# 解析式和生成器表达式

## 列表解析式

列表解析式List Comprehension，也叫列表推导式。

```
# 生成一个列表，元素0~9，将每一个元素加1后的平方值组成新的列表
x = []
for i in range(10):
    x.append((i+1)**2)
print(x)
```

```
# 列表解析式
print([(i+1)**2 for i in range(10)])
```

语法

- [返回值 for 元素 in 可迭代对象 if 条件]
- 使用中括号[]，内部是for循环，if条件语句可选
- 返回一个新的列表

列表解析式是一种语法糖

- 编译器会优化，不会因为简写而影响效率，反而因优化提高了效率
- 减少程序员工作量，减少出错
- 简化了代码，增强了可读性

```
[expr for item in iterable if cond1 if cond2]
等价于
ret = []
for item in iterable:
    if cond1:
        if cond2:
            ret.append(expr)
#
[expr for i in iterable1 for j in iterable2 ]
等价于
ret = []
for i in iterable1:
    for j in iterable2:
        ret.append(expr)
```

```
# 请问下面3种输出各是什么？为什么
[(i,j)  for i in range(7) if i>4 for j in range(20,25) if j>23]
[(i,j)  for i in range(7) for j in range(20,25) if i>4 if j>23]
[(i,j)  for i in range(7) for j in range(20,25) if i>4 and j>23]
```

## 集合解析式

语法

- {返回值 for 元素 in 可迭代对象 if 条件}
- 列表解析式的中括号换成大括号{}就变成了集合解析式
- 立即返回一个集合

```
{(x, x+1) for x in range(10)}
{[x] for x in range(10)} # 可以吗？
```

## 字典解析式

语法

- {key:value for 元素 in 可迭代对象 if 条件}
- 列表解析式的中括号换成大括号{}，元素的构造使用key:value形式
- 立即返回一个字典

```
{x:(x,x+1) for x in range(10)}
{x:[x,x+1] for x in range(10)}
{(x,):[x,x+1] for x in range(10)}
{[x]:[x,x+1] for x in range(10)} #

{str(x):y for x in range(3) for y in range(4)} # 输出多少个元素？
```

## 生成器表达式

语法

- (返回值 for 元素 in 可迭代对象 if 条件)
- 列表解析式的中括号换成小括号就行了
- 返回一个生成器对象

和列表解析式的区别

- 生成器表达式是**按需计算**（或称**惰性求值**、**延迟计算**），需要的时候才计算值
- 列表解析式是立即返回值

生成器对象

- 可迭代对象
- **迭代器**

```
x = (i+1 for i in range(10))
print(next(x))
for i in x:
    print(i)
print('-' * 30)
for i in x:
    print(i)
```

```
x = [i+1 for i in range(10)]

for i in x:
    print(i)
print('-' * 30)
for i in x:
    print(i)
```

| 生成器表达式 | 列表解析式 |
| --- | --- |
| 延迟计算<br>返回可迭代对象迭代器，可以迭代<br>只能迭代一次 | 立即计算<br>返回可迭代对象列表，不是迭代器<br>可反复迭代 |

## 生成器表达式和列表解析式对比

- 计算方式
  - 生成器表达式延迟计算，列表解析式立即计算
- 内存占用
  - 单从返回值本身来说，生成器表达式省内存，列表解析式返回新的列表
  - 生成器没有数据，内存占用极少，使用的时候，一次返回一个数据，只会占用一个数据的空间
  - 列表解析式构造新的列表需要为所有元素立即占用掉内存
- 计算速度
  - 单看计算时间看，生成器表达式耗时非常短，列表解析式耗时长
  - 但生成器本身并没有返回任何值，只返回了一个生成器对象
  - 列表解析式构造并返回了一个新的列表

## 总结

- Python2 引入列表解析式
- Python2.4 引入生成器表达式
- Python3 引入集合、字典解析式，并迁移到了2.7

一般来说，应该多应用解析式，简短、高效。如果一个解析式非常复杂，难以读懂，要考虑拆解成for循环。

生成器和迭代器是不同的对象，但都是可迭代对象。

如果不需要立即获得所有可迭代对象的元素，在Python 3中，推荐使用惰性求值的迭代器。

| 内建函数 | 函数签名 | 说明 |
| --- | --- | --- |
| sorted | sorted(iterable[, key][, reverse]) | 默认升序，对可迭代对象排序<br>立即返回列表 |

```
# 排序一定是容器内全体参与
print(sorted([1,2,3,4,5]))
print(sorted(range(10, 20), reverse=True))
print(sorted({'a':100, 'b':'abc'}))
print(sorted({'a':100, 'b':'abc'}.items()))
print(sorted({'a':'ABC', 'b':'abc'}.values(), key=str, reverse=True))
print(sorted({'a':2000, 'b':'201'}.values(), key=str))
print(sorted({'a':2000, 'b':'201'}.values(), key=int))
```

## 作业

1. 给出3个整数，使用if语句判断大小，并升序输出
2. 有一个列表lst = [1,4,9,16,2,5,10,15]，生成一个新列表，要求新列表元素是lst相邻2项的和
3. 随机生成100个产品ID，ID格式如下
    - 顺序的数字6位，分隔符点号，10个随机小写英文字符
    - 例如 000005.xcbaaduixy