8.循环语句

循环就是当条件满足的时候，就会重复的执行某些事情，直到条件不满足退出。python中的循环语句主要有while循环和for循环两种。

一、while循环语句

1. while 循环语句结构

```
while 条件表达式:
    循环体语句块
```

while循环中，当条件表达式为真时，就会重复的执行循环体内的语句块，直到条件为假。

示例代码：

```
# 计数循环
count = 1
while count <= 10:
    print "hello, I am %d loop." % count
    count += 1  # 自增运算
# 死循环(无限循环)
while True:  # 因为条件一直为True
    print "dead loop"
```

一般情况下，很少使用死循环，除非要求程序一直是执行的，比如持续的网络请求的时候。一般死循环里面都会结合条件判断语句，用以在某些适当的情况下推出循环体。

2. 使用break语句跳出循环体

```
while True:
    name = input("请输入你的名字(输入Q可退出程序):")
    if name == "Q":
        break
    print "你的名字是：%s" % name, "欢迎你的到来。"
```

break语句是跳出当前语句所在的循环体，如果有循环嵌套的且break是在内层循环的，那么只会跳出内层循环，不会跳出外层循环。

```
outer = 3
inner = 5
while outer > 0:
    while inner > 0:
        if inner < 3:
            break
        print "I am inner loop", inner
        inner -= 1
    print "I am outer loop", out
    outer -= 1
```

3. 使用continue语句跳出当次循环

continue 只是跳过当前次循环执行下次循环，并不是跳出整个循环体

```
count = 10
while count > 0:
    if count == 5:
        continue
    print "loop", count
    count -= 1
```

对于嵌套的循环，continue跳过的也是它所在的那层循环的当次循环，与break是相似的。

4. 神奇的else

   在其他的语言中，循环是没有else搭配的，但是python却出了这么个东西。当循环体是正常的结束的时候（没有break中断的）那么就会执行else语句的部分。

```
count = 0
while count < 10:
    print "hello", count
    count += 1
else:
    print "while loop done well"
```

二、for循环语句

python中for循环语句的主要作用就是访问可迭代对象（序列，迭代器，生成器，字典，集合等）中的所有元素，并在所有元素都处理之后自动的结束循环。

1. for语句的语法结构

```
for 变量名 in 可迭代对象:
    语句块
```

2. 用法示例

```
# 遍历序列类型
for char in "hello":
    print char
for num in [1,2,3,4,5]:
    print num * 10
# 遍历字典
d = {'a': 1, 'b': 2, 'c': 3}
for key in d:
    print key, d[key]

for key, val in d.items():
    print key, val
# 遍历集合
for s in set('abcdef'):
    print s
```

3. 再谈range函数

   range函数是用来生成一个由数字组成的列表的迭代器，它有三个参数，start、stop、step,看到这三个参数，相信大家也很熟悉，就是在切片的时候也有遇到过，因此range的这三个参数的功能是类似的。下面介绍一下用法：

```
>>> range(5)  # 生成从0-4的range列表迭代器
>>> range(1, 6) # 生成从1开始到5的列表迭代器
>>> range(1, 10, 2) # 生成从1开始到9结束的列表迭代器，每隔两步生成一个（可以理解为两个数之间相差2）
```

4. 结合range使用下标的方式遍历序列类型

```
chars = "hello world"
for i in range(len(chars)):
    print "chars[%d] = %s" % (i, chars[i])
```

5. 使用enumerate方法可以同时返回序列的下标和值

```
chars = "hello world"
for i, v in enumerate(chars):
    print i, v
```

6. for循环也有else语句

   for循环中的else语句跟while循环中的是一样的，只要for循环是正常结束的，那么就会执行else语句中的代码块

```
for i in range(10):
    print i
else:
    print "done"
```

三、迭代器和iter()函数（只做了解）

迭代器其实就是一个实现了迭代器协议的容器对象。它基于两个方法：

- __next__ 返回容器的下一个项目
- __iter__ 返回迭代器本身

迭代器可以通过使用一个iter内建函数和一个序列来创建，比如：

```
>>> i = iter('abc')
>>> i.__next__()  # 得到字符a
'a'
>>> i.__next__()
'b'
>>> i.__next__()
'c'
>>> i.__next__()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```

当序列遍历完成时，将抛出一个StopIteration异常。这个跟for循环是兼容的，for循环会自动的捕获这个异常，如果捕获到了那么就会停止循环并退出。

迭代器只能不断的往后迭代，而不能往前，也不能复制。如果你需要重复的迭代同一个对象的时候，你只能再重新创建一个新的迭代器了。不过一般情况下，我们很少自己去创建迭代器，因为很多的函数已经内置了。我们只需会用就好了。

比如我们前面学到过的: range()、enumerate()函数都是直接返回一个迭代器。

如何查看一个对象是否是可迭代的?

使用dir函数，查看对象的魔术方法，只要方法里面有__iter__方法的就是可以迭代的（可以通过循环来遍历出来）

四、使用for循环生成各种推导式

问题：想象一下，如果要创建一个列表，这个列表的每个元素分别是另一个列表元素的平方，根据之前学过的知识要怎么做?

```
list1 = range(1, 11)
list2 = []
for n in list1:
    list2.append(n ** 2)
```

是不是很麻烦?

解析器（推导式）：

就是利用for循环在列表、元组、集合、字典等容器类型中迭代一个可迭代对象的元素，将其中的元素放到对应的数据类型中，在迭代创建的过程中可以对数据进行二次加工后再存放。

1. 列表推导式
    - 语法：[变量名 for 变量名 in 可迭代对象]
    - 相关用法示例：

```
list1 = [x ** 2 for x in range(1, 11)]
list2 = [x for x in range(100) if x % 2 == 0]
list3 = ["stu-%d" % i for i in range(10)]
list4 = [x for x in range(3) for i in range(4)]
list5 = [[x for x in range(4)] for y in range(3)]
list6 = [('a%d' % i, i) for i in range(10)]
list7 = [{'stu%d' % i: i+10} for i in range(10)]
```

2. 集合推导式(用法跟列表推导式是一样的，只不过外层换成了集合)

```
s1 = {x for x in range(10)}
s2 = {x**2 for x in range(10) if x % 2 == 10}
```

3. 字典推导式

```
d1 = {x: 'a' for x in range(5)}
d2 = {'x%d' % i: i * 2 for i in range(5) }
```

4. 生成器表达式

    生成器表达式返回的是一个生成器(generator)，生成器的好处就是当你需要的时候才会去生成，不需要的时候是懒惰的。因此占用很少的内存空间。

    生成器表达式是用圆括号括起来的，里面的用法跟其它的推导式是一样的。

```
g1 = (x for x in range(10))
```

作业

1. 使用循环语句和前面学到的知识写一个网络爬虫程序，要求能够抓取指定数量的文章。提取文章的标题和内容。内容部分可以保留HTML源码也可以去除。至于目标站点可以自行去寻找。
2. 用while循环实现倒过来的99乘法表，用for循环也实现正序的99乘法表。