链滴

# Python 3 高级特性

作者：jiangyue

原文链接：https://ld246.com/article/1491125927184

来源网站：链滴

- list、tuple、str 都可以做切片操作，如 l 是一个 list，l[0:10] 表示取 l 中前 10 个元素，l[3:-2] 示取 l 中 第 4 到倒数第 2 个元素之前
- 当取得第一个元素为 list 中第一个元素或取得最后一个元素为 list 中的最后一个元素时，可省略 0 即 l[:5] 表示取前 5 个元素，l[-3:] 表示取 list 中最后 3 个元素
- l[:10:2] 表示对 l 的前 10 个元素，每 2 个取一个
- 'ABCDEF'[:4:2] 的结果为 'AC'
- Python 中的 for 循环可以作用于任何可迭代对象，如 list、tuple、str，甚至 dict
- 默认情况下 dict 迭代的是 key，如要迭代 value，可以用 for value in d.values()
- for k, v in d.items() 可用于同时迭代 key 和 value
- 判断一个对象是否为可迭代对象，可通过 collections 模块的 Iterable 类型判断
- 要使用 Iterable 判断对象是否可迭代，需先通过 from collections import Iterable 导入 Iterable 类型，再使用 isinstance(xxx, Iterable) 判断是否可迭代
- enumerate() 函数可以将 list 转为 **索引--元素** 对，可在 for 循环中同时迭代索引和元素本身，for i, value in enumerate(['A', 'B', 'C'])
- 列表生成式可以便捷地生成列表，如 [x * x for x in range(3)] 可生成列表 [0, 1, 4]
- 列表生成式中在 for 循环后可以加条件判断，如 [x for x in range(10) if x % 2 == 0] 可生成列表 0, 2, 4, 6, 8]
- 列表生成式中可以使用多成循环生成全排列，如 [m + n for m in 'AB' for n in 'XY'] 可生成列表 [ AX', 'AY', 'BX', 'BY']
- for 循环可同时迭代多个变量，所以列表生成式也可以使用两个变量生成列表，如 [k + '=' + v for k, v in d.items()]
- generator 为 生成式，可在循环中计算出后续的元素
- 将列表生成式中的 [] 改为 () 即可创建出 generator，如 g = (x for x in range(10))
- next() 函数用于获取 generator 的下一个返回值，如首次调用 next(g) 得到 0，再次调用得到 1，有更多元素时抛出 StopIteration 异常
- for 循环可用于迭代 generator 而避免 StopIteration 错误，如 for n in generator:
- yield 为 Python 中的一个关键字，用于定义 generator，使用了 yield 关键字的函数为 generator
- 使用 next() 函数获取 generator 的返回值时，每调用一次 next(g)，可获得一次 yield 的返回值
- 杨辉三角可使用 generator 生成

```
            1
          1   1
        1   2   1
      1   3   3   1
    1   4   6   4   1
  1   5   10  10  5   1

  ~   ~   ~   ~   ~   ~   ~   ~   ~   ~   ~
```

```
def triangles():
    l = [1]
    while True:
        yield l
```

```
l = [1] + [l[i-1] + l[i] for i in range(len(l)) if i > 0] + [1]
```

- list、tuple、dict、set、str 等数据类型都可用 for 循环处理，这些可以直接作用于 for 循环的对统称为可迭代对象：Iterable
- generator 不但可以作用于 for 循环，还可以被 next() 函数调用返回下一个值，为 Iterator
- Iterator 对象表示的是一个数据流，可以看做一个有序序列，在使用过程中不能提前知道序列的长，只能通过 next() 函数按需获取下一个数据
- Iterator 的计算是惰性的，只有在需要返回时才计算
- Iterable 对象可以通过 iter() 函数转为 Iterator 对象