14.11 输出警告信息¶

问题¶

你希望自己的程序能生成警告信息（比如废弃特性或使用问题）。

解决方案¶

要输出一个警告消息，可使用 `warning.warn()` 函数。例如：

```
import warnings

def func(x, y, logfile=None, debug=False):
    if logfile is not None:
         warnings.warn('logfile argument deprecated', DeprecationWarning)
    ...
```

`warn()` 的参数是一个警告消息和一个警告类，警告类有如下几种：UserWarning, DeprecationWarning, SyntaxWarning, RuntimeWarning, ResourceWarning, 或 FutureWarning.

对警告的处理取决于你如何运行解释器以及一些其他配置。 例如，如果你使用 `-W all` 选项去运行Python，你会得到如下的输出：

```
bash % python3 -W all example.py
example.py:5: DeprecationWarning: logfile argument is deprecated
  warnings.warn('logfile argument is deprecated', DeprecationWarning)
```

通常来讲，警告会输出到标准错误上。如果你想讲警告转换为异常，可以使用 `-W error` 选项：

```
bash % python3 -W error example.py
Traceback (most recent call last):
  File "example.py", line 10, in <module>
    func(2, 3, logfile='log.txt')
  File "example.py", line 5, in func
    warnings.warn('logfile argument is deprecated', DeprecationWarning)
DeprecationWarning: logfile argument is deprecated
bash %
```

讨论¶

在你维护软件，提示用户某些信息，但是又不需要将其上升为异常级别，那么输出警告信息就会很有用了。 例如，假设你准备修改某个函数库或框架的功能，你可以先为你要更改的部分输出警告信息，同时向后兼容一段时间。 你还可以警告用户一些对代码有问题的使用方式。

作为另外一个内置函数库的警告使用例子，下面演示了一个没有关闭文件就销毁它时产生的警告消息：

```
>>> import warnings
>>> warnings.simplefilter('always')
>>> f = open('/etc/passwd')
>>> del f
__main__:1: ResourceWarning: unclosed file <_io.TextIOWrapper name='/etc/passwd'
 mode='r' encoding='UTF-8'>
>>>
```

默认情况下，并不是所有警告消息都会出现。`-W` 选项能控制警告消息的输出。 `-W all` 会输出所有警告消息，`-W ignore` 忽略掉所有警告，`-W error` 将警告转换成异常。 另外一种选择，你还可以使用 `warnings.simplefilter()` 函数控制输出。 `always` 参数会让所有警告消息出现，`` `ignore `` 忽略调所有的警告，`error` 将警告转换成异常。

对于简单的生成警告消息的情况这些已经足够了。 `warnings` 模块对过滤和警告消息处理提供了大量的更高级的配置选项。 更多信息请参考 Python文档