**Python 中个括号代表的数据类型**

**（）小括号 元组**

**[ ] 中括号 列表**

**{} 大括号 字典**

**Python 中语句块的结束 是根据 缩进层次来判断的**

**当两个变量同时应用同一个列表的时候 改变一个变量 另一个变量同时改变**

**Example： names = [‘fate’,’a’,’b’];**

**N = names ;**

**N[0] = ‘ate’;**

**Print n;**

**Print names**

**Python 中交换两个值**

**（a，b） = (b,a) 等同于 a,b=b,a**

**等同于**

**Temp =b;**

**B =a ;**

**a = temp;**

**最大递归深度**

**经典递归 阶乘和二分查找**

# 引言汇总

ispace() 判断字符串是否为空格组成。

# 第一章 基础知识

1. 安装Python [www.python.org/download](http://www.python.org/download)
2. Windows Python2.5.x windows installer
3. Linux和Unix 大部分已经默认存在Python 在命令行 输入python 启动交互式python解释器
4. Python 不用分号 表示一行的结束 默认加上也没问题
5. 算法 告诉计算机做什么
6. 数字和表达式

默认为整数除法，参与除法的两个数中有一个数为浮点数 则运算结果为浮点数

如果希望Python 只执行普通的除法 可以加上 from \_\_future\_\_ import division

Python 提供另外一种 双斜线 实现整数除法 //

取模运算 %

幂运算 \*\* 2\*\*3 相当于 2的三次方 pow(2,3) 比取反的运算级高

**变量:**  x=3 在Python 中变量不需要 加$ 变量包括 数字 英文字母 下划线 不能以数字开头。**变量不能用关键字命名；**

表达式 可以比喻成一件事 语句 就是做一件事 print语句 和 赋值语句

**获取用户输入:**

Input (“what’s your telephone”)

raw\_input();

If 1==2 : print no;

**函数:**

像pow abs 这种标准函数称为内建函数。

**模块：**可以把模块想象成导入python的扩展功能 需要使用命令import 完成

Import math

Math.floor(1)

From math import floor 这种方法不能有同名函数

Ceil 大于或等于它的最小整数

可以使用变量 来引用函数 foo = math.floor PHP也可以

cmath 复数模块 此处跳过了

\_\_future\_\_ python 时光机 非标准python功能 可能在以后引入的功能

**新建可运行的程序：**以.py 结尾

若想直接运行python代码 可以再脚本的 头部 加上 #！/usr/bin/python2

**注释：**# PHP中为 # /\*\*/ //

**转义：** 在python中 双引号并不解释变量和单引号没有什么区别 \

**字符串：**长字符串和原始字符串，在python中 连着写两个字符串会默认拼接在一起如 “let’s go ” ‘”hello world”’ 变量拼接 类似 JS 用+ 号拼接

Str函数 它会把值转换为合理形式的字符串 reper函数 以合法的Python表达式的形势来表示值

Number = 42

Print ‘this is test’+number 这样是报错的 PHP支持这样写

Print ‘this is test’+`number` 这样是正确的 在python3.0 中已不使用 反引号 ``

print 'this number is '+repr(number) 应保持这样的代码

str ,repr,反引号 是 将python值转换为字符串的三种方法

raw\_input 是读取用户输入的原始字符串

长字符串 ‘’’ 类似 PHP的定界符 <<<# #;

原始字符串 print r’C:\\nwindows’ 请注意 r 原始字符串 最后一个字符不能使\ 除非对反斜杠进行了转义

**小结：算法，表达式，变量，语句，函数，模块程序，字符串。**

# 第二章 列表和元组

数据结构：通过某种方式组织在一起的数据元素的集合。

最基本的数据结构是序列，序列中每个元素被分配一个序号即元素的位置，也成为索引。第一个索引是0，第二个索引是1 以此类推。

Python 包含6中内建的序列。列表，元组，字符串，Unicode字串，buffer对象和xrange对象。

列表可以修改，元组不能修改。内建函数一般会返回元组。使用元组作为字典。

Xl = [‘fate’,27] 序列用中括号表示。序列也可以包含其他的序列。

容器的数据结构。序列和映射是两类主要的容器。

1. 通用的序列操作：加，乘，索引，分片，检查某个元素是否属于序列的成员。迭代。
2. 索引：xl = ‘abc’;

Xl[0]

字符串就是一个由字符组成的序列。使用复数索引，Python会从右边开始

Xl[-1

字符串字面值（序列字面量亦可）能够直接使用索引，而不需要一个变量引用。

‘hello’[4] o

>>> raw\_input("what's your name ")[3]

what's your name Fate

'e'

1. 分片：分片通过冒号隔开的两个索引来实现。

Tag = [‘abcdeefg’]

Tag[2:5] 分片的索引边界 第一个边界包含在分片内 第二个不包含在分片内

只要分片中最左边的索引比它右边的晚出现在序列中，结果就是一个空序列

优雅的捷径 tag[-3:] tag[:3]可以置空 复制一个序列 可以两边都置空[:]

1. 更大的步长 tag[0:6:2] 步长不能为0 但是可以是复数。将从右到左提取。

>>> numbers = [1,2,3,4,5,6,7,8,9,10]

>>> numbers[0:8:-2]

[]

>>> numbers[0:8]

[1, 2, 3, 4, 5, 6, 7, 8]

>>> numbers[8:0:-2]

[9, 7, 5, 3]

>>>

1. 相加：通过使用加运算符可以进行序列的连接操作。

[1,2,3] +[4,5,6] -> [1,2,3,4,5,6]

列表和字符串是无法连接在一起的 尽管它们都是序列 两种相同类型的序列才能进行连接操作。

1. 乘法：用数字乘以一个序列会生成新的序列，而在新的序列中，原来的序列被重复X次

>>> 'python'\*5

'pythonpythonpythonpythonpython'

[42]\*10 [42,42,42,42,42,42…]

NONE python内建值 空列表 [] 初始化 [NONE]\*10

1. 成员资格：检查一个值是否在序列中，可以使用in 运算符

>>> user = raw\_input("what's your name:"); user in users;

what's your name:fate

True

>>>

If [a,b] in [[a,b],[c,d]]: print yes

1. 长度，最大值，最小值

Len([1,2,3]) max([4,5,6]) min([7,8,9])

1. 列表 列表专门的方法

List 函数 把序列类型转换成列表 list(‘hello’) = [‘h’,’e’,’l’,’l’,’o’]

列表的基本操作

1. 元素赋值:x = [1,1,1]

X[1] = 2;

1. 删除元素

names = [‘fate’,’aa’,’bb’];

Del names[0]; del 不仅仅用于 删除列表元素 还能 用于删除其他元素。

1. 分片赋值：

Name = list(‘python’);

Name[-3:] = list[‘php’];

分片赋值时，可以使用与原序列不等长的序列将分片替换

Name = list(‘perl’);

Name[1:] = list(‘ython’);

分片赋值时，可以在不需要替换任何原有元素的情况下插入新的元素。

Numbers = [1,5];

Numbers[1:1] = [2,3,4];

Numbers -> [1,2,3,4,5,]

分片赋值时 删除

numbers = [1,5];

numbers[1:1] = [2,3,4]

numbers[1:-1] = []

print numbers;

列表的方法：

方法是一个与某些对象紧密联系的函数。对象被放在方法名之前 并用点好连接

1. append();用于在列表的末尾追加新的对象。直接修改原列表 而不是简单地返回一个修改过的新列表。

numbers.append(6);

1. count() 统计某个元素在列表中出现的次数。

numbers.append('6,6');

print numbers;

print numbers.count(6);

（3） extend 方法可以在列表的末尾一次性追加一个序列的多个值。可以扩展原有的列表。修改了 被扩展的序列 原始的连接操作 会返回一个全新的列表 不会改变原有的列表。

numbers.extend([2,3,4]);

（4）index 方法用于从列表中找出某个值第一个匹配项的索引位置。

numbers.index(4)

（5）insert 方法 用于将对象插入到列表中 只能插入一个元素

Numbers.insert(3,’four’) 第一个参数为列表索引 第二个参数为要插入的对象

（6） pop 方法 移除列表中的一个元素 默认是最后一个 并且返回该元素的值

X = [1,2,3]

x.pop();

x.pop(0) 0 指定索引位置。

利用 pop 方法可以实现一种常见的数据结构 栈 LIFO last in first out

入栈 append 出栈 pop

队列 FIFO

( 7 ) remove 移除列表中某个值的第一个匹配项。

Numbers.remove(2);

（8） reverse 方法 将列表中的元素反向存放。 返回一个迭代器对象

Numbers.reverse();

（9）sort

Y = x[:]

y.sort();

x = y

y.sort();

numbers.sort(len)

numbers.sort(key=len)

numbers.sort(revers=false)

元组：不可改变的序列 与列表唯一的不同 是不可以修改

创建元组 ： 1,3,4用逗号分隔一些值 自动创建了元组

元组 用圆括号括起来

空元组：（）

一个值得元组：（1，）

只有圆括号 没有 逗号是没有用的

元组 可以映射中当做键使用 而列表不行

元组作为很多内建函数和方法的返回值使用

zip函数接受任意多个（包括0个和1个）序列作为参数，返回一个tuple列表

# 第三章 字典

字典唯一内建的映射类型。

## 3.1 创建和使用字典

Phonebook = {‘name’:’Fate’,’phone’:13466687724}

Dict 函数

Items = [(‘name’,’fate’),(‘phone’,13466687724)]

通过关键字参数来创建字典

Dict(name=’Fate’,phone=13466687724)

## 3.2 字典基本操作

与序列的保持一致

区别 成员资格 k in d 查找的剑 列表为 v in l

在字典中检查键的成员资格比在列表中检查值得成员资格更高效

字典的格式化字符串 在每个转换说明符中的%字符后面，可以加上键（用圆括号括起来的）

‘my name is %(name)s’ % phonebook

## 3.3 字典方法

（1）clear() 清空字典

（2）copy() 浅复制 deepcopy() 深复制 理解两个的区别

From copy import deepcopy

（3）fromkeys 使用给定的键建立新的字典，每个键都对应一个默认的值 None

Fromkeys(字典，默认值)

（4）get 更宽松的方式访问字典的方法。若没有返回默认值 默认值为none

D.get(键名)

（5）has\_key(键名)

（6）popitem() 将字典的一组键值对 返回为元组

（7）items keys pop popitem setdefault update

# 第四章 字符串

1. **字符串的基本操作。**

所有标准的序列操作 对字符串同样适用 只是 字符串是属于元组不能改变值。

1. **字符串格式化**

字符串格式化操作符 % 在%的左侧放置一个字符串，右侧放置希望被格式化的值。可以是一个值 如果是多个值可以使用 元组

format\_string = "hello %s %s";

format\_values = ('world','!');

print format\_string % format\_values

如果 使用列表或者其他序列代替元组，那么序列会被解释为一个值。只有元组和字典可以格式化一个以上的值。

被转换字符串的%s部分称为转换说明符 如果格式化字符串例包括百分号 必须使用 %%

format\_pi = "Pi with three decimals:%.3f";

from math import pi

print format\_pi % pi

格式化操作符的右操作数可以是任意类型，如果右侧是元组或字典的话则序列的每一个元素都会被单独格式化。

如果元组作为转换表达式的一部分存在，必须用圆括号括起来。

%字符 标记转换说明符的开始

转换标志（可选）：- 左对齐 + 转换值之前要加上正负号 “”空白字符表示正数之前要保留空格。0表示转换值若位数不够则用0填充。

最小字段宽度（可选）：转换后的字符串至少应该具有该值指定的宽度。如果是\*则宽度会从值元组中读出。

点（.）后跟精度值（可选）：如果转换的是实数，精度就表示出现在小数点后的位数

转换类型 见 46页。

1. **模板字符串**

From string import Template Template 区分大小写 小写会报错

S = template(“ sting is $s”);

s.substitute(s=’nicai’’) 应该不会改变s 字符串 返回新的字符串

如果替换字段为单词的一部分 那么参数名就必须用括号括起来

S = Template(“It’s ${x}tastic”)

可以使用$$插入美元符号

除了关键字参数外 还可以使用字典变量提供值/名称对

D[‘s1’] = ‘s1’;

D(‘s2’)=’s2’;

S.substitute(d)

1. **字符串方法**

String 模块的常量

string.digits 包含0-9 的字符串

string.letters 包含所有字母的字符串 包含大小写 python3.0 以后移除了 和地区有关

string.lowercase 包含所有小写字母的字符串。

String.uppercase 所有大写字母

String.printable 包含所有可打印字符

String.punctuation 所有标点的字符串。

1. Find 方法

在一个较长的字符串查找字串 若有返回最左端的索引。若没有返回-1

S.find(‘aa’,0,1) 第二个参数查找起始位置 三个为结束位置。

1. join 方法

split方法 逆方法 用来连接序列中的元素。

需要连接的序列元素都必须是字符串。

print '/'.join(['c:','windows','progame']);

1. lower 方法

返回字符串的小写

1. title () 首字符大写
2. replace 替换

s.replace(‘is’,’am’)

1. split 分割

将字符串分割成序列

‘1+2+3.split(‘+’)

1. strip 去除字符串两侧的空格的字符串 也可以指定要去除的字符串
2. translate 用的时候在研究

# 第五章 语句

列表推导式,循环语句,条件语句。

Print 如果在结尾处加上逗号，那么语句会与后一条语句在同一行打印。

as 给要导入的模块起别名

import math as foobar

## 5.1 语句块

在代码前放置空格来缩进语句即可创建语句块，使用tab字符也可以缩进语句块。而一个tab字符位置为8个空格。

在python中使用冒号：来标识语句块的开始。

## 5.2 赋值语句

**序列解包或递归解包**

X,y,z =1,2,3 其实 （x,y,z）= (1,2,3)

当函数或者方法返回元组，这个特性尤其有用。

解包的序列中元素数量必须和放置在赋值符号=左边的变量数量完全一致

X,y,\*z = [1,2,3,4,5,6]

x->1 y->2 z->3,4,5,6

**链式赋值**

X = y= somefunction(); 等价

X = somefunction() x和下面的y不见得等价

Y = somefunction();

## 5.3 条件语句

下面的值在作为布尔值表达式的时候 会被解释器看做假。

False None 0 “” () [] {}

If true:

Elif:

断言

Age =10

Assert 0<age<100

## 5.4 循环语句

#while 循环

x=1

while x<100:

x+=1;

print

#for循环

numbers = range(0,11) #内建的范围函数 0-10 分片？ 和xrange的区别？

for number in numbers:

print number;

#遍历元组

dict = {('a','b'),('c','d')}

#for k,v in dict:

#print k+':'+v

#遍历列表 是遍历值

#遍历字典 是遍历键

#列表推导式 利用其他列表创建新列表 类似于数学中的集合推导式

print [x\*x for x in range(10)]

print [x\*x for x in range(10) if x%3==0]

[(x,y) for x in range(0,11) for y in range(11,21)]

girls = ['alice','bernice','clarice']

boys = ['chris','arnold','bob']

#方法1

[b+'+'+g for b in boys for g in girls if b[0]==g[0]]

#方法2 更效率的方法

letterGirls = {}

for girl in girls:

letterGirls.setdefault(girl[0],[]).append(girl)

[b+'+'+g for b in boys for g in letterGirls[b[0]]]

#pass del exec

#pass 是为了 解决空 代码块 什么都不执行

#del 不仅会移除一个对象的引用 也会移除那个名字本身

x =1

del x

x = ['hello','world']

y = x

y[1] = 'Fate';

del x

#这时候y并没有删除 删除的只是名称 而没有删除列表本身 所以y还是存在的

#某个值不在的时候 python解释器会负责内存的回收

#exec 动态执行 python语句 可以指定命名空间 即作用域 但不会有返回值

scope = {}

exec "sqrt=1" in scope

#eval 会计算python表达式 并返回结果值 可以指定命名空间内的变量print x

# 第六章 抽象

参数，作用域

## 6.1 懒惰即美德

斐波那契数列（任一个数都是前两数之和的数字序列）

numbers = int(raw\_input("please input your want number:"));

fibs\_list = [0,1]

for i in range(numbers-2):

fibs\_list.append(fibs\_list[-1]+fibs\_list[-2]);

print fibs\_list

## 6.2 创建函数

内建的callable函数可以用来判断函数是否可调用。

From math import sqrt

Callable(sqrt) 在Python3.0中不再可用 需要使用表达式 hasattr(func,\_\_call\_\_)

定义函数：

Def fibs\_list(number):

‘这是一行注释’

for i in range(numbers-2):

fibs\_list.append(fibs\_list[-1]+fibs\_list[-2]);

return fibs\_list

fibs\_list.\_\_doc\_\_ \_\_doc\_\_ 是函数属性

def test():

'this is test define function '

print 'this is test'

return

test()

help(test)

在python 中 如果函数没有返回值 默认是返回none

names = ['fate','a','b'];

def change\_list(list):

list[0] = 'changed';

change\_list(names);

print names;

使用函数改变数据结构是一种将程序抽象化的好方法.

def storage\_name(name):

name\_list = name.split();

if len(name\_list)==2:

name\_list.insert(1,'');

dictionary['first'].setdefault(name\_list[0],[]).append(name);

dictionary['middle'].setdefault(name\_list[1],[]).append(name);

dictionary['last'].setdefault(name\_list[2],[]).append(name);

return

def search\_middle\_name(middle\_name):

return dictionary['middle'][middle\_name];

def storage\_name(\*full\_names):

label\_tuple = ('first','middle','last');

for full\_name in full\_names:

name\_list = full\_name.split();

if len(name\_list)==2:

name\_list.insert(1,'');

zip\_tuple = zip(label\_tuple,name\_list);

for label,name in zip\_tuple:

people = search\_name(label,name)

if people:

people.append(full\_name)

else:

dictionary[label][name] = [full\_name]

return

def search\_name(lable,name):

return dictionary[lable].get(name);

在C++中重新绑定参数并且使这些改变影响到函数外的变量很平常的事情，但是在python中这是不可能的。(python中没有引用传递？)

如果真的想改变参数,那么将值放置在列表中

#位置参数函数

def hello\_one(greeting,name):

print '%s,%s' % (greeting,name);

def hello\_two(name,greeting):

print '%s,%s' % (name,greeting);

#关键字参数函数

def hello\_three(name="Fate",greeting="hello"):

print '%s,%s' % (greeting,name)

#收集位置参数函数 带了\*号后 会把输入的参数 放到一个元组中 收集其余的位置参数

def hello\_four(\*name):

print name

def hello\_five(greeting,\*name):

print '%s,%s' % (greeting,name)

#收集关键字参数函数 带了\*\* 、会把输入的参数 放到一个字典中

def hello\_six(\*\*name):

print name

hello\_six(name1='a',name2='b',name3='c')

#收集参数的逆过程

params = ('hello','Fate');

def hello\_seven(greeting,name):

print greeting+','+name;

如果在提供了greeting参数后 没有其他参数 那么name就是一个空元组

**关键字**：文档化函数，函数修改参数，函数不可修改参数，关键字参数和默认值，收集参数 参数收集的逆过程。

## 6.3 作用域

内建的vars函数可以返回这个字典

X=1

The\_real = vars();

The\_real[‘x’] = 2;

不可见字典 叫做命名空间或者作用域。除了全局作用域，每个函数调用都会创建一个新的作用域。

Globals 函数获取全局变量值。

Globals()[‘变量名字’] 获取全局变量 vars()[‘变量名字’] 获取局部变量的名字

在函数体内声明变量为全局变量：使用关键字 global

def hello\_night():

global name;

name = "Fate\_changed";

hello\_night();

print name;

# 第七章 更加抽象

多态，封装，方法，特性(属性)，超类(父类)，继承。

## 7.1 多态

多态来自希腊语。多态意味着就算不知道变量所引用的对象类型是什么，还是能对它进行操作。也会根据对象类型的不同而表现出不同的行为。

鸭子类型（duck typing）

## 7.2 封装

程序中的其他部分隐藏对象的具体实现细节二代原则。

## 7.3 继承

## 7.4创建自己的类

\_\_metaclass\_\_ = type