**[python：数据库连接操作入门](https://www.cnblogs.com/jyroy/p/9429897.html)**

**模块**[**#**](https://www.cnblogs.com/jyroy/p/9429897.html#idx_0)

1 import pymssql,pyodbc

**模块说明[#](https://www.cnblogs.com/jyroy/p/9429897.html" \l "idx_1)**

　　pymssql和pyodbc模块都是常用的用于SQL Server、MySQL等数据库的连接及操作的模块，当然一些其他的模块也可以进行相应的操作，类似adodbapi、mssql、mxODBC等，我们在实际用的时候选择其中一个模块就好，对于每一个模块都有相应的支持版本和支持平台，大家可以自行查阅文档<https://wiki.python.org/moin/SQL%20Server>

**模块安装**[**#**](https://www.cnblogs.com/jyroy/p/9429897.html#idx_2)

1 pip install pymssql

　　关于pip的使用我在我的另一篇博客里提到了https://www.cnblogs.com/jyroy/p/9410593.html

**模块使用**

　　我们利用python来进行数据库的操作，那么第一步就应该是连接数据库，这里我们用pymssql模块中的connect方法连连接，在pyodbc模块中同样也是利用connect方法。

* 使用connect创建连接对象
* connect.cursor创建游标对象，SQL语句的执行基本都在游标上进行
* cursor.executeXXX方法执行SQL语句，cursor.fetchXXX获取查询结果等
* 调用close方法关闭游标cursor和数据库连接

**pymssql模块连接**

'''

pymssql模块连接SQL Server数据库

'''

import pymssql

'''格式1'''

host= "XXXXXXXXXXXX" # 数据库服务器名称或IP

user = "test"

password = "123"

database = "test"

conn = pymssql.connect(host, user, password, database)

'''格式2'''

conn = pymssql.connect(host='XXXXXXXXXXXX', user="test", password="123", database="test")

**pyodbc模块连接**

import pyodbc

conn = pyodbc.connect(r'DRIVER={SQL Server Native Client 11.0};SERVER=test;DATABASE=test;UID=user;PWD=password')

　　不同的SQL server版本对应的DRIVER字段不同。对应关系如下：

* {SQL Server} - released with SQL Server 2000
* {SQL Native Client} - released with SQL Server 2005 (also known as version 9.0)
* {SQL Server Native Client 10.0} - released with SQL Server 2008
* {SQL Server Native Client 11.0} - released with SQL Server 2012

　　使用pyodbc需要安装微软官方的Native Client（没有安装会报错IM002)，安装SQL server management studio会自动附带安装（控制面板里可以看到安装的版本）。如果没有安装过需要在[https://msdn.microsoft.com/en-us/data/ff658533.aspx](https://msdn.microsoft.com/en-us/data/ff658533.aspx" \t "_blank)下载安装(sqlncli.msi)。建议选择与远程数据库版本相对应的Native Client。如果本地安装的Native Client是高版本，则DRIVER={SQL Server Native Client 11.0}需要填写的是本地的高版本。

**获取数据库内容**[**#**](https://www.cnblogs.com/jyroy/p/9429897.html#idx_3)

　　这里涉及到**游标的使用**（游标大家有不理解的可以参考一下这个大佬的博客<https://www.cnblogs.com/selene/p/4480328.html>）

1 '''

2 游标的使用

3 '''

4 cursor\_1 = conn.cursor() #获取游标

5

6 cursor\_1.execute("select Sno from student") #执行语句

7

8 print (cursor\_1.fetchone()) #结果是元组,fetchone()获取查询结果

9

**fetchone()**：返回单个的元组，也就是一条记录(row)，如果没有结果 则返回 None

**fetchall()** ：返回多个元组，即返回多个记录(rows),如果没有结果 则返回 ()

1 '''

2 fetchall()的使用

3 '''

4 cursor\_2 = conn.cursor()

5

6 cursor\_2.execute("select Sno, Sname from students")

7

8 rows = cursor\_2.fetchall()

9

10 for row in rows:

11 print(row.Sno, row.Sname)

[复制代码](javascript:void(0);)

　　由于execute返回的是cursor本身，所以如果你需要一次直接获取所有内容可以直接使用cursor本身来获取

1 cursor.execute("select Sno, Sname from students"):

2

3 for row in cursor:

4 print(row.Sno, row.Sname )

　　关于游标这里还存在一个要向大家专门说明的地方，

　　就是**一个连接一次只能有一个游标的查询处于活跃状态**，具体什么意思大家可以看下面的代码。

[复制代码](javascript:void(0);)

1 cursor\_1 = conn.cursor() #获取游标

2 cursor\_1.execute("select Sno from student") #执行语句

3

4 cursor\_2 = conn.cursor()

5 cursor\_2.execute("select \* from student where Sno=1")

6

7 print (cursor\_1.fetchall()) #显示的是cursor\_2的查询结果

8 print (cursor\_2.fetchall()) #不显示任何结果

[复制代码](javascript:void(0);)

　　提供一个解决的办法

[复制代码](javascript:void(0);)

1 '''解决上述问题'''

2

3 cursor\_1.execute("select Sno from student")

4 cursor\_list\_1 = cursor\_1.fetchall() #用list把结果存储下来

5

6 cursor\_2.execute("select \* from student where Sno=1")

7 cursor\_list\_2 = cursor\_2.fetchall()

8

9 print (cursor\_list\_1) #通过print list来获取cursor\_1的结果

10 print (cursor\_list\_2)

[复制代码](javascript:void(0);)

 　　在游标的正常使用中游标获取的查询结果，一行为一个元组。

　　我们在实际使用中可以根据需求，**用 as\_dict 方法返回一个字典变量**，其中字典的Key为数据表的列名

[复制代码](javascript:void(0);)

1 '''游标返回行为字典变量'''

2

3 cursor\_3 = conn.cursor(as\_dict=True) #创建游标时指定as\_dict参数来使游标返回字典变量

4 cursor\_3.execute("select \* from student")

5

6 for row in cursor\_3:

7 print('ID=%d' % row['Sno']) #键名为列表的列名 {Sno:ID}

8

9 print (cursor\_3.fetchone())

10

11 conn.close() #调用close方法关闭游标cursor和数据库连接

[复制代码](javascript:void(0);)

　　大家如果觉得上面的代码写起来看上去太长，给大家提供一个代码量小的数据库操作解决办法

　　就是 **with语句，即上下文管理器**，它的好处一个是代码的集成度高，一个是省去显示的调用close方法关闭连接和游标

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

1 '''

2 使用with语句（上下文管理器）

3 '''

4

5 '''通过使用with语句来省去显示的调用close方法关闭连接和游标'''

6 with pymssql.connect(host='LAPTOP-3OTJHAG9', user="sa", password="123", database="TEST") as conn:

7

8 #相当于 conn = pymssql.connect(host='XXXXXXXXXXXX', user="test", password="123", database="test")

9

10 with conn.cursor(as\_dict=True) as cursor\_4:

11 cursor\_4.execute('select \* from student')

12 for row in cursor\_4:

13 print ('ID=%d' %(row['Sno']))

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

**关于数据库的增删改[#](https://www.cnblogs.com/jyroy/p/9429897.html" \l "idx_4)**

　　增删改数据库的内容也是直接传递SQL语句给execute方法。但要注意运行之后需要用**commit提交变更**

1 cursor.execute("insert into students(id, name) values ('123', 'My name')")

2 conn.commit()

3 cursor.execute("delete from studnts where id=‘123’")  
  
4 conn.commit()

关于pymssql模块，还有不明白的大家可以看下官网http://www.pymssql.org/