Python基础内容

第三章　运算与控制结构

目录

运算符

算术运算符，比较运算符，逻辑运算符

算术运算符

常用的算术运算符

运算符	描述	实例
+	两个对象相加	10+20 输出结果 30
–	得到负数或是一个数减去另一个数	10–20 输出结果 -10
*	两个数相乘或是返回一个被重复若干次的字符串	10 * 20 输出结果 200
/	x除以y	20/10 输出结果 2.0
%	取余，返回除法的余数	20%10 输出结果 0
**	幂，返回x的y次幂	10**2 输出结果 100
//	取整，返回商的整数部分	9//2 输出结果 4

比较运算符

常用的比较运算符

运算符	描述	实例
==	等于，比较对象是否相等	(a==b) 判断a、b是否相等
!=	不等于，比较两个案例是否不相等	(a != b) 判断a、b是否不等
<>	不等于，比较两个案例是否不相等	(a <> b) 判断a、b是否不等
>	大于，返回x是否大于y	(a > b) 判断a是否大于b
<	小于，返回x是否小于y	(a < b) 判断a是否小于b
>=	大于等于，返回x是否大于等于y	(a >= b) 判断a是否大于等于b
<=	小于等于，返回x是否小于等于y	(a <= b) 判断a是否小于等于b
is	判断两个变量所引用的对象是否相同	(a is b) 判断a是否相同
Is not	判断两个变量所引用的对象是否不相同	(a is not b) 判断a是否相同

比较运算符

比较的结果返回的是布尔值

```
In [23]: 2.3>7
Out[23]: False

In [24]: "a">"b"#字符串比较，按照字典顺序进行比较
Out[24]: False

In [25]: "ABC">"AB"
Out[25]: True

In [26]: "abc">123#比较必须在同种对象类型中进行
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-26-fa6b1571dcea> in <module>
----> 1 "abc">123

TypeError: '>' not supported between instances of 'str' and 'int'
```

比较运算符

== 和 is 用于比较两个对象 (=是赋值不是比较)

== 用来比较两个变量的值是否相等，而 is 则用来比对两个变量引用的是否是同一个对象

```
In [9]: import time#引入time模块
        t1 = time.gmtime() # gmtime()用来获取当前时间
        t2 =  time.gmtime()
        print(t1 == t2)
        print(t1 is t2)

        True
        False
```

```
In [10]: import time#引入time模块
         t1 = time.gmtime() # gmtime()用来获取当前时间
         t2 =  time.gmtime()
         print(t1 == t2)
         print(t1 is t2)
         id(t1),id(t2)

         True
         False

Out[10]: (1747822394048, 1747822393536)
```

time 模块的 gmtime() 方法用来获取当前的系统时间，精确到秒级，因为程序运行非常快，所以 t1 和 t2 得到的时间是一样的。== 用来判断 t1 和 t2 的值是否相等，所以返回 True。

虽然 t1 和 t2 的值相等，但它们是两个不同的对象（每次调用 gmtime() 都返回不同的对象），所以t1 is t2返回 False。这就好像两个双胞胎姐妹，虽然她们的外貌是一样的，但它们是两个人。

那么，如何判断两个对象是否相同呢？答案是判断两个对象的内存地址。如果内存地址相同，说明两个对象使用的是同一块内存，当然就是同一个对象了；这就像两个名字使用了同一个身体，当然就是同一个人了。

思考为什么会出现下面的结果?

```
In [17]: c = 256
         d = 256
         c == d
Out[17]: True

In [18]: c is d
Out[18]: True

In [19]: id(c),id(d)
Out[19]: (140715253192448, 140715253192448)
```

```
In [20]: c = 257
         d = 257
         c == d
Out[20]: True

In [21]: c is d
Out[21]: False

In [22]: id(c),id(d)
Out[22]: (1747821506128, 1747821506000)
```

Python 小整数与大整数小整数」和「大整数」，前者的数值范围在 [-5, 257) 之间，其余的数值均归为后者

最新版本已经变了！ 257==257

逻辑运算符

- bool()函数：对于内容为空或0的对象均输出False

```
In [10]: bool(" ")
Out[10]: True

In [11]: bool("")
Out[11]: False

In [14]: bool([])
Out[14]: False

In [15]: bool(set())
Out[15]: False

In [16]: bool(None)
Out[16]: False
```

```
In [2]: bool()
Out[2]: False

In [3]: bool(0)
Out[3]: False

In [4]: bool(1)
Out[4]: True

In [5]: bool(2)
Out[5]: True

In [6]: issubclass(bool, int)  # 判断bool 是否是int 子类
Out[6]: True
```

逻辑运算符

- bool()函数：运算过程 A(and，or) B, not (A)
- and

```
if bool(A) == False:
    return False
else:
    return bool(B)
```

- or

```
if bool(A) == True:
    return True
else:
    return bool(B)
```

- not

```
In [18]: 4 < 3 and 4 < 9
Out[18]: False

In [19]: 4 > 3 and 4 < 9
Out[19]: True

In [20]: 4 < 3 or 4< 9
Out[20]: True

In [21]: not(4 > 3)
Out[21]: False
```

逻辑运算符

运算符	逻辑表达式	描述	符号
and	x and y	并且	&
or	x or b	或者	|
not	not x	不是	

```
print(1 > 2 and 1 > 0)
print(1 > 2 or 1 > 0)
print(1==2)
print(1!=2)
print(not False)
```

```
False
True
False
True
True
```

```
In [1]: print(1 > 2 and 1 > 0)
        print(1 > 2 or 1 > 0)
        print(1==2)
        print(1!=2)
        print(not False)

        False
        True
        False
        True
        True
```

优先级

运算符说明	Python运算符	优先级	结合性	优先级顺序
小括号	()	19	无	
索引运算符	x[i] 或 x[i1: i2 [:i3]]	18	左	高
属性访问	x.attribute	17	左	^
乘方	**	16	右	|
按位取反	~	15	右	|
符号运算符	+（正号）、-（负号）	14	右	|
乘除	*、/、//、%	13	左	|
加减	+、-	12	左	|
位移	>>、<<	11	左	|
按位与	&	10	右	|
按位异或	^	9	左	|
按位或	|	8	左	|
比较运算符	==、!=、>、>=、<、<=	7	左	|
is 运算符	is、is not	6	左	|
in 运算符	in、not in	5	左	|
逻辑非	not	4	右	|
逻辑与	and	3	左	|
逻辑或	or	2	左	低
逗号运算符	exp1, exp2	1	左	

下列关于布尔代数的运算，正确的是：

【单选】以下哪个选项不是布尔表达式?

- 选项1
  `not (10 == 1 or 1000 == 1000)`
- 选项2
  `e > 5 and 4 == f`
- 选项3
  `(Z-3) > 6`
- 选项4
  `a = 2`

【单选】以下哪个选项不是布尔表达式?

- 选项1
  `4 == f`
- 选项2
  `e > 5 and f = 2`
- 选项3
  `not (10 == 1 or 1000 == 1000)`
- 选项4
  `a == 3 and (Z-2) > 5`

下列关于布尔代数的运算，正确的是：

- 选项1
  True| False =False
- 选项2
  not (True | False) | True = False
- 选项3
  True & (False & True | False) = True
- 选项4
  False | (not True | False) & True= True

2

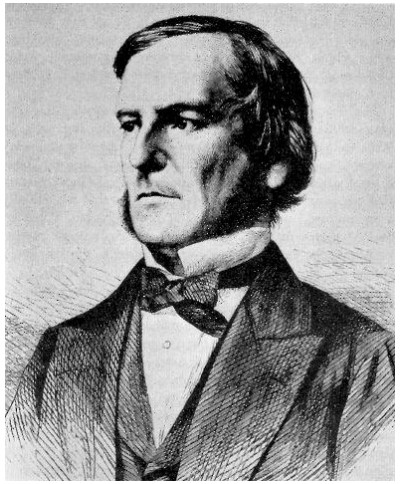
【多选】下列输出结果为True的有?

- 选项1
  `True + False + 21 == 22`
- 选项2
  `bool(0) == False`
- 选项3
  `bool('this is a test')==False`
- 选项4
  `bool(-25)==False`

1和2

true + false = true
逻辑运算，1 + 0 = 1

励 志 敬 业 榜 样



乔治·布尔（George Boole，1815.11.2～1864），1815年11月2日生于英格兰的林肯。乔治·布尔是皮匠的儿子，由于家境贫寒，布尔不得不在协助养家的同时为自己能受教育而奋斗，不管怎么说，他成了19世纪最重要的数学家之一。 尽管他考虑过以牧师为业，但最终还是决定从教，1835年他开办了自己的学校。 在备课的时候，布尔不满意当时的数学课本，便决定阅读伟大数学家的论文。在阅读伟大的法国数学家拉格朗日的论文时，布尔有了变分法方面的新发现。变分法是数学分析的分支，它处理的是寻求优化某些参数的曲线和曲面。

1847年，布尔出版了《逻辑的数学分析》（The Mathematical Analysis of Logic），这是他对符号逻辑诸多贡献中的第一次。1849年，他被任命位于爱尔兰科克的皇后学院(现National University of Ireland,College Cork或UCC)担任数学教授。1854年，他出版了《思维规律的研究》，（An Investigation of The Laws of Thought），这是他最著名的著作。 在这本书中布尔介绍了现在以他的名字命名的布尔代数。布尔撰写了微分方程和差分方程的课本，这些课本在英国一直使用到19世纪末。1857年布尔当选为伦敦皇家学会会员，不久荣获该会的皇家奖章。1864年，布尔(49岁)死于肺炎，肺炎是他在暴风雨天气中尽管已经湿淋淋的了仍坚持上课引起的。由于其在符号逻辑运算中的特殊贡献，很多计算机语言中将逻辑运算称为布尔运算，将其结果称为布尔值。

在python中运行发现：

>>> 1 in [1,0] == True # This is strangeFalse
>>> False
Python实际上在这里应用比较运算符链接。表达式被翻译成

(1 in [1, 0]) and ([1, 0] == True)
这显然是False。

这也适用于像这样的表达式

a < b < c
转化为

(a < b) and (b < c)
以上就是为何1 in [1,0] == True执行结果是False

```
>>> 1 in [1,0]            # This is expected
True
>>> 1 in [1,0] == True    # This is strange
False
>>> (1 in [1,0]) == True  # This is what I wanted it to be
True
>>> 1 in ([1,0] == True)  # But it's not just a precedence issue!
                          # It did not raise an exception on the second example.

Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    1 in ([1,0] == True)
TypeError: argument of type 'bool' is not iterable
```

'bool' object is not iterable:bool'对象不可迭代

简单语句

简单语句

- 简单语句由（逻辑上的）一行组成，例如
  - 赋值语句：

```
a = 12
```

  - import语句：

```
import math
```

  - print语句：

```
print ('Hello, world')
```

  - del语句：

```
del seq[42:] #删除序列切片
```

赋值语句:

```
In [6]: s = 1

In [7]: x, y, z = 1, 2, 3

In [8]: x
Out[8]: 1

In [9]: y
Out[9]: 2

In [10]: z
Out[10]: 3
```

```
In [11]: a = "python", "php"
         a
Out[11]: ('python', 'php')

In [12]: m = n = "python"

In [13]: m
Out[13]: 'python'

In [14]: n
Out[14]: 'python'
```

```
In [15]: x = 9
         x = x + 1
         x
Out[15]: 10

In [16]: x += 1
         x
Out[16]: 11

In [17]: m = "python"
         m += "ok"
         m
Out[17]: 'pythonok'
```

```
x
4

x, y=y, x

x
5

y
4
```

多个语句 x=4;y=5;z=6

import语句

```
In [1]: import math

In [2]: from math import pow

In [3]: from math import pow as pingfang

In [4]: from math import pow, e, pi

In [5]: from math import *
```

使用*导入模块中的所有函数

条件判断语句

条件判断语句

- 布尔值
  - 对于计算机而言，布尔值 True 和 False 就表示真 和 假
  - True 、 False 是比较显式的真和假, 在Python中以下值也会被看作是真（True）：

```
print(not False)
print(not 0)
print(not '')
print(not "")
print(not ())
print(not {})
print(not [])
print(not ())
print(not {})
print(not [])
```

```
True
True
True
True
True
True
True
True
True
True
```

条件判断语句

- if选择结构
  - 判断条件为真（True）的时候才执行:冒号后下面的语句
  - 比如现在已有一个精灵宝贝的 HP 值，而只希望当这个 HP 值大于 20 的时候才打印出来

```
HP=30
if HP > 20:#判断条件
    print('HP大于20')#执行语句
```

```
HP大于20
```

```
In [7]: HP=30
        if HP>20:#注意有：号，换行判断条件
            print("HP大于20")#自动缩进，执行语句

HP大于20
```

if选择结构

- 除了 if 语句外，还有 if-else 、 if-elif 语句

```
HP=30
if HP > 20:#判断语句
    print('HP大于20)#执行语句
else:
    print('HP小于20')#执行语句
```

```
HP大于20
```

```
HP=12
if HP > 20:#判断语句
    print('HP大于20')#执行语句
elif HP > 10:#判断语句
    print('HP大于10，小于20')#执行语句
```

```
HP大于10，小于20
```

```
if bool(conj):
    something 1
elif:
    something 2
elif:
    something 3
else:
    something 4
```

循环控制语句

while循环语句

- While循环语句的停止条件是需要设定的
  - 在确定满足条件而不确定需要的循环次数时，使用while语句是最好的选择

```
while 判断条件:
    执行语句
```

```
while bool(conj):

    something
```

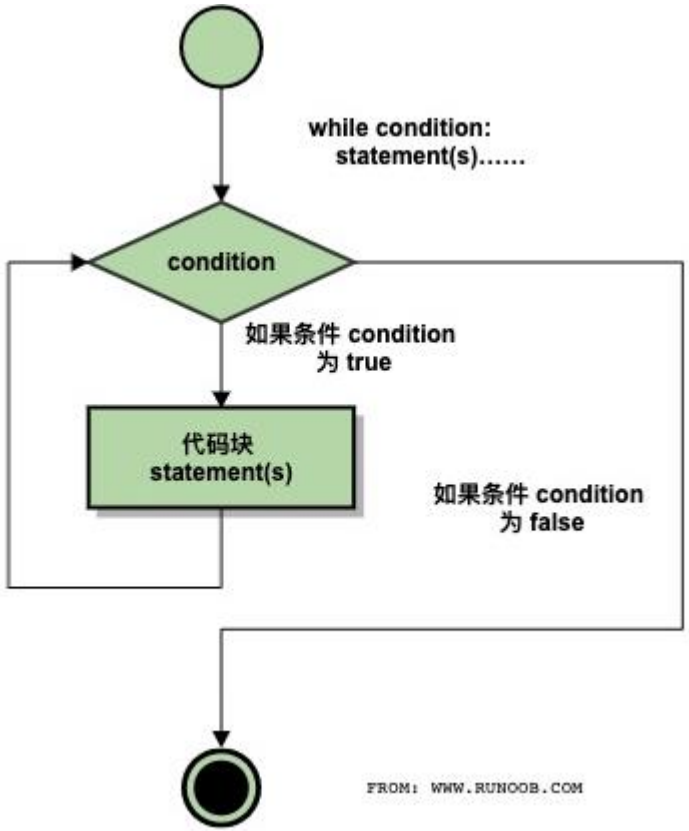
#现在有一个HP数值变量，我希望它在大于“20”的时候，逐次变小，直到等于“20”为止，注意print的位置的影响。

```
HP = 30
while HP>20:
    HP = HP - 1
print(HP)
```

```
20
```

```
In [3]: HP = 30
        while HP>20:
            HP = HP - 1
            print(HP)
```

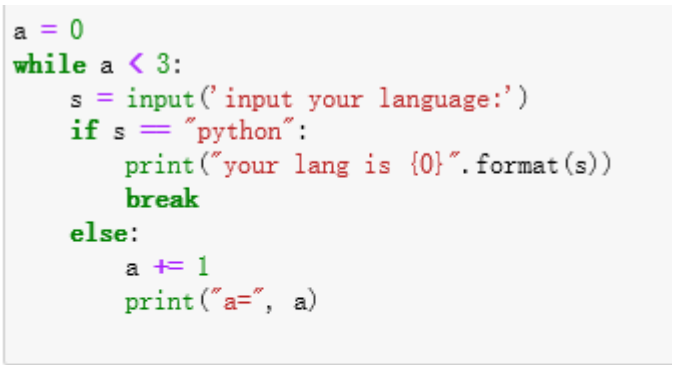
```
29
28
27
26
25
24
23
22
21
20
```



•判断条件可以是任何表达式，任何非零、或非空（null）的值均为true。
•当判断条件假 false 时，循环结束。
•执行语句可以是单个语句或语句块。

```
a = 0
while a < 3:
    s = input('input your language:')
    if s == "python":
        print("your lang is {0}".format(s))
        break
    else:
        a += 1
        print("a=", a)
```

```
input your language:python
your lang is python
```



```
input your language:Python
a= 1
input your language:python
your lang is python
```

for循环语句

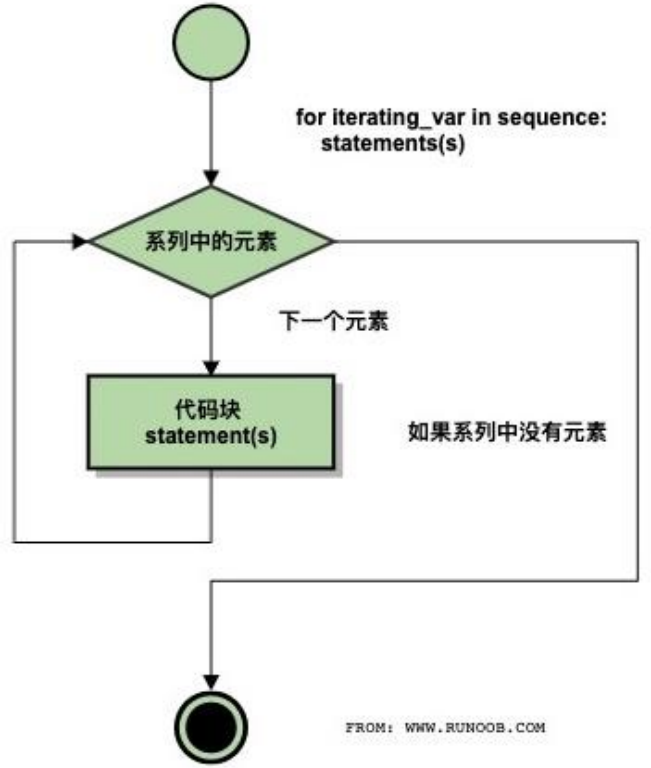
- for 循环是可以依次得到序列循环中每个元素，并依次处理
  - •可遍历任意序列，例如：一个字符串，一个列表
  - •遍历，就是查看序列中的每个元素（for循环、遍历、迭代，是自动播放所有序列当中的元素）
  - •可迭代的对象可以使用for循环进行遍历，例如：字符串、列表、字典、元组和集合
  - •for循环里面有一个隐藏的机制，就是自动执行index+1，直到遍历完整个序列

- 基本样式　for 循环规则:
  语句块

```
pokemon = ['Pikachu','Bulbasaur','Squirtle']#存放姓名
for name in pokemon:
    print(name)
```

```
Pikachu
Bulbasaur
Squirtle
```

#for循环中，for 后面的name为变量，用来依次接收序列中的元素



for循环语句

- 现在有一个列表，这个列表存放的是 10 个的 HP 数值，现在希望得到里面每个数字都乘以2的新表

```
HP = [110,103,103,75,85,105,50,75,105,120,75,45,55,75]
HP_new = []#新列表存储乘以2后的数
for n in HP:
    HP_new.append(n*2) #形成新列表
print(HP_new)
```

```
[220, 206, 206, 150, 170, 210, 100, 150, 210, 240, 150, 90, 110, 150]
```

for循环语句

使用列表的切片，对列表中的字符串进行遍历

#先使用切片将列表中的第一个元素取出，然后再对其进行遍历取值
list=['python','good','very']
for i in list[0]:
    print(i)#注意print的位置

输出结果:
p
y
t
h
o
n

```
In [5]: list=['python','good','very']
        for i in list[0]:
        print(i)

  File "<ipython-input-5-b8f977135c12>", line 3
    print(i)
    ^
IndentationError: expected an indented block
```

for循环语句

使用for循环遍历字典

- for循环遍历字典，默认是遍历字典的所有key值
- 如果想要取字典的value值，需要先取出所有的value值后，进行遍历取值，或者遍历key，打印的时候打印value值
- 也可以使用for循环取字典的键值对

```
dict={"name":"小花","age":18,"性别":"女"}
#默认遍历字典中的key值
for i in dict:
    print(i)
```

输出结果：

```
name
age
性别
```

```
#直接遍历字典中的value值
for m in dict.values():
    print(m)
```

输出结果：

```
小花
18
女
```

```
#直接遍历字典中的key值，打印出对应的value值
for j in dict:
    print(dict[j])
```

输出结果：

```
小花
18
女
```

```
#取字典中的键值对

for h in dict.items():
    print(h)

for key,value in dict.items():
    print('{} 的值为 {}'.format(key,value))
```

输出结果：

```
('name', '小花')
('age', 18)
('性别', '女')
name 的值为 小花
age 的值为 18
性别 的值为 女
```

for循环语句

for循环的嵌套：主要用来自动化测试中的用例获取
（九九乘法表、冒泡排序的实现）
range() 函数可创建一个整数列表，一般用在 for 循环中
使用方法：range(start, stop,step)
参数说明：
start:计数从start开始，默认为从0开始，eg：range(5)--->等价于range(0,5)
stop:计数到stop计数，但是不包含stop。eg：range(0,5)-->[0,1,2,3,4]（现在需要转换）
step:步长，默认为1，eg：range(5)--->等价于range(0,5,1)
range()函数 用在你想重复执行多少次计算的时候，没有现存的列表可以循环，就可以使用range()，返回的是一个可迭代对象
range(1,10) 表示从1开始到10结束，但不包括10. 输出结果为：1,2,3,4,5,6,7,8,9
range(1,10,2) 表示从1开始到10结束，但不包括10，步长为2 输出结果为：1,3,5,7,9 左闭右开！！

```
In [8]: range(5)
Out[8]: range(0, 5)

In [12]: for i in range(5):
             print(i)

0
1
2
3
4

In [15]: #如果不传起始值，默认是从0开始，如果设置起始值，则就从设置的起始值开始
         for i in range(10):
             print(i,end=' ') #输出结果添加位置参数，返回结果之间用空格连接

0 1 2 3 4 5 6 7 8 9
```

for循环语句

for循环嵌套实例

In [16]:
```
list=[1,2,3,4,5]
list2=["A","B","C","D","E"]

for i in list:
    print(i,end=' ')
    for j in list2:
        print(j,end=' ')
```

```
1 A B C D E 2 A B C D E 3 A B C D E 4 A B C D E 5 A B C D E
```

总结：for循环的嵌套，外层取一个值，内层取全部值，然后再返回外层继续变量，直到遍历完所有值。（外层走一个，内层走一圈）

for循环语句

思考与练习：使用for打印九九乘法表
提示：

输出九九乘法表，格式如下：（每项数据之间空一个Tab键，可以使用"\t"）

```
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
```

for循环语句

使用for打印九九乘法表
生成一个1到9的整数列表 range(1,10)
当i=1时，j=1
当i=2时，j=1,2
当i=3时，j=1,2,3
当i=4时，j=1,2,3,4
....依次类推
当i=n时，j=n ，所以j的取值范围由i的取值决定。
所以j的取值范围为range(1,i+1)
依次进行遍历，生成如下的乘法口诀表

```
for i in range(1,10):
    for j in range(1,i+1):
        print("{}*{}={}".format(j,i,i*j),end='\t')
    #此处的print()输出每执行完一轮后，打印计算结果到控制台
    print()#作用是什么？没有会怎么样？
```

```
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
```


```
for i in range(1,10):
    for j in range(1,i+1):
        print("{}*{}={}".format(j,i,i*j),end='\t')
    #此处的print()输出每执行完一轮后，打印计算结果到控制台
    print()#位置会影响输出结果，尝试一下位置的影响
```

break/continue/pass关键字

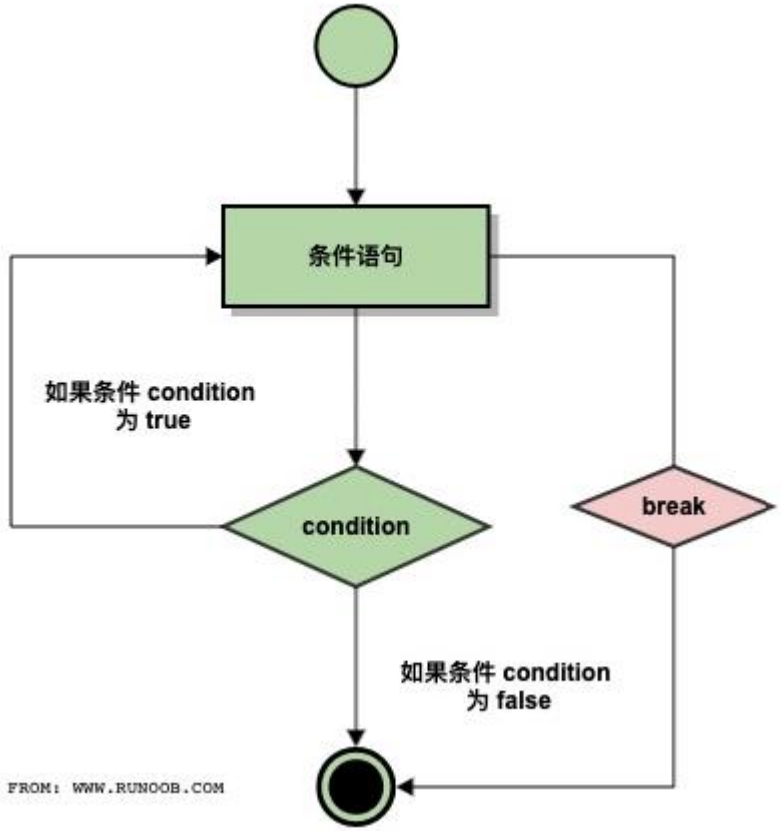
- break
  - break是终止本循环，以简单的for循环来举例
  - 如果您使用嵌套循环，break语句将停止执行最深层的循环，并开始执行下一行代码

```
for letter in 'Python':     # 第一个实例
  if letter == 'h':
    break
  print('当前字母:',letter)

var = 10                    # 第二个实例
while var > 0:
  print('当前变量值 :',var)
  var = var -1
  if var == 5:  # 当变量 var 等于 5 时退出循环
    break

print("Good bye!")
```

```
当前字母 : P
当前字母 : y
当前字母 : t
当前变量值 : 10
当前变量值 : 9
当前变量值 : 8
当前变量值 : 7
当前变量值 : 6
Good bye!
```



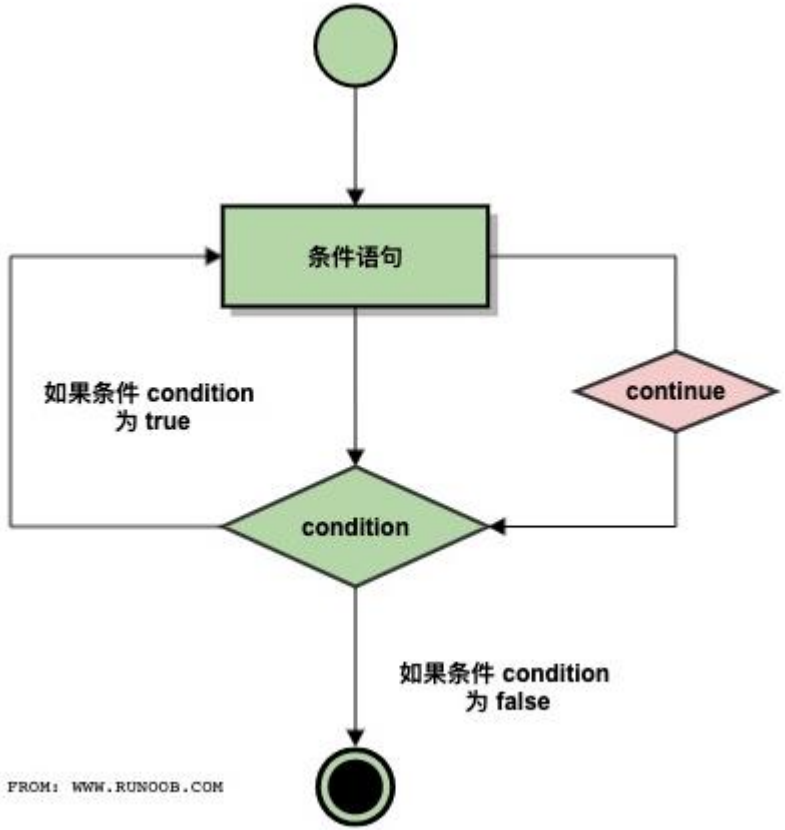
break/continue/pass关键字

- Continue
- Python continue 语句跳出本次循环，而break跳出整个循环。
- continue 语句用来告诉Python跳过当前循环的剩余语句，然后继续进行下一轮循环。
- continue语句用在while和for循环中。

```
for letter in 'Python':     # 第一个实例
   if letter == 'h':
      continue
   print( '当前字母:', letter)

var = 10                    # 第二个实例
while var > 0:
   var = var -1
   if var == 5:
      continue
   print( '当前变量值 :', var)
print( "Good bye!")
```

```
当前字母 : P
当前字母 : y
当前字母 : t
当前字母 : o
当前字母 : n
当前变量值 : 9
当前变量值 : 8
当前变量值 : 7
当前变量值 : 6
当前变量值 : 4
当前变量值 : 3
当前变量值 : 2
当前变量值 : 1
当前变量值 : 0
Good bye!
```



break/continue/pass关键字

continue 语句具有删除的效果，它可以删除满足循环条件下的某些不需要的成分

```
var = 10
while var > 0:
    var = var -1
    if var == 5 or var == 8:
        continue
    print( '当前值 :', var)
print("Good bye!")
```

```
当前值 : 9
当前值 : 7
当前值 : 6
当前值 : 4
当前值 : 3
当前值 : 2
当前值 : 1
当前值 : 0
Good bye!
```

只打印0-10之间的奇数，可以用continue语句跳过某些循环：

```
n = 0
while n < 10:
    n = n + 1
    if n % 2 == 0:      # 如果n是偶数，执行continue语句
        continue        # continue语句会直接继续下一轮循环，后续的print()语句不会执行
    print(n)
```

```
1
3
5
7
9
```

break/continue/pass关键字

- pass
  - pass 是空语句，是为了保持程序结构的完整性
  - pass 不做任何事情，一般用做占位语句

```
for letter in 'Python':
  if letter == 'h':
    pass
    print ('这是 pass 块')
  print ('当前字母 :', letter)
```

```
当前字母 : P
当前字母 : y
当前字母 : t
这是 pass 块
当前字母 : h
当前字母 : o
当前字母 : n
```

组合使用

- 除了单独使用上述控制结构之外，还可以嵌套使用
- 比如说在前面的HP序列，将其中的偶数乘以“2”，奇数不变

```
HP = [110,103,103,75,85,105,50,75,105,120,75,45,55,75]
HP_new = []
for n in HP:
    if not n%2:
        HP_new.append(n*2)
    else:
        HP_new.append(n)
print(HP_new)
```

```
[220, 103, 103, 75, 85, 105, 100, 75, 105, 240, 75, 45, 55, 75]
```

列表推导式

- 相关函数
  - range()
  - zip()
  - enumerate()

列表推导式

- 列表推导式也称为列表解析，效率高于使用循环语句
- 程序更方便简洁地实现功能
- 在list/dict内部使用for循环来构造list/dict的方法
- 将一个列表转换成另一个列表
- 生成另一个新列表，原列表保持不变

列表推导式

- 含义解析：
  - 关键词for循环后跟的是循环语法，这部分不变；而在for循环真正表达式部分则在列表推导式中移前，运算结果直接添加入列表中

for循环起始关键词

```
s = [x ** 2 for x in range(1, 10)]
print(s)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```

for循环起始关键词

```
s = []
for x in range(1,10):
    s.append(x ** 2)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```

range(stop)
range(start, stop[, step])

参数说明:
start: 计数从 start 开始。默认是从 0 开始。例如range（5）等价于range（0， 5）;
stop: 计数到 stop 结束，但不包括 stop。例如：range（0， 5） 是[0, 1, 2, 3, 4]没有5
step：步长，默认为1。例如：range（0， 5） 等价于 range(0, 5, 1)

```
In [3]: a=range(5,1,-2)

In [4]: a
Out[4]: range(5, 1, -2)

In [5]: print(a)
        range(5, 1, -2)

In [6]: type(a)
Out[6]: range

In [7]: list(a)
Out[7]: [5, 3]
```

```
In [8]: range(5)
Out[8]: range(0, 5)

In [9]: list(range(5))
Out[9]: [0, 1, 2, 3, 4]
```

```
>>> c= range(5,1,-1)
>>> list(c)
[5, 4, 3, 2]
>>>
```

```
range(5, 1, -2)
>>> list(a)
[5, 3]
```

2，3，4，5

列表推导式

- 基本样式：**[e for e in seq]**

```
multiples = [i*2 for i in range(30) if i % 3 is 0]
print (multiples)
```

```
[0, 6, 12, 18, 24, 30, 36, 42, 48, 54]
```

列表推导式

- 列表推导式也称为列表解析，效率高于使用循环语句
- 可以把列表推导式理解成一种集合了变换和筛选功能的函数，通过这个函数把一个列表转换成另一个列表
- 注意：是生成另一个新列表，原列表保持不变
- 基本样式：**[e for e in seq]**

```
multiples = [i*2 for i in range(30) if i % 3 is 0]
print (multiples)
```

```
[0, 6, 12, 18, 24, 30, 36, 42, 48, 54]
```

列表推导式

zip()

有两个列表，第一个列表包含了名，第二个列表包含了姓。使用 zip() 函数，如下我们可以将它们拼接在一起。

```
first_names = ["Peter", "Christian", "Klaus"]

last_names = ["Jensen", "Smith", "Nistrup"]

print([' '.join(x) for x in zip(first_names,last_names)])
```

```
['Peter Jensen', 'Christian Smith', 'Klaus Nistrup']
```

zip 将两个等长的列表变为了一对一对的

不限于两个可迭代对象作为参数传递-可以添加任意多个

zip() 是可以接受多于两个的序列的参数，不仅仅是两个

```
uppercase = ['A', 'B', 'C']
lowercase = ['a', 'b', 'c']
for x, y in zip(uppercase, lowercase):
    print(x, y)
```

```
A a
B b
C c
```

```
uppercase = ['A', 'B', 'C']

lowercase = ['a', 'b', 'c']

numbers = [1, 2, 3]

for x, y, z in zip(uppercase, lowercase, numbers):
    print(x, y, z)
```

```
A a 1
B b 2
C c 3
```

字典推导式

字典推导式

- 字典也可以使用推导式
- 字典推导式和列表推导式的使用方法是类似的，只不过中括号该改成大括号
- 字典推导式多用于需要元素有一一对应关系

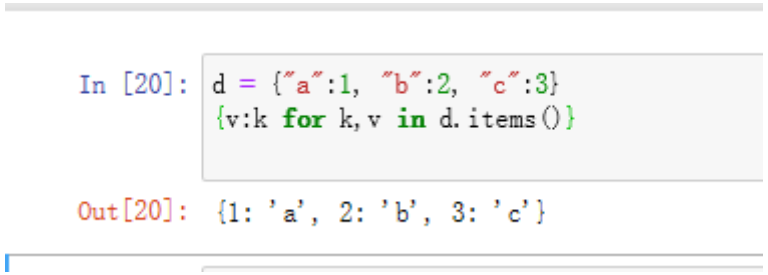
字典推导式

- 基本样式：**{e for e in seq}**

```
d = {"a":1, "b":2, "c":3}

{v:k for k,v in d.items()}
```

```
{1: 'a', 2: 'b', 3: 'c'}
```



字典推导式

- enumerate()是Python的一个常用内置函数，它用在列表中时，不但会产生列表内的元素，并且会从“0”开始按顺序生成序号。enumerate() 函数用于将一个可遍历的数据对象(如列表、元组或字符串)组合为一个索引序列，同时列出数据和数据下标，一般用在 for 循环当中。
- enumerate方法的语法:
- enumerate(sequence, [start=0]) # start= 是定义下标起始数字，默认是0。

```
{i:j for i,j in enumerate(["a","b","c"])}
```

```
{0: 'a', 1: 'b', 2: 'c'}
```

```
In [34]: direction = ['east', 'south', 'west', 'north']
         {i:j for i, j in enumerate(direction)}

Out[34]: {0: 'east', 1: 'south', 2: 'west', 3: 'north'}
```

应用：为文本文件每一行的末尾增加行号

集合推导式

集合推导式十分简单，和列表推导式是同一个用法，但是我们使用集合推倒式对列表推导式进行遍历之后，最后形成的是一个集合，而不是一个列表，而集合当中的数字是不重复的。如下所示：

```
In [35]: list=[1, 2, 54, 67, 2, 3, 5, 32, 2, 2, 4, 4, 4, 4]

In [36]: {num for num in list}
Out[36]: {1, 2, 3, 4, 5, 32, 54, 67}
```

```
ff = '你是不是来这里买东西的？买啥？'
set_ = {w for w in ff} # type(set_) is: set
print(set_)
```

{'是', '的', '东', '啥', '你', '？', '西', '买', '不', '里', '这', '来'}

```
squared = {x**2 for x in [1, 1, 2]}

print(squared)
```

```
{1, 4}
```

练习

1.请通过 for 循环结构依次将列表HP的每个元素加2，并将计算结果存放到新的列表HP_new中

HP = [110, 103, 103, 75, 85, 105, 50, 75, 105, 120, 75, 45, 55, 75]

2.现在有两个列表，一个是模型的列表models，另一个是模型评分的列表scores

使用字典推导式快速创建模型:评价指标一一对应的字典，将结果存为model_evaluate

models = ['decision trees', 'svm', 'random forest', 'neural network']

scores = [0.92, 0.94, 0.97, 0.96]

model_evaluate = None

第三章 结 束