**pymysql模块**

pymysql安装

使用pypi

pypi即python package index

是python语言的软件库

官方地址为https://pypi.python.org

通过pip安装pymysql模块

安装依赖包

yum -y install gcc

本地安装

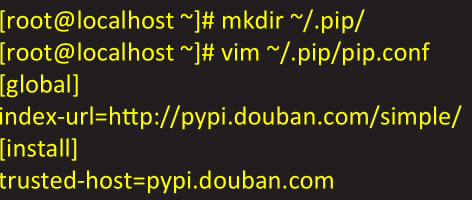
pip3 install pymysql-0.8.0.tar.gz

在线安装

pip3 install pymysql

使用国内镜像站点

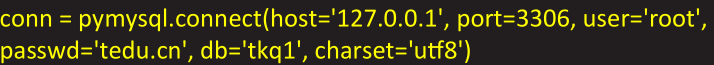
为了实现安装加速，可以配置pip安装时采用国内镜像站点

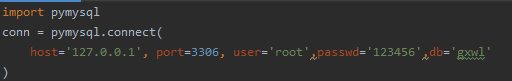


pymysql应用

连接数据库

创建连接是访问数据库的第一步





游标

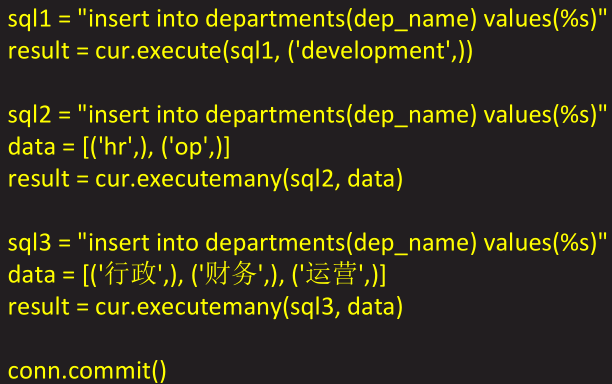
游标就是游动的标识

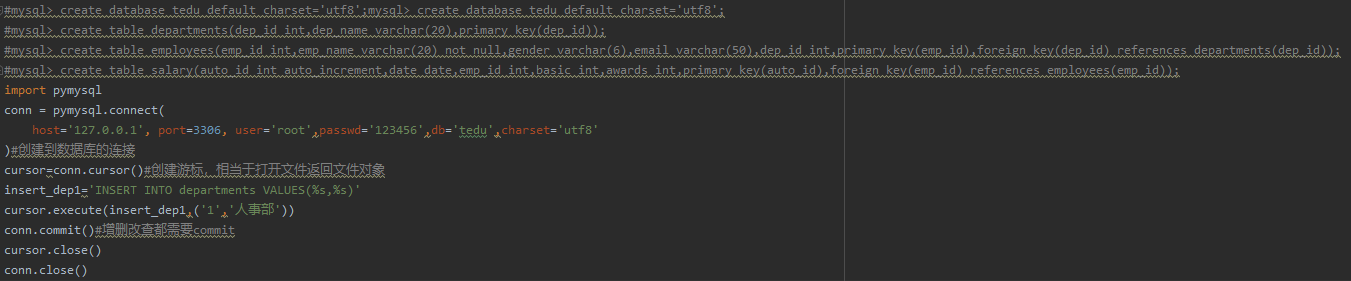
通俗的说，一条sql取出对应n条结果资源的接口/句柄，就是游标，沿着游标可以一次取出一行

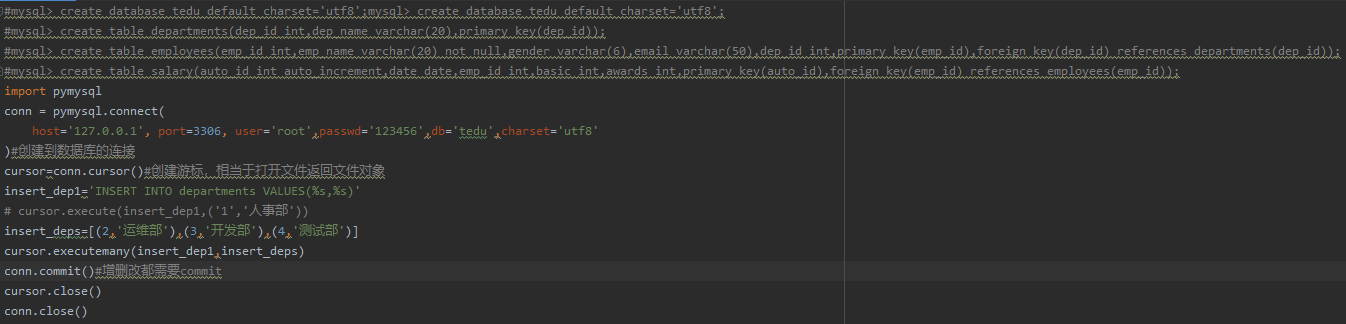


插入数据

对数据表做修改操作，必须要commit

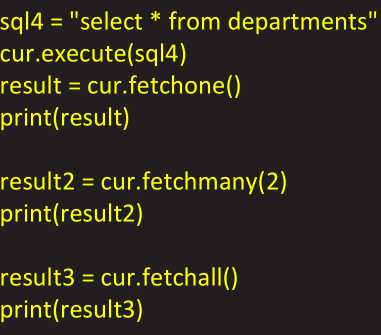






查询数据

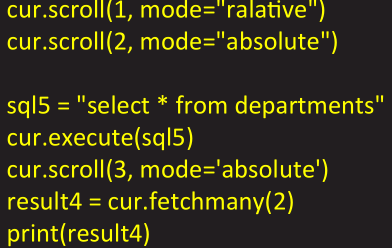
可以取出表中的一条、多条或全部记录





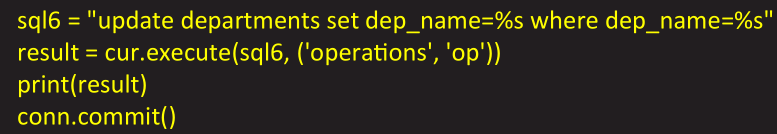
移动游标

如果不希望从头取数据，可以先移动游标



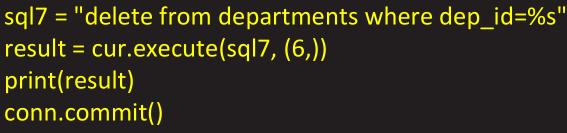
修改数据

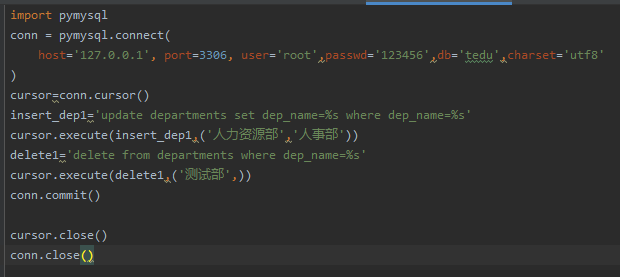
通过update修改某一字段的值



删除记录

通过delete删除记录





**sqlalchemy基础**

sqlalchemy概述

sqlalchemy安装

sqlalchemy由官方收录，可以直接安装

pip3 install sqlalchemy



sqlalchemy简介

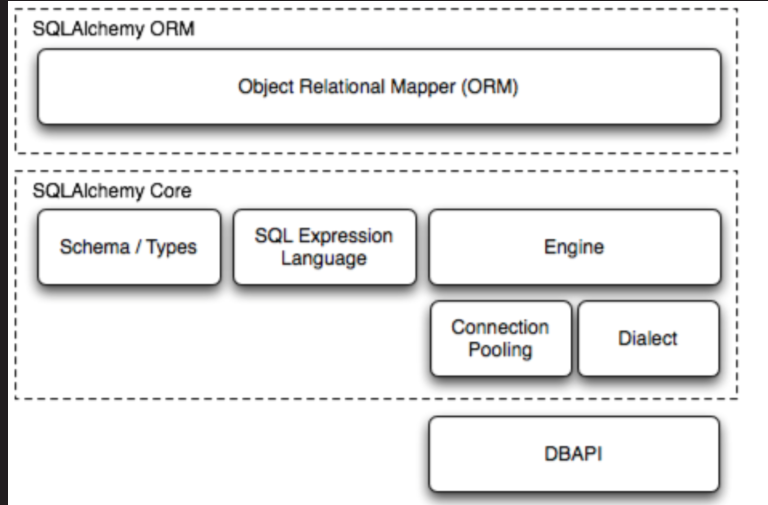
sqlalchemy是python编程语言下的一款开源软件。提供sql包及对应关系的映射（ORM）工具，使用MIT许可证发

sqlalchemy“才用简单的python语言，为高效何高性能的数据库访问设计，实现了完整的企业级持久模型”

sqlalchemy的理念是，sql数据库的量级喝性能重要于对象集合；而对象集合的抽象又重于表和行

目标是提供能兼容众多数据库（如SQLite、mysql、Postgresql、oracle、ms-sql、sqlserver喝firebird）的企业级持久性模型

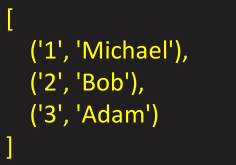
sqlalchemy架构



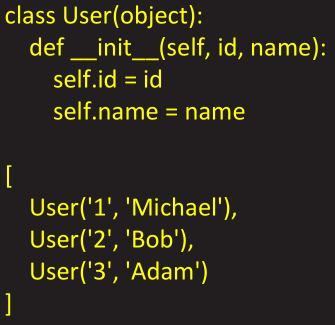
ORM模型

ORM即对象映射

数据库表是一个二维表，包含多行多列。把一个表的内容用python的数据结构表示出来的话，可以用一个List表示多行，list的每一个元素是tuple，表示一行记录



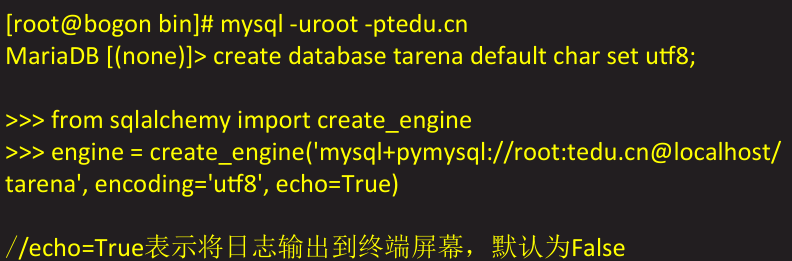
用tuple表示一行很难看出来的结构。如果把一个tuple用class实例来表示，就更容易地看出来表的结构



数据库对象管理

连接mysql

通过create\_engine实现数据库的连接



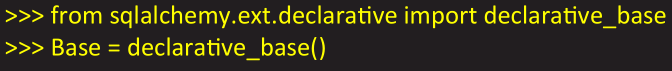
申明映射

当使用ORM的时候，配置过程从秒数数据库表开始

通过自定义类映射相应的表

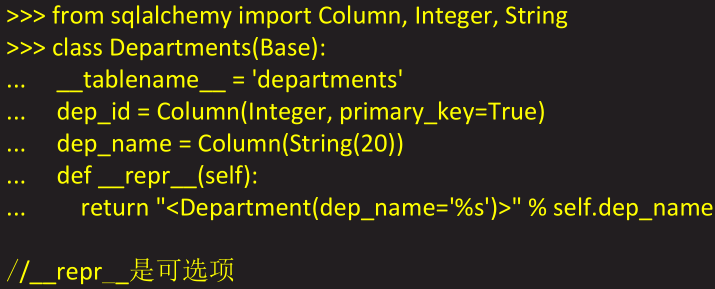
通过声明系统实现类映射

首先通过声明系统，定义基类



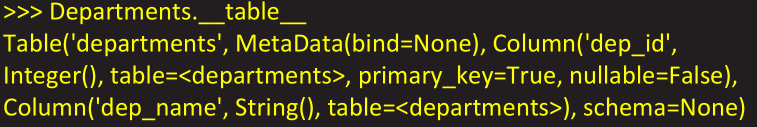
创建映射类

一旦创建了基类，就可以创建自定义映射类了



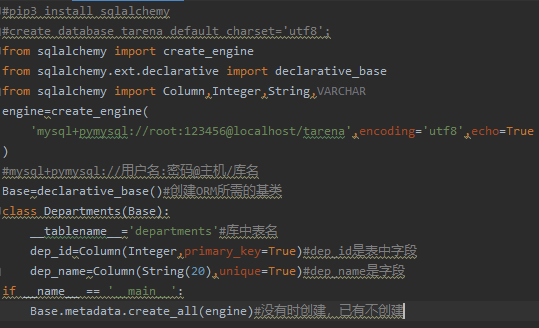
创建架构

类构建完成后，表的信息将写入到表的元数据（metadata）



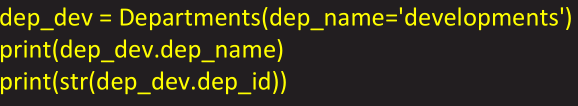
通过表的映射类，在数据库中创建表





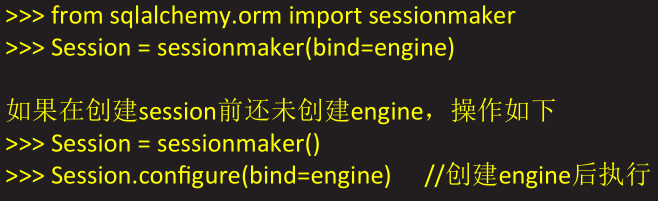
创建映射类的实例

创建实例时，并不会真正在表中添加记录



创建会话类

ORM访问数据库的句柄被称作session



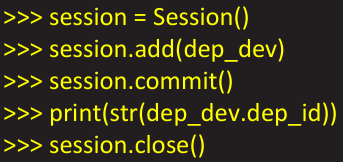
添加新对象

会话类的实例对象用于捆绑到数据库

实例化类的对象，并不打开任何连接

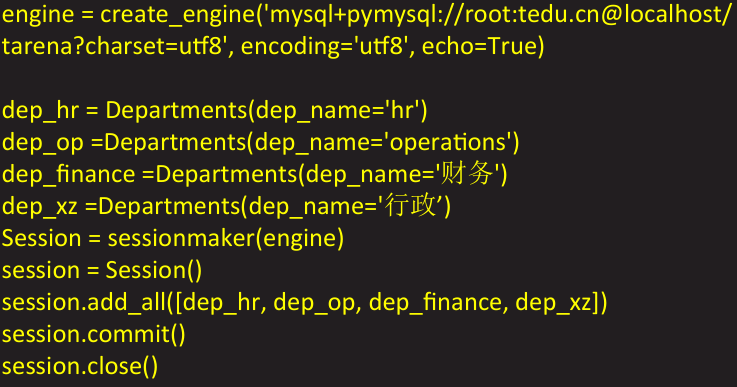
当实例初次使用，它将从engine维护的连接池中获得一个连接

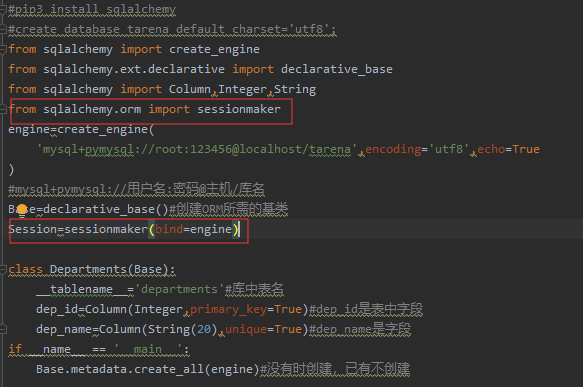
当所有的事务均被commit或会话对象被关闭时，连接结束

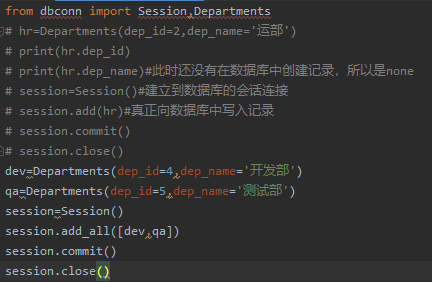


可以创建多个实例，批量添加记录

如果有中文文件，注意engine的参数

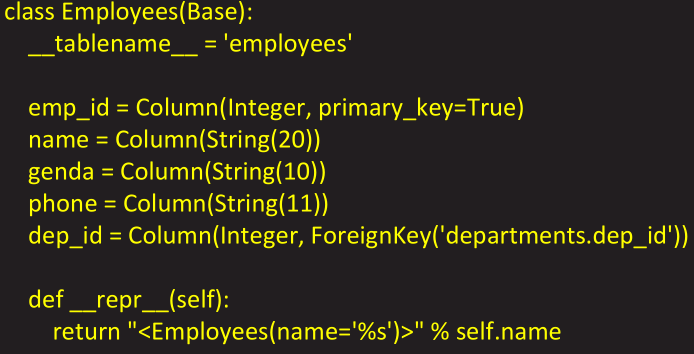






外键约束

ORM映射关系也可用于表间创建外键约束



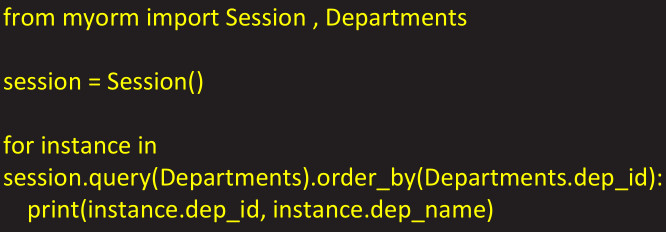
**sqlalchemy进阶**

查询操作

基本查询

通过作用于session的query()函数创建查询对象

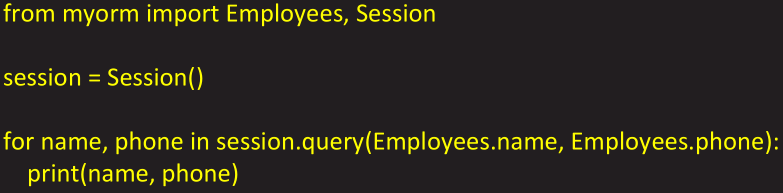
query()函数可以接收多种参数



使用ORM描述符进行查询

使用ORM描述符进行查询

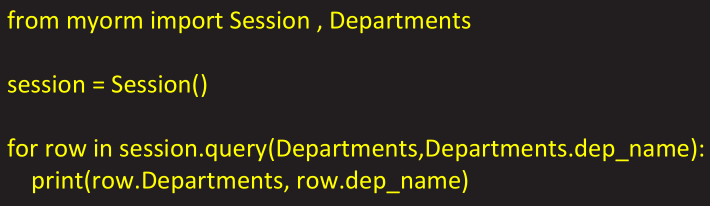
返回值是元组



使用命名元组

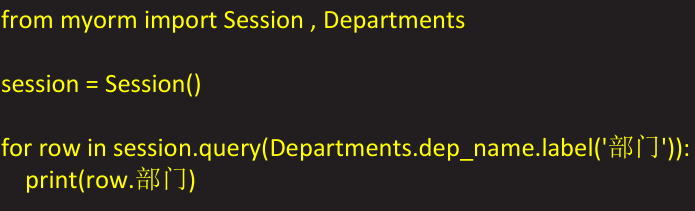
查询对象返回的是一个命名元组

名称是类的名字，或是类中属性的名字



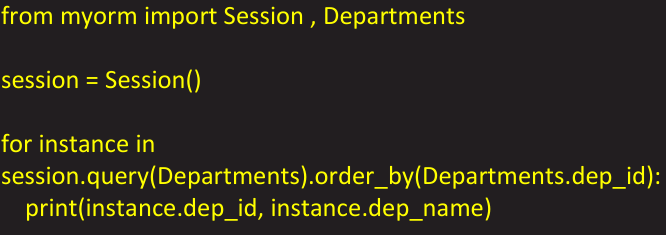
修改显示字段名

显示的字段名可以通过Label()函数进行修改



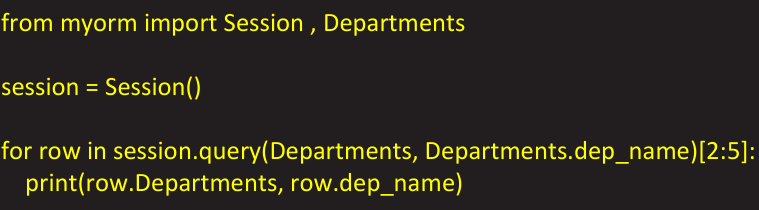
排序

通过order\_by()函数可以实现按指定字段排序



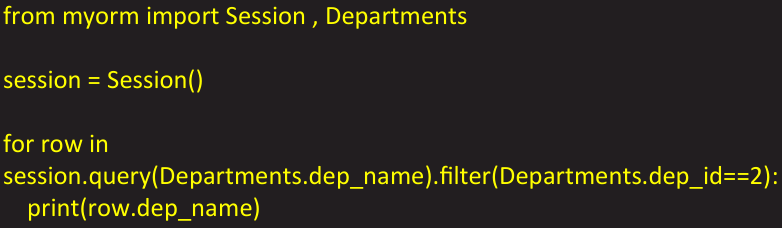
提取部分数据

通过“切片”的方式，实现部分数据的提取

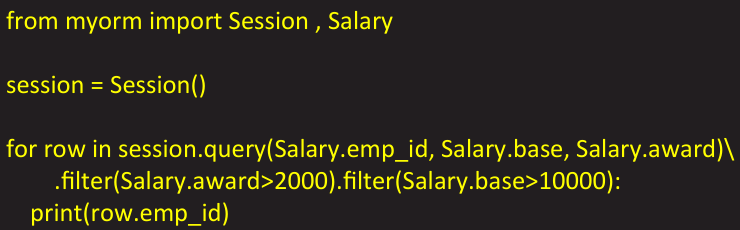


结果过滤

通过filter()函数实现结果过滤



filter()函数可以叠加使用



常用过滤操作符

相等

query.filter(Employees.name==’join’)

不相等

query.filter(Employees.name!=’join’)

模糊查询

query.filter(Employees.name.like(' %j '))

in

query.filter(new\_emp.name.in\_(['bob', 'john'])

not in

query.filter(~new\_emp.name.in\_(['bob', 'john'])

字段为空

query.filter(new\_emp.name.is\_(None))

字段不为空

query.filter(new\_emp.name.isnot(None))

多重条件and

from sqlalchemy import and\_

query.filter(and\_(new\_sal.basic>=10000,new\_sal.extra>=2000))

多重条件or

from sqlalchemy import or\_

query.filter(or\_(new\_sal.basic>=10000,new\_sal.extra>=3000))

查询对象返回值

all()返回列表

first()返回结果中的第一条记录

one()取出所有记录，如果不是一条记录则抛出异常

scalar()调用one()，返回第一列的值

聚合

通过count()方法，统计行数



多表查询

通过join()方法实现多表查询



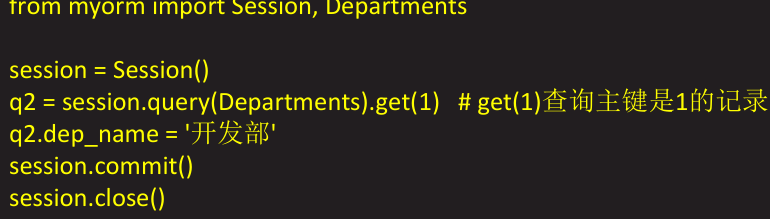
修改操作

更新数据

通过会话的update()方法更新



通过会话的字段赋予值更新



删除记录

通过会话的delete()方法进行删除记录

