目 录

1 生成器

在 python 中，生成器用于一边循环一遍计算，在循环的过程中不断推算出后续的元素。

1.1 创建生成器

1.1.1 第一种方法

将列表生成式的 [] 换成 ()，就能创建一个生成器了：

```
L = [x * x for x in range(10)]
g = (x * x for x in range(10))
```

生成器自带 next() 方法，可以通过 next() 方法访问后续的元素：

```
g.next() # 输出0
g.next() # 输出1
g.next() # 输出4
```

还可以通过 for 循环去访问生成器的元素：

```
g = (x * x for x in range(10))
for n in g:
    print n
```

1.1.2 第二种方法

如果一个函数定义中包含 yield 关键字，那么这个函数就不再是一个普通函数，而是一个 generator。

举个例子：

```
# 斐波那契数列的普通函数
def fib(max):
    n, a, b = 0, 0, 1
    while n < max:
        print b
        a, b = b, a+b
        n = n + 1

# 斐波那契数列的生成器
def fib(max):
    n, a, b = 0, 0, 1
    while n < max:
        yield b
        a, b = b, a+b
        n = n + 1
```

变成 generator 的函数，将在 yield 处停止执行，每次调用 next() 函数或者 for 循环时，将继续执行直到在遇到 yield 语句。如下所示：

```
g = fib(6)
g.next() # 输出1
g.next() # 输出1
g.next() # 输出2

for n in fib(6):
    print n
```