# 第四章 Python运算与流程控制循环

本节所讲内容：

4.1 Python的运算（布尔值、自增、比较、逻辑）

4.2 Python流程控制介绍

4.3 Python 推导式

4.4 深浅拷贝

## 4.1 python的运算

### 4.1.1 布尔值（bool）

True :

非0的数字，非空的（字符串、列表、元组、字典）

False:

0 ‘ ’() [] {} None

布尔值可以逻辑运算符and（且）、or（或）、not（非）来运算

and(且)运算，所有条件都符合才为True

>>> True and True

True

>>> True and False

False

>>> False and False

False

>>>

or（或）运算，只要其中有一个为True，都为True

>>> True or False

True

>>> True or True

True

>>> False or False

False

>>>

not(非)运算

>>> not True

False

>>> not False

True

>>>

空值:空值为None；  
 None不能理解为0；0是有意义的，None是一个特殊的空值；  
 变量；变量必须是大小写字母，数字和\_的组合；

非 且 或 （执行先后顺序）。

### 4.1.2 自增运算

其他语言的自增运算

Var int i

i = 0

i++ 等同后面 i = i+1

printf(i)

python

i = 10

i += 1 等同后面 i = i+1

i += 2 等同后面 i = i+2

i -= 1 等同后面 i = I-1

i \*= 1 等同后面 i = i\*1

i /= 1 等同后面 i = i/1

### 4.1.3 比较(关系)运算符

| 运算符 | | 描述 | 示例 |
| --- | --- | --- | --- |
| == | 检查两个操作数的值是否相等，如果是则条件变为真。 | | 如a=3,b=3则（a == b) 为 True. |
| != | 检查两个操作数的值是否相等，如果值不相等，则条件变为真。 | | 如a=1,b=3则(a != b) 为 True. |
| <> | 检查两个操作数的值是否相等，如果值不相等，则条件变为真。Python和在[Pascal](https://baike.baidu.com/item/Pascal)等特有方式，java和c没有,在Python3中废弃了 | | 如a=1,b=3则(a <> b) 为 True。这个类似于 != 运算符 |
| > | 检查左操作数的值是否大于右操作数的值，如果是，则条件成立。 | | 如a=7,b=3则(a > b) 为 True. |
| < | 检查左操作数的值是否小于右操作数的值，如果是，则条件成立。 | | 如a=7,b=3则(a < b) 为 False. |
| >= | 检查左操作数的值是否大于或等于右操作数的值，如果是，则条件成立。 | | 如a=3,b=3则(a >= b) 为 True. |
| <= | 检查左操作数的值是否小于或等于右操作数的值，如果是，则条件成立。 | | 如a=3,b=3则(a <= b) 为 True. |

python中的比较运算符如下表

| 运算符 | 逻辑表达式 | 描述 | 提问 |
| --- | --- | --- | --- |
| and | x and y | 布尔"与" - 如果 x 为 False，x and y 返回 False，否则它返回 y 的计算值。 | 那如果x总为False呢？ |
| or | x or y | 布尔"或" - 如果 x 是 True，它返回 True，否则它返回 y 的计算值。 | 那如果x总为True呢？ |
| not | not x | 布尔"非" - 如果 x 为 True，返回 False 。如果 x 为 False，它返回 True。 |  |

### 4.1.4 逻辑运算符

看一下输出什么：

or 和 and 顺序比较（and>or）

>>>True or False and False

True

如果先算or 就会输出：False

not 和 and 顺序比较(not>and)

>>>not False and False

False  
 如果先算and 就会输出：True

not 和 or 顺序比较（not>or）

>>>not False or True

True  
 如果先算 or 就会输出：False

无意义逻辑：

>>> True or False

True

后面如何写如何改变输出都是True

>>>False and True  
 False

后面的结果如何写如何更改都是False.

通过以上的对比逻辑运算的执行顺序 not>and>or

执行顺序(由高到低):非 且 或

看一下这个输出什么？

not False and False or True True

实战（面试题）：

闰年是公历中的名词：

普通年:能被4整除但不能被100整除的年份为普通闰年。（如2004年就是闰年，1999年不是闰年）；

世纪年:能被400整除的为世纪闰年。（如2000年是闰年，1900年不是闰年）；

year = int(input('请输入一个年份:'))  
*#判断！ 判断是不是能够被4整除！ 100不整除或者是 被400整除*if (year % 4) == 0 and (year % 100) != 0 or (year % 400) == 0:  
 print('{0}是闰年'.format(year))  
else:  
 print('{0}不是闰年'.format(year))

运行结果如下：

请输入一个年份：100

100不是闰年

## 4.2 python流程控制介绍

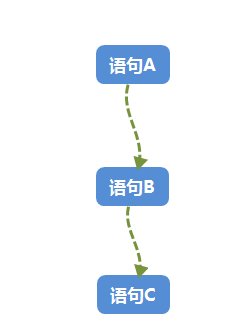
我问一下同学们，你们去五道口地铁站看美女的时候是不是要有一整套流程？美不美看什么？

流程控制:

个别的指令（或是陈述、子程序）运行或求值的顺序。

Python 代码正常情况下执行的顺序是从左到右从上到下的，但是在流程控制当中不是这样的

正常流程：

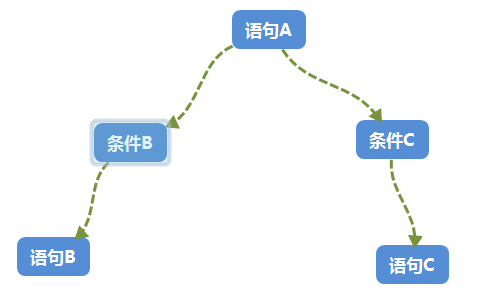


流程控制三兄弟：

If（判断），for（循环：广度循环），while（循环：深度循环）

### 4.2.1 if 语句（判断）

1、 if 语句判断语句，只有满足if条件才会执行if 下的语句块。



关键字：if elif else

elif 的 else if 的缩写，完全可以有多个elif，只有一个else

if语句格式：

if (条件1):

block 语句块

elif(条件2):

block

else:

block

有人会问 ‘ : ’有什么用？：用来区分代码之间的层次

例子：

age = 18  
if age >= 18:  
 print('你已经长大了')  
elif age >= 6:  
 print('孩子 你还小')  
else:  
 print('hello 小baby 叔叔带你去……')

在这里你一定要处理好你的判断语句，减少无意义的判断

反面案例：

age = 18  
if age >= 6:  
 print('你已经长大了')  
elif age >= 18:  
 print('孩子 你还小')  
else:  
 print('hello 小baby 叔叔带你去……')

如果不管age是不是大于6 那么 elif这个语句一直都不会执行

2、 一定要区分if elif 和 if if的区别。

num = 10

#if elif 是一个判断，一个条件执行之后就不会有第二个条件执行  
if num < 30:  
 print('< 30')  
elif num < 100:  
 print(' < 100')  
# if if 是两个不同的判断，互不影响  
if num < 30:  
 print('< 30')  
if num < 100:  
 print('< 100')

嵌套判断

内层是在外层成立的前提下成立

num = 18  
if num < 15:  
 if num > 8:  
 print('15 > num > 8')  
else:  
 print('111')  
 if num < 30:  
 print('15 < num < 30')

3、 for 循环的格式：

for 变量 in 可迭代对象(序列):

block循环语句块

什么叫可迭代对象？

我们把可以通过for...in...这类语句迭代读取⼀条数据供我们使用的对象称之为可迭代对象（Iterable）

普通的循环示例如下

#普通的range范围，print（end=‘ ’）不换行，引号中可传入自定义符号。for i in range(10): #一个参数  
 print(i,end='-')

for i in range(2,10): #两个参数  
 print(i,end='-')  
#带有步长range  
for i in range(2,10,2): #三个参数  
 print(i,end='-')

#字符串的遍历

name = 'xuegod'  
for i in name:  
 print(i)

运行结果如下：

x

u

e

g

o

d

特殊的for 循环：

a= [(1,2),(3,4)]  
for i in a:  
 print(i[0])  
#i,j = (1,2) 序列解包赋值  
for i,j in a:  
 print('这是i %s'%i)  
 print('这是j %s'%j)

运行结果如下：

1

3

这是i 1

这是j 2

这是i 3

这是j 4

4、 嵌套循环（重要）

for循环的格式：

for 条件1:

条件1满足时，做的事情1

for 条件2:

条件2满足时，做的事情

外层循环一次

内层循环一遍

例子：

打印九成九乘法：

#循环九次

for i in range(1,10):*# i =1 i = 2* for j in range(1,i+1):*#(1,2) i1,2* print('%d x %d = %d\t'%(j,i,i\*j),end='')  
 print()

注意：外层循环一次，内层循环一遍

运行结果如下：

1 \* 1 = 1

1 \* 2 = 2 2 \* 2 = 4

1 \* 3 = 3 2 \* 3 = 6 3 \* 3 = 9

1 \* 4 = 4 2 \* 4 = 8 3 \* 4 = 12 4 \* 4 = 16

1 \* 5 = 5 2 \* 5 = 10 3 \* 5 = 15 4 \* 5 = 20 5 \* 5 = 25

1 \* 6 = 6 2 \* 6 = 12 3 \* 6 = 18 4 \* 6 = 24 5 \* 6 = 30 6 \* 6 = 36

1 \* 7 = 7 2 \* 7 = 14 3 \* 7 = 21 4 \* 7 = 28 5 \* 7 = 35 6 \* 7 = 42 7 \* 7 = 49

1 \* 8 = 8 2 \* 8 = 16 3 \* 8 = 24 4 \* 8 = 32 5 \* 8 = 40 6 \* 8 = 48 7 \* 8 = 56 8 \* 8 = 64

1 \* 9 = 9 2 \* 9 = 18 3 \* 9 = 27 4 \* 9 = 36 5 \* 9 = 45 6 \* 9 = 54 7 \* 9 = 63 8 \* 9 = 72 9 \* 9 = 81

### 4.2.3 while循环

while 循环，while循环和for循环不同，for循环是一个广度遍历，而while循环是一个深度遍历，while循环，指当满足while的条件的时候，就一直循环执行while的语句块，直到不满足。

while格式

while（条件）：

条件满足时，做的事情

else：

条件不满足时做的事情。

假设：条件一直满足，那么就形成了死循环。

在Python当中默认的死循环的条件是True。

while True:  
 print('风中一匹狼')  
 ctrl+c(停止)  
a = 0  
while a <10:  
 print(a)  
 a += 1

1、死循环会阻塞程序运行

2、死循环多数用于图形化、游戏等可视化的主循环

ATM



1. while普通示例如下：

i = 1  
while i >5:  
 print('我是for')  
else:  
 print('我不是For')

运行结果如下：

我不是For

4. while 嵌套

while 条件1:

条件1满足时，做的事情1

while 条件2:

条件2满足时，做的事情

while嵌套和for嵌套的类型一致

例子：

n = int(input('请输入你想要的数字'))  
i = 1  
while i <= n:  
 j = 1  
 while j <= i:  
 print('\*',end='')  
 j += 1  
 print()  
 i += 1

运行结果如下：

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

### 4.2.4 流程控制词（面试题）

pass，continue ，break

循环是让计算机做重复任务的有效的方法。

pass写在任何缩进的语句块部分，只是占位，什么事情都不做。为了满足python的语法要求。

break语句可以在循环过程中直接退出循环，continue语句可以提前结束本轮循环，并直接开始下一轮循环。这两个语句通常都必须配合if语句使用。

要特别注意，不要滥用break和continue语句。break和continue会造成代码执行逻辑分叉过多，容易出错。大多数循环并不需要用到break和continue语句，上面的两个例子，都可以通过改写循环条件或者修改循环逻辑，去掉break和continue语句。

有些时候，如果代码写得有问题，会让程序陷入“死循环”，也就是永远循环下去。这时可以用Ctrl+C退出程序，或者强制结束Python进程。

1. pass 占位符的

name = 'xuegod'  
for i in name:  
 if i == 'e':  
 pass  
 print(i,end=' ')

运行结果如下：

x u e g o d

1. for 带有break的循环示例如下:

name = 'xuegod'  
for i in name:  
 if i == 'e':  
 break  
 print(i,end=' ')

运行结果如下：

x u

1. for 带有continue示例

name = 'xuegod'  
for i in name:  
 if i == 'e':  
 continue  
 print(i,end=' ')

运行结果如下：

x u g o d

1. while带有break的循环示例如下:

i = 0  
while i <10:  
 i = i+1  
 if i == 5:  
 break  
 print(i,end=' ')

运行结果如下：

1 2 3 4

1. while带有continue的循环示例如下:

i = 0  
while i <10:  
 i = i+1  
 if i == 5:  
 continue  
 print(i,end=' ')

运行结果如下：

1 2 3 4 6 7 8 9 10

## 4.3 推导式（面试题）

### 4.3.1 列表推导式

列表推导能非常简洁的构造一个新列表:只用一条简洁的表达式即可对得到的元素进行转换变形。

语法：[变量for 变量 in 可迭代对象]

比如现在我们要生成一个列表，列表当中的元素为0-10，首先我们不用列表推导式，该如何来实现？

result1 = []  
for i in range(10):  
 result1.append(i)  
print(result1)

那如何用列表推导式呢？

result = [i for i in range(10)]  
print(result)

分析：相比是不是代码简洁很多

那继续，如何利用列表推导式求出对象的偶数呢

result = [i for i in range(10)if i % 2 ==0 ]

那么问题来了，比如现在有一列表L = [[1,2,3],[4,5,6],[7,8,9]]

要求出1/4/7 和 1/5/9 元素，思考如何取出？

看代码：

L = [[1,2,3],[4,5,6],[7,8,9]]

#len(L)获取长度为3，第一次i = 0,L[i][0]取值下标为0 中的第一个元素

result = [L[i][0] for i in range(len(L))]

print(result)

运行结果如下：

[1,4,7]

求出1、 5 、 9

L = [[1,2,3],[4,5,6],[7,8,9]]

#len(L)获取长度为3，第一次i = 0,L[i][0]取值下标为0 中的第一个元素

result = [L[i][i] for i in range(len(L))]

print(result)

运行结果如下：

[1,5,9]

2个for循环

a=[(i,j) for i in range(1,5) for j in range(6,10)]

3个for循环

a= [(x,y,z) for x in range(2) for y in range(2) for z in range(2)]

### 4.3.2 字典推导式

字典和集合推导式是该思想的延续，语法差不多，只不过产生的是集合和字典而已。

字典推导式

dic2 ={k:v for k,v in {'name':'for','age':18}.items()}

### 4.3.3 集合推导式

集合推导式跟列表推导式非常相似，唯一区别在于用{}代替[]

L = [1,2,3,1,2,3,4]  
set1 = {i for i in L}  
print(set1)

### 4.3.4 元组生成式

a = (i for i in range(5))  
for i in a:  
 print(i)

## 4.4 深浅拷贝（面试题）

### 4.4.1 深浅拷贝定义

深拷贝：

拷贝所有对象，包括顶级对象以及嵌套对象，所以原始对象的改变不会造成深拷贝里任何子元素的改变

浅拷贝：

只拷贝顶级对象, 没有拷贝嵌套对象，所以原始数据改变，嵌套对象会改变!

应用场景：

比如在CMDB系统中，我们定义了一个报警模版call给所有的服务器使用，此时有一批特殊应用的服务器需要不通的报警参数，我们既不想单独新建模版来一个一个添加报警参数，又不想修改默认模版而影响其他机器的报警阈值。此时我们就需要用深拷贝来完成。

我们在维护服务器信息的时候，经常会要更新服务器信息，这时我们重新一个一个添加是比较麻烦的，我们可以把原数据类型拷贝一份，在它的基础上做修改。

import copy

a = [1,2,3,4,[5,6,7]]  
b = a  
print('a的id:%s,b的id:%s'%(id(a),id(b)))  
c = copy.copy(a)  
print(id(c))  
d = copy.deepcopy(a)  
print(id(d))  
a.append(8)  
a[4].append(9)  
print(a)  
print(b)

print(c)

print(d)

输出：

a:2015893011080,d:2015893011080

2015891565512

2015893011592

[1, 2, 3, 4, [5, 6, 7, 9], 8]

[1, 2, 3, 4, [5, 6, 7, 9], 8]

[1, 2, 3, 4, [5, 6, 7, 9]]

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

总结：

4.1 python的运算（布尔值、自增、比较、逻辑、）

4.2 python流程控制介绍

4.3 python 推导式

4.4 深浅拷贝

练习:

给你一个字符串例如：’abcsadfadsfgfegegegea’，经过计算返回一个字典，字典中包含每个元素当中的个数，结合if 判断，for 循环，完成本次代码！  
 例如：‘abb’ 🡺 {‘a’:1,’b’:2}

输入任意行文字，存于列表L中,当不输入任何容直接回车后结束输入

1) 打印L列表中的内容

2) 计算您共输入了几行内容

3) 计算您共输入了多少个字符

L = []  
while True:  
 text = input('请输入想要输入的内容:')  
 if text:  
 L.append(text)  
 print(L)  
 print('计算输入的几行内容')  
 len\_text = len(L)  
 print(len\_text)  
 print('开始计算总共字符')  
 num\_str = 0  
 for data in L:  
 num\_str += len(data)  
 print(num\_str)  
 if text=='':  
 break

输入一个整数（代表树干的高度)

打印出如下一棵树

输入: 2

打印:

\*

\*\*\*

\*

\*

输入:3

\*

\*\*\*

\*\*\*\*\*

\*

\*

\*

答案：

num = int(input('请输入一个整数：'))  
for i in range(1,num+1):  
 print(' '\* (num - i )+(2\*i-1)\*'\*')  
a = num  
while a > 0:  
 print((num-1)\*' '+'\*')  
 a -= 1

写一个程序，任意输入一个整数，判断这个数是否是素数(prime)

素数(也叫质数), 是只能被1和自身整除的正整数:

如: 2 3 5 7 11 等

提示:

用排除法: 当判断x是否是素数是，只要让x分别除以:

2, 3, 4, 5, .... x-1, 只要整除了，那x不是素数，否则x是素数

4.算出100~1000以内的水仙化数(Naricissistic Number)

水仙花数是指百位的3次方 加上 十位的3次方 加上 个位的3次方等于原数的数

如:

153 = 1\*\*3 + 5\*\*3 + 3\*\*3

答案:

153, 370, ....

a = 678  
print(a//100)  
print(a%10)  
print(a//10%10)