Console下的数据管理

第三章

目标

- 掌握Python模块的定义及使用
- 掌握Python的函数
- 掌握Python的字符串操作
- 理解正则表达式

案例

- Console下的信息管理系统；
- Console下的记事本

函数

第一节

邹琪鲜

案例

- 工具集;
- 排序;
- 极值

函数

- 函数就是完成特定功能的一个语句组，这组语句可以作为一个单位使用，并且给它取一个名字。
- 可以通过函数名在程序的不同地方多次执行（这通常叫做函数调用），却不需要在所有地方都重复编写这些语句。
- 自定义函数
  - 用户自己编写的
- 预定义的Python函数
  - 系统自带的一些函数，还有一些第三方编写的函数，如其他程序员编写的一些函数，对于这些现成的函数用户可以直接拿来使用。

函数

- 为什么使用函数
- 降低编程的难度
  - 通常将一个复杂的大问题分解成一系列更简单的小问题，然后将小问题继续划分成更小的问题，当问题细化为足够简单时，我们就可以分而治之。这时，我们可以使用函数来处理特定的问题，各个小问题解决了，大问题也就迎刃而解了。
- 代码重用
  - 我们定义的函数可以在一个程序的多个位置使用，也可以用于多个程序。此外，我们还可以把函数放到一个模块中供其他程序员使用，同时，我们也可以使用其他程序员定义的函数。这就避免了重复劳动，提供了工作效率。

函数

- 函数的定义和调用
- 当我们自己定义一个函数时，通常使用def语句，其语法形式如下所示：

```
def 函数名（参数列表）： #可以没有参数
    函数体
```

函数

- 调用函数的一般形式是：
  - 函数名（参数表）

函数

第二节

邹琪鲜

函数

- 形式参数和实际参数
  - 在定义函数时函数名后面圆括号中的变量名称叫做“形式参数”，或简称为“形参”；
  - 在调用函数时，函数名后面圆括号中的变量名称叫做“实际参数”，或简称为“实参”。

函数

- 调用函数的一般形式是：
  - 函数名（参数表）

```
def  machine(x,y):
    print  "制作出一个",x,"元",y,"口味冰激凌"

mashine（5, "巧克力"）
```

函数

- 缺省参数（默认参数）

```
def  machine(x,y=“奶油”):
    print  "制作出一个",x,"元",y,"口味冰激凌"

mashine（5, "巧克力"）
mashine（5）
```

函数

第三节

邹琪鲜

函数

- 局部变量和全局变量
  - Python中的任何变量都有其特定的作用域
  - 在函数中定义的变量一般只能在该函数内部使用，这些只能在程序的特定部分使用的变量我们称之为局部变量；
  - 在一个文件顶部定义的变量可以供该文件中的任何函数调用，这些可以为整个程序所使用的变量称为全局变量

函数

- 局部变量和全局变量

```
# -*- coding: cp936 -*-
#定义全局变量
globalint = 9

#定义一个函数
def myAdd():
    localint = 3     #定义局部变量
    print globalint
    print localint

#测试变量的局部性和全局性
myAdd()
print globalInt
print localInt
```

```
# -*- coding: cp936 -*-
#定义全局变量
g = 9

#定义一个函数
def myf():
    g = 3     #定义局部变量，并且与全局变量重名
    print 'g =', g

#测试变量g的局部性和全局性
myf()
```

函数

- global语句

```
global  变量名
```

- 强制声明为全局变量

函数

第四节

邹琪鲜

案例

- 返回数字绝对值
- 取列表最大最小值

函数

- 函数返回值
  - 函数被调用后会返回一个指定的值
  - 函数调用后默认返回None
  - return 返回值
  - 返回值可以是任意类型
  - return执行后，函数终止
  - 区分返回值和打印

函数

第五节

邹琪鲜

案例

- 多类型传值
- 传值冗余

函数

- 向函数传元组和字典
- 处理多余实参

函数

- 向函数传元组和字典
- fun（*args）
- fun（**kwords）

函数

- 处理多余实参
- def fun（*args，**kw）

Lambda表达式

第六节

邹琪鲜

lambda

- 匿名函数
  - lambda函数是一种快速定义单行的最小函数，是从 Lisp 借用来的，可以用在任何需要函数的地方。

lambda

```
>>> def f(x,y):
...     return x*y

>>> g = lambda x,y: x*y
```

- 1. 使用Python写一些执行脚本时，使用lambda可以省去定义函数的过程，让代码更加精简。
- 2. 对于一些抽象的，不会别的地方再复用的函数，有时候给函数起个名字也是个难题，使用lambda不需要考虑命名的问题。
- 3. 使用lambda在某些时候让代码更容易理解。

lambda基础

- lambda语句中，冒号前是参数，可以有多个，用逗号隔开，冒号右边的返回值。lambda语句构建的其实是一个函数对象:

- g = lambda x : x**2
- print g
- <function <lambda> at 0x00AFAAF0>

lambda应用实例

- reduce为逐次操作list里的每项，接收的参数为 2个,最后返回的为一个结果
- >>> def myadd(x,y):
- >>> return x+y
- >>> sum=reduce(myadd,(1,2,3))
- >>> 6

```
>>>   sum=reduce(lambda x,y:x+y,(1,2,3))
```

switch

第七节

邹琪鲜

switch语句

- switch语句用于编写多分支结构的程序，类似与if... elif... else 语句。
- switch语句表达的分支结构比 if... elif... else 语句表达的更清晰，代码的可读性更高。

- 但是**python**并没有提供**switch** 语句

switch实现

- python可以通过字典实现switch语句的功能。
- 实现方法分为两步。
  - 首先，定义一个字典。
  - 其次，调用字典的get()获取相应的表达式。

函数调用

- 通过字典调用函数
- {1:case1, 2:case2}.get(x,lambda *arg,**key:)()

```
# -*- coding:UTF-8 -*-
# 使用字典实现switch语句

from __future__ import division
x=1
y=2
operator = "/"
result = {
        "+" : x+y,
        "-" : x-y,
        "*" : x*y,
        "/" : x/y
        }
print result.get(operator)
```

小结

- ;

总结

- 掌握Python的函数