**对学过的对象类型做个归纳整理：**

能够索引的，如list/str，其中的元素可以重复。

可变的，如list/dict，即其中的元素/键值对可以原地修改。

不可变的，如str/int，即不能进行原地修改。

无索引序列的，如dict，即其中的元素（键值对）没有排列顺序。

# 数值int:不可修改

# 字符串str:不可修改

# 列表list:元素可修改

[1,”22”]

List.pop(index)

List.remove(x)

# 元组tuple:元素不可修改

(1,”22”)

索引

切片

Tuple()/list()

写保护/遍历/字符串格式化输出/字典中的key（要求不可变）

# 字典dict元素可修改

{“1”:”22”}

Key为不变对象，Value可变

不能索引/切片

Dict.Copy()

Dict.Deepcopy()

Dict.Clear()

Dict[key]

Dict.Get(key[,d])

Dict.setdefault(key[,d])

Dict.pop(key)

Dict.popitem(key)

Dict1.update(dict2)

Dict.has\_key()

# 集合set：元素可修改

([1,”22”])

Set.add(x)

Set.update(set2)

Set.pop()

Set.remove(obj)

Set.discard(obj)

Set.clear()

Set.Frozenset(obj) --- 不可修改的集合

Set < set2 等同于Set.issubset(set2)

Set == set2

Set | set2 等同于 set,union(set2)

Set & set2 等同于 a.intersection(set2)

a.difference(set2)

a.symmetric\_difference(set2)

# 运算符

算术运算符

+

\_

\*

/

% 取余

\*\* 幂

// 取商的整数

比较运算符

==

！=

>

<

>=

<=

逻辑运算符

布尔运算：and or not

# 语句

Print

Import：from math import pow as pingfang; from math import \*

赋值语句：a,b = b,a(a b互换)

循环语句

1. for循环

for 循环规则：

操作语句

条件语句

If 条件1：

语句块1

Elif 条件2：

语句2

Elif 条件3：

语句3

Else：

语句4

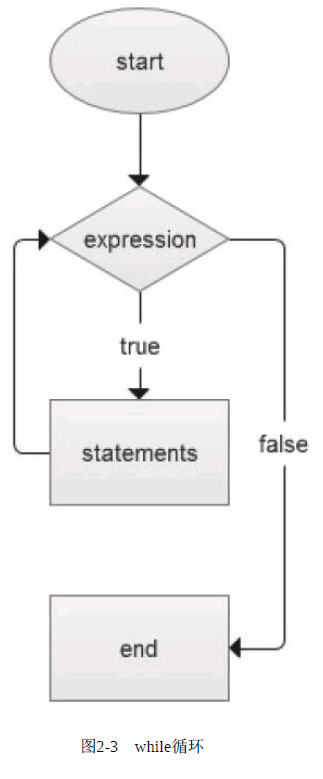
Range(start,stop,step)

Zip()

Enumerate()

列表解析

1. While循环



Break

Continue

While...else

For...else

无条件分支语句

文件：

读取文件

f = open(“filepath/filename.后缀”)

创建文件

nf = open(“filepath/filename.后缀”,权限；r,w,a,r+,w+,a+,b)

nf.write(content)

关闭文件

nf.close()

with open(“file”,”权限”) as nf

文件状态

Import os

Os.stat(“file”)

Nf.read/readline/readlines()

读取大文件

Import fileinput

Fileinput.input(“file”)

换行

Seek(offset[whence])

# 迭代

循环（loop），指的是在满足条件的情况下，重复执行同一段代

码。比如，while语句。

迭代（iterate），指的是按照某种顺序逐个访问对象中的每一项。

比如，for语句。

递归（recursion），指的是一个函数不断调用自身的行为。比如，

以编程方式输出著名的斐波纳契数列。

遍历（traversal），指的是按照一定的规则访问树形结构中的每个

节点，而且每个节点都只访问一次。

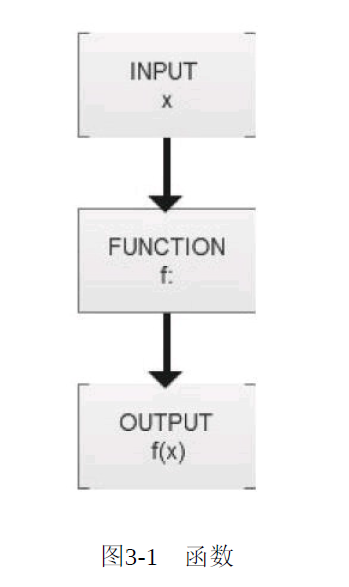
（list.iter()）.next()

文件迭代

文件是可迭代对象，无需iter()转换

**列表解析也是迭代工具**

# 函数



定义：

Def 函数名（参数1、参数2...）：（参数可以赋值默认值，调用时有传值则替换）

函数体（语句块）

函数命名

文件名：全小写，可使用下画线。

函数名：小写，可以用下画线风格单词以增加可读性。

函数的参数：如果一个函数的参数名称和保留的关键字冲突，通常

使用一个后缀下画线。

变量：变量名全部小写，由下画线连接各个单词。

关于命名的问题，还有不少争论，最典型的是所谓匈牙利命

名法、驼峰命名等

http://legacy.python.org/dev/peps/pep-0008/#prescriptive-namingconventions

**函数调用**

1. 按参数位置顺序赋值
2. 调用时直接给参数赋值（此时无需考虑顺序）

#\*\*\*冒号使用场景：if/for/while等语句，函数定义\*\*\*

#\*\*\*\*\*\*

L=[]

for i in range(1,11):

L.append(i\*\*2)

Print L

L = [i\*\*2 for i in range(1,11)]

L=[]

for i in range(1,11):

if i > 4:

L.append(i\*\*2)

Print L

L = [i\*\*2 for i in range(1,11) if i > 4]

L = [(a,b) for a in L1 and for b in L2 ]

九九乘法表：

print('\n'.join([''.join(['%s\*%s=%-2s '%(y,x,x\*y) for y in range(1,x+1)])for x in range(1,10)]))

返回值用return

#! /usr/bin/env python

# coding:utf-8

def fib(n):

result = ['a', 'b']

for i in range(n-2):

result.append(result[-2]+'+'+result[-1])

return result #返回值用return

if \_\_name\_\_ == "\_\_main\_\_":

lst = fib(10)

print lst

return还可中止之后续语句执行

# 参数和变量

1、局部变量/全局变量

2、命名空间（作用域）

3、参数收集

\*args 得到tuple格式的多参数

\*\*kargs 得到字典格式的多参数

可一起使用 \*args \*\*kargs

也可使用变量bars赋值为\*args或\*\*kargs，再传给函数：赋值数目要和形参一致

常用函数参数定义方式：

Def func(p1,p2,p3,.....)---调用时按顺序赋值

Def func(p1=value1,p2=value2,p3=value3,.....)---调用时无需考虑顺序

Def func(\*args)---适用于不知道参数个数时

Def func(\*\*args)---调用时必需arg=value

**特殊函数**

递归：函数内调用自己

**编程范型**

1、函数式编程

filter、map、reduce、lambda、yield

* 1. lambda

lambda arg1,arg2,arg3,…:expression using args

exp:lambda x,y:x+y;(lambda x:x\*\*2)(4)

* 1. map

map(func,seq1(,seq2))

exp:

items = [1,2,3,4,5]

def sqr(x):

return x\*\*2

map(sqr,items)

* 1. filter
  2. reduce

reduce(function, iterable, initializer=None)

* 1. yield