

# Python 3 程序设计

## Python的数据库编程

主讲教师:

武汉纺织大学

### Python的数据库编程

- ◆ 本章内容
- ◆ 数据库的基础知识
- ◆ SQLite数据库
- ◆ 关系数据库语言SQL
- ◆ Python的SQLite3编程
- ◆ SQLite编程应用

Python支持Sybase、SQL Server、SQLite等多种数据库,本章主要介绍Python自带的关系型数据库SQLite的应用



## 数据库的基础知识

### ◇ 数据库的概念

- ◇ 数据库（**Data Base, DB**）将大量数据按照一定的方式组织并存储起来，是相互关联的数据的集合。
- ◇ 数据库为用户提供安全、高效、快速检索和修改的数据集合。
- ◇ 数据库在数据库系统中使用，其核心是数据库管理系统。
  1. 数据库系统
  2. 数据库管理系统
    - ◇ 数据定义功能。
    - ◇ 数据操纵功能。
    - ◇ 数据库的运行管理。
    - ◇ 数据通信功能。

## 数据库的基础知识

### ◆ 关系型数据库

关系型数据库是目前的主流数据库。

#### 1. 关系型数据库的基本概念

◆ 关系、元组、属性、域、关键字。

#### 2. 实体间联系的类型

◆ 实体是指客观世界的事物，实体的集合构成实体集，在关系数据库中用二维表来描述实体。

◆ 实体之间有各种各样的联系，归纳起来有以下3种类型。

一对一联系（1:1），一对多联系（1: $n$ ），多对多联系（ $m:n$ ）

### SQLite数据库

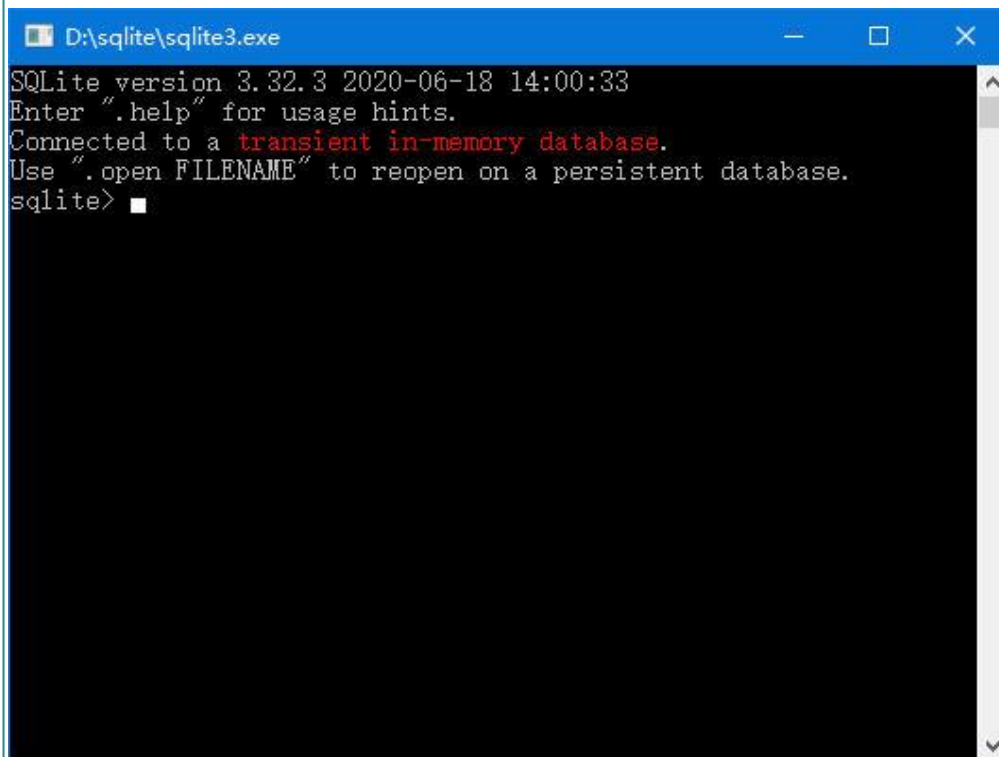
#### ◆ SQLite数据库简介

- ◆ SQLite是用C语言编写的嵌入式数据库，体积小。
- ◆ SQLite不需要一个单独的服务器进程或操作系统（无服务器的），也不需要配置。一个完整的 **SQLite**数据库存储在单一的跨平台的磁盘文件中。
- ◆ SQLite支持**SQL92（SQL2）**标准的大多数查询语言的功能，并提供了简单和易于使用的**API**。

### SQLite数据库

#### ◆ 下载和安装SQLite数据库

- ◆ SQLite是开源的数据库，可以在其官网免费下载。
- ◆ SQLite3是SQLite的第3个版本。
- ◆ SQLite数据库不需要安装，直接运行sqlite3.exe，即可打开SQLite数据库的命令行窗口。



```
D:\sqlite\sqlite3.exe
SQLite version 3.32.3 2020-06-18 14:00:33
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> ■
```

## SQLite数据库

### ◆ SQLite3常用命令

- ◆ SQLite3命令分为两类，一类是SQLite3交互模式命令，另一类是SQL命令。
- ◆ SQLite3交互模式常用的命令。

交互命令	功能
<b>sqlite3.exe [dbname]</b>	启动sqlite3的交互模式，并创建dbname数据库
<b>.open dbname</b>	创建数据库或打开数据库
<b>.databases</b>	显示当前打开的数据库文件
<b>.tables</b>	查看当前数据库下的所有表
<b>.schema [tbname]</b>	查看表结构信息
<b>.exit</b>	退出交互模式
<b>.help</b>	列出命令的提示信息

### SQLite数据库

#### ◆ SQLite3的数据类型

- ◆ SQLite数据库中的数据分为整数、小数、字符、日期、时间等类型。
- ◆ 动态的数据类型，数据库管理系统会根据列值自动判断列的数据类型。
- ◆ 静态数据类型取决于它的存储单元（所在列）的类型。
- ◆ SQLite3的动态数据类型向后兼容其他数据库普遍使用的静态类型。
- ◆ SQLite3使用弱数据类型。SQLite3的表可以不声明列的类型。



## SQLite数据库

### ◆ SQLite3的函数

- ◆ SQLite数据库提供算术、字符串、日期、时间等操作函数，方便用户处理数据库中的数据。
- ◆ 函数需要在SQLite的命令窗口使用select命令运行。

#### SQLite3算术函数

<b>abs(x)</b>	返回绝对值
<b>max(x,y,.....)</b>	返回最大值
<b>min(x,y,.....)</b>	返回最小值
<b>random(*)</b>	返回随机数
<b>round(x[,y])</b>	四舍五入

#### SQLite3字符串函数

<b>length(x)</b>	返回字符个数
<b>lower(x)</b>	大写转小写
<b>upper(x)</b>	小写转大写
<b>substr(x,y,Z)</b>	截取子串
<b>like(A,B)</b>	确定给定的字符串与指定的模式是否匹配

#### SQLite3时间/日期函数

<b>date()</b>	产生日期
<b>datetime()</b>	产生日期和时间
<b>time()</b>	产生时间
<b>strftime()</b>	格式化字符串

### SQLite数据库

#### ◆ Python的sqlite3模块

- ◆ Python内置了SQLite数据库，通过内置的sqlite3模块可以直接访问数据库。
- ◆ sqlite3提供的Python程序上遵守Python DB-API规范。
- ◆ Python DB-API是为不同的数据库提供的访问接口规范。
- ◆ 该接口定义了一系列必需的对象和数据库存取方式，以便为各种底层数据库系统和多样的数据库接口程序提供一致的访问接口。

### SQLite数据库

#### ◆ sqlite3模块中的对象

◆ 下面是sqlite3模块中的部分常量、函数或对象。

- (1) `sqlite3.version`: 常量, 返回sqlite3模块的版本号。
- (2) `sqlite3.sqlite_version`: 常量, 返回sqlite数据库的版本号。
- (3) `sqlite3.Connection`: 数据库连接对象。
- (4) `sqlite3.Cursor`: 游标对象。
- (5) `sqlite3.Row`: 行对象。
- (6) `sqlite3.connect(dbname)`: 函数, 链接到数据库, 返回Connection对象。

### SQLite数据库

#### ◆ 创建SQLite3数据库

- ◆ 运行SQLite数据库的同时，通过参数创建Sqlite数据库，方法如下。

`sqlite3 dbname`

- ◆ 数据库文件的扩展名为.db。如果数据库文件存在，则打开该数据库；否则创建该数据库。

### 关系数据库语言SQL

- ◆ SQL是Structured Query Language的缩写，即结构化查询语言
- ◆ SQL命令的执行，需要注意下面的问题。
  - ◆ SQL命令需要在数据库管理系统中运行。
  - ◆ 在SQLite窗口运行SQL命令，需要在SQL语句后加英文的分号后回车执行。
  - ◆ SQL命令不区分大小写。

## 关系数据库语言SQL

### ◇ 数据表的建立和删除

◇ 在SQL中，使用 `create table` 语句创建表，`delete table` 删除表。

### 创建表的语法结构和示例

```
create table <表名> (
```

列名 1 数据类型和长度 1 列属性 1,

列名 2 数据类型和长度 2 列属性 2,

.....

列名 n 数据类型和长度 n 列属性 n

)

```
create table employee (  
    emp_id integer primary key,  
    emp_name varchar(20) NOT NULL,  
    sex char(2) default('男'),  
    title varchar(20),  
    wage float,  
    dep_id integer  
)
```

## 关系数据库语言SQL

### ◆ 向表中添加列

◆ alter table语句向表中添加列。

`alter table <表名> add column<字段名>[ <类型>]`

例12-3 为表 `employee` 中增加一列，列名为 `tele`，数据类型为 `varchar`，长度为50，列属性不允许为空。

`alter table employee add column tele varchar(50) not null`

◆ `schema employee`

用于查看表 `employee` 的结构。

## 关系数据库语言SQL

### ◆ 向表中插入数据

◆ insert语句向表中插入数据。

insert into <表名>[<字段名表>] values (<表达式表>)

◆ 例12-4 将下面数据插入到employee表中。

1132, 李四, 男, 部门经理, 7548.6, 11

```
insert into employee(emp_id,emp_name,sex,title,wage,dep_id)
values(1132,'李四','男','部门经理',7548.6,11)
```



## 关系数据库语言SQL

### ◆ 修改表中的数据

◆ update语句修改表中的数据。

update <表名> set <字段名1>=<表达式1> [,<字段名2>=<表达式2>...][where<条件表达式>]

◆ 例12-5 将employee表中，李四的工资改为7550元。

update employee set wage=7550 where emp\_name="李四"

◆ 将李四的工资增加550元

update employee set wage=wage+550 where emp\_name="李四"

## 关系数据库语言SQL

### ◆ 删除数据

◆ DELETE语句删除表中的数据。

`delete from <表名> [where <条件表达式>]`

`from`指定从哪个表中删除数据，`where`指定被删除的记录所满足的条件，如果省略`where`子句，则删除该表中的全部记录。

◆ 例12-6 删除表employee中性别为女的记录。

`delete from employee where sex='女'`

## 关系数据库语言SQL

### ◆ 查询数据

- ◆ SQL语句创建查询使用的是select命令，基本形式是由select-from-where子句组成。

select <字段名表>[\*] from <表名> [join <表名> on <联接条件>][where <条件表达式>][group by <分组字段名>][having <条件表达式>][order by <排序选项>[asc|desc]]

- ◆ 例12-7 检索工资高于6000元的雇员的雇员号和姓名信息。

```
select emp_id,emp_name from employee where wage>6000
```

- ◆ 例12-8 检索性别为“男”，并且工资高于5500的的雇员信息。

```
select * from employee where sex="男" and wage>5500
```

### Python的SQLite3编程

#### ◇ 访问数据库的步骤

◇ 访问SQLite3数据库主要过程如下。

- (1) 导入 Python sqlite3模块
- (2) 建立数据库连接的Connection对象
- (3) 创建游标对象
- (4) 使用Cursor对象的execute()方法执行SQL命令返回结果集
- (5) 获取游标的查询结果集
- (6) 数据库的提交和回滚
- (7) 关闭Cursor对象和Connection对象

## Python的SQLite3编程

```
>>> import sqlite3
>>> dbstr="d:/sqlite/test.db"
#连接到数据库，还回sqlite3.Connection对象
>>> con=sqlite3.connect(dbstr)
>>> cur=con.cursor()
>>> cur.execute("insert into emp values(101,'Jack',23)")
>>> cur.execute("select * from emp")
>>> print(cur.fetchall()) #提取查询到的数据
>>> con.commit() #事务提交。
>>> cur.close() #关闭Cursor对象。
```

## 12.4 Python的SQLite3编程

### ◆ 数据库的插入、更新和删除操作

◆ 在数据库中插入、更新、删除记录的一般步骤如下。

（1）建立数据库连接。

（2）创建游标对象`cur`，使用`cur.execute(sql)`方法执行SQL的`insert`、`update`、`delete`等语句，完成数据库记录的插入、更新、删除操作，并根据返回值判断操作结果。

（3）提交操作。

（4）关闭数据库。

◆ 例12-12 在`goods`表中完成记录的插入、更新和删除操作。

### Cursor对象

#### ▪ Cursor对象常用方法:

- ✓ `close(...)`: 关闭游标
- ✓ `execute(...)`: 执行SQL语句
- ✓ `executemany(...)`: 重复执行多次SQL语句
- ✓ `executescript(...)`: 一次执行多条SQL语句
- ✓ `fetchall(...)`: 从结果集中返回所有行记录
- ✓ `fetchmany(...)`: 从结果集中返回多行记录
- ✓ `fetchone(...)`: 从结果集中返回一行记录

## Sqlite3 执行带参数的SQL语句

- ◆ `c = conn.cursor()` # 获取游标
- ◆ `sql1 = "insert into company (id , name , age , address , salary) values ( ? , ? , ? , ? , ? )"`
- ◆ `arg = (20202, '王五', 32, '天堂', 5555)`
- ◆ `c.execute(sql1, arg)` # 执行sql
- ◆ `conn.commit()` # 提交数据库

在执行的时候，使用**arg**参数化带入数据，执行即可  
**sql**语句中用“？”占位，执行的时候用**arg**中的参数代替。  
**sqlite3**模块支持两种占位符：**？占位符** 和 **有名占位符**。



## Cursor两种占位符


- `execute(sql[, parameters])`: 该方法用于执行一条SQL语句，下面的代码演示了用法，以及为SQL语句传递参数的两种方法，分别使用问号和命名变量作为占位符。

```
import sqlite3
conn = sqlite3.connect(":memory:")
cur = conn.cursor()
cur.execute("CREATE TABLE people (name_last, age)")
who = "Dong"
age = 38
# 使用问号作为占位符
cur.execute("INSERT INTO people VALUES (?, ?)", (who, age))
# 使用命名变量作为占位符
cur.execute("SELECT * FROM people WHERE name_last=:who AND age=:age",
            {"who": who, "age": age})
print(cur.fetchone())
```



## Cursor游标fetchall()用法

- fetchone()、fetchmany(size=cursor.arraysize)、fetchall(): 用来读取数据。假设数据库通过下面代码插入数据:

  
import sqlite3

```
conn = sqlite3.connect("D:/addressBook.db")
```

```
cur = conn.cursor()          #创建游标
```

```
cur.execute("INSERT INTO addressList(name , sex , phon , QQ , address) VALUES('王小丫', '女', '13888997011', '66735', '北京市')")
```

```
cur.execute("INSERT INTO addressList(name, sex, phon, QQ, address) VALUES('李莉', '女', '15808066055', '675797', '天津市')")
```

```
cur.execute("INSERT INTO addressList(name, sex, phon, QQ, address) VALUES('李星草', '男', '15912108090', '3232099', '昆明市')")
```

```
conn.commit()                #提交事务，把数据写入数据库
```

## Cursor游标fetchall()用法

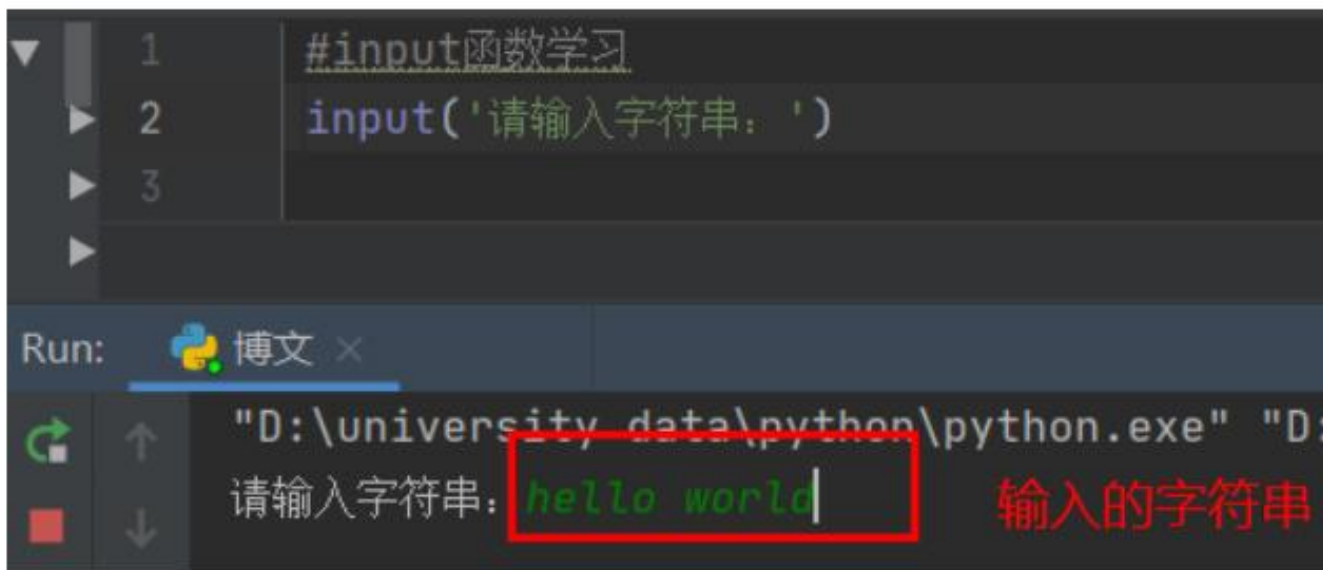
✓ 使用fetchall()读取数据:

```
import sqlite3

conn = sqlite3.connect('D:/addressBook.db')
cur = conn.cursor()
cur.execute('SELECT * FROM addressList')
li = cur.fetchall()                                #返回所有查询结果
for line in li:
    for item in line:
        print(item, end=' ')
    print()
```

**Input函数：**显示用户输入的值，可以包含提示性文字

```
input('请输入字符串：')
```



The screenshot shows a Python IDE with a dark theme. The editor window displays three lines of code: line 1 is a comment `#input函数学习`, line 2 is the function call `input('请输入字符串：')`, and line 3 is empty. Below the editor is a 'Run' button with a Python logo and the text '博文 ×'. At the bottom, a terminal window shows the command prompt `"D:\university_data\python\python.exe" "D:` followed by the prompt `请输入字符串：` and the user input `hello world`. The input `hello world` is highlighted with a red rectangle. To the right of the terminal, the text `输入的字符串` is displayed in red.

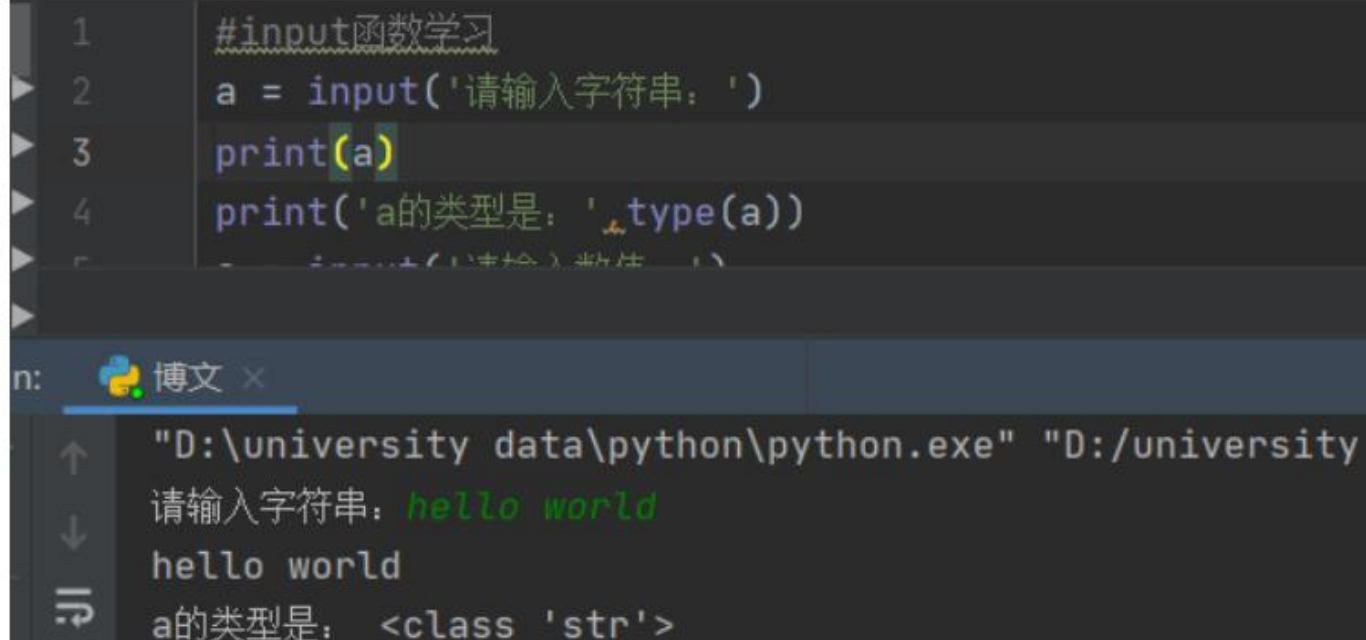
```
1 #input函数学习
2 input('请输入字符串：')
3
```

Run: 博文 ×

"D:\university\_data\python\python.exe" "D:  
请输入字符串： **hello world** 输入的字符串

### Input函数

```
1 a = input('请输入字符串: ')
2 print(a)
3 print('a的类型是: ', type(a))
```



The screenshot shows a Python IDE with a dark theme. The editor window displays the following code:

```
1 #input函数学习
2 a = input('请输入字符串: ')
3 print(a)
4 print('a的类型是: ', type(a))
```

Below the editor, the terminal window shows the execution output:

```
n: "D:\university data\python\python.exe" "D:/university
请输入字符串: hello world
hello world
a的类型是: <class 'str'>
```

## `eval()`函数

`eval()`函数功能：将引号去掉，把括号内部的字符串当成命令执行。

```
eval(<字符串>)
```

```
print('2*3')
```

2\*3

```
print(eval('2*3'))
```

6

## `eval()`函数

```
print("结果是: ", eval(input("请输入算式: ")))
```

请输入算式:  $2*(3+5)$

结果是: 16

- ◆ 用户希望输入一个数字，并用程序对这个数字进行计算，可采用 `eval(input(<输入提示字符串>))` 的组合

## 12.5 SQLite编程的应用

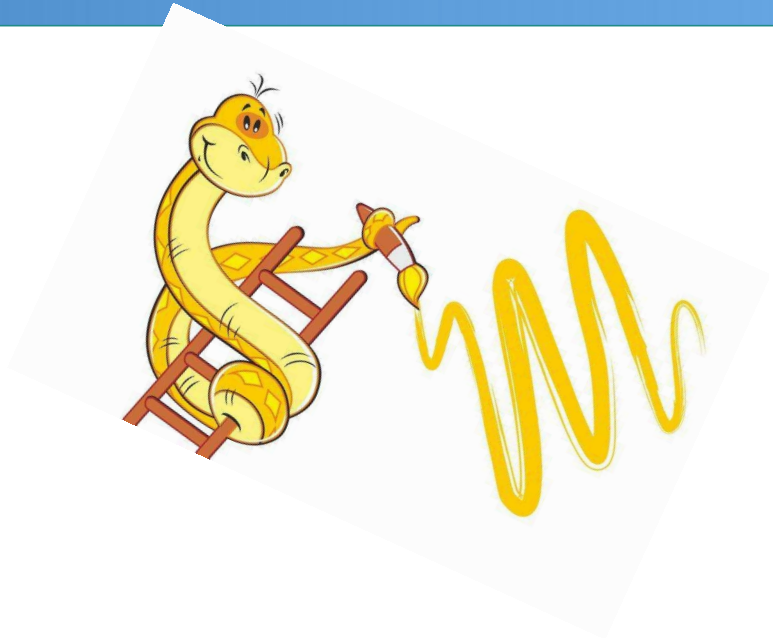
- ◆ 使用SQLite数据库实现一个简单的订单管理系统。
- ◆ 数据库名称为test.db，订单数据保存在order1表中，实现的是订单数据的增删改查的功能。
- ◆ 应用程序中涉及的函数及功能如表。

函数名称	函数功能
<b>getConnection()</b>	连接数据库的通用函数
<b>showAllData()</b>	显示所有记录
<b>getOrderListInfo()</b>	获得用户输入数据
<b>addRec()</b>	增加记录
<b>delRec()</b>	删除记录
<b>modifyRec()</b>	修改记录
<b>searchRec()</b>	查找记录



### 小结

- ◆ 数据库、数据库系统、数据库管理系统等基本概念
- ◆ 关系型数据库是目前的主流数据库，关系与二维表是等价的
- ◆ 实体之间的对应关系称为实体间的联系
- ◆ Python自带的关系型数据库SQLite是一种开源的、嵌入式数据库
- ◆ SQLite3交互模式常用的命令，SQLite3数据库使用动态的数据类型
- ◆ SQL语言基本知识
- ◆ 用SQLite数据库实现一个简单的订单管理系统



# Thanks