13.12 给函数库增加日志功能¶

问题¶

你想给某个函数库增加日志功能，但是又不能影响到那些不使用日志功能的程序。

解决方案¶

对于想要执行日志操作的函数库而已，你应该创建一个专属的 `logger` 对象，并且像下面这样初始化配置：

```
# somelib.py

import logging
log = logging.getLogger(__name__)
log.addHandler(logging.NullHandler())

# Example function (for testing)
def func():
    log.critical('A Critical Error!')
    log.debug('A debug message')
```

使用这个配置，默认情况下不会打印日志。例如：

```
>>> import somelib
>>> somelib.func()
>>>
```

不过，如果配置过日志系统，那么日志消息打印就开始生效，例如：

```
>>> import logging
>>> logging.basicConfig()
>>> somelib.func()
CRITICAL:somelib:A Critical Error!
>>>
```

讨论¶

通常来讲，你不应该在函数库代码中自己配置日志系统，或者是已经假定有个已经存在的日志配置了。

调用 `getLogger(__name__)` 创建一个和调用模块同名的logger模块。 由于模块都是唯一的，因此创建的logger也将是唯一的。

`log.addHandler(logging.NullHandler())` 操作将一个空处理器绑定到刚刚已经创建好的logger对象上。 一个空处理器默认会忽略调用所有的日志消息。 因此，如果使用该函数库的时候还没有配置日志，那么将不会有消息或警告出现。

还有一点就是对于各个函数库的日志配置可以是相互独立的，不影响其他库的日志配置。 例如，对于如下的代码：

```
>>> import logging
>>> logging.basicConfig(level=logging.ERROR)

>>> import somelib
>>> somelib.func()
```

```
CRITICAL:somelib:A Critical Error!

>>> # Change the logging level for 'somelib' only
>>> logging.getLogger('somelib').level=logging.DEBUG
>>> somelib.func()
CRITICAL:somelib:A Critical Error!
DEBUG:somelib:A debug message
>>>
```

在这里，根日志被配置成仅仅输出ERROR或更高级别的消息。 不过， `somelib` 的日志级别被单独配置成可以输出debug级别的消息，它的优先级比全局配置高。 像这样更改单独模块的日志配置对于调试来讲是很方便的， 因为你无需去更改任何的全局日志配置——只需要修改你想要更多输出的模块的日志等级。

Logging HOWTO 详细介绍了如何配置日志模块和其他有用技巧，可以参阅下。