In [1]:

```python
import datetime

#创建一个列表来存储随机数
randomlist = []

#设置一个数字的随机数作为种子
#seed = 6406

#以randomlist的id后4位为随机数种子
#seed = id(randomlist) % 10000

#以系统时间毫秒数的后4位为随机数种子
seed = datetime.datetime.now().microsecond % 10000

while(len(randomlist) <= 1000):

    #计算随机数的平方
    seedsquare = seed ** 2

    #获取随机数平方的长度,如果长度不是8，前面用0补齐
    seedsquare_str = str(seedsquare)
    if len(seedsquare_str) != 8:
        seedsquare_str = seedsquare_str.rjust(8,'0')

    #获取该随机数平方的中间四位
    seedsquare_4 = seedsquare_str[2:6]

    #将取出的随机数平方的中间四位作为随机数加入随机数列表
    randomlist.append('0.' + seedsquare_4)

    #将取出的随机数平方的中间四位作为新的种子
    seed = int(seedsquare_4)


for randint in randomlist:
    print(randint,end='\t\t')
```

```
0.5352          0.6439          0.4607          0.2244          0.0355          0.
1260            0.5876          0.5273          0.8045          0.7220          0.
1284            0.6486          0.0681          0.4637          0.5017          0.
1702            0.8968          0.4250          0.0625          0.3906          0.
2568            0.5946          0.3549          0.5954          0.4501          0.
2590            0.7081          0.1405          0.9740          0.8676          0.
2729            0.4474          0.0166          0.0275          0.0756          0.
5715            0.6612          0.7185          0.6242          0.9625          0.
6406            0.0368          0.1354          0.8333          0.4388          0.
2545            0.4770          0.7529          0.6858          0.0321          0.
1030            0.0609          0.3708          0.7492          0.1300          0.
6900            0.6100          0.2100          0.4100          0.8100          0.
6100            0.2100          0.4100          0.8100          0.6100          0.
2100            0.4100          0.8100          0.6100          0.2100          0.
4100            0.8100          0.6100          0.2100          0.4100          0.
8100            0.6100          0.2100          0.4100          0.8100          0.
6100            0.2100          0.4100          0.8100          0.6100          0.
2100            0.4100          0.8100          0.6100          0.2100          0.
4100            0.8100          0.6100          0.2100          0.4100          0.
```

下面介绍一些上面的代码涉及到的Python语法

Python中使用变量无需提前声明，直接用赋值语句即可创建一个对象，例如第7行

In [2]:

```
1 seed = 6406
```

创建的对象类型会根据赋值语句的内容自动选择

In [3]:

```
1 a = 1
2 type(a)
```

Out[3]:

```
int
```

In [4]:

```
1 a = 0.5
2 type(a)
```

Out[4]:

```
float
```

In [5]:

```
1 a = []
2 type(a)
```

Out[5]:

```
list
```

In [6]:

```
1 a = ()
2 type(a)
```

Out[6]:

```
tuple
```

In [7]:

```
1 a = ''
2 type(a)
```

Out[7]:

```
str
```

前面创建的5种对象中，后3个可以包含其他的对象，类似于C语言中的数组
用方括号框住的是列表**list**，用圆括号框住的是元组**tuple**，用引号框住的是字符串**str**

列表中的对象可以用索引来访问

In [8]:

```
1 a = [1, 2, 3, 4]
2 print(a[2])
```

3

列表的索引从0开始，所以a[2]是列表a的第3个对象
索引也可以是负数，a[-1]是最后1个

In [9]:

```
1 print(a[-1])
```

4

在索引中使用冒号可以选取多个对象，从冒号左边的索引开始，直到冒号右边的索引的前一个为止
冒号两边的数字可以略去，表示从头开始或者到末尾结束

In [10]:

```
1 print(a[1:3])
2 print(a[2:])
3 print(a[:-2])
```

[2, 3]
[3, 4]
[1, 2]

列表可以做加法，表示合并两个列表

In [11]:

```
1 a = [1, 2, 3]
2 b = ['a', 'b']
3 a + b
```

Out[11]:

[1, 2, 3, 'a', 'b']

字符串和元组也有类似的操作
代码的第26行中用索引截取了字符串的一部分，29行用加法合并了2个字符串

使用append方法可以在列表中添加对象,代码的第29行使用了这种用法

In [14]:

```
a = [1, 2]
a.append('abc')
print(a)
```

```
[1, 2, 'abc']
```

使用len方法可以得到容器对象的长度

In [19]:

```
a = [1, 2, 3]
print(len(a))
a = (2, 3, 5, 1)
print(len(a))
a = 'abcde'
print(len(a))
```

```
3
4
5
```

Python中if和while结构与C语言中的相似

In [20]:

```
x = 5
if x < 1:
    print('x<1')
elif x < 10:
    print('1<x<10')
else:
    print('x>10')
```

```
1<x<10
```

In [21]:

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

```
10
```

for循环需要一个可遍历的对象

In [22]:

```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

for循环把可遍历对象range(5)中的每一个对象依次赋值给i，并执行循环结构中的语句

Python通过缩进表示循环和判断等结构
结构中的语句需要对齐并比for,while,if等关键字所在行多缩进数格