT INTO cities (state\_id, name) VALUES (5, "Las Vegas"), (5, "Reno"), (5, "Henderson"), (5, "Carson City");

guillaume@ubuntu:~/0x0F$ cat 14-model\_city\_fetch\_by\_state.sql | mysql -uroot -p

Enter password:

guillaume@ubuntu:~/0x0F$ ./14-model\_city\_fetch\_by\_state.py root root hbtn\_0e\_14\_usa

California: (1) San Francisco

California: (2) San Jose

California: (3) Los Angeles

California: (4) Fremont

California: (5) Livermore

Arizona: (6) Page

Arizona: (7) Phoenix

Texas: (8) Dallas

Texas: (9) Houston

Texas: (10) Austin

New York: (11) New York

Nevada: (12) Las Vegas

Nevada: (13) Reno

Nevada: (14) Henderson

Nevada: (15) Carson City

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: model\_city.py, 14-model\_city\_fetch\_by\_state.py

 Done! Help Check your code Get a sandbox QA Review

15. City relationship

**#advanced**

Score: 65.0% (*Checks completed: 100.0%*)

Improve the files model\_city.py and model\_state.py, and save them as relationship\_city.py and relationship\_state.py:

* City class:
  + No change
* State class:
  + In addition to previous requirements, the class attribute cities must represent a relationship with the class City. If the State object is deleted, all linked City objects must be automatically deleted. Also, the reference from a City object to his State should be named state
* You must use the module SQLAlchemy

Write a script that creates the State “California” with the City “San Francisco” from the database hbtn\_0e\_100\_usa: (100-relationship\_states\_cities.py)

* Your script should take 3 arguments: mysql username, mysql password and database name
* You must use the module SQLAlchemy
* Your script should connect to a MySQL server running on localhost at port 3306
* You must use the cities relationship for all State objects
* Your code should not be executed when imported

guillaume@ubuntu:~/0x0F$ cat 100-relationship\_states\_cities.sql

-- Create the database hbtn\_0e\_100\_usa

CREATE DATABASE IF NOT EXISTS hbtn\_0e\_100\_usa;

USE hbtn\_0e\_100\_usa;

SELECT \* FROM states;

SELECT \* FROM cities;

guillaume@ubuntu:~/0x0F$ cat 100-relationship\_states\_cities.sql | mysql -uroot -p

Enter password:

ERROR 1146 (42S02) at line 5: Table 'hbtn\_0e\_100\_usa.states' doesn't exist

guillaume@ubuntu:~/0x0F$ ./100-relationship\_states\_cities.py root root hbtn\_0e\_100\_usa

guillaume@ubuntu:~/0x0F$ cat 100-relationship\_states\_cities.sql | mysql -uroot -p

Enter password:

id name

1 California

id name state\_id

1 San Francisco 1

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: relationship\_city.py, relationship\_state.py, 100-relationship\_states\_cities.py

 Done! Help Check your code Get a sandbox QA Review

16. List relationship

**#advanced**

Score: 65.0% (*Checks completed: 100.0%*)

Write a script that lists all State objects, and corresponding City objects, contained in the database hbtn\_0e\_101\_usa

* Your script should take 3 arguments: mysql username, mysql password and database name
* You must use the module SQLAlchemy
* The connection to your MySQL server must be to localhost on port 3306
* You must only use one query to the database
* You must use the cities relationship for all State objects
* Results must be sorted in ascending order by states.id and cities.id
* Results must be displayed as they are in the example below
* Your code should not be executed when imported

<state id>: <state name>

<tabulation><city id>: <city name>

guillaume@ubuntu:~/0x0F$ cat 101-relationship\_states\_cities\_list.sql

-- Create states table in hbtn\_0e\_101\_usa with some data

CREATE DATABASE IF NOT EXISTS hbtn\_0e\_101\_usa;

USE hbtn\_0e\_101\_usa;

CREATE TABLE IF NOT EXISTS states (

id INT NOT NULL AUTO\_INCREMENT,

name VARCHAR(256) NOT NULL,

PRIMARY KEY (id)

);

INSERT INTO states (name) VALUES ("California"), ("Arizona"), ("Texas"), ("New York"), ("Nevada");

CREATE TABLE IF NOT EXISTS cities (

id INT NOT NULL AUTO\_INCREMENT,

state\_id INT NOT NULL,

name VARCHAR(256) NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY(state\_id) REFERENCES states(id)

);

INSERT INTO cities (state\_id, name) VALUES (1, "San Francisco"), (1, "San Jose"), (1, "Los Angeles"), (1, "Fremont"), (1, "Livermore");

INSERT INTO cities (state\_id, name) VALUES (2, "Page"), (2, "Phoenix");

INSERT INTO cities (state\_id, name) VALUES (3, "Dallas"), (3, "Houston"), (3, "Austin");

INSERT INTO cities (state\_id, name) VALUES (4, "New York");

INSERT INTO cities (state\_id, name) VALUES (5, "Las Vegas"), (5, "Reno"), (5, "Henderson"), (5, "Carson City");

guillaume@ubuntu:~/0x0F$ cat 101-relationship\_states\_cities\_list.sql | mysql -uroot -p

guillaume@ubuntu:~/0x0F$ ./101-relationship\_states\_cities\_list.py root root hbtn\_0e\_101\_usa

1: California

1: San Francisco

2: San Jose

3: Los Angeles

4: Fremont

5: Livermore

2: Arizona

6: Page

7: Phoenix

3: Texas

8: Dallas

9: Houston

10: Austin

4: New York

11: New York

5: Nevada

12: Las Vegas

13: Reno

14: Henderson

15: Carson City

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: 101-relationship\_states\_cities\_list.py

 Done! Help Check your code Get a sandbox QA Review

17. From city

**#advanced**

Score: 65.0% (*Checks completed: 100.0%*)

Write a script that lists all City objects from the database hbtn\_0e\_101\_usa

* Your script should take 3 arguments: mysql username, mysql password and database name
* You must use the module SQLAlchemy
* Your script should connect to a MySQL server running on localhost at port 3306
* You must use only one query to the database
* You must use the state relationship to access to the State object linked to the City object
* Results must be sorted in ascending order by cities.id
* Results must be displayed as they are in the example below
* Your code should not be executed when imported

<city id>: <city name> -> <state name>

guillaume@ubuntu:~/0x0F$ ./102-relationship\_cities\_states\_list.py root root hbtn\_0e\_101\_usa

1: San Francisco -> California

2: San Jose -> California

3: Los Angeles -> California

4: Fremont -> California

5: Livermore -> California

6: Page -> Arizona

7: Phoenix -> Arizona

8: Dallas -> Texas

9: Houston -> Texas

10: Austin -> Texas

11: New York -> New York

12: Las Vegas -> Nevada

13: Reno -> Nevada

14: Henderson -> Nevada

15: Carson City -> Nevada

guillaume@ubuntu:~/0x0F$

**No test cases needed**

**Repo:**

* GitHub repository: alx-higher\_level\_programming
* Directory: 0x0F-python-object\_relational\_mapping
* File: 102-relationship\_cities\_states\_list.py

 Done! Help Check your code Get a sandbox QA Review

Copyright © 2023 ALX, All rights reserved.