

# python logging模块使用教程



好吃的野菜 (/u/62bbd7b8d80c)

+ 关注

2015.11.26 17:07\* 字数 2854 阅读 32197 评论 10 喜欢 71 赞赏 2

(/u/62bbd7b8d80c)

## 简单使用

```
#!/usr/local/bin/python
# -*- coding:utf-8 -*-
import logging

logging.debug('debug message')
logging.info('info message')
logging.warn('warn message')
logging.error('error message')
logging.critical('critical message')
```

输出：

```
WARNING:root:warn message
ERROR:root:error message
CRITICAL:root:critical message
```

默认情况下，logging模块将日志打印到屏幕上(stdout)，日志级别为WARNING(即只有日志级别高于WARNING的日志信息才会输出)，日志格式如下图所示：



问题来了

1. 日志级别等级及设置是怎样的？
2. 怎样设置日志的输出方式？比如输出到日志文件中？

## 简单配置

### 日志级别

级别	何时使用
DEBUG	详细信息，典型地调试问题时会感兴趣。
INFO	证明事情按预期工作。
WARNING	表明发生了一些意外，或者不久的将来会发生问题（如‘磁盘满了’）。软件还是在正常工作。
ERROR	由于更严重的问题，软件已不能执行一些功能了。
CRITICAL	严重错误，表明软件已不能继续运行了。

### 简单配置

```
#!/usr/local/bin/python
# -*- coding:utf-8 -*-
import logging

# 通过下面的方式进行简单配置输出方式与日志级别
logging.basicConfig(filename='logger.log', level=logging.INFO)

logging.debug('debug message')
logging.info('info message')
logging.warn('warn message')
logging.error('error message')
logging.critical('critical message')
```

输出：

标准输出(屏幕)未显示任何信息，发现当前工作目录下生成了logger.log，内容如下：

```
INFO:root:info message
WARNING:root:warn message
ERROR:root:error message
CRITICAL:root:critical message
```

因为通过`level=logging.INFO`设置日志级别为INFO，所以所有的日志信息均输出出来了。

问题又来了

1. 通过上述配置方法都可以配置那些信息？

在解决以上问题之前，需要先了解几个比较重要的概念，**Logger**，**Handler**，**Formatter**，**Filter**

## 几个重要的概念

- Logger 记录器，暴露了应用程序代码能直接使用的接口。
- Handler 处理器，将（记录器产生的）日志记录发送至合适的目的地。
- Filter 过滤器，提供了更好的粒度控制，它可以决定输出哪些日志记录。
- Formatter 格式化器，指明了最终输出中日志记录的布局。

### Logger 记录器

Logger是一个树形层级结构，在使用接口debug，info，warn，error，critical之前必须创建Logger实例，即创建一个记录器，如果没有显式的进行创建，则默认创建一个root logger，并应用默认的日志级别(WARN)，处理器Handler(StreamHandler，即将日志信息打印输出在标准输出上)，和格式化器Formatter(默认的格式即为第一个简单使用程序中输出的格式)。

创建方法: `logger = logging.getLogger(logger_name)`

创建Logger实例后，可以使用以下方法进行日志级别设置，增加处理器Handler。

- `logger.setLevel(logging.ERROR)` # 设置日志级别为ERROR，即只有日志级别大于等于ERROR的日志才会输出
- `logger.addHandler(handler_name)` # 为Logger实例增加一个处理器
- `logger.removeHandler(handler_name)` # 为Logger实例删除一个处理器



## Handler 处理器

Handler处理器类型有很多种，比较常用的有三个，**StreamHandler**，**FileHandler**，**NullHandler**，详情可以访问Python logging.handlers (http://python.usyiyi.cn/python\_278/library/logging.handlers.html#)

创建StreamHandler之后，可以通过使用以下方法设置日志级别，设置格式化器Formatter，增加或删除过滤器Filter。

- `ch.setLevel(logging.WARN)` # 指定日志级别，低于WARN级别的日志将被忽略
- `ch.setFormatter(formatter_name)` # 设置一个格式化器formatter
- `ch.addFilter(filter_name)` # 增加一个过滤器，可以增加多个
- `ch.removeFilter(filter_name)` # 删除一个过滤器

### StreamHandler

创建方法: `sh = logging.StreamHandler(stream=None)`

### FileHandler

创建方法: `fh = logging.FileHandler(filename, mode='a', encoding=None, delay=False)`

### NullHandler

NullHandler类位于核心logging包，不做任何的格式化或者输出。  
本质上它是个“什么都不做”的handler，由库开发者使用。

## Formatter 格式化器

使用Formatter对象设置日志信息最后的规则、结构和内容，默认的时间格式为%Y-%m-%d %H:%M:%S。

创建方法: `formatter = logging.Formatter(fmt=None, datefmt=None)`

其中，fmt是消息的格式化字符串，datefmt是日期字符串。如果不指明fmt，将使用'%(message)s'。如果不指明datefmt，将使用ISO8601日期格式。

## Filter 过滤器

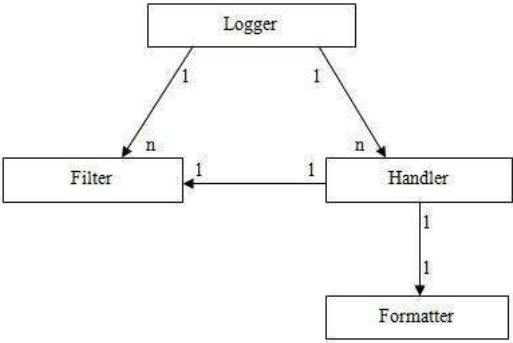
Handlers和Loggers可以使用Filters来完成比级别更复杂的过滤。Filter基类只允许特定Logger层次以下的事件。例如用'A.B'初始化的Filter允许Logger 'A.B', 'A.B.C', 'A.B.C.D', 'A.B.D'等记录的事件，logger'A.BB', 'B.A.B' 等就不行。如果用空字符串来初始化，所有的事件都接受。

创建方法: `filter = logging.Filter(name='')`

以下是相关概念总结:



熟悉了这些概念之后，有另外一个比较重要的事情必须清楚，即**Logger是一个树形层级结构**；  
Logger可以包含一个或多个Handler和Filter，即Logger与Handler或Filter是一对多的关系；  
一个Logger实例可以新增多个Handler，一个Handler可以新增多个格式化器或多个过滤器，而且日志级别将会继承。



element\_relation.jpg

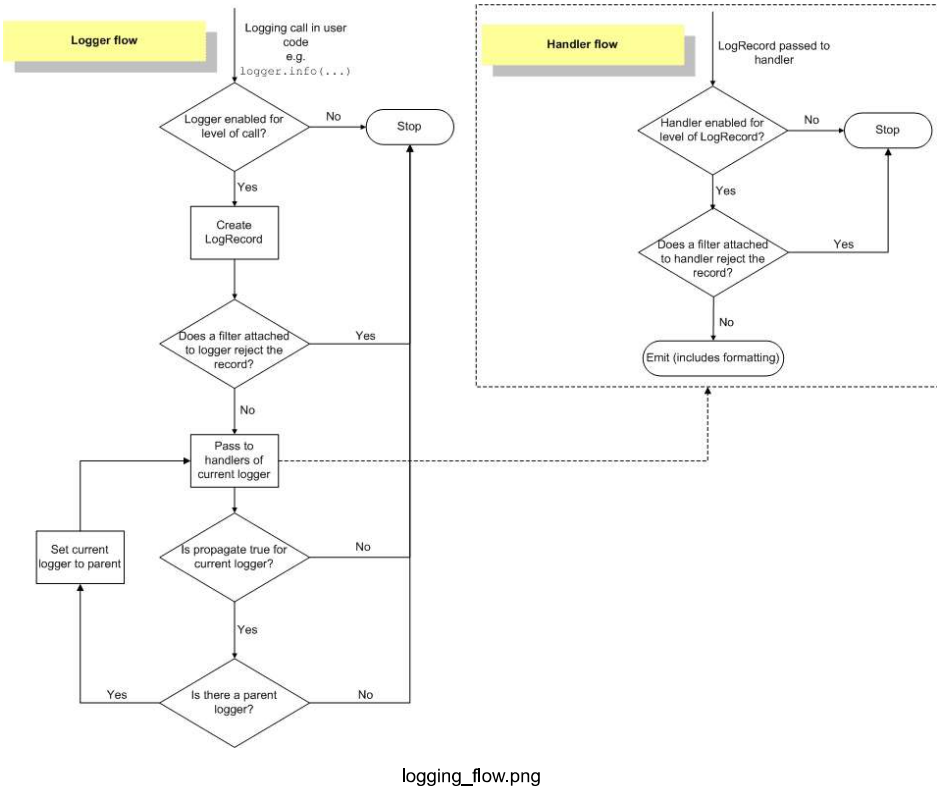
Logging工作流程

logging模块使用过程

- 1. 第一次导入logging模块或使用reload函数重新导入logging模块，logging模块中的代码将被执行，这个过程中将产生logging日志系统的默认配置。
- 2. 自定义配置(可选)。logging标准模块支持三种配置方式: dictConfig，fileConfig，listen。其中，dictConfig是通过一个字典进行配置Logger，Handler，Filter，Formatter；fileConfig则是通过一个文件进行配置；而listen则监听一个网络端口，通过接收网络数据来进行配置。当然，除了以上集体化配置外，也可以直接调用Logger，Handler等对象中的方法在代码中来显式配置。
- 3. 使用logging模块的全局作用域中的getLogger函数来得到一个Logger对象实例(其参数即是一个字符串，表示Logger对象实例的名字，即通过该名字来得到相应的Logger对象实例)。
- 4. 使用Logger对象中的debug，info，error，warn，critical等方法记录日志信息。

logging模块处理流程





- 判断日志的等级是否大于Logger对象的等级，如果大于，则往下执行，否则，流程结束。
- 产生日志。第一步，判断是否有异常，如果有，则添加异常信息。第二步，处理日志记录方法(如debug，info等)中的占位符，即一般的字符串格式化处理。
- 使用注册到Logger对象中的Filters进行过滤。如果有多个过滤器，则依次过滤；只要有一个过滤器返回假，则过滤结束，且该日志信息将丢弃，不再处理，而处理流程也至此结束。否则，处理流程往下执行。
- 在当前Logger对象中查找Handlers，如果找不到任何Handler，则往上到该Logger对象的父Logger中查找；如果找到一个或多个Handler，则依次用Handler来处理日志信息。但在每个Handler处理日志信息过程中，会首先判断日志信息的等级是否大于该Handler的等级，如果大于，则往下执行(由Logger对象进入Handler对象中)，否则，处理流程结束。
- 执行Handler对象中的filter方法，该方法会依次执行注册到该Handler对象中的Filter。如果有一个Filter判断该日志信息为假，则此后的所有Filter都不再执行，而直接将该日志信息丢弃，处理流程结束。
- 使用Formatter类格式化最终的输出结果。注：Formatter同上述第2步的字符串格式化不同，它会添加额外的信息，比如日志产生的时间，产生日志的源代码所在的源文件的路径等等。
- 真正地输出日志信息(到网络，文件，终端，邮件等)。至于输出到哪个目的地，由Handler的种类来决定。

注：以上内容摘抄自第三条参考资料，内容略有改动，转载特此声明。

再看日志配置

配置方式

- 显式创建记录器Logger、处理器Handler和格式化器Formatter，并进行相关设置；

- 通过简单方式进行配置，使用basicConfig()  
([http://python.usyiyi.cn/python\\_278/library/logging.html#logging.basicConfig](http://python.usyiyi.cn/python_278/library/logging.html#logging.basicConfig))函数直接进行配置；
- 通过配置文件进行配置，使用fileConfig()  
([http://python.usyiyi.cn/python\\_278/library/logging.config.html#logging.config.fileConfig](http://python.usyiyi.cn/python_278/library/logging.config.html#logging.config.fileConfig))函数读取配置文件；
- 通过配置字典进行配置，使用dictConfig()  
([http://python.usyiyi.cn/python\\_278/library/logging.config.html#logging.config.dictConfig](http://python.usyiyi.cn/python_278/library/logging.config.html#logging.config.dictConfig))函数读取配置信息；
- 通过网络进行配置，使用listen()  
([http://python.usyiyi.cn/python\\_278/library/logging.config.html#logging.config.listen](http://python.usyiyi.cn/python_278/library/logging.config.html#logging.config.listen))函数进行网络配置。

basicConfig关键字参数

关键字	描述
filename	创建一个FileHandler，使用指定的文件名，而不是使用StreamHandler。
filemode	如果指明了文件名，指明打开文件的模式（如果没有指明filemode，默认为'a'）。
format	handler使用指明的格式化字符串。
datefmt	使用指明的日期 / 时间格式。
level	指明根logger的级别。
stream	使用指明的流来初始化StreamHandler。该参数与'filename'不兼容，如果两个都有，'stream'被忽略。

有用的format格式

格式	描述
%(levelno)s	打印日志级别的数值
%(levelname)s	打印日志级别名称
%(pathname)s	打印当前执行程序的路径
%(filename)s	打印当前执行程序名称
%(funcName)s	打印日志的当前函数
%(lineno)d	打印日志的当前行号
%(asctime)s	打印日志的时间
%(thread)d	打印线程id
%(threadName)s	打印线程名称
%(process)d	打印进程ID
%(message)s	打印日志信息

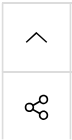
有用的datefmt格式

参考time.strftime (<https://docs.python.org/2/library/time.html?highlight=strftime#time.strftime>)

配置示例

显式配置

使用程序logger.py如下:



```
# -*- encoding:utf-8 -*-
import logging

# create logger
logger_name = "example"
logger = logging.getLogger(logger_name)
logger.setLevel(logging.DEBUG)

# create file handler
log_path = "./log.log"
fh = logging.FileHandler(log_path)
fh.setLevel(logging.WARN)

# create formatter
fmt = "%(asctime)-15s %(levelname)s %(filename)s %(lineno)d %(process)d %(message)s"
datefmt = "%a %d %b %Y %H:%M:%S"
formatter = logging.Formatter(fmt, datefmt)

# add handler and formatter to logger
fh.setFormatter(formatter)
logger.addHandler(fh)

# print log info
logger.debug('debug message')
logger.info('info message')
logger.warn('warn message')
logger.error('error message')
logger.critical('critical message')
```

## 文件配置

配置文件logging.conf如下:

```
keys=root,example01

[logger_root]
level=DEBUG
handlers=hand01,hand02

[logger_example01]
handlers=hand01,hand02
qualname=example01
propagate=0

[handlers]
keys=hand01,hand02

[handler_hand01]
class=StreamHandler
level=INFO
formatter=form02
args=(sys.stderr,)

[handler_hand02]
class=FileHandler
level=DEBUG
formatter=form01
args=('log.log', 'a')

[formatters]
keys=form01,form02

[formatter_form01]
format=%(asctime)s %(filename)s[line:%(lineno)d] %(levelname)s %(message)s
```

使用程序logger.py如下:



下载简书App  
创作你的创作

(/apps/download?  
utm\_source=stc)



```
# -*- encoding:utf-8 -*-
import logging
import logging.config

logging.config.fileConfig("./logging.conf")

# create logger
logger_name = "example"
logger = logging.getLogger(logger_name)

logger.debug('debug message')
logger.info('info message')
logger.warn('warn message')
logger.error('error message')
logger.critical('critical message')
```

## 字典配置

有兴趣的童鞋可以使用 `logging.config.dictConfig(config)` 编写一个示例程序发给我，以提供给我进行完善本文。

## 监听配置


有兴趣的童鞋可以使用 `logging.config.listen(port=DEFAULT_LOGGING_CONFIG_PORT)` 编写一个示例程序发给我，以提供给我进行完善本文。

更多详细内容参考logging.config日志配置

([http://python.usyiyi.cn/python\\_278/library/logging.config.html#module-logging.config](http://python.usyiyi.cn/python_278/library/logging.config.html#module-logging.config))

## 参考资料

- 英文Python logging HOWTO (<https://docs.python.org/2/howto/logging.html#logging-basic-tutorial>)
- 中文Python 日志 HOWTO ([http://python.usyiyi.cn/python\\_278/howto/logging.html#logging-basic-tutorial](http://python.usyiyi.cn/python_278/howto/logging.html#logging-basic-tutorial))
- Python日志系统Logging (<http://www.52ij.com/jishu/666.html>)
- logging模块学习笔记：basicConfig配置文件 (<http://www.cnblogs.com/bjdx/archive/2013/04/12/3016820.html>)
- 其他一些前辈博客相关文章

 Python (/nb/2550665)

[举报文章](#) © 著作权归作者所有



好吃的野菜 (/u/62bbd7b8d80c)

写了 3828 字，被 42 人关注，获得了 71 个喜欢  
(/u/62bbd7b8d80c)

+ 关注

我很懒，什么都不想写

♡ 喜欢 (/sign\_in?utm\_source=desktop&utm\_medium=not-signed-in-like-button) | 71



更多分享



(<http://cwb.assets.jianshu.io/notes/images/2448840>)