17丨如何使用Python操作MySQL？

2019-07-19 陈旸

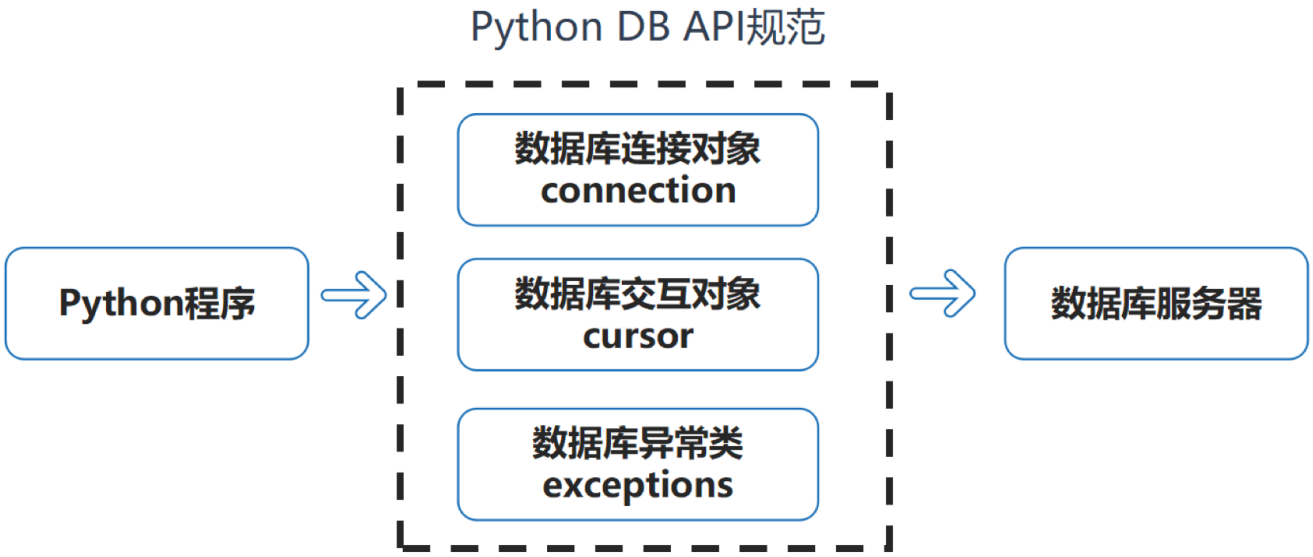
我们之前都是直接在DBMS里面进行SQL的操作，实际上我们还可以通过后端语言对DBMS进行访问以及进行相应的操作，这样更具有灵活性，可以实现一些较为复杂的操作。作为一个后端开发人员，掌握一些SQL技术是必须的；作为一个数据库管理人员，了解后端语言如何开发和管理数据库也是很有必要的。

今天我以Python为例，讲解下如何对MySQL数据库进行操作。你需要掌握以下几个方面的内容：

1. Python的DB API规范是什么，遵守这个规范有什么用？
2. 基于DB API，MySQL官方提供了驱动器mysql-connector，如何使用它来完成对数据库管理系统的操作？
3. CRUD是最常见的数据库的操作，分别对应数据的增加、读取、修改和删除。在掌握了mysql-connector的使用方法之后，如何完成对数据表的CRUD操作？

Python DB API规范

Python可以支持非常多的数据库管理系统，比如MySQL、Oracle、SQL Server和PostgreSQL等。为了实现对这些DBMS的统一访问，Python需要遵守一个规范，这就是DB API规范。我在下图中列出了DB API规范的作用，这个规范给我们提供了数据库对象连接、对象交互和异常处理的方式，为各种DBMS提供了统一的访问接口。这样做的好处就是如果项目需要切换数据库，Python层的代码移植会比较简单。



我们在使用Python对DBMS进行操作的时候，需要经过下面的几个步骤：

1. 引入API模块；
2. 与数据库建立连接；
3. 执行SQL语句；
4. 关闭数据库连接。

如何使用mysql-connector

使用Python对数据库进行访问需要基于DB API规范，这里有不少库供我们选择，比如MySQLdb、mysqlclient、PyMySQL、peewee和SQLAlchemy等。今天我讲解的是mysql-connector，它是MySQL 官方提供的驱动器，用来给后端语言，比如Python提供连接。

下面我们看下如何用Python使用mysql-connector，以完成数据库的连接和使用。

首先安装mysql-connector。在使用前，你需要先使用下面这句命令进行安装：

```
pip install mysql-connector
```

在安装之后，你可以创建数据库连接，然后查看下数据库的版本号，来验证下数据库是否连接成功。代码如下：

```
# -*- coding: UTF-8 -*-
import mysql.connector
# 打开数据库连接
db = mysql.connector.connect(
        host="localhost",
        user="root",
        passwd="XXX", # 写上你的数据库密码
        database='wucai',
        auth_plugin='mysql_native_password'
)
# 获取操作游标
cursor = db.cursor()
# 执行SQL语句
cursor.execute("SELECT VERSION()")
# 获取一条数据
data = cursor.fetchone()
print("MySQL版本: %s " % data)
# 关闭游标&数据库连接
cursor.close()
db.close()
```

运行结果：

```
MySQL版本: 8.0.13
```

上面这段代码中有两个重要的对象你需要了解下，分别是Connection和Cursor。

Connection就是对数据库的当前连接进行管理，我们可以通过它来进行以下操作：

1. 通过指定host、user、passwd和port等参数来创建数据库连接，这些参数分别对应着数据库IP地址、用户名、密码和端口号；
2. 使用db.close()关闭数据库连接；
3. 使用db.cursor()创建游标，操作数据库中的数据；
4. 使用db.begin()开启事务；
5. 使用db.commit()和db.rollback()，对事务进行提交以及回滚。

当我们通过cursor = db.cursor()创建游标后，就可以通过面向过程的编程方式对数据库中的数据

进行操作：

1. 使用cursor.execute(query_sql)，执行数据库查询；
2. 使用cursor.fetchone()，读取数据集中的一条数据；
3. 使用cursor.fetchall()，取出数据集中的所有行，返回一个元组tuples类型；
4. 使用cursor.fetchmany(n)，取出数据集中的多条数据，同样返回一个元组tuples；
5. 使用cursor.rowcount，返回查询结果集中的行数。如果没有查询到数据或者还没有查询，则结果为-1，否则会返回查询得到的数据行数；
6. 使用cursor.close()，关闭游标。

对数据表进行增删改查

了解了Connection和Cursor的使用方式之后，我们来看下如何来对heros数据表进行CRUD的操作，即增加、读取、更新和删除。

首先是增加数据。

假设我们想在player表中增加一名新球员，姓名为“约翰·科林斯”，球队ID为1003（即亚特兰大老鹰），身高为2.08m。代码如下：

```
# 插入新球员
sql = "INSERT INTO player (team_id, player_name, height) VALUES (%s, %s, %s)"
val = (1003, "约翰-科林斯", 2.08)
cursor.execute(sql, val)
db.commit()
print(cursor.rowcount, "记录插入成功。")
```

我们使用cursor.execute来执行相应的SQL语句，val为SQL语句中的参数，SQL执行后使用db.commit()进行提交。需要说明的是，我们在使用SQL语句的时候，可以向SQL语句传递参数，这时SQL语句里要统一用（%s）进行占位，否则就会报错。不论插入的数值为整数类型，还是浮点类型，都需要统一用（%s）进行占位。

另外在用游标进行SQL操作之后，还需要使用db.commit()进行提交，否则数据不会被插入。

然后是读取数据。我们来看下数据是否被插入成功，这里我们查询下身高大于等于2.08m的球员都有哪些，代码如下：

```
# 查询身高大于等于2.08的球员
sql = 'SELECT player_id, player_name, height FROM player WHERE height>=2.08'
cursor.execute(sql)
data = cursor.fetchall()
for each_player in data:
  print(each_player)
```

运行结果：

```
(10003, '安德烈-德拉蒙德', 2.11)
(10004, '索恩-马克', 2.16)
(10009, '扎扎-帕楚里亚', 2.11)
(10010, '乔恩-洛伊尔', 2.08)
(10011, '布雷克-格里芬', 2.08)
(10015, '亨利-埃伦森', 2.11)
(10023, '多曼塔斯-萨博尼斯', 2.11)
(10024, '迈尔斯-特纳', 2.11)
(10032, 'TJ-利夫', 2.08)
(10033, '凯尔-奥奎因', 2.08)
(10037, '伊凯·阿尼博古', 2.08)
(10038, '约翰-科林斯', 2.08)
```

你能看到球员约翰·科林斯被正确插入。

那么如何修改数据呢？

假如我想修改刚才插入的球员约翰·科林斯的身高，将身高修改成2.09，代码如下：

```
# 修改球员约翰-科林斯
sql = 'UPDATE player SET height = %s WHERE player_name = %s'
val = (2.09, "约翰-科林斯")
cursor.execute(sql, val)
db.commit()
print(cursor.rowcount, "记录被修改。")
```

最后我们看下如何删除约翰·科林斯这个球员的数据，代码如下：

```
sql = 'DELETE FROM player WHERE player_name = %s'
val = ("约翰-科林斯",)
cursor.execute(sql, val)
db.commit()
print(cursor.rowcount, "记录删除成功。")
```

最后都执行完了，我们来关闭游标和数据库的连接，使用以下代码即可：

```
cursor.close()
db.close()
```

针对上面的操作过程，你可以模拟下数据的CRUD操作，但有几点你需要注意。

1.打开数据库连接以后，如果不再使用，则需要关闭数据库连接，以免造成资源浪费。
2.在对数据进行增加、删除和修改的时候，可能会出现异常，这时就需要用try...except捕获异常信息。比如针对插入球员约翰·科林斯这个操作，你可以写成下面这样：

```
import traceback
try:
  sql = "INSERT INTO player (team_id, player_name, height) VALUES (%s, %s, %s)"
  val = (1003, "约翰-科林斯", 2.08)
  cursor.execute(sql, val)
  db.commit()
  print(cursor.rowcount, "记录插入成功。")
except Exception as e:
  # 打印异常信息
  traceback.print_exc()
  # 回滚
  db.rollback()
finally:
  # 关闭数据库连接
  db.close()
```

运行结果告诉我们记录插入成功。

3.如果你在使用mysql-connector连接的时候，系统报的错误为authentication plugin caching_sha2，这时你需要下载最新的版本更新来解决，点击这里进行更新。

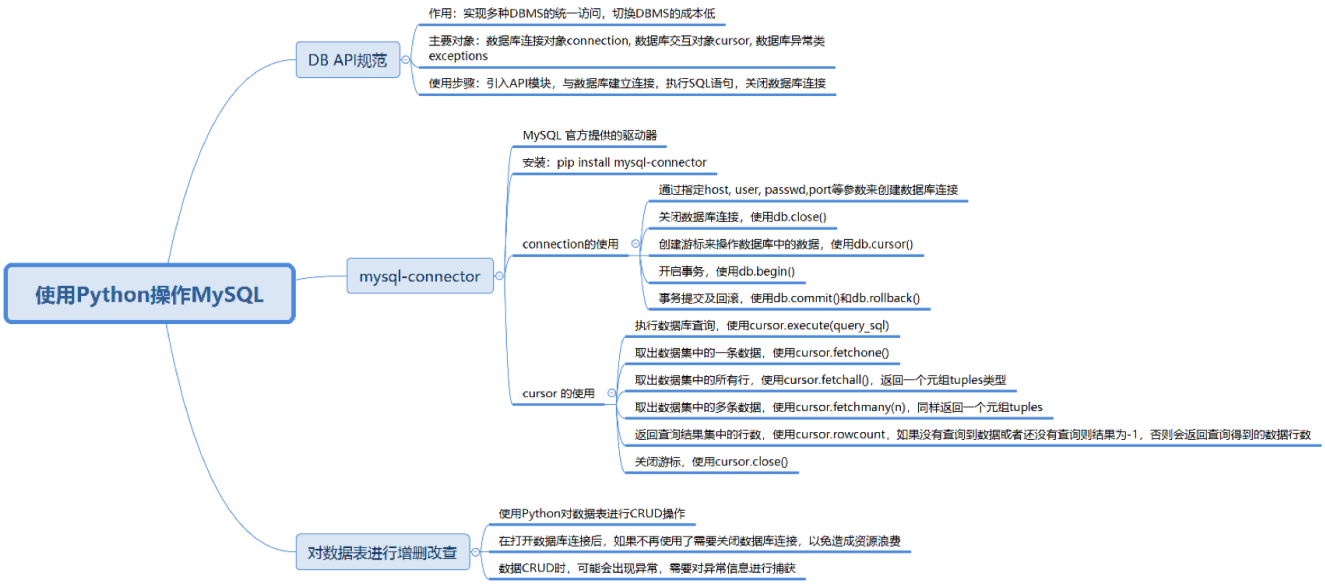
总结

我今天讲解了如何使用Python来操作MySQL，这里我们使用的是官方提供的mysql-connector，当然除了它之外，还有很多库可以进行选择。

在使用基于DB API规范的协议时，重点需要掌握Connection和Cursor这两个对象，Connection就是对数据库的连接进行管理，而Cursor是对数据库的游标进行管理，通过它们，我们可以执行具体的SQL语句，以及处理复杂的数据。

用Python操作MySQL，还有很多种姿势，mysql-connector只是其中一种，实际上还有另外一种方式，就是采用ORM框架。ORM的英文是Object Relational Mapping，也就是采用对象关系映射的模式，使用这种模式可以将数据库中各种数据表之间的关系映射到程序中的对象。这种模式可以屏蔽底层的数据库的细节，不需要我们与复杂的SQL语句打交道，直接采用操作对象的形式操作就可以。

不过如果应用数据实体少，其实没有必要使用ORM框架，针对少量对象的管理，自己实现起来也很简单，比如本篇文章中我讲到的采用官方提供的mysql-connector驱动的方式来实现CRUD。引入一个框架的学习成本很高，代码膨胀也很厉害，所以如果是相对简单的操作，完全可以自己动手来实现。



使用Python对数据库进行操作，关键在于实战，所以这里我出一个练习题。请你使用Python对heros表中最大生命值大于6000的英雄进行查询，并且输出相应的属性值。

欢迎在评论区写下你的答案，我会与你一起交流。也欢迎把这篇文章分享给你的朋友或者同事，与它们一起交流一下。

精选留言

ttttt

7

```
import json
import traceback
import mysql.connector

# 读取数据库链接配置文件
with open('mysql.json', encoding='utf-8') as con_json:
con_dict = json.load(con_json)

# 打开数据库链接
db = mysql.connector.connect(
host=con_dict['host'],
user=con_dict['user'],
passwd=con_dict['passwd'],
database=con_dict['database'],
auth_plugin=con_dict['auth_plugin'],
)

# 获取操作游标
cursor = db.cursor()
try:
```

```
sql = 'SELECT id, name, hp_max FROM heros WHERE hp_max>6000'
cursor.execute(sql)
data = cursor.fetchall()
print(cursor.rowcount, '查询成功。')
for each_hero in data:
print(each_hero)
except Exception as e:
# 打印异常信息
traceback.print_exc()
finally:
cursor.close()
db.close()
# 建议吧数据库链接信息写到配置文件里，防止密码泄露。
```

2019-07-19

一叶知秋

2

sqlalchemy用习惯了。。。献丑来一段Python代码吧

```Python
# -*- coding:utf-8 -*-
from sqlalchemy import and_
from sqlalchemy import Column, INT, FLOAT, VARCHAR
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqlalchemy.ext.declarative import declarative_base
Base = declarative_base()


class Test_db:
def __init__(self):
"""此处填上自己的连接配置"""
self.engine = create_engine(
'mysql+pymysql://username:password@host:port/db_name?charset=utf8')
db_session = sessionmaker(bind=self.engine)
self.session = db_session()

def query_all(self, target_class, query_filter):
result_list = self.session.query(target_class).filter(query_filter).all()
self.session.close()
return result_list


class Heros(Base):
"""定义表结构"""
```

```
__tablename__ = 'heros'
id = Column(INT(), primary_key=True)
name = Column(VARCHAR(255))
hp_max = Column(FLOAT())
mp_max = Column(FLOAT())

def __init__(self, id, name, hp_max, mp_max):
self.id = id
self.name = name
self.hp_max = hp_max
self.mp_max = mp_max


if __name__ == '__main__':
db_obj = Test_db()
query_filter = and_(Heros.hp_max > 6000)
heros = db_obj.query_all(Heros, query_filter)
for hero_info in heros:
print("id:{}, name:{}, hp_max:{}, mp_max:{}".format(hero_info.id, hero_info.name,
hero_info.hp_max, hero_info.mp_max))
```

```
id:10000, name:夏侯惇, hp_max:7350.0, mp_max:1746.0
id:10046, name:钟馗, hp_max:6280.0, mp_max:1988.0
id:10048, name:鬼谷子, hp_max:7107.0, mp_max:1808.0
id:10051, name:赵云, hp_max:6732.0, mp_max:1760.0
id:10052, name:橘石京, hp_max:7000.0, mp_max:0.0
id:10055, name:杨戬, hp_max:7420.0, mp_max:1694.0
id:10056, name:达摩, hp_max:7140.0, mp_max:1694.0
id:10057, name:孙悟空, hp_max:6585.0, mp_max:1760.0
id:10058, name:刘备, hp_max:6900.0, mp_max:1742.0
.....执行结果有点多字数限制了
Process finished with exit code 0
```

2019-07-19

Destroy、

2

```
sql = 'DELETE FROM player WHERE name = %s'
val = (" 约翰 - 科林斯 ")
cursor.execute(sql)
db.commit()
print(cursor.rowcount, " 记录删除成功。")
```

这里写错了哇，这样写才不会报错：

```
sql = 'DELETE FROM player WHERE player_name = %s'
val = (" 约翰 - 科林斯 ", )
```

```
cursor.execute(sql, val)
db.commit()
print(cursor.rowcount, " 记录删除成功。")
```

2019-07-19

mickey

2

```
# -*- coding: UTF-8 -*-
import mysql.connector
import traceback

# 打开数据库连接
db = mysql.connector.connect(
host="localhost",
user="root",
passwd="123456", # 写上你的数据库密码
database='nba',
auth_plugin='mysql_native_password'
)

# 获取操作游标
cursor = db.cursor()

try:

# 查询heros 表中最大生命值大于 6000 的英雄进行查询，并且输出相应的属性值。
sql = 'SELECT name, hp_max FROM heros WHERE hp_max > %s ORDER BY hp_max DES
C'
val = (6000,)
cursor.execute(sql, val)
data = cursor.fetchall()
for each_player in data:
print(each_player)
except Exception as e:
# 打印异常信息
traceback.print_exc()
# 回滚
db.rollback()
finally:
# 关闭游标 & 数据库连接
cursor.close()
db.close()
```

输出：

('廉颇', 9328.0)
('白起', 8638.0)
('程咬金', 8611.0)
('刘禅', 8581.0)
('牛魔', 8476.0)
('张飞', 8341.0)
('庄周', 8149.0)
('刘邦', 8073.0)
('项羽', 8057.0)
('亚瑟', 8050.0)
('东皇太一', 7669.0)
('典韦', 7516.0)
('曹操', 7473.0)
('杨戬', 7420.0)
('夏侯惇', 7350.0)
('吕布', 7344.0)
('哪吒', 7268.0)
('墨子', 7176.0)
('老夫子', 7155.0)
('达摩', 7140.0)
('鬼谷子', 7107.0)
('关羽', 7107.0)
('钟无艳', 7000.0)
('橘石京', 7000.0)
('刘备', 6900.0)
('太乙真人', 6835.0)
('孙膑', 6811.0)
('赵云', 6732.0)
('扁鹊', 6703.0)
('铠', 6700.0)
('露娜', 6612.0)
('孙悟空', 6585.0)
('钟馗', 6280.0)
('雅典娜', 6264.0)
('兰陵王', 6232.0)
('宫本武藏', 6210.0)
('娜可露露', 6205.0)
('高渐离', 6165.0)
('芈月', 6164.0)
('不知火舞', 6014.0)
('孙尚香', 6014.0)

Process finished with exit code 0

2019-07-19

林彦 2

try...except...那部分代码没有关闭游标的语句。关闭数据库连接的语句执行时一般都会先隐式关闭并释放当前的游标吗？

2019-07-19

mickey 1

sql = 'DELETE FROM player WHERE player_name = " 约翰-科林斯 "'

2019-07-19

极客酱 1

删除约翰·科林斯这个球员的数据代码里面，excute那个函数缺少了val的参数吧？

2019-07-19

一步 1

看目录，我以为到 SQL刷题了。。。

2019-07-19

发条 0

使用8.0以上版本mysql的同学，在连接数据库的时候可能会受到quth_plugin不支持mysql_native_password的报错，
可以用ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'root';语句把auth_plugin改掉

2019-07-26

大斌 0

核心代码：

```
cursor = db.cursor()

sql = "select name,hp_max from heros where hp_max > %s"
val = (6000,)
cursor.execute(sql,val)
data = cursor.fetchall()
```

注意：val里面的元素后面必须要加英文逗号，不加或者中文逗号都会报错

2019-07-23

大斌 0

python连接mysql时报错【mysql.connector.errors.NotSupportedError: Authentication plugin 'caching_sha2_password' is not support】。
原因是：mysql8.0.11使用了Use Strong Password Encryption for Authentication即强密码加密

。
通常的处理方法是：重装mysql【装更低版本的或者将Use Strong Password Encryption for Authentication改为Use Legacy Authentication Method(在Authentication Method中改)】
还有一种更好的方案，那就是使用【pymysql】库来连接，代码如下：

```
db_host = "localhost"
db_username = "root"
db_password = "123456"
db_name = "database_name"
conn = pymysql.connect(
host=db_host,
user=db_username,
passwd=db_password,
database=db_name,
)
```

2019-07-22

Geek_5d805b

0

请问老师关于sql语句中的%s占位问题，如果我要修改多项值可否用变量替代呢？

2019-07-22

cricket1981

0

```
# -*- coding: UTF-8 -*-

import mysql.connector
# 打开数据库连接
db = mysql.connector.connect(
host="localhost",
user="root",
passwd="root", # 写上你的数据库密码
database='test',
auth_plugin='mysql_native_password'
)
# 获取操作游标
cursor = db.cursor()
sql = 'SELECT * FROM heros WHERE hp_max>=6000'
cursor.execute(sql)
data = cursor.fetchall()
for each_hero in data:
print(each_hero)
cursor.close()
db.close()
```

2019-07-22

cricket1981

0

python print语句出来的中文是乱码要怎么处理？python程序第一行加过 # -*- coding: UTF-8 -*-
(10003, u'\u5b89\u5fb7\u70c8-\u5fb7\u62c9\u8499\u5fb7', 2.11)

2019-07-22

华夏 0

pip install mysql-connector之后连接数据库会报错，更新到最新版还是一样报错，pip install mysql-connector-python之后可以正常运行。大家如果有遇到这个问题的可以参考@林彦-广州-数据分析师 这个办法。

2019-07-20

lmingzhi 0

老师，你好，我一般使用SQLAlchemy连接数据库:
from sqlalchemy import create_engine
db = create_engine('mysql://root@localhost/test_database')
一般如你在文中提到的话如果使用结束后，不想再浪费资源，需要执行什么操作？

2019-07-20

ABC 0

执行结果:

('夏侯', 7350.0)
('钟无艳', 7000.0)
('张飞', 8341.0)
('牛魔', 8476.0)
('吕布', 7344.0)
('亚瑟', 8050.0)
('芈月', 6164.0)
('程咬金', 8611.0)
('廉颇', 9328.0)
('东皇太一', 7669.0)
('庄周', 8149.0)
('太乙真人', 6835.0)
('白起', 8638.0)
('雅典娜', 6264.0)
('刘邦', 8073.0)
('刘禅', 8581.0)
('墨子', 7176.0)
('项羽', 8057.0)
('关羽', 7107.0)
('孙尚香', 6014.0)
('露娜', 6612.0)
('不知火舞', 6014.0)

('孙膑', 6811.0)
('高渐离', 6165.0)
('扁鹊', 6703.0)
('钟馗', 6280.0)
('鬼谷子', 7107.0)
('赵云', 6732.0)
('橘石京', 7000.0)
('杨戬', 7420.0)
('达摩', 7140.0)
('孙悟空', 6585.0)
('刘备', 6900.0)
('曹操', 7473.0)
('典韦', 7516.0)
('宫本武藏', 6210.0)
('老夫子', 7155.0)
('哪吒', 7268.0)
('娜可露露', 6205.0)
('兰陵王', 6232.0)
('铠', 6700.0)

2019-07-19

ABC

0

练习答案:

```
# -*- coding: UTF-8 -*-
import mysql.connector

db = mysql.connector.connect(
host="localhost",
user="root",
passwd="123456",
database='geektime-sql',
auth_plugin='mysql_native_password'
)

cursor = db.cursor()
try:
sql = 'SELECT name,hp_max from heros where hp_max>6000;'
cursor.execute(sql)
data = cursor.fetchall()
for each_player in data:
print(each_player)
```

finally:
cursor.close()
db.close()

2019-07-19

谷径

0

auth_plugin='mysql_native_password' 这句报错，是什么意思呢？

2019-07-19

华夏

0

('夏侯惇', 7350.0, 1746.0, 321.0, 397.0)
('钟无艳', 7000.0, 1760.0, 318.0, 409.0)
('张飞', 8341.0, 100.0, 301.0, 504.0)
('牛魔', 8476.0, 1926.0, 273.0, 394.0)
('吕布', 7344.0, 0.0, 343.0, 390.0)
('亚瑟', 8050.0, 0.0, 346.0, 400.0)
('芈月', 6164.0, 100.0, 289.0, 361.0)
('程咬金', 8611.0, 0.0, 316.0, 504.0)
('廉颇', 9328.0, 1708.0, 286.0, 514.0)
('东皇太一', 7669.0, 1926.0, 286.0, 360.0)
('庄周', 8149.0, 1694.0, 297.0, 497.0)
('太乙真人', 6835.0, 1680.0, 284.0, 396.0)
('白起', 8638.0, 1666.0, 288.0, 430.0)
('雅典娜', 6264.0, 1732.0, 327.0, 418.0)
('刘邦', 8073.0, 1940.0, 302.0, 504.0)
('刘禅', 8581.0, 1694.0, 295.0, 459.0)
('墨子', 7176.0, 1722.0, 328.0, 475.0)
('项羽', 8057.0, 1694.0, 306.0, 494.0)
('关羽', 7107.0, 10.0, 343.0, 386.0)
('孙尚香', 6014.0, 1756.0, 411.0, 346.0)
('露娜', 6612.0, 1836.0, 335.0, 375.0)
('不知火舞', 6014.0, 200.0, 293.0, 336.0)
('孙膑', 6811.0, 1926.0, 328.0, 413.0)
('高渐离', 6165.0, 1988.0, 290.0, 343.0)
('扁鹊', 6703.0, 2016.0, 309.0, 374.0)
('钟馗', 6280.0, 1988.0, 278.0, 390.0)
('鬼谷子', 7107.0, 1808.0, 297.0, 394.0)
('赵云', 6732.0, 1760.0, 380.0, 394.0)
('橘石京', 7000.0, 0.0, 347.0, 392.0)
('杨戬', 7420.0, 1694.0, 325.0, 428.0)
('达摩', 7140.0, 1694.0, 355.0, 415.0)
('孙悟空', 6585.0, 1760.0, 349.0, 385.0)
('刘备', 6900.0, 1742.0, 363.0, 381.0)

('曹操', 7473.0, 0.0, 361.0, 371.0)
('典韦', 7516.0, 1774.0, 345.0, 402.0)
('宫本武藏', 6210.0, 0.0, 330.0, 391.0)
('老夫子', 7155.0, 5.0, 329.0, 409.0)
('哪吒', 7268.0, 1808.0, 320.0, 408.0)
('娜可露露', 6205.0, 1808.0, 385.0, 359.0)
('兰陵王', 6232.0, 1822.0, 388.0, 342.0)
('铠', 6700.0, 1784.0, 328.0, 388.0)
41 查询成功。

2019-07-19