条件判断和循环

4143次阅读

条件判断

计算机之所以能做很多自动化的任务，因为它可以自己做条件判断。

比如，输入用户年龄，根据年龄打印不同的内容，在Python程序中，用if语句实现：

```
age = 20
if age >= 18:
    print 'your age is', age
    print 'adult'
```

根据Python的缩进规则，如果if语句判断是True，就把缩进的两行print语句执行了，否则，什么也不做。

也可以给if添加一个else语句，意思是，如果if判断是False，不要执行if的内容，去把else执行了：

```
age = 3
if age >= 18:
    print 'your age is', age
    print 'adult'
else:
    print 'your age is', age
    print 'teenager'
```

注意不要少写了冒号:。

当然上面的判断是很粗略的，完全可以用elif做更细致的判断：

```
age = 3
if age >= 18:
    print 'adult'
elif age >= 6:
    print 'teenager'
else:
    print 'kid'
```

elif是else if的缩写，完全可以有多个elif，所以if语句的完整形式就是：

```
if <条件判断1>:
    <执行1>
elif <条件判断2>:
    <执行2>
elif <条件判断3>:
    <执行3>
else:
    <执行4>
```

if语句执行有个特点，它是从上往下判断，如果在某个判断上是True，把该判断对应的语句执行后，就忽略掉剩下的elif和else，所以，请测试并解释为什么下面的程序打印的是teenager：

```
age = 20
if age >= 6:
    print 'teenager'
elif age >= 18:
```

```
    print 'adult'
else:
    print 'kid'
```

if判断条件还可以简写，比如写：

```
if x:
    print 'True'
```

只要x是非零数值、非空字符串、非空list等，就判断为True，否则为False。

循环

Python的循环有两种，一种是for...in循环，依次把list或tuple中的每个元素迭代出来，看例子：

```
names = ['Michael', 'Bob', 'Tracy']
for name in names:
    print name
```

执行这段代码，会依次打印names的每一个元素：

```
Michael
Bob
Tracy
```

所以for x in ...循环就是把每个元素代入变量x，然后执行缩进块的语句。

再比如我们想计算1-10的整数之和，可以用一个sum变量做累加：

```
sum = 0
for x in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
    sum = sum + x
print sum
```

如果要计算1-100的整数之和，从1写到100有点困难，幸好Python提供一个range()函数，可以生成一个整数序列，比如range(5)生成的序列是从0开始小于5的整数：

```
>>> range(5)
[0, 1, 2, 3, 4]
```

range(101)就可以生成0-100的整数序列，计算如下：

```
sum = 0
for x in range(101):
    sum = sum + x
print sum
```

请自行运行上述代码，看看结果是不是当年高斯同学心算出的5050。

第二种循环是while循环，只要条件满足，就不断循环，条件不满足时退出循环。比如我们要计算100以内所有奇数之和，可以用while循环实现：

```
sum = 0
n = 99
while n > 0:
    sum = sum + n
    n = n - 2
print sum
```

在循环内部变量n不断自减，直到变为-1时，不再满足while条件，循环退出。

再议raw_input

最后看一个有问题的条件判断。很多同学会用raw_input()读取用户的输入，这样可以自己输入，程序运行得更有意思：

```
birth = raw_input('birth: ')
if birth < 2000:
    print '00前'
else:
    print '00后'
```

输入1982，结果却显示00后，这么简单的判断Python也能搞错？

当然不是Python的问题，在Python的交互式命令行下打印birth看看：

```
>>> birth
'1982'
>>> '1982' < 2000
False
>>> 1982 < 2000
True
```

原因找到了！原来从raw_input()读取的内容永远以字符串的形式返回，把字符串和整数比较就不会得到期待的结果，必须先用int()把字符串转换为我们想要的整型：

```
birth = int(raw_input('birth: '))
```

再次运行，就可以得到正确地结果。但是，如果输入abc呢？又会得到一个错误信息：

```
Traceback (most recent call last):
  ...
ValueError: invalid literal for int() with base 10: 'abc'
```

原来int()发现一个字符串并不是合法的数字时就会报错，程序就退出了。

如何检查并捕获程序运行期的错误呢？后面的错误和调试会讲到。

小结

条件判断可以让计算机自己做选择，Python的if...elif...else很灵活。

```
if salary >= 10000:
    print
elif salary >=5000:
    print
else:
    print
```

循环是让计算机做重复任务的有效的方法，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。

请试写一个死循环程序。
