容器（container）

1. 容器：把一种多个元素组织在一起的数据结构，容器中的元素都可以迭代获取，可以用成员关系操作符来判断元素是否在容器中---容器不是类型
2. 常见的容器：

```
list tuple str set frozenset dict
```

可以是异构的也可以是同构

可迭代对象（iterable）

1. 很多容器都是可迭代对象（容器包含可迭代对象）
2. 一个可迭代对象是不能独自进行迭代，Python中，迭代是通过for来完成的for循环做了一下两件事：

```
>>> k=l.__iter__()----k=iter(l)
>>> k.__next__()  -----next(k)
1
>>> k.__next__()
2
```

迭代器（iterator）

1. iter()---返回对象的迭代器
2. 他是一个带状对象，可以在调用next()时，返回对象的一个值
3. 只要实现了__iter__(),__next__()的对象，就是迭代器
4. 迭代器也是可迭代对象

序列

```
list
str
tuple
```

1. 可迭代对象包含序列
2. 序列可以被iter()next()使用
3. 生成无限序列

```
import itertools
a=itertools.count()
print(next(a))
print(next(a))
```

4. 从一个有限序列生成无限序列

```
import itertools
a=itertools.cycle([1,2,3])
print(next(a))
print(next(a))
```

5. next()

```
懒加载机制（什么时候使用，什么时候调）
```

```
        懒汉式加载
6. for
        饿汉式加载
```

列表推导式

```
1. 语法：
        l=[i for i in 可迭代对象]
        l=[i for i in range(10) if 4>5]  #如果if为真则执行前面的for
2. 变形：
        l=[i*2 for i in range(10)]
        l=[i*2 for i in 'ABCD']
        l=[i*2 for i in 'ABCD' if 0==True]
        l=[fun(i) for i in range(5)]
        l=[fun1(i) if fun(i) else fun2(i) for i in range(5)]
```

集合推导式

```
语法：
    l={fun(i) for i in range(5)}
```

生成器

```
1. 生成器表达式（生成式）
        将列表的推导式方括号改为圆括号
        生成器直接通过next（）--__next__()进行调用
        是一个特殊的迭代器
2. yield 关键字
        def fun():
            a=10
            while 1:
                yield a
                a+=1
a=fun()
print(a)
3. 生成器是可迭代对象
```

- 小结

```
容器>可迭代对象（迭代器（生成器---1. 生成器表达式2. yield关键字））>序列
1. 迭代器：实现__iter__()和__next__()
2. 生成器---1. 生成器表达式2. yield关键字
```