一 生成器和迭代器

```
s = [i for i in range(10)]
print(s)
```

```
a = (i for i in range(10000000000000000000000000000000000))
print(a)
```

生成器不会直接给你开辟内存空间

--next--()

```
a = (i for i in range(10))
print(a.__next__())

next()

a = (i for i in range(10))
print(next(a))
```

```
a = (i for i in range(10))
print(next(a))
print(next(a))
print(next(a))
print(next(a))
print(next(a))
print(next(a))
```

```

a = (i for i in range(1,6))
print(next(a))
print(next(a))
print(next(a))
print(next(a))
print(next(a))
print(next(a))
```

生成器是一个可迭代对象

```
r = (i for i in range(1,6))
for x in r:
    print(x)  # 第一次: x = 1  第二次 : x = 2
```

只要没有被对象引用，那么python就会有垃圾回收机制将其回收掉

for循环的本质就是在调用next()

```
生成器有两种创建方式:
    1.(i for i in range(1,6))
    2.yield 这个关键字实现
```

```
def fn():
    print('hello world')
    yield 1


print(fn())
```

```
def fn():
    print('hello world')
    yield 1


g = fn()
print(next(g))
```

```
def fn():
    print('f1')
    yield 1
    print('f2')
    yield 2

g = fn()
print(next(g))
print(next(g))
```

```
def fn():
    print('f1')
    yield 1
    print('f2')
    yield 2

for i in fn():
    print(i)
```

生成器是可迭代对象

可迭代对象不是迭代器的意思

```
for i in [2,4,6]:
    print(i)
```
[2,4,6]是可迭代对象
从现象来看，只要是可以for循环的都是可迭代对象
从本质来看，是内置有iter方法的是可迭代对象

```
l = [2,4,6]
l.__iter__()
```

能循环是因为内部有一个iter方法
可迭代对象：对象拥有iter方法

yield 和 return的区别：
return：在函数中返回某个值，然后函数结束运行
yield：带yield的函数是一个迭代器，在函数内部碰到yield的时候，函数会返回某个值，并停留在这个位置，当下次执行函数后，会在上次停留的位置继续运行

1.2 迭代器

生成器就是迭代器

iter 方法为我们做了一个事情: 返回一个迭代器对象

```
l = [1,2,3,4]
d = iter(l)
print(d)

<list_iterator object at 0x0000027C7F5980A0>
```

iterator 是迭代器的意思
iterable 可迭代对象

什么是迭代器， 满足两个条件:

1. 有iter方法
2. 有next方法

```
l = [1,2,3,4]
d = iter(l)
print(d)
print(next(d))
print(next(d))
print(next(d))
print(next(d))
```

```
for i in [1,2,3]:
    print(i)

a = iter([1,2,3])
while 1:
    print(next(a))
```

```
for循环内部本质是3个事情:
    1.调用可迭代对象的iter方法返回一个迭代器
    2.不断调用迭代器的next方法
    3.处理异常

a = iter([1,2,3])
while 1:
    try:
        print(next(a))
    except StopIteration:
        break
```

二 模块 module

定义: 包含一些列数据，函数，类的文件，通常以.py结尾

模块和模块之间互相调用怎么办?

2.1 模块导入

```
import 导入

import module01

module01.f1()

from ...  import ...

print('模块1')
def f1():
    print('模块1的f1')

from module01 import f1
f1()


from ... import *
```

```
第一种: import
本质;  使用变量名比如 module01关联的模块地址，所以将来你用import导的时候要求大家还要用这个变量名才能点出东西
第二种: from ... import ...
本质: 将指定的成员导入到当前模块作用域中
第三种: from...import *
本质: 将指定模块的所有成员导入到当前模块作用域中
```

注意：导入进来的成员不要与当前模块成员名称相同
就近原则

```
from module01 import f as f1
from module02 import f as f2
from module03 import f as f3


f2()
```