高级特性

迭代

- 可以通过 for 循环来遍历这个 list 或 tuple,这种遍历我们称为迭代(Iteration)
- 只要是可迭代对象,无论有无下标,都可以迭代,比如 dict 就可以迭代:

迭代

- 默认情况下,dict 迭代的是 key。如果要迭代 value，怎么办?

```
for k,v in d.iteritems():
        print k,v
```

迭代

- 如何判断一个对象是可迭代对象呢?

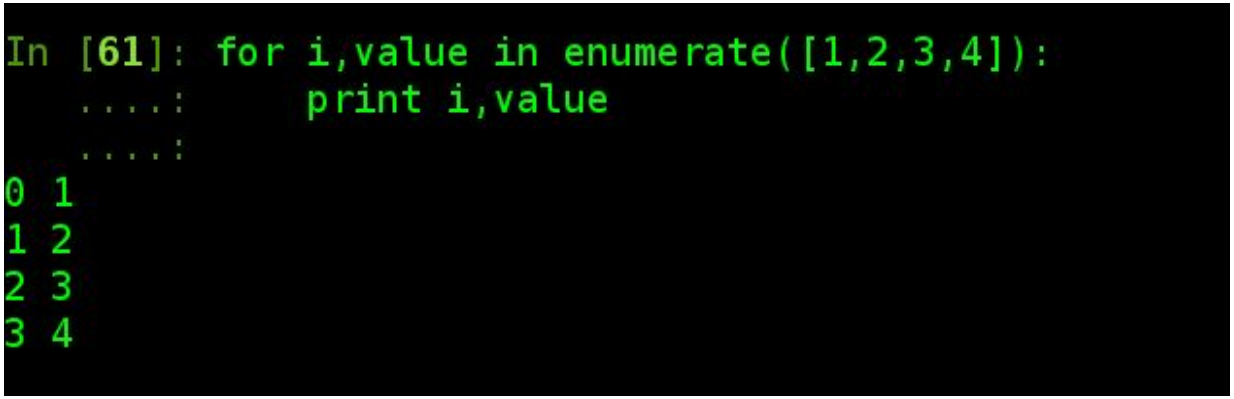
方法是通过 collections 模块的 Iterable 类型判断

```
In [56]: from collections  import Iterable

In [57]: isinstance('abc',Iterable)
Out[57]: True

In [58]: isinstance('{"name":"fentiao"}',Iterable)
Out[58]: True

In [59]: isinstance([1,2,3],Iterable)
Out[59]: True

In [60]: isinstance(1,Iterable)
Out[60]: False
```

迭代

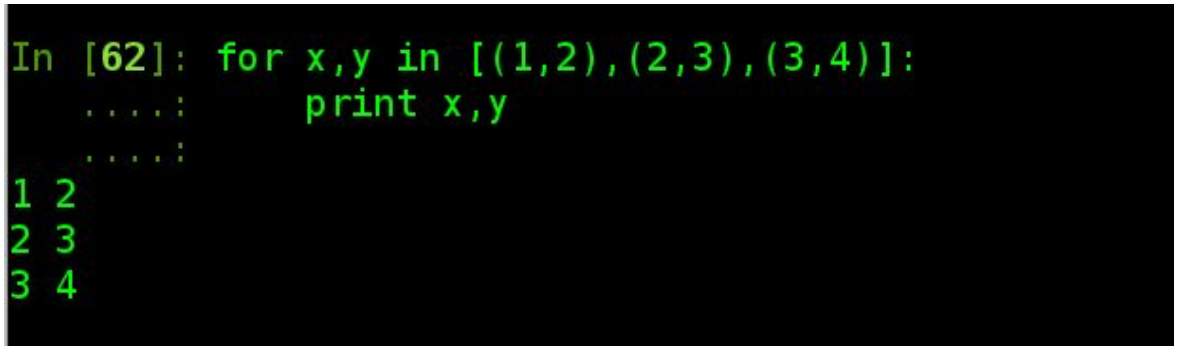
- 如果要对 list 实现类似 Java 那样的下标循环怎么办?

  python内置的枚举方法enumerate,把一个 list 变成索引-元素对



迭代

- for 循环里,同时引用了两个变量,在 Python 里是很常见的;
- 如果要显示[(1,2),(2,3),(3,4)]怎么实现迭代显示?



列表生成式

列表生成式是Python 内置的非常简单却强大的可以用来创建 list的生成式

思考:

要生成 list [1, 2, 3, 4, 5, 6, 7] 可以用 range(1, 8)

但如果要生成 [1x1, 2x2, 3x3, ..., 7x7] 怎么做?

列表生成式

方法一：for循环

```
In [63]: L = []

In [64]: for x in range(1,8):
   ....:     L.append( x * x )
   ....:

In [65]: L
Out[65]: [1, 4, 9, 16, 25, 36, 49]
```

方法二：列表生成式

```
In [66]: [x*x for x in range(1,8)]
Out[66]: [1, 4, 9, 16, 25, 36, 49]
```

循环太繁琐,而列表生成式则可以用一行语句代替循环生成上面的 list.

列表生成式

列表生成式可以嵌套if语句和for语句么?

- 生成100以内所有偶数的平方;
- 生成‘ABC’与‘123’的全排列;
- 列出当前目录下的所有文件和目录名;

<参考：os.listdir(".")>

列表生成式

列表生成式也可以使用两个变量来生成 list么?

- 生成字典的的内容，格式为‘key=value’,返回其列表格式;

- 将list中所有的字符串变为小写字母;

<参考：s.lower()>

列表生成式

列表生成式也可以使用两个变量来生成 list么?

- 生成字典的的内容，格式为‘key=value’,返回其列表格式;

- 将list中所有的字符串变为小写字母;

<参考：s.lower()>

生成器

为什么需要生成器？

- 通过列表生成式,我们可以直接创建一个列表,受到内存限制,列表容量肯定是有限的;
- 创建一个包含 100 万个元素的列表,占用很大的存储空间;

生成器

生成器是什么？

- 在循环的过程中不断推算出后续的元素呢?这样就不必创建完整的 list,从而节省大量的空间。在 Python 中,这种一边循环一边计算的机制,称为生成器(Generator)

生成器

怎么创建生成器？把一个列表生成式的 [] 改成 ()

- 使用g.next()方法依次读取元素（麻烦）
- 使用 for 循环（推荐）

```
In [10]: g = (x for x in range(4))

In [11]: for x in g:
   ....:     print x
   ....:
0
1
2
3
```

```
In [4]: g = (x for x in range(4))

In [5]: g.next()
Out[5]: 0

In [6]: g.next()
Out[6]: 1

In [7]: g.next()
Out[7]: 2

In [8]: g.next()
Out[8]: 3

In [9]: g.next()
---------------------------------------
StopIteration
<ipython-input-9-d7e53364a9a7> in <module
----> 1 g.next()

StopIteration:
```

生成器

怎么创建生成器？把一个列表生成式的 [] 改成 ()

- 使用g.next()方法依次读取元素（麻烦）
- 使用 for 循环（推荐）

```
In [10]: g = (x for x in range(4))

In [11]: for x in g:
   ....:     print x
   ....:
0
1
2
3
```

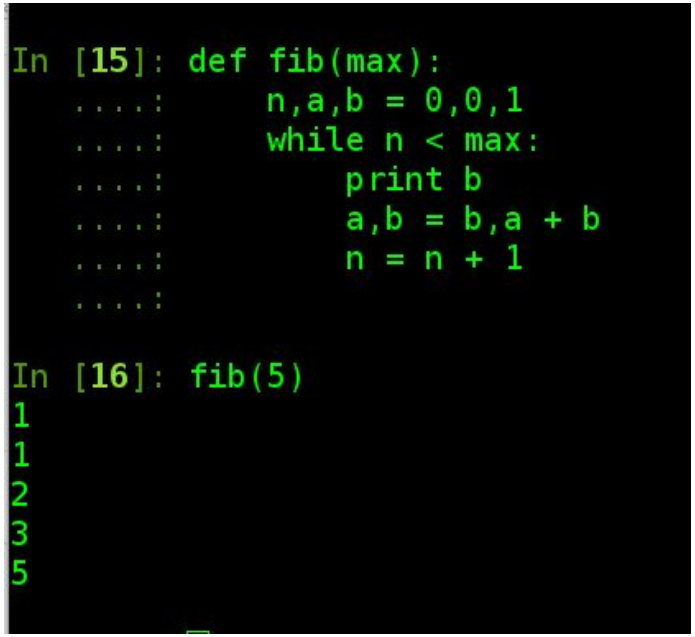
```
In [4]: g = (x for x in range(4))

In [5]: g.next()
Out[5]: 0

In [6]: g.next()
Out[6]: 1

In [7]: g.next()
Out[7]: 2

In [8]: g.next()
Out[8]: 3

In [9]: g.next()
---------------------------------------
StopIteration
<ipython-input-9-d7e53364a9a7> in <module
----> 1 g.next()

StopIteration:
```

生成器

- 理解生成器的实质，当无法通过列表生成式表述问题时，如何通过函数实现生成式的功能。

python编程:著名的斐波拉契数列(Fibonacci),除第一个和第二个数外,任意一个数都可由前两个数相加得到:1, 1, 2, 3, 5, 8, 13, 21,...

生成器



fib 函数定义了斐波拉契数列的推算规则,可以从第一个元素开始,推算出后续任意的元素,逻辑非常类似 generator。

要把 fib 函数变成 generator,只需要把print b 改为 yield b 就可以.

生成器

```
In [17]: def fib(max):
   ....:     n,a,b = 0,0,1
   ....:     while n < max:
   ....:         yield b
   ....:         a,b = b, a+b
   ....:         n = n + 1
   ....:

In [18]: fib(3)
Out[18]: <generator object fib at 0x21a1e60>

In [19]: g = fib(3)

In [20]: g.next()
Out[20]: 1

In [21]: g.next()
Out[21]: 1

In [22]: g.next()
Out[22]: 2
```

- 函数顺序执行,遇return语句或最后一行函数语句就返回。
- generator函数在每次调用next() 的时候执行,遇到 yield 语句返回,再次执行时从上次返回的yield 语句处继续执行。

over !