Python循环控制语句

一.什么是循环?

循环就是不断重复动作的语句! Python的两个主要循环结构:

1. for循环 遍历序列对象内的元素, 对每个元素运行一个代码块!

伪代码:

```
for .. in ...:
     do something
```

2. while循环 编写通用循环的一种方法; while循环只要条件为真, 就不断循环, 条件为假时退出循环, 之所以称为循环, 是因为控制权会持续返回到语句开头部分.

伪代码:

```
while <test>:
    statements1
else: #可选(当循环正常退出时, 执行此处的语句)
    statement2
```

前面我们已经在非正式的环境下见过这两种循环了, 在这里我们将会说一些细节, 此外我们还会研究循环中的continue和 break关键字, 并且还会介绍循环中常用的一些内置函数 zip map range !

while和for语句是用来编写循环操作的主要语法, 后面我们会说到迭代器 列表解析 函数式编程(filter map reduce)等.

In [1]:
```
#示例代码1:

n = 5
while n > 1:
    print '123'
    n  = n - 1
else:
    print 'abc'
```

```
123
123
123
123
abc
```

In [ ]:
```
#示例代码2: 一个简单的死循环

while 1:
    print 'hello world'
```

二.循环控制语句中的关键字 break continue

一般会出现在while或for循环主体的任何地方, 但通常会进一步嵌套在if语句中, 根据某些条件来才去对应的操作.

break 跳出整个循环

continue 跳出本次循环 来到循环的首行

pass 什么也不做 占位

else 只有当程序正常离开时才执行(没有碰到break语句)

In [4]:
```
#1) break关键字
n  = 10
while n >= 0:
    if n == 8:  ##当n = 2 时跳出本次循环
        break
    print n
    n = n - 1
```

```
10
9
```

In [6]:
```
#2) continue关键字

n = 10
while n >= 0:
    if n == 8:
        n = n - 1
        continue ##跳出循环
    print n
    n = n - 1
```

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

In [8]:
```
#3) pass关键字

n = 10
while n >= 5:
    if n == 2:
        pass
    elif n == 4:
        pass
    print n
    n = n - 1
```

```
10
9
8
7
6
5
```

In [9]:
```
#4)else关键字
#循环正常退出时(没有遇到break),执行else下面的语句
n = 10
while n >= 5:
    if n == 2:
        pass
    elif n == 4:
        pass
    print n
    n = n - 1
else:
    print 'hello'
```

```
10
9
8
7
6
5
hello
```

In [10]:
```
#循环非正常退出时(遇到break),不执行else下面的语句
n  = 10
while n >= 0:
    if n == 8:
        break
    print n
    n = n - 1
else:
    print 'hello'
```

```
10
9
```

三. for 是Python中一个通用的序列迭代器, 可以遍历任何可迭代对象内的元素.如字符串 列表 元组 集合 字典

In [11]:
```
#例1:
names = ['Michael', 'Bob', 'Tracy']
for name in names:
    print(name)
```

```
Michael
Bob
Tracy
```

In [14]:
```
#例2:
for i in range(5):
    print i
else:
    print 'finish!'

#在for循环中, 关键字break continue else 的含义与while循环中相同
```

```
0
1
2
3
4
finish!
```

In [15]:
```
#例3. 同时获取元素和偏移 enumerate()

for j in enumerate('apple'):
    print j
```

```
(0, 'a')
(1, 'p')
(2, 'p')
(3, 'l')
(4, 'e')
```

In [ ]: