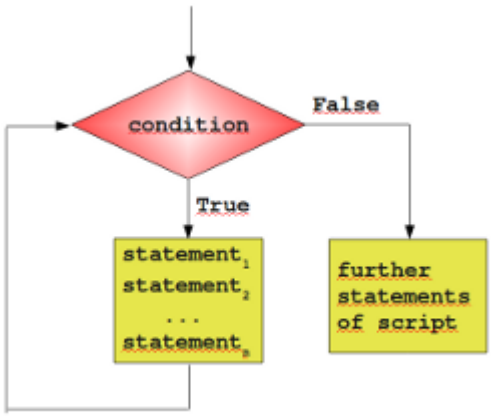
Python3 循环语句

本章节将为大家介绍Python循环语句的使用。

Python中的循环语句有 for 和 while。

Python循环语句的控制结构图如下所示：



while 循环

Python中while语句的一般形式：

```
while 判断条件:
    语句
```

同样需要注意冒号和缩进。另外，在Python中没有do..while循环。

以下实例使用了 while 来计算 1 到 100 的总和：

```
#!/usr/bin/env python3

n = 100

sum = 0
counter = 1
while counter <= n:
    sum = sum + counter
    counter += 1

print("1 到 %d 之和为: %d" % (n,sum))
```

执行结果如下：

```
1 到 100 之和为: 5050
```

无限循环

我们可以通过设置条件表达式永远不为 false 来实现无限循环，实例如下：

```
#!/usr/bin/python3

var = 1
while var == 1 :  # 表达式永远为 true
   num = int(input("输入一个数字  :"))
   print ("你输入的数字是: ", num)

print ("Good bye!")
```

执行以上脚本，输出结果如下：

```
输入一个数字  :5
你输入的数字是:  5
输入一个数字  :
```

你可以使用 **CTRL+C** 来退出当前的无限循环。

无限循环在服务器上客户端的实时请求非常有用。

while 循环使用 else 语句

在 while ... else 在条件语句为 false 时执行 else 的语句块：

```
#!/usr/bin/python3

count = 0
while count < 5:
   print (count, " 小于 5")
   count = count + 1
else:
   print (count, " 大于或等于 5")
```

执行以上脚本，输出结果如下：

```
0  小于 5
1  小于 5
2  小于 5
3  小于 5
4  小于 5
5  大于或等于 5
```

简单语句组

类似if语句的语法，如果你的while循环体中只有一条语句，你可以将该语句与while写在同一行中， 如下所示：

```
#!/usr/bin/python

flag = 1

while (flag): print ('欢迎来到联航!')

print ("Good bye!")
```

注意：以上的无限循环你可以使用 CTRL+C 来中断循环。

执行以上脚本，输出结果如下：

```
欢迎来到联航!
欢迎来到联航!
欢迎来到联航!
欢迎来到联航!
欢迎来到联航!
.....
```

---

for 语句

Python for循环可以遍历任何序列的项目，如一个列表或者一个字符串。

for循环的一般格式如下：

```
for <variable> in <sequence>:
    <statements>
else:
    <statements>
```

Python loop循环实例：

```
#!/usr/bin/python3

languages = ["C", "C++", "Perl", "Python"]
for x in languages:
    print(x)
```

```
C
C++
Perl
Python
```

以下 for 实例中使用了 break 语句，break 语句用于跳出当前循环体：

```
#!/usr/bin/python3

sites = ["Baidu", "Google", "Unigress", "Taobao"]
for site in sites:
    if site == "Unigress":
        print("联航科技!")
        break
    print("循环数据 " + site)
else:
    print("没有循环数据!")
print("完成循环!")
```

执行脚本后，在循环到 "Unigress"时会跳出循环体：

```
循环数据 Baidu
循环数据 Google
联航科技!
完成循环!
```

---

range()函数

如果你需要遍历数字序列，可以使用内置range()函数。它会生成数列，例如：

```
>>>for i in range(5):
...     print(i)
...
0
1
2
3
4
```

你也可以使用range指定区间的值：

```
>>>for i in range(5,9) :
    print(i)
5
6
7
8
>>>
```

也可以使range以指定数字开始并指定不同的增量(甚至可以是负数，有时这也叫做'步长'):

```
>>>for i in range(0, 10, 3) :
    print(i)
0
3
6
9
>>>
```

负数:

```
>>>for i in range(-10, -100, -30) :
    print(i)
-10
-40
-70
>>>
```

您可以结合range()和len()函数以遍历一个序列的索引,如下所示:

```
#!/usr/bin/python3

a = ['Google', 'Baidu', 'Unigress', 'Taobao', 'QQ']
for i in range(len(a)):
    print(i, a[i])

0 Google
1 Baidu
2 Unigress
3 Taobao
4 QQ
```

还可以使用range()函数来创建一个列表:

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

break和continue语句及循环中的else子句

break 语句可以跳出 for 和 while 的循环体。如果你从 for 或 while 循环中终止，任何对应的循环 else 块将不执行。 实例如下：

```
#!/usr/bin/python3

for letter in 'Unigress':     #第一个实例
    if letter == 's':
        break
    print ('当前字母为 :', letter)

var = 10                      #第二个实例
while var > 0:
    print ('当期变量值为 :', var)
    var = var -1
    if var == 5:
        break

print ("Good bye!")
```

执行以上脚本输出结果为：

```
当前字母为 : U
当前字母为 : n
当前字母为 : i
当前字母为 : g
当前字母为 : r
当前字母为 : e
当期变量值为 : 10
当期变量值为 : 9
当期变量值为 : 8
当期变量值为 : 7
当期变量值为 : 6
Good bye!
```

continue语句被用来告诉Python跳过当前循环块中的剩余语句，然后继续进行下一轮循环。

```
#!/usr/bin/python3

for letter in 'Lenovo':     # 第一个实例
    if letter == 'o':        # 字母为 o 时跳过输出
        continue
    print ('当前字母 :', letter)

var = 10                    # 第二个实例
while var > 0:
    var = var -1
    if var == 5:             # 变量为 5 时跳过输出
```

```
        continue
    print ('当前变量值 :', var)

print ("Good bye!")
```

执行以上脚本输出结果为:

```
当前字母 : L
当前字母 : e
当前字母 : n
当前字母 : v
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!
```

循环语句可以有 else 子句，它在穷尽列表(以for循环)或条件变为 false (以while循环)导致循环终止时被执行,但循环被break终止时不执行。

如下实例用于查询质数的循环例子:

```
#!/usr/bin/python3

for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print(n, ' 等于', x, '*', n//x)
            break
    else:
        # 循环中没有找到元素
        print(n, ' 是质数')
```

执行以上脚本输出结果为:

```
2  是质数
3  是质数
4  等于 2 * 2
5  是质数
6  等于 2 * 3
7  是质数
8  等于 2 * 4
9  等于 3 * 3
```

pass 语句

Python 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，如下实例

```
>>>while True:
...     pass  # 等待键盘中断 (Ctrl+C)
```

最小的类:

```
>>>class MyEmptyClass:
...     pass
```

以下实例在字母为 o 时 执行 pass 语句块:

```
#!/usr/bin/python3

for letter in 'Lenovo':
    if letter == 'o':
        pass
        print('执行 pass 块')
    print('当前字母 :', letter)

print("Good bye!")
```

执行以上脚本输出结果为:

```
当前字母 : L
当前字母 : e
当前字母 : n
执行 pass 块
当前字母 : o
当前字母 : v
执行 pass 块
当前字母 : o
Good bye!
```