python连接数据库

```
1.python连接mysql数据库需要下载pymysql模块
2.步骤:
    1.连接数据库
    2.创建游标对象
    3.通过游标对象操作数据库
    4.关闭数据库

前提:
1.你的电脑要安装好mysql数据库
2.你有数据库的用户名和密码

#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====
#需要引入pymysql模块,没有这个模块的话需要下载
import pymysql

#连接数据库
'''
host=数据库所在的电脑的ip地址
port=3306
user=登录数据库的用户名
passwd=登录数据库的密码
db=要连接的数据库名,mysql是默认的数据库名字
charset=字符集
返回值:db是数据库对象,通过这对象创建游标
'''
db=pymysql.connect(host='127.0.0.1',port=3306,user="root",passwd='123456',db="mysql"
,charset='utf8')

#创建游标
cur=db.cursor()

#查看数据库系统的版本
cur.execute('select version()')
#获取查询的结果
data=cur.fetchall()
print("数据库系统的版本为:",data)#数据库系统的版本为: (('5.6.25-log',),)

#创建数据库
cur.execute('drop database if exists mytestcn')#如果有这个名字的数据库就删除
cur.execute('create database mytestcn')#创建数据库

#查看所在的数据库名字
```

```
cur.execute("select database()")
#获取查询的结果
data=cur.fetchall()
print("所在的数据库名字为:",data)#所在的数据库名字为: (('mysql',),)

#切换数据库,切换到mytestcn数据库中
cur.execute('use mytestcn')

#查看所在的数据库名字
cur.execute("select database()")
#获取查询的结果
data=cur.fetchall()
print("所在的数据库名字为:",data)#所在的数据库名字为: (('mytestcn',),)


#创建表格
#表格要存储(用户名,密码,手机号码,邮箱)
#创建表格的sql语句
sql="create table usertable(" \
    "name varchar(20),passwd varchar(20)," \
    "iph varchar(20),email varchar(20))"

cur.execute(sql)

#往表格添加数据(增)
sql="insert into usertable values('李四','123456','13833445566','2342423@qq.com')"
cur.execute(sql)
db.commit()

#改表格中的数据(改)
sql="update usertable set iph='138888888' where name='李四'"
cur.execute(sql)
db.commit()

#查看表格的内容
sql="select *from usertable"
cur.execute(sql)
data=cur.fetchall()
print(data)

#删除数据库(删)
sql="delete from usertable where name='李四'"
cur.execute(sql)
db.commit()

#关闭数据库
db.close()



```

迭代器的介绍

一、什么是迭代器
迭代是python中访问集合元素的一种非常强大的一种方式。迭代器是一个可以记住遍历位置的对象，因此不会像列表那样一次性全部生成，而是可以等到用的时候才生成，因此节省了大量的内存资源。迭代器对象从集合中的第一个元素开始访问，直到所有的元素被访问完。迭代器有两个方法：iter（ ）和next（ ）方法。

二、可迭代的对象
类似于list、tuple、str 等类型的数据可以使用for …… in…… 的循环遍历语法从其中依次拿到数据并进行使用，我们把这个过程称为遍历，也称迭代。python中可迭代的对象有list（列表）、tuple（元组）、dirt（字典）、str（字符串）等。

从字面来理解，迭代器指的就是支持迭代的容器，更确切的说，是支持迭代的容器类对象，这里的容器可以是列表、元组等这些 Python 提供的基础容器，也可以是自定义的容器类对象，只要该容器支持迭代即可。

三.Python 迭代器的好处
使用迭代器的好处是可以节省资源。
代码减少。
代码冗余得到极大解决。
降低代码复杂度。
它为编码带来了更多的稳定性。

如果我们的列表只有10个数据，那么读取的速度会很快。可能对资源（也就是我们的内存）占用的消耗不是很大；如果我们有一个成千上万甚至十万数据的列表，那么这些数据都要一次性的写入内存里，这么多的数据所消耗占用的资源必然会很大，甚至会撑爆我们的内存造成内存溢出，程序就会报错了。
所以如果通过迭代器的方式，我们只需要用到一个数据就将一个数据扔到内存里并且被使用。这样既可以提高我们内存使用的效率，又可以减少我们内存的消耗。这也是我们平时使用迭代器的目的。

生成迭代器-方法1

```
1.iter（可迭代数据类型）:生成一个迭代器对象
2.next（ ）:返回迭代器中的数据

#生成迭代器对象
obj_iter=iter([1,2,3,4,5,6,7,8,9])
#注意,iter参数中的列表被迭代器迭代后,不是一次性把数据放到内存
print(next(obj_iter))

#next获取不到数据时,会报StopIteration异常
# for i in range(10):
#     print(next(obj_iter))
```

生成迭代器-方法2

```
1.使用函数,在函数中使用for循环,结合yield函数
2.yield函数就是在for循环中,把数据放到一个迭代对象中,只不过是被调用了才放入

# def mytest():
```

```
#     for i in range(10):
#         yield i
#
# #生成迭代对象
# res=mytest()
#
# print(next(res))
# print(next(res))


#生成迭代对象
res=(i for i in [1,2,3,4])

# print(next(res))
# print(next(res))
for i in res:
    print(i)
```

自定义迭代器对象

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====

class Maker():
    def __init__(self,n):
        self.n=n #[0,n-1]
        self.index=0

    def __next__(self):
        if self.index<self.n:
            res=self.index
            self.index+=1
            return res
        else:
            raise StopIteration

    def __iter__(self):
        return self


m=Maker(5)
print(next(m))
print(next(m))

# for i in m:
#     print(i)
```

自定义一个可迭代对象，从0开始的一个连续的整数序列

```
"""
可以通过for...in...进行遍历的对象
实现了\__iter\_\_() 魔术方法的类的对象
并且__iter__()魔术方法的返回结果是一个迭代器对象
"""

作业,自己写个类,实现迭代器对象
```