

## 循环

[2.7旧版教程](#)

阅读: 304817

## 循环

要计算 $1+2+3$ ，我们可以直接写表达式：

```
>>> 1 + 2 + 3
6
```

要计算 $1+2+3+...+10$ ，勉强也能写出来。

但是，要计算 $1+2+3+...+10000$ ，直接写表达式就不可能了。

为了让计算机能计算成千上万次的重复运算，我们就需要循环语句。

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，再通过 `list()` 函数可以转换为list。比如 `range(5)` 生成的序列是从0开始小于5的整数：

```
>>> list(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。

## 练习

请利用循环依次对 `list` 中的每个名字打印出 `Hello, xxx!`：

```
# -*- coding: utf-8 -*-
L = ['Bart', 'Lisa', 'Adam']
```

▶ Run

## break

在循环中，`break` 语句可以提前退出循环。例如，本来要循环打印**1~100**的数字：

```
n = 1
while n <= 100:
    print(n)
    n = n + 1
print('END')
```

上面的代码可以打印出**1~100**。

如果要提前结束循环，可以用 `break` 语句：

```
n = 1
while n <= 100:
```

```
> if n > 10: # 当n = 11时，条件满足，执行break语句
    break # break语句会结束当前循环
print(n)
n = n + 1
print('END')
```



执行上面的代码可以看到，打印出1~10后，紧接着打印 `END`，程序结束。

可见 `break` 的作用是提前结束循环。

## continue

在循环过程中，也可以通过 `continue` 语句，跳过当前的这次循环，直接开始下一次循环。

```
n = 0
while n < 10:
    n = n + 1
    print(n)
```

上面的程序可以打印出1~10。但是，如果我们想只打印奇数，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)
```

执行上面的代码可以看到，打印的不再是1~10，而是1，3，5，7，9。

可见 `continue` 的作用是提前结束本轮循环，并直接开始下一轮循环。

## 小结

循环是让计算机做重复任务的有效的办法。

`break` 语句可以在循环过程中直接退出循环，而 `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。

请试写一个死循环程序。

## 参考源码

[do\\_for.py](#)

[do\\_while.py](#)

您见本站内容不错，以后有收获：

>

🏠

≡

¥ 我要小额赞助，鼓励作者写出更好的教程



还可以分享给朋友

分享 赶快成为第一个分享的人吧

珠峰NODE.JS全栈开发

Angular

React

Gulp

Meteor

Vue

微信开发

技术陪伴成长社区

珠峰培训

麦子学院

www.maiziedu.com

百万级python导师亲身指导

保你120天

变身python大牛

有时候，你需要的只是一句点拨

立即咨询

掘金

一个只有高手分享

的技术社区

立即加入

深度学习在线课程

通向无人驾驶的必经之路

< [条件判断](#)

[使用dict和set](#) >

评论

发表评论

Sign In to Make a Comment

