Using_PyMySQL

November 16, 2016

```
In [1]: # Just to know last time this was run:
        import time
        print time.ctime()
```

```
Wed Nov 16 17:51:50 2016
```

1 J Using PyMySQL to access MySQL databases

This package contains a pure-Python MySQL client library. In this sense, it does not need to have access to mysql header or library, which is the case for the mysqldb package. The goal of PyMySQL is to be a drop-in replacement for MySQLdb and work on CPython, PyPy, IronPython and Jython.

It is installed with "pip install pymysql"

We first import the usual libraries

```
In [2]: %matplotlib inline
        import numpy as np
        import matplotlib.pyplot as plt
```

This is the import of the library used to connect to MySQl database

```
In [3]: import pymysql
```

First you need to connect to a database. In our example, we will use the 3MdB database, which needs a password. https://sites.google.com/site/mexicanmillionmodels/

1.0.1 Connect to the database

```
In [4]: user_password = '********' # ask me for the password :-)
```

```
In [5]: # We create a connector to the database
        connector = pymysql.connect(host='132.248.1.102', port=3306, user='OVN_user
```

1.0.2 Use a cursor to send query and receive results

```
In [6]: # The cursor is used to send and receive the quesries to the databse
        cur = connector.cursor()

In [7]: # Send the query to be executed. It returns the number of lines of the resu
        cur.execute('select * from `lines` limit 15')

Out[7]: 15

In [8]: # get a description of the columns of the query results
        cur.description

Out[8]: ((u'Nl', 8, None, 20, 20, 0, False),
         (u'label', 253, None, 15, 15, 0, True),
         (u'id', 253, None, 20, 20, 0, True),
         (u'lambda', 5, None, 22, 22, 31, True),
         (u'name', 253, None, 40, 40, 0, False),
         (u'used', 3, None, 2, 2, 0, True))

In [9]: # fech all the resulting data into a variable
        lines = cur.fetchall()

In [10]: # close the cursor once used
         cur.close()

In [11]: # the result is in a form of tuple of tuples
         print lines

((1, 'BAC___3646A', 'Bac ', 3646.0, 'BalmHead', 1), (2, 'COUT__3646A', 'cout', 3646


In [12]: # Each element of the first level tuple is a tuple corresponding to a row
         print len(lines)
         print lines[0]

15
(1, 'BAC___3646A', 'Bac ', 3646.0, 'BalmHead', 1)
```

1.0.3 Using a cursor that returns a dictionary

```
In [13]: cur_dic = connector.cursor(pymysql.cursors.DictCursor)

In [14]: cur_dic.execute('select * from `lines` limit 15')

Out[14]: 15

In [15]: lines_dic = cur_dic.fetchall()

In [16]: print lines_dic
```

```
[{u'used': 1, u'Nl': 1, u'name': 'BalmHead', u'label': 'BAC___3646A', u'id': 'Bac '
```

```
In [17]: # Each element of the table is a dictionary corresponding to a row od the
         print lines_dic[0]
```

```
{u'used': 1, u'Nl': 1, u'name': 'BalmHead', u'label': 'BAC___3646A', u'id': 'Bac ',
```

```
In [18]: # One can easily create a new dictionary than hold the data in columns, be
         new_dic = {k:np.array([d[k] for d in lines_dic]) for k in lines_dic[0].key
```

```
In [19]: # The names of the columns are the names use in the database
         new_dic['lambda']
```

```
Out[19]: array([  3.64600000e+03,   3.64600000e+03,   3.64600000e+03,
                  4.86100000e+03,   4.86100000e+03,   6.56300000e+03,
                  4.34000000e+03,   4.10200000e+03,   3.97000000e+03,
                  3.83500000e+03,   1.21600000e+03,   4.05100000e+00,
                  2.62500000e+00,   7.45800000e+00,   5.87600000e+03])
```

```
In [20]: # One can also transform the results into a numpy recarray.
         # First step: create a table from the dictionnary
         lines_tab = [e.values() for e in lines_dic]
         lines_tab
```

```
Out[20]: [[1, 1, 'BalmHead', 'BAC___3646A', 'Bac ', 3646.0],
          [1, 2, 'OutwardBalmPeak', 'COUT__3646A', 'cout', 3646.0],
          [1, 3, 'ReflectedBalmPeak', 'CREF__3646A', 'cref', 3646.0],
          [1, 4, 'H I 4861', 'H__1__4861A', 'H  1', 4861.0],
          [1, 5, 'H I 4861', 'TOTL__4861A', 'TOTL', 4861.0],
          [1, 6, 'H I 6563', 'H__1__6563A', 'H  1', 6563.0],
          [1, 7, 'H I 4340', 'H__1__4340A', 'H  1', 4340.0],
          [1, 8, 'H I 4102', 'H__1__4102A', 'H  1', 4102.0],
          [1, 9, 'H I 3970', 'H__1__3970A', 'H  1', 3970.0],
          [1, 10, 'H I 3835', 'H__1__3835A', 'H  1', 3835.0],
          [1, 11, 'H I 1216', 'H__1__1216A', 'H  1', 1216.0],
          [1, 12, 'H I 4.051m', 'H__1_4051M', 'H  1', 4.051],
          [1, 13, 'H I 2.625m', 'H__1_2625M', 'H  1', 2.625],
          [1, 14, 'H I 7.458m', 'H__1_7458M', 'H  1', 7.458],
          [1, 15, 'He I 5876', 'HE_1__5876A', 'He 1', 5876.0]]
```

```
In [21]: # Second step: transform the table into a numpy recarray, using the names
         res = np.rec.fromrecords(lines_tab, names = lines_dic[0].keys())
```

```
In [22]: res
```

```
Out[22]: rec.array([(1, 1, 'BalmHead', 'BAC___3646A', 'Bac ', 3646.0),
          (1, 2, 'OutwardBalmPeak', 'COUT__3646A', 'cout', 3646.0),
```

```
 (1, 3, 'ReflectedBalmPeak', 'CREF__3646A', 'cref', 3646.0),
 (1, 4, 'H I 4861', 'H__1__4861A', 'H  1', 4861.0),
 (1, 5, 'H I 4861', 'TOTL__4861A', 'TOTL', 4861.0),
 (1, 6, 'H I 6563', 'H__1__6563A', 'H  1', 6563.0),
 (1, 7, 'H I 4340', 'H__1__4340A', 'H  1', 4340.0),
 (1, 8, 'H I 4102', 'H__1__4102A', 'H  1', 4102.0),
 (1, 9, 'H I 3970', 'H__1__3970A', 'H  1', 3970.0),
 (1, 10, 'H I 3835', 'H__1__3835A', 'H  1', 3835.0),
 (1, 11, 'H I 1216', 'H__1__1216A', 'H  1', 1216.0),
 (1, 12, 'H I 4.051m', 'H__1_4051M', 'H  1', 4.051),
 (1, 13, 'H I 2.625m', 'H__1_2625M', 'H  1', 2.625),
 (1, 14, 'H I 7.458m', 'H__1_7458M', 'H  1', 7.458),
 (1, 15, 'He I 5876', 'HE_1__5876A', 'He 1', 5876.0)],
          dtype=[(u'used', '<i8'), (u'Nl', '<i8'), (u'name', 'S17'), (u'la
```

```
In [23]: res['lambda']
```

```
Out[23]: array([  3.64600000e+03,   3.64600000e+03,   3.64600000e+03,
                  4.86100000e+03,   4.86100000e+03,   6.56300000e+03,
                  4.34000000e+03,   4.10200000e+03,   3.97000000e+03,
                  3.83500000e+03,   1.21600000e+03,   4.05100000e+00,
                  2.62500000e+00,   7.45800000e+00,   5.87600000e+03])
```

1.0.4 Example of plotting the result of a query

```
In [24]: # Send the query
         N = cur_dic.execute('select O__3__5007A, N__2__6584A, H__1__6563A, oxygen
```

```
In [25]: print N
```

```
7854
```

```
In [26]: # obtain the results as a dictionnary
         res = cur_dic.fetchall()
```

```
In [27]: # transform the disctionary into a recarray
         data = np.rec.fromrecords([e.values() for e in res], names = res[0].keys()
```

```
In [28]: # check the data
         data
```

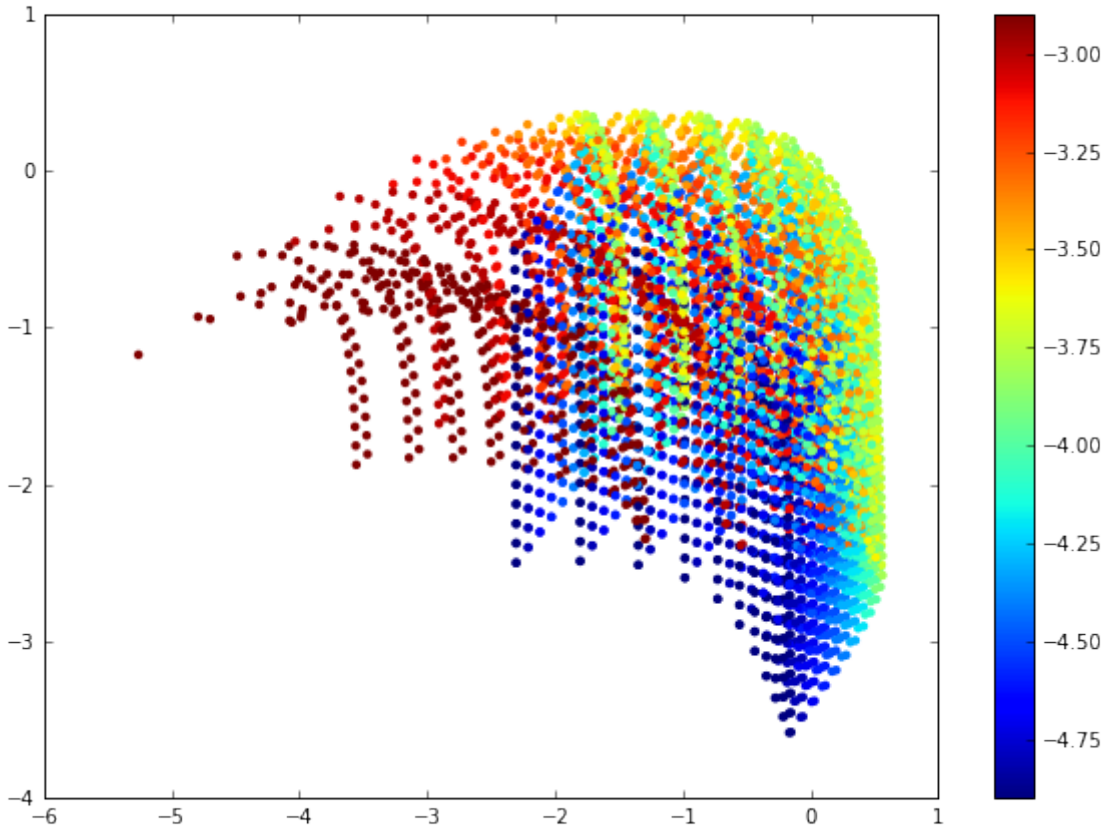
```
Out[28]: rec.array([(1.13306243836e+58, 8.465943086e+58, -3.1, 3.15741653467e+58),
          (3.42011987292e+59, 3.82678097448e+59, -4.7, 1.96658128904e+58),
          (1.9919317079e+55, 2.95364632532e+58, -2.9, 8.79993595982e+57), ...,
          (1.75269190656e+60, 5.79356475056e+59, -3.7, 5.08981089096e+58),
          (1.37202884837e+60, 5.15976659165e+59, -4.1, 3.20261785304e+57),
          (1.52244147812e+60, 5.27404255136e+59, -4.0, 3.89222406128e+58)],
                   dtype=[(u'O__3__5007A', '<f8'), (u'H__1__6563A', '<f8'), (u'oxyg
```

```
In [29]: data['O__3__5007A']
```

```
Out[29]: array([  1.13306244e+58,   3.42011987e+59,   1.99193171e+55, ...,
                  1.75269191e+60,   1.37202885e+60,   1.52244148e+60])
```

```
In [30]: # Plot the results, using a column as color code
         fig, ax = plt.subplots(figsize=(10,7))
         scat = ax.scatter(np.log10(data['O__3__5007A'] / data['H__1__6563A']), np.
                           c=data['oxygen'], edgecolor='none')
         fig.colorbar(scat)
```

```
Out[30]: <matplotlib.colorbar.Colorbar at 0x1106dc7d0>
```




```
In [31]: # Disconnect cursor and connector
         cur_dic.close()
         connector.close()
```

1.0.5 Using pandas library

```
In [32]: import pandas as pd
         import pymysql
```

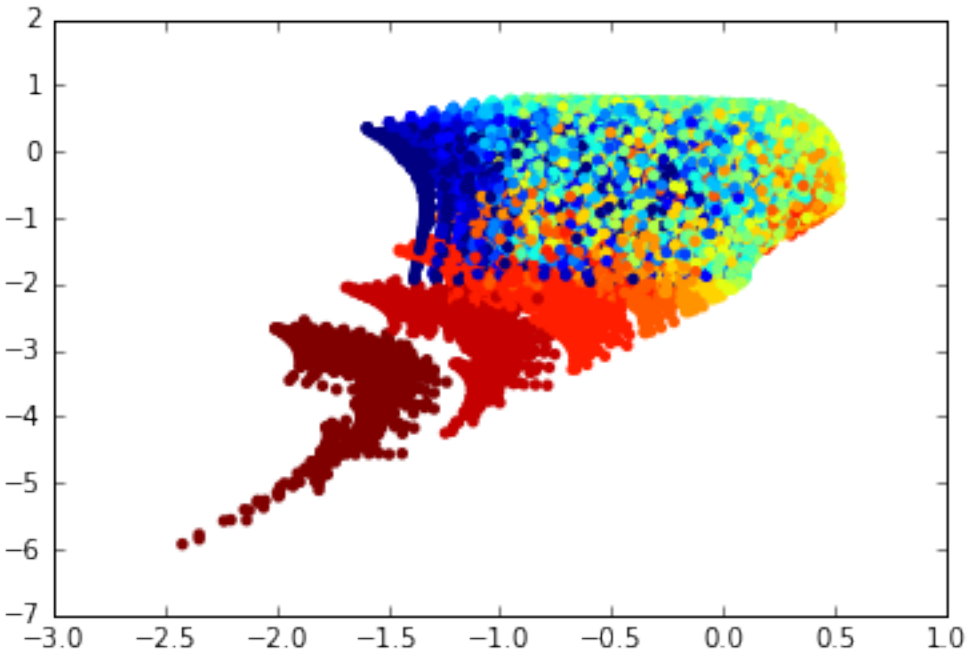
```
import matplotlib.pyplot as plt

co = pymysql.connect(host='132.248.1.102', db='3MdB', user='OVN_user', pas
res = pd.read_sql("select log10(N__2__6584A/H__1__6563A) as n2, log10(O__3
co.close()
```

In [33]:
```
print(len(res))
```

```
36193
```

In [34]:
```
plt.scatter(res['n2'], res['o3'], c=res['O'], edgecolor='None')
```

Out[34]: <matplotlib.collections.PathCollection at 0x119b5b290>



In [ ]: