迭代协议

有next方法，以及__iter__的对象，及实现了Python的迭代协议，成为可以被for循环调用的迭代器。
for循环会调用__iter__方法获取一个迭代对象，并每次调用next方法会返回下一个迭代值，next方法在迭代器末尾需要抛出StopIteration异常。

```
class Fibonacci:
    def __init__(self, max_num=100):
        self.MAX_NUM = max_num
        self.num = 1

    def next(self):
        if self.num < self.MAX_NUM:
            ans = self.num ** 2 + self.num
            self.num += 1
            return ans
        raise StopIteration

    def __iter__(self):
    #表示对象是自身的迭代器
        return self

if __name__ == "__main__":
    for i in Fibonacci(5000000):
        print i
```

通过iter方法可以的对象的迭代器

```
I = iter(Fibonacci())
```

上面的for循环等价于这么写：

```
I = iter(Fibonacci())
```

```
while True:
    try:
        X = next(I)
    except StopIteration:
        break
    print X
```

列表解析

在一个序列的值上应用一个任意表达式，将其结果收集到一个新的列表中并返回。
它的基本形式是一个方括号里面包含一个for语句对一个iterable对象迭代。
基本语法：

```
fibonacci_sq = [x ** 2 for x in Fibonacci()]
```

扩展语法：

```
fibonacci_sq = [x ** 2 for x in Fibonacci() if x % 100 == 0]
```

扩展语法2：

```
[x+y for x in "abc" for y in "xyz"]
```

上面表达式要从右边向左看。