This document is a summary of the methods defined by the Python Database API Specification V2.0 and used in this course. The full document can be viewed at http://legacy.python.org/dev/peps/pep-0249/.

Introduction

The API was defined to offer a common interface between Python and a database. That means that you can easily switch an application from MySql to Postgress or another database without complex recoding of your application.

But like all good things, there is a caveat. Not all databases offer the same range of features so not all the methods defined by the API are supported by all implementation of the API.

Therefore, this document will confine itself to the methods used by PyMySQL **AND taught in this course.**

Constructor

The constructor establishes the connection to the database. It's syntax is

connect()

Example: conn = connect(parameters) where parameters is database dependent.

To connect to a MySQL database with the PyMySQL implementation, the syntax is as follows:

conn = pymysql. connect(host='localhost', user='root', passwd='password', database='DBNAME')

There are also several optional parameters that can be passed to MySQL with the connect constructor. They are defined in the MySQL documentation (http://dev.mysql.com/doc/.)

Connection Methods

.close()

Closes the database immediately.

Example: conn.close()

.commit()

Commits all pending transactions to the database. If a commit is not done, data is not written to the database. MySQL supports autocommit=TRUE as an optional parameter in the connect constructor.

Example: conn.commit()

.rollback()

This method allows you to roll back all transactions that have not already been committed. It can only work if autocommit has not been set to TRUE. It is useful for recovering from error situations to prevent database corruption.

If you close the database without doing a commit, it will automatically do a rollback on all pending transactions.

Example conn. rollback()

.cursor()

Returns a cursor object using the connection.

Example cur = conn.cursor()

Cursor Methods

.close()

Close the cursor.

Example: cur.close()

.execute(operation [, parameters])

Prepare and execute a MySQL database operation. It allows for optional parameters to be passed.

Example:

```
sql = 'UPDATE inventory SET QtyOnHand = %s where InvID = %s'
cur.execute(sql, [10,  425])
```

.executemany(operation, sequence_of_parameter)

Prepares and executes a database operation against a series of parameter sequences

Example:

```
sql = 'UPDATE inventory SET QtyOnHand = %s where InvID = %s'
cur.execute(sql, ([10,  425], [24, 938], [3, 3910])
```

updates the quantity of inventory on hand for three inventory items with one statement

.fetchone()

Fetches the next row of a MySQL query or returns None if there are no more records to return.

Example:

```
        sql = 'SELECT * from inventory where InvID = %s'
        cur.execute(sql, record)
        row = cur.fetchone()
```

.fetchall()

Fetches all the rows returned by a MySQL query.

Example:

```
       sql = 'SELECT * from inventory'
       cur.execute(sql)
       for row in cur.fetchall():
          print ('<input type="radio" name="record" value="', row[0], '"> SKU: ', row[1], ' - ', row[2],
'<br>', sep='')
```