前言

广大Python+Java网友好，最近锋哥上线了 www.python222.com 网站，后面主要分享python相关干货学习资源，以及录制python相关干货课程，希望对大家有帮助。

关于python课程，大伙先学习python3 7天基础课程（已经更新完）：https://www.bilibili.com/video/BV1o84y1Z7J1/

课程简介

本课程讲解python通过pymysql模块技术操作mysql数据库，课程全免费，后期会继续推出进阶，高级课程，敬请期待。

高清视频+源码+配套文档获取

关注 **python222** 公众号，回复 **888**

获取本套课程的高清视频+源码+配套文档

进本课程专属微信交流群

加上面锋哥微信，**java3459** ，备注暗号：**222**

备用微信： **java9266**

即可进入Python222课程的专属技术交流群。

作者简介

我叫曹锋，网名：python222_小锋 江苏南通人，12年毕业于江苏师范大学，计算机与科学技术专业，12年Java+Python老兵，资深讲师，技术自媒体人，南通小锋网络科技有限公司光杆司令员，司令部：www.java1234.vip www.python222.com 还有一个网站：www.java1234.com

爆丑照，美颜后，依然很丑。

pymysql简介以及安装

前面基础课程介绍了使用文件来保存数据，这种方式虽然简单、易用，但只适用于保存一些格式简单、数据量不太大的数据。对于数据量巨大且具有复杂关系的数据，当然还是推荐使用数据库进行保存。

大伙先需要学习下mysql数据库基础，可以学习我的mysql课程：http://vip.java1234.com/course/87

推荐大家用mysql5.7或者mysql8，我们课程演示用mysql5.7

python操作mysql数据库有好几个第三方模块，比较出名是pymysql和mysql-connector-python，用法差不多。

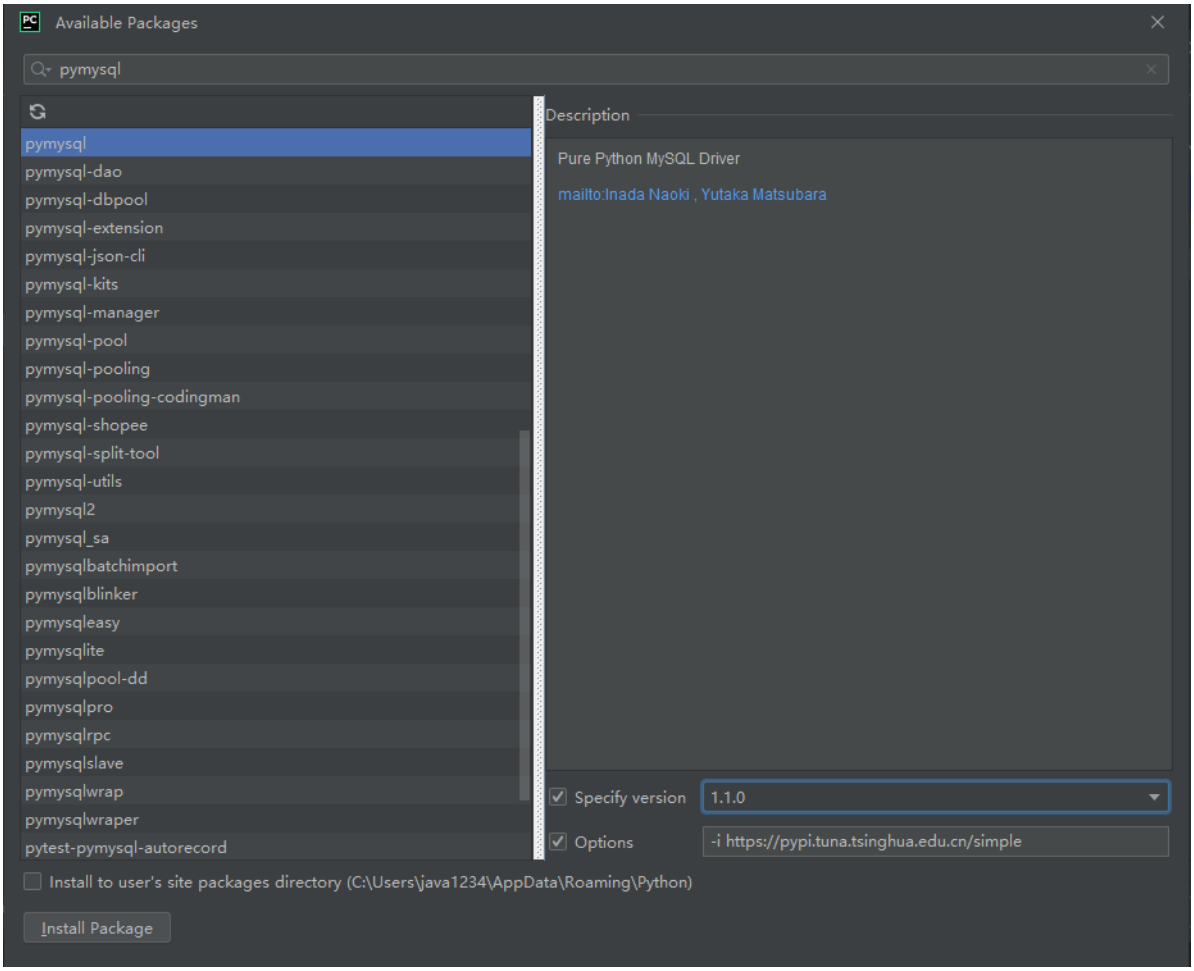
mysql-connector-python，这个是mysql官方提供的驱动模块。

我们以pymysql模块为例，操作mysql5.7；

pymsyql安装：

```
pip install pymysql -i https://pypi.tuna.tsinghua.edu.cn/simple
```

或者直接用pycharm安装，



pymysql创建数据库连接

通过pymysql的Connection类创建数据库连接，用完一定要关闭连接。

```
from pymysql import Connection

# 创建数据库连接
con = Connection(
    host="localhost",  # 主机名
    port=3306,  # 端口
    user="root",  # 账户
    password="123456"  # 密码
)
print(type(con))
print(con.get_host_info())
print(con.get_server_info())

# 关闭连接
con.close()
```

运行输出：

```
<class 'pymysql.connections.Connection'>
socket localhost:3306
5.7.18-log
```

改进，可能会出现异常，我们加上 try except finally

```
from pymysql import Connection

con = None

try:
    # 创建数据库连接
    con = Connection(
        host="localhost",  # 主机名
        port=3306,  # 端口
        user="root",  # 账户
        password="123456"  # 密码
    )
    print(type(con))
    print(con.get_host_info())
    print(con.get_server_info())
except Exception as e:
    print("异常：", e)
finally:
    if con:
        # 关闭连接
        con.close()
```

pymysql执行DDL语句

pymysql对操作mysql *DDL*(Data Definition Language) 数据库模式定义语言，提供了很好的支持。

连接MySQL数据库后，可以使用cursor()方法创建一个游标对象。游标对象用于执行MySQL语句并返回结果。

建表DDL：

```
"""
    pymysql执行DDL语句
"""
from pymysql import Connection

con = None

try:
    # 创建数据库连接
    con = Connection(
        host="localhost",  # 主机名
        port=3306,  # 端口
        user="root",  # 账户
        password="123456",  # 密码
        database="db_python"  # 指定操作的数据库
    )
    # 创建游标对象
    cursor = con.cursor()

    # 定义建表sql语句
    sql = """CREATE TABLE `t_student3` (
      `id` int(11) NOT NULL AUTO_INCREMENT,
      `name` varchar(10) DEFAULT NULL,
      `age` int(11) DEFAULT NULL,
      PRIMARY KEY (`id`)
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8
    """

    # 选择要操作的数据库
    # con.select_db("db_python")

    # 使用游标对象，执行sql
    cursor.execute(sql)
except Exception as e:
    print("异常", e)
finally:
    if con:
        # 关闭连接
        con.close()
```

```
#对表中的字段进行增，删，改的操作
#格式：alter table 表名 add/drop/modify/change [column] .......

#向表中添加字段:alter table 表名 add [column] 字段名 字段类型;
ALTER TABLE stu ADD COLUMN age INT;

#删除表中的字段:alter table 表名 drop [column] 字段名:
ALTER TABLE stu DROP age;
```

```
#修改字段的名字:alter table 表名 change [column] 旧字段名 新字段名 字段类型;
ALTER TABLE stu CHANGE COLUMN sname s2name VARCHAR(20);

#修改字段的类型:alter table 表名 modify [column] 字段名 字段类型;
ALTER TABLE stu MODIFY id VARCHAR(20);


#修改表的名字: alter table 旧表名 rename to 新表名:
ALTER TABLE stu4 RENAME TO stu5;
```

增加字段

```
ALTER TABLE t_student3 ADD COLUMN sex VARCHAR(2)
```

修改字段

```
ALTER TABLE t_student3 MODIFY NAME VARCHAR(12)
```

```
from pymysql import Connection

con = None

try:
    # 创建数据库连接
    con = Connection(
        host="localhost",  # 主机名
        port=3306,  # 端口
        user="root",  # 账户
        password="123456"  # 密码
        # database="db_python"  # 指定操作的数据库
    )
    # 创建游标对象
    cursor = con.cursor()

    # 定义建表sql语句
    sql = "ALTER TABLE t_student2 MODIFY NAME VARCHAR(12)"

    # 选择要操作的数据库
    con.select_db("db_python")

    # 使用游标对象，执行sql
    cursor.execute(sql)
except Exception as e:
    print("异常", e)
finally:
    if con:
        # 关闭连接
        con.close()
```

pymysql执行DML语句

MySQL 数据库模块同样可以使用游标的execute()方法执行DML（Data Manipulation Language, 数据操纵语言）的 insert、update、delete语句，对数据库进行插入、修改和删除数据操作。

pymysql执行select查询操作

```
con = None

try:
    # 创建数据库连接
    con = Connection(
        host="localhost",  # 主机名
        port=3306,  # 端口
        user="root",  # 账户
        password="123456",  # 密码
        database="db_python"  # 指定操作的数据库
    )
    # 获取游标对象
    cursor = con.cursor()
    # 使用游标对象，执行sql语句
    cursor.execute("select * from t_student")
    # 获取查询所有结果
    result = cursor.fetchall()
    print(type(result), result)
    for row in result:
        print(row)
except Exception as e:
    print("异常: ", e)
finally:
    if con:
        # 关闭连接
        con.close()
```

pymysql执行insert插入操作

执行修改操作，需要通过Connection对象调用commit()方法确认提交，或者 构造方法里面，autocommit设置True，自动提交

```
from pymysql import Connection

con = None

try:
    # 创建数据库连接
    con = Connection(
        host="localhost",  # 主机名
        port=3306,  # 端口
        user="root",  # 账户
        password="123456",  # 密码
        database="db_python",  # 指定操作的数据库
        autocommit=True  # 设置自动提交
```

```
    )
    # 获取游标对象
    cursor = con.cursor()
    # 使用游标对象，执行sql语句
    cursor.execute("insert into t_student values(null,'赵六2',25)")
    # 获取主键
    print("主键id=", con.insert_id())
    # 确认提交
    # con.commit()
except Exception as e:
    print("异常：", e)
finally:
    if con:
        # 关闭连接
        con.close()
```

pymysql执行update更新操作

执行update操作，雷同前面的insert操作

```
from pymysql import Connection

con = None

try:
    # 创建数据库连接
    con = Connection(
        host="localhost",  # 主机名
        port=3306,  # 端口
        user="root",  # 账户
        password="123456",  # 密码
        database="db_python",  # 指定操作的数据库
        autocommit=True  # 设置自动提交
    )
    # 获取游标对象
    cursor = con.cursor()
    # 使用游标对象，执行sql语句
    cursor.execute("UPDATE t_student SET age=19 WHERE id=5")
    # 确认提交
    # con.commit()
except Exception as e:
    print("异常：", e)
finally:
    if con:
        # 关闭连接
        con.close()
```

pymysql执行delete删除操作

执行delete操作，雷同前面的update操作

```
from pymysql import Connection

con = None

try:
    # 创建数据库连接
    con = Connection(
        host="localhost",  # 主机名
        port=3306,  # 端口
        user="root",  # 账户
        password="123456",  # 密码
        database="db_python",  # 指定操作的数据库
        autocommit=True  # 设置自动提交
    )
    # 获取游标对象
    cursor = con.cursor()
    # 使用游标对象，执行sql语句
    cursor.execute("delete from t_student  WHERE id=5")
    # 确认提交
    # con.commit()
except Exception as e:
    print("异常：", e)
finally:
    if con:
        # 关闭连接
        con.close()
```

pymysql调用存储过程

我们首先创建一个简单的存储过程

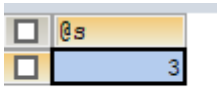
```
DELIMITER //

CREATE PROCEDURE test_add(m INT,n INT, OUT result INT)
BEGIN
SET result=m+n;

END; //
```

测试：

```
SET @s=0;
CALL test_add(1,2,@s);
SELECT @s
```



Pymysql调用存储过程实现：

```
from pymysql import Connection

con = None

try:
    # 创建数据库连接
    con = Connection(
        host="localhost",  # 主机名
        port=3306,  # 端口
        user="root",  # 账户
        password="123456",  # 密码
        database="db_python",  # 指定操作的数据库
        autocommit=True  # 设置自动提交
    )
    # 获取游标对象
    cursor = con.cursor()
    # 使用游标对象，调用存储过程
    cursor.execute("CALL test_add(1,2,@s);")
    cursor.execute("select @s;")
    result = cursor.fetchone()
    print(result[0])
    # 确认提交
    # con.commit()
except Exception as e:
    print("异常：", e)
finally:
    if con:
        # 关闭连接
        con.close()
```