### 变量和数据类型

* 1. 布尔类型

and 与运算

or 或运算

not 非运算

0 ，空字符串‘’ ，None 为False

其他数值、非空字符都是True

|  |
| --- |
| a=’t’  b=’e’  #若a为True，与运算结果取决于b,则结果将返回b的值；若a为False，与运算结果一定为False，短路计算，不考虑b了，所以返回a的值  print(a and b) # 输出e  print(a or b) # 输出t  a=’’  print a and b #输出 e |

* 1. print

print 可以打印多个变量，用逗号分开，一个逗号显示一个空格

数字跟字符串不能加号连接

### List和Tuple类型

list为有序集合。类似于php数组，支持重复元素

list=[‘a’,’b’]

* 1. list中添加新元素

list.append(‘ele’) 在list末尾添加ele元素

list.insert(index,’ele’) 把新元素ele添加到list中索引时index的位置。

insert为前插，即插入到原来index那个元素的前面，所以list.insert(-1,’ele’),会插入到倒数第二个位置。

* 1. list中删除元素

del list[2] 删除第三个元素

list.pop() 删除list最后一个元素，并返回这个元素

list.pop(index) 删除index索引位置的元素，并返回这个元素

list.remove(obj) 移除列表中第一个obj匹配项

|  |
| --- |
| L = ['apple',100,0.01,['pear','A','b'],'C']  L.remove('apple')  print(L) |

* 1. 列表支持+ \*

+：合并两个list 返回一个新的list

|  |
| --- |
| list1=[1,2,3]  list2=[100,200,300]  list3=list1+list2  print(list3)  [1, 2, 3, 100, 200, 300] |

\*：复制列表

|  |
| --- |
| list4=['a','b','c']  list5=list4\*3  print(list5)  ['a', 'b', 'c', 'a', 'b', 'c', 'a', 'b', 'c'] |

* 1. 列表长度

len(list)

* 1. 列表中的最大值，最小值

max(list)

min(list)

max(list, key, default)

求迭代器的最大值, max会for i in … 遍历一遍这个迭代器,然后将迭代器的每一个返回值当做参数传给key=func中的func（一般是lambda表达式）然后将执行结果传给key，然后以key为标准进行大小判断。

|  |
| --- |
| d1 = {'name': 'egon', 'price': 100}  d2 = {'name': 'rdw', 'price': 666}  d3 = {'name': 'zat', 'price': 1}  L1 = [d1, d2, d3]  a = max(L1, key=lambda x: x['name'])  print(a)  b = max(L1, key=lambda x: x['price'])  print(b)  结果：  {'name': 'zat', 'price': 1}  {'name': 'rdw', 'price': 666}  例2：  L = [100,30,’40’]  max(L,key=int)  当没有最大值时，必须设置默认值default=xxx |

对于字符串的比较，是从左边第一个字符开始比，按其ASKII码大小比较，若第一个字符相同，再去比第二个字符。

* 1. 列表扩展 改变原来的list

list.extend(extendList)

|  |
| --- |
| aList=[1,2,3]  b=[4,5,6]  aList.extend(b)  print(aList) 直接改变了原list |

* 1. 返回某个元素的索引

list.index(obj)

|  |
| --- |
| list = ['apple',100,0.01,['pear','A','b'],'C']  print(list.index('apple')) |

* 1. 统计某个元素在列表中有多少个

|  |
| --- |
| aList=['to','do','or','not','to','do','to']  print(aList.count('to')) |

* 1. 创建tuple

tuple也是有序列表，中文翻译为‘元祖’，一旦创建不能修改。

t = ('Adam', 'Lisa', 'Bart')

tuple也可以用t[2]这样访问，就是不能修改。

创建单元素tuple时，元素后要多加个“，”

t=(1,) 跟运算符（）以示区分

tuple的不可修改，是指向不变，但是指向的内容可以更改。

|  |
| --- |
| t = ('a', 'b', ['A', 'B'])  L[0] = 'X'  L[1] = 'Y'  print t  ('a', 'b', ['X', 'Y'])  #这里第三个元素还是指向list，只是list里的元素变了，但指向并没改变 |

1. 条件判断和循环

关于缩进，python中，相同缩进的代码视为代码块，缩进使用四个空格，不要使用Tab，更不要混合使用。

* 1. if

|  |
| --- |
| if a>b:  print(…) |

* 1. if else

|  |
| --- |
| if a>b:  …  elif a>c:  ….  else:  …. |

* 1. for循环

|  |
| --- |
| L = ['a','b','c','d']  for name in L: #name为每个元素的变量  print(name) |

* 1. while

|  |
| --- |
| while |

### dict和set类型

* 1. dict 字典

带索引的集合，形式：

|  |
| --- |
| d = {  'Adam': 95,  'Lisa': 85,  'Bart': 59  } |

访问时可以用d[‘Adan’] ,但是若这个key不存在，直接报错。

所以可以使用：

if ‘Adam’in d:

…

d.get(‘Adam’) //存在这个key，返回value，不存在，返回None

d.get(‘Adam’,’Not Found’) //第二个参数为默认值，找不到时返回

* 1. dict特点

1. 查找速度快
2. key不能重复
3. key-value序对是没有顺序的
4. 作为key的元素必须不可变，list不能作为key
   1. dict是无序的，若想让他按添加的顺序进行有序排列：

|  |
| --- |
| from collections import OrderedDict    orderDict=OrderedDict()  orderDict['a']=1  orderDict['b']=2  orderDict['c']=3  print(orderDict)  >>>OrderedDict([('a', 1), ('b', 2), ('c', 3)]) |

* 1. 字典推导式

|  |
| --- |
| students\_score={'jack':80,'james':91,'leo':100,'sam':60}  print({key:val for key,val in students\_score.items() if val>90})  类似于列表生成式:第一部分是字典的中每个元素的形式，第二部分是迭代，第三部分是if语句过滤 |

* 1. set

无序不重复的集合

s = set([‘A’,’B’,’C’]) //set()里传入一个list

‘A’ in s 返回True或False ,判断一个元素是否在set中

特点：

1. 元素必须是不可变对象
2. 无序的

向set添加元素：

s.add(‘D’) //若添加的元素已存在在set中，不会报错，但也不会再添加进去一次了

从set删除元素：

s.remove(‘D’) //若要删除的元素不存在，会报错

所以，add（）可以直接添加，remove()前需要先判断

### 函数

* 1. 定义函数

def 函数名(形参):

…

return …

* 1. 可变参数

def fn(\*args):

print args

Python解释器会把传入的一组参数组装成一个tuple传递给可变参数，因此，在函数内部，直接把变量 args 看成一个 tuple 就好了。

def fn(\*args,\*\*kwargs)

print kwargs

kwargs 关键字参数 key word arguments

kwargs 是个dict集合

### 切片

* 1. 对list切片

前包含后不包含

L[0:3] 从索引0开始，到索引3，但不包括3

L[:3] 若第一个索引是0，也可以省略

L[:] 表示从头到尾

L[::3]表示每三个元素取出一个，即隔两个取一个（三个参数）

* 1. 对字符串切片

字符串也可以看成list进行切片

'ABCDEFG'[:3] 返回结果还是字符串

### 迭代

* 1. 集合

集合是指包含一组元素的数据结构，我们已经介绍的包括：

1. 有序集合：list，tuple，str和unicode；

2. 无序集合：set

3. 无序集合并且具有 key-value 对：dict

他们都可以用for in 来迭代

* 1. for value in L

迭代取出来的永远是元素本身，不是索引

要想取索引：使用enumerate(L) L作为参数

|  |
| --- |
| L = ['Adam', 'Lisa', 'Bart', 'Paul']  for t in enumerate(L):  print t  输出：  (0, 'Adam')  (1, 'Lisa')  (2, 'Bart')  (3, 'Paul') |

enumerate将list有序集合每个元素都变成了tuple （index,value）

可以进一步简化：

|  |
| --- |
| L = ['Adam', 'Lisa', 'Bart', 'Paul']  for index,value in enumerate(L):  print index,'-',value |

index代表索引，value代表元素

* 1. zip（）

将两个list合并为一个list

|  |
| --- |
| zip([10, 20, 30], ['A', 'B', 'C'])  结果：[(10, 'A'), (20, 'B'), (30, 'C')] |

* 1. dict迭代

|  |
| --- |
| d = { 'Adam': 95, 'Lisa': 85, 'Bart': 59, 'Paul': 74 }  for i in d:  print i  结果：Lisa  Paul  Adam  Bart |

dict 迭代里的i，是key，并不是value

要想获取value，使用dict的values()方法，实际上是将所有的value值组合成一个list；

要获取key，使用dict.keys()

items() 作为字典的一个方法，字典在它前面

将dict转化成包含tuple的list

|  |
| --- |
| d = { 'Adam': 95, 'Lisa': 85, 'Bart': 59 }  print d.items()  [('Lisa', 85), ('Adam', 95), ('Bart', 59)]  然后通过  for key,value in d.items():  …  循环 |

* 1. 列表生成式

range(1,101,2) 生成一个list集合，从1到100（不含101），且间隔是1

第一个参数是从几开始，

第二个参数是到几结束（不包含），

第三个参数是每几个数取一个出来

列表生成式格式：[x\*x for x in L]

前面是列表中的元素，后面跟for循环

后面还可以跟if判断语句

[x\*x for x in L if x%2==0] 只有满足if条件才将x返回到列表

还可以循环嵌套，只要在后面继续跟for in 即可

[x+y+z for x in L for y in M for z in N]

等同于

|  |
| --- |
| A=[]  for x in L:  for y in M:  for z in N:  A.append(x+y+z) |

* 1. 生成器

通过列表生成器可以直接创建一个列表，但是受到内存限制，列表容量是有限的，若要创建一个含有100万个元素的列表，不仅占用很大的存储空间，如果我们仅仅需要访问前面几个元素，后面绝大多数的占用空间都白白浪费了。

所以，如果列表元素可以按照某种算法推算出来，那我们就可以在循环的过程中不断推算出后续的元素，这样就不必创建完整的list，从而节省大量的空间。

在Python中，这种一边循环一边计算的机制，称为生成器：generator

第一种创建方法：

把列表生成器的 [] 改成（）

|  |
| --- |
| L = [x \* x for x in range(10)]  print(L)  [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]  g = (x \* x for x in range(10))  print(g)  <generator object <genexpr> at 0x1022ef630>  print(next(g))  print(next(g))  print(next(g))  print(next(g))  print(next(g))  print(next(g)) |

我们可以直接打印出list的所有元素，但是只能一个一个打印generator的每一个元素。

使用next（generator）来获得generator的下一个返回值。

到最后一个元素之后，会抛出StopIteration错误。

改进：(for循环，因为generator也是可迭代对象)

|  |
| --- |
| g = (x \* x for x in range(10))  for n in g:  print(n)  #到最后也不会抛出错误 |

复杂的生成器：

|  |
| --- |
| def fib(max):  n, a, b = 0, 0, 1  while n < max:  yield b  a, b = b, a + b  n = n + 1  return 'done' |

若一个函数中包含了yield关键字，那么这个函数就是一个generator

|  |
| --- |
| f = fib(5)  print(f)  <generator object fib at 0x104feaaa0> |

函数是顺序执行，遇到return或者最后一行语句就返回；而变成generator的函数，每次调用next()的时候执行，遇到yield语句返回，再次执行时从上次返回的yield语句处继续执行。

|  |
| --- |
| 使用for循环来获取返回值：  for n in fib(5):  print(n) |

此时发现拿不到generator的return的返回值。若想拿到返回值，必须捕获StopIteration错误，返回值存在StopIteration的value值中：

|  |
| --- |
| while True:  try:  print(next(fib))  except StopIteration as e:  print('return :'+e.value)  break |

1. 字符串
   1. 连接

str1 + str2 两个字符串用加号连接

‘,’.join(list) 将列表的元素用，连接用join

* 1. 字符串切片

str = ‘Today is a good day’

str[0:3] 表示截取字符串第一到第三个字符 与list切片类似

* 1. 字符串相乘

str = '-'\*20 表示20个‘-’相连接

打印：--------------------

* 1. 字符串分割

phone='400-800-800-1234'

print(phone.split('-'))

>>['400', '800', '800', '1234']

复杂的分割：

phone='hello world; python, I ,like, it'

import re

print(re.split('[;,]\s\*',phone)) 正则匹配

* 1. 字符串开头和结尾处理

filename='trace.h'

print(filename.endswith('e.h')) 字符串以什么结束

print(filename.startswith('tra')) 字符串以什么开始

* 1. 字符串查找和匹配

find在字符串里查找子字符串，返回子字符串所在位置的索引，找不到返回-1

title = 'python is a good language'

print(title.find('a good')) 返回10

复杂匹配：

|  |
| --- |
| import re  mydate = '11/27/2016'  if re.match('\d+\/\d+\/\d+',mydate): #正则匹配，匹配返回True，不匹配返回False  print('match')  else:  print('not match') |

* 1. 字符串替换

普通：str.replace(原子串，替换子串)

title = 'python is a good language'

print(title.replace('a good','a bad'))

复杂：re.sub(正则，替换子串，原字符串)

import re

content = 'boys 90,girls 200'

print(re.sub('\d+','100',content)) # boys 100,girls 100

* 1. 去除字符串两端的空格、制表符、换行符

str.strip()

注意：只能去除两侧的，不能去除中间的，去除中间的要用re模块

* 1. is和==的不同

is用来比较两个对象的内存空间是否一样，是不是用的同一块空间地址

== 比较两个对象的内容是否相等

* 1. 字符串连接

join的执行效率比+号高

若是连接大规模的字符串，比如要连接10万左右的字符串的时候，join的方法效率就会快很多(甚至相差百倍)

若要连接字符串:S1+S2+S3+....+SN,由于字符串是不可变的对象,执行一次就要申请一块新的内存，这样的话在N个字符串连接的过程中，会产生N-1个中间结果,每产生一个中间结果就要申请一次内存，这样会严重影响执行效率.

而join不一样，它是一次性申请总的内存，然后把字符串里面的每一个元素复制到内存中去，所以join会快很多.

1. 注意事项
   1. python中的拷贝

a=[1,2,3]  
b=a  
b.append(4)  
print id(a),a  
print id(b),b

>>

39161288 [1, 2, 3, 4]

39161288 [1, 2, 3, 4]

用id()函数来查看对象的唯一识别号,发现a,b是一样的,原因是因为b只是a的引用，都是同一个地址，并没有实现真正的copy

注意：这里跟php有所不同，php里若b的值改变了，则意味着b不再指向a指向的内存地址了，而是指向新值得内存地址；但是在python中，b的值改变了，还是指向同一个地址，因此a的值也会跟着改变

如果你想修改一个对象，但是又不需要改动原对象,必须要引入copy模块

import copy  
a=[1,2,3]  
b=copy.copy(a)  
b.append(4)  
print 'a:',a  
print 'b:',b

>>

a: [1, 2, 3]

b: [1, 2, 3, 4]

* 1. 关于深拷贝和浅拷贝

http://python.jobbole.com/82294/