



马哥教育

IT 人的高薪职业学院

Python解析式、生成器

讲师：Wayne

从业十余载，漫漫求知路

标准库datetime

- datetime模块

- 对日期、时间、时间戳的处理

- datetime类

- 类方法

- today() 返回本地时区当前时间的datetime对象

- now(tz=None) 返回当前时间的datetime对象，时间到微秒，如果tz为None，返回和today()一样

- utcnow() 没有时区的当前时间

- fromtimestamp(timestamp, tz=None) 从一个时间戳返回一个datetime对象

- datetime对象

- timestamp() 返回一个到微秒的时间戳。

- 时间戳：格林威治时间1970年1月1日0点到现在的秒数

标准库datetime

□ datetime对象

- 构造方法 `datetime.datetime(2016, 12, 6, 16, 29, 43, 79043)`
- `year`、`month`、`day`、`hour`、`minute`、`second`、`microsecond`，取datetime对象的年月日时分秒及微秒
- `weekday()` 返回星期的天，周一0，周日6
- `isoweekday()` 返回星期的天，周一1，周日7
- `date()` 返回日期date对象
- `time()` 返回时间time对象
- `replace()` 修改并返回新的时间
- `isocalendar()` 返回一个三元组(年，周数，周的天)

标准库datetime

□ 日期格式化*

- 类方法 `strptime(date_string, format)` , 返回datetime对象
- 对象方法 `strftime(format)` , 返回字符串
- 字符串format函数格式化

```
import datetime
```

```
dt = datetime.datetime.strptime("21/11/06 16:30", "%d/%m/%y %H:%M")
```

```
print(dt.strftime("%Y-%m-%d %H:%M:%S"))
```

```
print("{0:%Y}/{0:%m}/{0:%d} {0:%H}::{0:%M}::{0:%S}".format(dt))
```

标准库datetime

□ timedelta对象

□ `datetime2 = datetime1 + timedelta`

□ `datetime2 = datetime1 - timedelta`

□ `timedelta = datetime1 - datetime2`

□ 构造方法

□ `datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)`

□ `year = datetime.timedelta(days=365)`

□ `total_seconds()` 返回时间差的总秒数

标准库time

- time

- time.sleep(secs) 将调用线程挂起指定的秒数



列表解析

□ 举例

- 生成一个列表，元素0~9，对每一个元素自增1后求平方返回新列表



列表解析

□ 举例

- 生成一个列表，元素0~9，对每一个元素自增1后求平方返回新列表

```
l1 = list(range(10))
```

```
l2 = []
```

```
for i in l1:
```

```
    l2.append((i+1)**2)
```

```
print(l2)
```

□ 列表解析式

```
l1 = list(range(10))
```

```
l2 = [(i+1)**2 for i in l1]
```

```
print(l2)
```

```
print(type(l2))
```



列表解析List Comprehension

□ 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，但可读性增强

列表解析

□ 举例

- 获取10以内的偶数，比较执行效率

```
even = []
```

```
for x in range(10):
```

```
    if x % 2 == 0:
```

```
        even.append(x)
```

```
even = [x for x in range(10) if x%2==0]
```

□ 思考

- 有这样的赋值语句`newlist = [print(i) for i in range(10)]`，请问`newlist`的元素打印出来是什么？
- 获取20以内的偶数，如果数是3的倍数也打印`[i for i in range(20) if i%2==0 elif i%3==0]` 行吗？

列表解析进阶

□ `[expr for item in iterable if cond1 if cond2]`

□ 等价于

```
ret = []
```

```
for item in iterable:
```

```
    if cond1:
```

```
        if cond2:
```

```
            ret.append(expr)
```

□ 举例

20以内，既能被2整除又能被3整除的数

```
[i for i in range(20) if i%2==0 and i%3==0]
```

```
[i for i in range(20) if i%2==0 if i%3==0]
```



列表解析进阶

□ `[expr for i in iterable1 for j in iterable2]`

□ 等价于

```
ret = []
```

```
for i in iterable1:
```

```
    for j in iterable2:
```

```
        ret.append(expr)
```

□ 举例

```
[(x, y) for x in 'abcde' for y in range(3)]
```

```
[[x, y] for x in 'abcde' for y in range(3)]
```

```
{x: y for x in 'abcde' for y in range(3)}
```



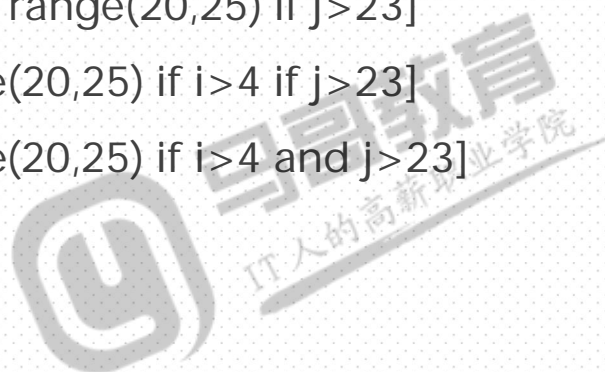
列表解析进阶

□ 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]`



列表解析练习

□ 练习（要求使用列表解析式完成）

- 返回1-10平方的列表
- 有一个列表`lst = [1,4,9,16,25,10,15]`，生成一个新列表，要求新列表元素是`lst`相邻2项的和
- 打印九九乘法表
- "0001.abadicddws" 是ID格式，要求ID格式是以点号分割，左边是4位从1开始的整数，右边是10位随机小写英文字母。请依次生成前100个ID的列表

```
'0001.ingbocjsem'  
'0002.ykjixnhzqj'  
'0003.nvuslmqrrn'  
'0004.qbfjdtcxue'  
'0005.ahxjrfpikv'  
'0006.oihpdykejt'  
'0007.ipvltoinic'  
'0008.orizapbgmv'
```



马哥教育
IT人的高薪职业学院

生成器表达式Generator expression

□ 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

□ 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

- **可迭代对象**
- **迭代器**

生成器表达式**

□ 举例

```
g = ("{:04}".format(i) for i in range(1,11))
next(g)
for x in g:
    print(x)
print('~~~~~')
for x in g:
    print(x)
```

□ 总结

- 延迟计算
- 返回迭代器，可以迭代
- 从前到后走完一遍后，不能回头

□ 对比列表

```
g = ["{:04}".format(i) for i in range(1,11)]
for x in g:
    print(x)
print('~~~~~')
for x in g:
    print(x)
```

□ 总结

- 立即计算
- 返回的不是迭代器，返回可迭代对象列表
- 从前到后走完一遍后，可以重新回头迭代



生成器表达式

□ 习题

```
it = (print("{}".format(i+1)) for i in range(2))
```

```
first = next(it)
```

```
second = next(it)
```

```
val = first + second
```

□ val的值是什么？

□ val = first + second 语句之后能否再次next(it)？

生成器表达式

□ 习题

```
it = (x for x in range(10) if x % 2)
```

```
first = next(it)
```

```
second = next(it)
```

```
val = first + second
```

□ val的值是什么？

□ val = first + second 语句之后能否再次next(it)？

生成器表达式

□ 和列表解析式的对比

□ 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，但是使用的时候，虽然一个个返回数据，但是合起来占用的内存也差不多
- 列表解析式构造新的列表需要占用内存

□ 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
- 但是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

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{}就行了
- 立即返回一个集合

□ 用法

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



字典解析式

□ 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{}就行了
- 使用key:value形式
- 立即返回一个字典

□ 用法

- {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)} #
- {chr(0x41+x):x**2 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？



字典解析式

□ 用法

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

□ 等价于

```
ret = {}
```

```
for x in range(3):
```

```
    for y in range(4):
```

```
        ret[str(x)] = y
```



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

马哥教育
IT人的高薪职业学院

谢谢

咨询热线 400-080-6560

