Python 语言基础

Python 的数据类型

变量的定义。在 python 中，变量的类型是由赋给它的数值定义的。

其为数值型变量

```
q = 7                #q
```

为字符串型变量

```
q = “Seven” #q
```

基本数据类型:字符串，整数，浮点数，虚数，布尔型。

集合类型：列表（List),元组（Tuple),字典（Dictionary 或 Hash)

Python 的数据类型:列表（List)

List 的定义。

或者

```
aList = [23]    bList = [1,2,3]
```

List 的使用。可以像 c 语言中数据一样引用 list 中的元素。

```
print bList[1]
```

List 常用操作：append, del, + ,*, len(list)

```
[0] * 5
```

Python 的数据类型:列表（方法)

Table 3.3. 列表对象支持的方法(演示）

append(x) count(x) extend(L)
Index(x) insert(i,x) pop(x)
remove(x) reverse() sort()

Python 的数据类型:元组（Tuple)

Tuple 的定义。

```
aTuple = (1, 3, 5)
print aTuple
```

List 的使用。

1. 元组可以用方括号括起下标做索引
2. 元组一旦创建就不能改变
3. 列表大部分操作同样适用于元组

Python 的数据类型:字典（Hash)

字典是一个用大括号括起来的键值对，字典元素分为两部份，键(key)和值。字典是 python 中唯一内置映射数

据类型。通过指定的键从字典访问值。

字典的使用:

```
a = {„a‟:‟aa‟, „b‟:‟bb‟}
a[„c‟]=„cc‟
a.has_key(„a‟)
```

Python 的数据类型:字典（常用方法)

字典的常用方法（演示）：

has_key(x) keys() values()
items() clear() copy()
update(x) get(x[,y])

Python 控制语句 if

Python 支持三种不同的控制结构：**if**，**for** 和 **while**，不支持 **C** 语言中的 **switch** 语句。

(1)if 语句的用法：

```
if EXPRESSION1:
    STATEMENT1
elif EXPRESSION2:
    STATEMENT2
else:
    STATEMENT3
```

Python 控制语句 for

for 语句的用法：

```
mylist = "for statement"
for word in mylist:
    print word
else:
    print "End list"
```

Python 控制语句 while

while 语句的用法：

```
a = 0
while a > 5:
        a = a + 1
        print a
else:
        print "a's value is five"
```

Python 循环中的控制语句

循环中的控制语句

break: 终止当前循环

continue: 终止本次循环

pass: 什么事都不错

Python 函数

函数定义：

```
def function_name(arg1,arg2[,...]):
        statement
          [return value]
```

函数名：

1. 函数名必须以下划线或字母开头，可以包含任意字母、数字或下划线的组合。不能使用任何的标点符号；

2. 函数名是区分大小写的。

3. 函数名不能是保留字。

Python 函数

作用域：Python 使用名称空间的概念存储对象，这个名称空间就是对象作用的区域， 不同对象存在于不同的作

用域。下面是不同对象的作用域规则：

1. 每个模块都有自已的全局作用域。

2. 函数定义的对象属局部作用域，只在函数内有效，不会影响全局作用域中的对象。

3. 赋值对象属局部作用域，除非使用 global 关键字进行声明。

LGB 规则

大多数名字引用在三个作用域中查找：先局部(Local)，次之全局(Global)，再次之内置(Build-in)。

Python 函数

函数的参数的分类：

默认参数：**def function(ARG=VALUE)**

元组参数：**def function(*ARG)**

字典参数：**def function(**ARG)**

一些规则：

1. 默认值必须在非默认参数之后；

2. 在单个函数定义中，只能使用一个 tuple 参数（*ARG）和一个字典参数（**ARG）。

3. tuple 参数必须在连接参数和默认参数之后。

4. 字典参数必须在最后定义。

Python 模块

模块：模块可把一个复杂的程序按功能分开，分别存放到不同文件中，使程序更容易维护和管理。在 Python 中

的模块是一个以.py 结尾的 Python 代码文件。可通过 import 命令输入，如：

（和中语句似乎相似

```
import syscinclude)
```

该 import 语句共执行三步操作：

1. 创建新的名称空间（namespace），该名称空间中拥有输入模块中定义的所有对象；
2. 执行模块中的代码；
3. 创建该名称空间的变量名。

Python 模块

import 的使用:

```
import ftplib as ftp
from ftplib import FTP
```

Python 脚本与模块

python 脚本和模块都是一个以.py 结束的文件，那程序是如何判

断一个.py 文件是作为脚本还是模块呢？关键

是一个名为__name__的变量，如果它的值是__main__，则是作为脚本直接运行，否则是做为模块运行的。

```
if __name__ == “__main__”:
    main()
```

Python 包(package)

我们可以把几个功能相近的模块组成一个 Python 包，存放到一个目录结构中，通过输入包的路径来调用对对象。

例子：

```
/WebDesign
    __init__.py
    design.py
    draw.py
```

其中__init__.py 是包的初始化文件，可以为空，但是必不可少的。

可以以下列方式引用 design 模块：

```
import WebDesign.design
```

Python 类

一个简单的例子：

```
#!/usr/bin/python
#-*- encoding:utf-8 -*-
```

定义一个类

```
class test: #test
```

这是一个测试类。在类中定义一个属性

```
    desc = "" #desc
```

对象构造函数，初始化类

```
def __init__(self,name1): #
    self.name1 = name1
```

在类中定义一个方法

```
def show(self,name2): #show()
    print "hello world"
    print 'name1:',self.name1
    print 'name2:',name2
```

这是传递给的值生成类的实例对象

```
obj = test(„name1“) #test
```

调用类中的属性

```
print obj.desc #desc
```

这是传递给的值调用类中的方法

```
obj.show('name2') #show()
```