Python DB-API

Georgia Tech

Database Programming in Python

- DB-API: https://www.python.org/dev/peps/pep-0249/
- SQLite3 is built-in: `import sqlite3`
- MySQL requires third-party library

```
1 $ conda install pymysql
2 ...
3 $ python
4 Python 3.6.0 |Continuum Analytics, Inc.| (default, Dec 23 2016,
      13:19:00)
5 [GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
6 Type "help", "copyright", "credits" or "license" for more
      information.
7 >>> import pymysql
```

Key point: most database APIs, including Python's DB-API, are simply ways of executing SQL statements and getting the results of SQL statements. You can't use DB-API without knowing SQL.

Georgia Tech

Working With Databases

- A connection objects represent a connection to a database

```
connection = pymysql.connect(...)
```

- Cursor object is a stateful pointer to a part of the database

```
cursor = connection.cursor()
```

- SQL Statements are submitted to cursor's `execute()` method
  - `execute` returns the number of rows in the statement's result

```
>>> cursor.execute('insert into table values (%s, %s)', ('field1',
    'field2'))
1
```

  - If the statement was a select, then the cursor points at the first row of the result

```
- The cursor object is an iterator over the results (preferred
   method)
- `fetchall()` , `fetchone()` , etc. return results in Python data
   structures
```

```
>>> cursor.execute('select * from table where column1 = %s',
    ('field1'))
1
>>> for row in cursor: print(row)
{'column1': 'field1', 'column2': 'field'2'}
```

Georgia Tech

Connecting to a MySQL Database

If you configured your MySQL server without a root password, this will work:

```
>>> import pymysql
>>> connection = pymysql.connect(host='localhost',
                                 user='root',
                                 password='',
                                 db='pubs',
                                 charset='utf8mb4',
                                 cursorclass=pymysql.cursors.DictCursor)
```

- Substitute your own root password if you have one
- Notice that the cursor class id `DictCursor` - pymysql will return rows of dabases as dictionaries

Georgia Tech

Inserting Data into a Database Table

```
>>> cursor = connection.cursor()
>>> cursor.execute('insert into author (first_name, last_name) values
    (%s, %s)',
                   ('Jenny', 'McCarthy'))
1
```

To get the primary key of the row you just inserted, use `cursor.lastrowid`. This is especially helpful for getting the id of an `AUTO_INCREMENT` primary key. (Note that the `1` returned by `cursor.execute` is the number of rows affected, not an id.)

```
>>> # Oops! Not the anti-vaxxer, the Lisp inventor
>>> mccarthy_id = cursor.lastrowid
>>> cursor.execute('update author set first_name = %s where author_id =
    %s',
                   ('John', mccarthy_id))
```

Georgia Tech

Executing Queries on a MySQL Database

```
>>> cursor = connection.cursor()
>>> query = "select * from author where last_name = %s"
>>> cursor.execute(query, ('McCarthy'))
2
>>> for row in cursor: print(row)
{'author_id': 1, 'first_name': 'John', 'last_name': 'McCarthy'}
```

- In the query string, use placeholders with same syntax as `%` -based string interpolation
- In call to `cursor.execute`, supply values for placeholders in a tuple.
- After calling `cursor.execute`, cursor is an iterator over the result rows.

Georgia Tech