# Python logging模块详解

**简单将日志打印到屏幕：**

```
import logging
logging.debug('debug message')
logging.info('info message')
logging.warning('warning message')
logging.error('error message')
logging.critical('critical message')
```

输出：

WARNING:root:warning message
ERROR:root:error message
CRITICAL:root:critical message

可见，默认情况下python的logging模块将日志打印到了标准输出中，且只显示了大于等于WARNING级别的日志，这说明默认的日志级别设置为WARNING（日志级别等级CRITICAL > ERROR > WARNING > INFO > DEBUG > NOTSET），默认的日志格式为日志级别：Logger名称：用户输出消息。

### 灵活配置日志级别，日志格式，输出位置

```
import logging
logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s %(filename)s[line:%(lineno)d] %(levelname)s %(message)s',
                    datefmt='%a, %d %b %Y %H:%M:%S',
                    filename='/tmp/test.log',
                    filemode='w')

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

查看输出：
cat /tmp/test.log
Mon, 05 May 2014 16:29:53 test_logging.py[line:9] DEBUG debug message
Mon, 05 May 2014 16:29:53 test_logging.py[line:10] INFO info message
Mon, 05 May 2014 16:29:53 test_logging.py[line:11] WARNING warning message
Mon, 05 May 2014 16:29:53 test_logging.py[line:12] ERROR error message
Mon, 05 May 2014 16:29:53 test_logging.py[line:13] CRITICAL critical message

可见在logging.basicConfig()函数中可通过具体参数来更改logging模块默认行为，可用参数有
filename：用指定的文件名创建FiledHandler（后边会具体讲解handler的概念），这样日志会被存储在指定的文件中。
filemode：文件打开方式，在指定了filename时使用这个参数，默认值为“a”还可指定为“w”。
format：指定handler使用的日志显示格式。
datefmt：指定日期时间格式。
level：设置rootlogger（后边会讲解具体概念）的日志级别
stream：用指定的stream创建StreamHandler。可以指定输出到sys.stderr,sys.stdout或者文件，默认为sys.stderr。若同时列出了filename和stream两个参数，则stream参数会被忽略。

format参数中可能用到的格式化串：
%(name)s Logger的名字
%(levelno)s 数字形式的日志级别
%(levelname)s 文本形式的日志级别
%(pathname)s 调用日志输出函数的模块的完整路径名，可能没有
%(filename)s 调用日志输出函数的模块的文件名
%(module)s 调用日志输出函数的模块名
%(funcName)s 调用日志输出函数的函数名
%(lineno)d 调用日志输出函数的语句所在的代码行
%(created)f 当前时间，用UNIX标准的表示时间的浮 点数表示
%(relativeCreated)d 输出日志信息时的，自Logger创建以 来的毫秒数
%(asctime)s 字符串形式的当前时间。默认格式是 “2003-07-08 16:49:45,896”。逗号后面的是毫秒
%(thread)d 线程ID。可能没有
%(threadName)s 线程名。可能没有
%(process)d 进程ID。可能没有
%(message)s用户输出的消息

### 若要对logging进行更多灵活的控制有必要了解一下Logger，Handler，Formatter，Filter的概念

上述几个例子中我们了解到了logging.debug()、logging.info()、logging.warning()、logging.error()、logging.critical()（分别用以记录不同级别的日志信息），logging.basicConfig()（用默认日志格式（Formatter）为日志系统建立一个默认的流处理器（StreamHandler），设置基础配置（如日志级别等）并加到root logger（根Logger）中）这几个logging模块级别的函数，另外还有一个模块级别的函数是logging.getLogger([name])（返回一个logger对象，如果没有指定名字将返回root logger）

先看一个具体的例子

```
#coding:utf-8
import logging

# 创建一个logger
logger = logging.getLogger()

logger1 = logging.getLogger('mylogger')
logger1.setLevel(logging.DEBUG)

logger2 = logging.getLogger('mylogger')
logger2.setLevel(logging.INFO)

logger3 = logging.getLogger('mylogger.child1')
logger3.setLevel(logging.WARNING)

logger4 = logging.getLogger('mylogger.child1.child2')
logger4.setLevel(logging.DEBUG)

logger5 = logging.getLogger('mylogger.child1.child2.child3')
logger5.setLevel(logging.DEBUG)

# 创建一个handler，用于写入日志文件
fh = logging.FileHandler('/tmp/test.log')

# 再创建一个handler，用于输出到控制台
ch = logging.StreamHandler()

# 定义handler的输出格式formatter
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
fh.setFormatter(formatter)
ch.setFormatter(formatter)

#定义一个filter
#filter = logging.Filter('mylogger.child1.child2')
#fh.addFilter(filter)

# 给logger添加handler
#logger.addFilter(filter)
logger.addHandler(fh)
logger.addHandler(ch)

#logger1.addFilter(filter)
logger1.addHandler(fh)
logger1.addHandler(ch)

logger2.addHandler(fh)
logger2.addHandler(ch)

#logger3.addFilter(filter)
logger3.addHandler(fh)
logger3.addHandler(ch)

#logger4.addFilter(filter)
logger4.addHandler(fh)
logger4.addHandler(ch)

logger5.addHandler(fh)
logger5.addHandler(ch)

# 记录一条日志
logger.debug('logger debug message')
logger.info('logger info message')
logger.warning('logger warning message')
logger.error('logger error message')
logger.critical('logger critical message')

logger1.debug('logger1 debug message')
logger1.info('logger1 info message')
logger1.warning('logger1 warning message')
logger1.error('logger1 error message')
logger1.critical('logger1 critical message')

logger2.debug('logger2 debug message')
```

```
logger2.info('logger2 info message')
logger2.warning('logger2 warning message')
logger2.error('logger2 error message')
logger2.critical('logger2 critical message')

logger3.debug('logger3 debug message')
logger3.info('logger3 info message')
logger3.warning('logger3 warning message')
logger3.error('logger3 error message')
logger3.critical('logger3 critical message')

logger4.debug('logger4 debug message')
logger4.info('logger4 info message')
logger4.warning('logger4 warning message')
logger4.error('logger4 error message')
logger4.critical('logger4 critical message')

logger5.debug('logger5 debug message')
logger5.info('logger5 info message')
logger5.warning('logger5 warning message')
logger5.error('logger5 error message')
logger5.critical('logger5 critical message')
```

输出：

```
2014-05-06 12:54:43,222 - root - WARNING - logger warning message
2014-05-06 12:54:43,223 - root - ERROR - logger error message
2014-05-06 12:54:43,224 - root - CRITICAL - logger critical message
2014-05-06 12:54:43,224 - mylogger - INFO - logger1 info message
2014-05-06 12:54:43,224 - mylogger - INFO - logger1 info message
2014-05-06 12:54:43,225 - mylogger - WARNING - logger1 warning message
2014-05-06 12:54:43,225 - mylogger - WARNING - logger1 warning message
2014-05-06 12:54:43,226 - mylogger - ERROR - logger1 error message
2014-05-06 12:54:43,226 - mylogger - ERROR - logger1 error message
2014-05-06 12:54:43,227 - mylogger - CRITICAL - logger1 critical message
2014-05-06 12:54:43,227 - mylogger - CRITICAL - logger1 critical message
2014-05-06 12:54:43,228 - mylogger - INFO - logger2 info message
2014-05-06 12:54:43,228 - mylogger - INFO - logger2 info message
2014-05-06 12:54:43,229 - mylogger - WARNING - logger2 warning message
2014-05-06 12:54:43,229 - mylogger - WARNING - logger2 warning message
2014-05-06 12:54:43,230 - mylogger - ERROR - logger2 error message
2014-05-06 12:54:43,230 - mylogger - ERROR - logger2 error message
2014-05-06 12:54:43,231 - mylogger - CRITICAL - logger2 critical message
2014-05-06 12:54:43,231 - mylogger - CRITICAL - logger2 critical message
2014-05-06 12:54:43,232 - mylogger.child1 - WARNING - logger3 warning message
2014-05-06 12:54:43,232 - mylogger.child1 - WARNING - logger3 warning message
2014-05-06 12:54:43,232 - mylogger.child1 - WARNING - logger3 warning message
2014-05-06 12:54:43,234 - mylogger.child1 - ERROR - logger3 error message
2014-05-06 12:54:43,234 - mylogger.child1 - ERROR - logger3 error message
2014-05-06 12:54:43,234 - mylogger.child1 - ERROR - logger3 error message
2014-05-06 12:54:43,235 - mylogger.child1 - CRITICAL - logger3 critical message
2014-05-06 12:54:43,235 - mylogger.child1 - CRITICAL - logger3 critical message
2014-05-06 12:54:43,235 - mylogger.child1 - CRITICAL - logger3 critical message
2014-05-06 12:54:43,237 - mylogger.child1.child2 - DEBUG - logger4 debug message
2014-05-06 12:54:43,237 - mylogger.child1.child2 - DEBUG - logger4 debug message
2014-05-06 12:54:43,237 - mylogger.child1.child2 - DEBUG - logger4 debug message
2014-05-06 12:54:43,237 - mylogger.child1.child2 - DEBUG - logger4 debug message
2014-05-06 12:54:43,239 - mylogger.child1.child2 - INFO - logger4 info message
2014-05-06 12:54:43,239 - mylogger.child1.child2 - INFO - logger4 info message
2014-05-06 12:54:43,239 - mylogger.child1.child2 - INFO - logger4 info message
2014-05-06 12:54:43,239 - mylogger.child1.child2 - INFO - logger4 info message
2014-05-06 12:54:43,240 - mylogger.child1.child2 - WARNING - logger4 warning message
2014-05-06 12:54:43,240 - mylogger.child1.child2 - WARNING - logger4 warning message
2014-05-06 12:54:43,240 - mylogger.child1.child2 - WARNING - logger4 warning message
2014-05-06 12:54:43,240 - mylogger.child1.child2 - WARNING - logger4 warning message
2014-05-06 12:54:43,242 - mylogger.child1.child2 - ERROR - logger4 error message
2014-05-06 12:54:43,242 - mylogger.child1.child2 - ERROR - logger4 error message
2014-05-06 12:54:43,242 - mylogger.child1.child2 - ERROR - logger4 error message
2014-05-06 12:54:43,242 - mylogger.child1.child2 - ERROR - logger4 error message
2014-05-06 12:54:43,243 - mylogger.child1.child2 - CRITICAL - logger4 critical message
2014-05-06 12:54:43,243 - mylogger.child1.child2 - CRITICAL - logger4 critical message
2014-05-06 12:54:43,243 - mylogger.child1.child2 - CRITICAL - logger4 critical message
```

2014-05-06 12:54:43,243 - mylogger.child1.child2 - CRITICAL - logger4 critical message
2014-05-06 12:54:43,244 - mylogger.child1.child2.child3 - DEBUG - logger5 debug message
2014-05-06 12:54:43,244 - mylogger.child1.child2.child3 - DEBUG - logger5 debug message
2014-05-06 12:54:43,244 - mylogger.child1.child2.child3 - DEBUG - logger5 debug message
2014-05-06 12:54:43,244 - mylogger.child1.child2.child3 - DEBUG - logger5 debug message
2014-05-06 12:54:43,244 - mylogger.child1.child2.child3 - DEBUG - logger5 debug message
2014-05-06 12:54:43,246 - mylogger.child1.child2.child3 - INFO - logger5 info message
2014-05-06 12:54:43,246 - mylogger.child1.child2.child3 - INFO - logger5 info message
2014-05-06 12:54:43,246 - mylogger.child1.child2.child3 - INFO - logger5 info message
2014-05-06 12:54:43,246 - mylogger.child1.child2.child3 - INFO - logger5 info message
2014-05-06 12:54:43,246 - mylogger.child1.child2.child3 - INFO - logger5 info message
2014-05-06 12:54:43,247 - mylogger.child1.child2.child3 - WARNING - logger5 warning message
2014-05-06 12:54:43,247 - mylogger.child1.child2.child3 - WARNING - logger5 warning message
2014-05-06 12:54:43,247 - mylogger.child1.child2.child3 - WARNING - logger5 warning message
2014-05-06 12:54:43,247 - mylogger.child1.child2.child3 - WARNING - logger5 warning message
2014-05-06 12:54:43,247 - mylogger.child1.child2.child3 - WARNING - logger5 warning message
2014-05-06 12:54:43,249 - mylogger.child1.child2.child3 - ERROR - logger5 error message
2014-05-06 12:54:43,249 - mylogger.child1.child2.child3 - ERROR - logger5 error message
2014-05-06 12:54:43,249 - mylogger.child1.child2.child3 - ERROR - logger5 error message
2014-05-06 12:54:43,249 - mylogger.child1.child2.child3 - ERROR - logger5 error message
2014-05-06 12:54:43,249 - mylogger.child1.child2.child3 - ERROR - logger5 error message
2014-05-06 12:54:43,250 - mylogger.child1.child2.child3 - CRITICAL - logger5 critical message
2014-05-06 12:54:43,250 - mylogger.child1.child2.child3 - CRITICAL - logger5 critical message
2014-05-06 12:54:43,250 - mylogger.child1.child2.child3 - CRITICAL - logger5 critical message
2014-05-06 12:54:43,250 - mylogger.child1.child2.child3 - CRITICAL - logger5 critical message
2014-05-06 12:54:43,250 - mylogger.child1.child2.child3 - CRITICAL - logger5 critical message

先简单介绍一下，logging库提供了多个组件：Logger、Handler、Filter、Formatter。Logger对象提供应用程序可直接使用的接口，Handler发送日志到适当的目的地，Filter提供了过滤日志信息的方法，Formatter指定日志显示格式。

**Logger**

Logger是一个树形层级结构，输出信息之前都要获得一个Logger（如果没有显示的获取则自动创建并使用root Logger，如第一个例子所示）。
logger = logging.getLogger()返回一个默认的Logger也即root Logger，并应用默认的日志级别、Handler和Formatter设置。
当然也可以通过Logger.setLevel(lel)指定最低的日志级别，可用的日志级别有logging.DEBUG、logging.INFO、logging.WARNING、logging.ERROR、logging.CRITICAL。
Logger.debug()、Logger.info()、Logger.warning()、Logger.error()、Logger.critical()输出不同级别的日志，只有日志等级大于或等于设置的日志级别的日志才会被输出。

我们看到程序中

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

只输出了
2014-05-06 12:54:43,222 - root - WARNING - logger warning message
2014-05-06 12:54:43,223 - root - ERROR - logger error message
2014-05-06 12:54:43,224 - root - CRITICAL - logger critical message
从这个输出可以看出logger = logging.getLogger()返回的Logger名为root。这里没有用logger.setLevel()显示的为logger设置日志级别，所以使用默认的日志级别WARNIING，故结果只输出了大于等于WARNIING级别的信息。

另外，我们明明通过logger1.setLevel(logging.DEBUG)将logger1的日志级别设置为了DEBUG，为何显示的时候没有显示出DEBUG级别的日志信息，而是从INFO级别的日志开始显示呢？原来logger1和logger2对应的是同一个Logger实例，只要logging.getLogger（name）中名称参数name相同则返回的Logger实例就是同一个，且仅有一个，也即name与Logger实例一一对应。在logger2实例中通过logger2.setLevel(logging.INFO)设置mylogger的日志级别为logging.INFO，所以最后logger1的输出遵从了后来设置的日志级别。

```
logger1 = logging.getLogger('mylogger')
logger1.setLevel(logging.DEBUG)
logger2 = logging.getLogger('mylogger')
logger2.setLevel(logging.INFO)
```

为什么logger1、logger2对应的每个输出分别显示两次，logger3对应的输出显示3次，logger4对应的输出显示4次.....呢？
这是因为我们通过logger = logging.getLogger()显示的创建了root Logger，而logger1 = logging.getLogger('mylogger')创建了root Logger的孩子(root.)mylogger,logger2同样。
logger3 = logging.getLogger('mylogger.child1')创建了(root.)mylogger.child1
logger4 = logging.getLogger('mylogger.child1.child2')创建了(root.)mylogger.child1.child2
logger5 = logging.getLogger('mylogger.child1.child2.child3')创建了(root.)mylogger.child1.child2.child3
而孩子,孙子，重孙……既会将消息分发给他的handler进行处理也会传递给所有的祖先Logger处理。

试着注释掉如下一行程序，观察程序输出

```
#logger.addHandler(fh)
```

发现标准输出中每条记录对应两行（因为root Logger默认使用StreamHandler）
2014-05-06 15:10:10,980 - mylogger - INFO - logger1 info message
2014-05-06 15:10:10,980 - mylogger - INFO - logger1 info message
2014-05-06 15:10:10,981 - mylogger - WARNING - logger1 warning message
2014-05-06 15:10:10,981 - mylogger - WARNING - logger1 warning message
2014-05-06 15:10:10,982 - mylogger - ERROR - logger1 error message
2014-05-06 15:10:10,982 - mylogger - ERROR - logger1 error message
2014-05-06 15:10:10,984 - mylogger - CRITICAL - logger1 critical message
2014-05-06 15:10:10,984 - mylogger - CRITICAL - logger1 critical message
……
而在文件输出中每条记录对应一行（因为我们注释掉了logger.addHandler(fh)，没有对root Logger启用FileHandler）
2014-05-06 15:10:10,980 - mylogger - INFO - logger1 info message
2014-05-06 15:10:10,981 - mylogger - WARNING - logger1 warning message
2014-05-06 15:10:10,982 - mylogger - ERROR - logger1 error message
2014-05-06 15:10:10,984 - mylogger - CRITICAL - logger1 critical message

孩子,孙子，重孙……可逐层继承来自祖先的日志级别、Handler、Filter设置，也可以通过Logger.setLevel(lel)、Logger.addHandler(hdlr)、Logger.removeHandler(hdlr)、Logger.addFilter(filt)、Logger.removeFilter(filt)。设置自己特别的日志级别、Handler、Filter。若不设置则使用继承来的值。

**Handler**
上述例子的输出在标准输出和指定的日志文件中均可以看到，这是因为我们定义并使用了两种Handler。

```
fh = logging.FileHandler('/tmp/test.log')
ch = logging.StreamHandler()
```

Handler对象负责发送相关的信息到指定目的地，有几个常用的Handler方法：
Handler.setLevel(lel):指定日志级别，低于lel级别的日志将被忽略
Handler.setFormatter()：给这个handler选择一个Formatter
Handler.addFilter(filt)、Handler.removeFilter(filt)：新增或删除一个filter对象

可以通过addHandler()方法为Logger添加多个Handler：

```
logger.addHandler(fh)
logger.addHandler(ch)
```

有多中可用的 Handler ：

logging.StreamHandler 可以向类似与sys.stdout或者sys.stderr的任何文件对象(file object)输出信息
logging.FileHandler 用于向一个文件输出日志信息
logging.handlers.RotatingFileHandler 类似于上面的FileHandler，但是它可以管理文件大小。当文件达到一定大小之后，它会自动将当前日志文件改名，然后创建一个新的同名日志文件继续输出
logging.handlers.TimedRotatingFileHandler 和RotatingFileHandler类似，不过，它没有通过判断文件大小来决定何时重新创建日志文件，而是间隔一定时间就自动创建新的日志文件
logging.handlers.SocketHandler 使用TCP协议，将日志信息发送到网络。
logging.handlers.DatagramHandler 使用UDP协议，将日志信息发送到网络。
logging.handlers.SysLogHandler 日志输出到syslog
logging.handlers.NTEventLogHandler 远程输出日志到Windows NT/2000/XP的事件日志
logging.handlers.SMTPHandler 远程输出日志到邮件地址
logging.handlers.MemoryHandler 日志输出到内存中的制定buffer
logging.handlers.HTTPHandler 通过"GET"或"POST"远程输出到HTTP服务器
各个Handler的具体用法可查看参考书册：
https://docs.python.org/2/library/logging.handlers.html#module-logging.handlers

**Formatter**
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。

```
#定义Formatter
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
#为Handler添加Formatter
fh.setFormatter(formatter)
ch.setFormatter(formatter)
```

Formatter参数中可能用到的格式化串参见上文（logging.basicConfig()函数format参数中可能用到的格式化串：）

**Filter**

限制只有满足过滤规则的日志才会输出。
比如我们定义了filter = logging.Filter('a.b.c'),并将这个Filter添加到了一个Handler上，则使用该Handler的Logger中只有名字带a.b.c前缀的Logger才能输出其日志。

取消下列两行程序的注释

```
#filter = logging.Filter('mylogger.child1.child2')
#fh.addFilter(filter)
```

标准输出中输出结果并没有发生变化，但日志文件输出中只显示了如下内容：

```
2014-05-06 15:27:36,227 - mylogger.child1.child2 - DEBUG - logger4 debug message
2014-05-06 15:27:36,227 - mylogger.child1.child2 - DEBUG - logger4 debug message
2014-05-06 15:27:36,227 - mylogger.child1.child2 - DEBUG - logger4 debug message
2014-05-06 15:27:36,227 - mylogger.child1.child2 - DEBUG - logger4 debug message
2014-05-06 15:27:36,228 - mylogger.child1.child2 - INFO - logger4 info message
2014-05-06 15:27:36,228 - mylogger.child1.child2 - INFO - logger4 info message
2014-05-06 15:27:36,228 - mylogger.child1.child2 - INFO - logger4 info message
2014-05-06 15:27:36,228 - mylogger.child1.child2 - INFO - logger4 info message
2014-05-06 15:27:36,230 - mylogger.child1.child2 - WARNING - logger4 warning message
2014-05-06 15:27:36,230 - mylogger.child1.child2 - WARNING - logger4 warning message
2014-05-06 15:27:36,230 - mylogger.child1.child2 - WARNING - logger4 warning message
2014-05-06 15:27:36,230 - mylogger.child1.child2 - WARNING - logger4 warning message
2014-05-06 15:27:36,232 - mylogger.child1.child2 - ERROR - logger4 error message
2014-05-06 15:27:36,232 - mylogger.child1.child2 - ERROR - logger4 error message
2014-05-06 15:27:36,232 - mylogger.child1.child2 - ERROR - logger4 error message
2014-05-06 15:27:36,232 - mylogger.child1.child2 - ERROR - logger4 error message
2014-05-06 15:27:36,233 - mylogger.child1.child2 - CRITICAL - logger4 critical message
2014-05-06 15:27:36,233 - mylogger.child1.child2 - CRITICAL - logger4 critical message
2014-05-06 15:27:36,233 - mylogger.child1.child2 - CRITICAL - logger4 critical message
2014-05-06 15:27:36,233 - mylogger.child1.child2 - CRITICAL - logger4 critical message
2014-05-06 15:27:36,235 - mylogger.child1.child2.child3 - DEBUG - logger5 debug message
2014-05-06 15:27:36,235 - mylogger.child1.child2.child3 - DEBUG - logger5 debug message
2014-05-06 15:27:36,235 - mylogger.child1.child2.child3 - DEBUG - logger5 debug message
2014-05-06 15:27:36,235 - mylogger.child1.child2.child3 - DEBUG - logger5 debug message
2014-05-06 15:27:36,235 - mylogger.child1.child2.child3 - DEBUG - logger5 debug message
2014-05-06 15:27:36,236 - mylogger.child1.child2.child3 - INFO - logger5 info message
2014-05-06 15:27:36,236 - mylogger.child1.child2.child3 - INFO - logger5 info message
2014-05-06 15:27:36,236 - mylogger.child1.child2.child3 - INFO - logger5 info message
2014-05-06 15:27:36,236 - mylogger.child1.child2.child3 - INFO - logger5 info message
2014-05-06 15:27:36,236 - mylogger.child1.child2.child3 - INFO - logger5 info message
2014-05-06 15:27:36,238 - mylogger.child1.child2.child3 - WARNING - logger5 warning message
2014-05-06 15:27:36,238 - mylogger.child1.child2.child3 - WARNING - logger5 warning message
2014-05-06 15:27:36,238 - mylogger.child1.child2.child3 - WARNING - logger5 warning message
2014-05-06 15:27:36,238 - mylogger.child1.child2.child3 - WARNING - logger5 warning message
2014-05-06 15:27:36,238 - mylogger.child1.child2.child3 - WARNING - logger5 warning message
2014-05-06 15:27:36,240 - mylogger.child1.child2.child3 - ERROR - logger5 error message
2014-05-06 15:27:36,240 - mylogger.child1.child2.child3 - ERROR - logger5 error message
2014-05-06 15:27:36,240 - mylogger.child1.child2.child3 - ERROR - logger5 error message
2014-05-06 15:27:36,240 - mylogger.child1.child2.child3 - ERROR - logger5 error message
2014-05-06 15:27:36,240 - mylogger.child1.child2.child3 - ERROR - logger5 error message
2014-05-06 15:27:36,242 - mylogger.child1.child2.child3 - CRITICAL - logger5 critical message
2014-05-06 15:27:36,242 - mylogger.child1.child2.child3 - CRITICAL - logger5 critical message
2014-05-06 15:27:36,242 - mylogger.child1.child2.child3 - CRITICAL - logger5 critical message
2014-05-06 15:27:36,242 - mylogger.child1.child2.child3 - CRITICAL - logger5 critical message
2014-05-06 15:27:36,242 - mylogger.child1.child2.child3 - CRITICAL - logger5 critical message
```

当然也可以直接给Logger加Filter。若为Handler加Filter则所有使用了该Handler的Logger都会受到影响。而为Logger添加Filter只会影响到自身。
注释掉

```
#fh.addFilter(filter)
```

并取消如下几行的注释

```
#logger.addFilter(filter)
#logger1.addFilter(filter)
#logger3.addFilter(filter)
#logger4.addFilter(filter)
```

输出结果

```
2014-05-06 15:32:10,746 - mylogger.child1.child2 - DEBUG - logger4 debug message
2014-05-06 15:32:10,746 - mylogger.child1.child2 - DEBUG - logger4 debug message
2014-05-06 15:32:10,746 - mylogger.child1.child2 - DEBUG - logger4 debug message
2014-05-06 15:32:10,746 - mylogger.child1.child2 - DEBUG - logger4 debug message
2014-05-06 15:32:10,748 - mylogger.child1.child2 - INFO - logger4 info message
2014-05-06 15:32:10,748 - mylogger.child1.child2 - INFO - logger4 info message
2014-05-06 15:32:10,748 - mylogger.child1.child2 - INFO - logger4 info message
2014-05-06 15:32:10,748 - mylogger.child1.child2 - INFO - logger4 info message
2014-05-06 15:32:10,751 - mylogger.child1.child2 - WARNING - logger4 warning message
2014-05-06 15:32:10,751 - mylogger.child1.child2 - WARNING - logger4 warning message
2014-05-06 15:32:10,751 - mylogger.child1.child2 - WARNING - logger4 warning message
2014-05-06 15:32:10,751 - mylogger.child1.child2 - WARNING - logger4 warning message
2014-05-06 15:32:10,753 - mylogger.child1.child2 - ERROR - logger4 error message
2014-05-06 15:32:10,753 - mylogger.child1.child2 - ERROR - logger4 error message
2014-05-06 15:32:10,753 - mylogger.child1.child2 - ERROR - logger4 error message
2014-05-06 15:32:10,753 - mylogger.child1.child2 - ERROR - logger4 error message
2014-05-06 15:32:10,754 - mylogger.child1.child2 - CRITICAL - logger4 critical message
2014-05-06 15:32:10,754 - mylogger.child1.child2 - CRITICAL - logger4 critical message
2014-05-06 15:32:10,754 - mylogger.child1.child2 - CRITICAL - logger4 critical message
2014-05-06 15:32:10,754 - mylogger.child1.child2 - CRITICAL - logger4 critical message
2014-05-06 15:32:10,755 - mylogger.child1.child2.child3 - DEBUG - logger5 debug message
2014-05-06 15:32:10,755 - mylogger.child1.child2.child3 - DEBUG - logger5 debug message
2014-05-06 15:32:10,755 - mylogger.child1.child2.child3 - DEBUG - logger5 debug message
2014-05-06 15:32:10,755 - mylogger.child1.child2.child3 - DEBUG - logger5 debug message
2014-05-06 15:32:10,755 - mylogger.child1.child2.child3 - DEBUG - logger5 debug message
2014-05-06 15:32:10,757 - mylogger.child1.child2.child3 - INFO - logger5 info message
2014-05-06 15:32:10,757 - mylogger.child1.child2.child3 - INFO - logger5 info message
2014-05-06 15:32:10,757 - mylogger.child1.child2.child3 - INFO - logger5 info message
2014-05-06 15:32:10,757 - mylogger.child1.child2.child3 - INFO - logger5 info message
2014-05-06 15:32:10,757 - mylogger.child1.child2.child3 - INFO - logger5 info message
2014-05-06 15:32:10,759 - mylogger.child1.child2.child3 - WARNING - logger5 warning message
2014-05-06 15:32:10,759 - mylogger.child1.child2.child3 - WARNING - logger5 warning message
2014-05-06 15:32:10,759 - mylogger.child1.child2.child3 - WARNING - logger5 warning message
2014-05-06 15:32:10,759 - mylogger.child1.child2.child3 - WARNING - logger5 warning message
2014-05-06 15:32:10,759 - mylogger.child1.child2.child3 - WARNING - logger5 warning message
2014-05-06 15:32:10,761 - mylogger.child1.child2.child3 - ERROR - logger5 error message
2014-05-06 15:32:10,761 - mylogger.child1.child2.child3 - ERROR - logger5 error message
2014-05-06 15:32:10,761 - mylogger.child1.child2.child3 - ERROR - logger5 error message
2014-05-06 15:32:10,761 - mylogger.child1.child2.child3 - ERROR - logger5 error message
2014-05-06 15:32:10,761 - mylogger.child1.child2.child3 - ERROR - logger5 error message
2014-05-06 15:32:10,762 - mylogger.child1.child2.child3 - CRITICAL - logger5 critical message
2014-05-06 15:32:10,762 - mylogger.child1.child2.child3 - CRITICAL - logger5 critical message
2014-05-06 15:32:10,762 - mylogger.child1.child2.child3 - CRITICAL - logger5 critical message
2014-05-06 15:32:10,762 - mylogger.child1.child2.child3 - CRITICAL - logger5 critical message
2014-05-06 15:32:10,762 - mylogger.child1.child2.child3 - CRITICAL - logger5 critical message
```

发现root、mylogger、mylogger.child1的输出全部被过滤掉了。

**除了直接在程序中设置Logger，Handler,Filter,Formatter外还可以将这些信息写进配置文件中。**

例如典型的logging.conf

```
[loggers]
keys=root,simpleExample

[handlers]
keys=consoleHandler

[formatters]
keys=simpleFormatter

[logger_root]
level=DEBUG
handlers=consoleHandler

[logger_simpleExample]
level=DEBUG
handlers=consoleHandler
qualname=simpleExample
propagate=0

```

```
[handler_consoleHandler]
class=StreamHandler
level=DEBUG
formatter=simpleFormatter
args=(sys.stdout,)

[formatter_simpleFormatter]
format=%(asctime)s - %(name)s - %(levelname)s - %(message)s
datefmt=
```

程序可以这么写

```
import logging
import logging.config

logging.config.fileConfig("logging.conf")    # 采用配置文件

# create logger
logger = logging.getLogger("simpleExample")

# "application" code
logger.debug("debug message")
logger.info("info message")
logger.warn("warn message")
logger.error("error message")
logger.critical("critical message")
```

**多模块使用logging**

logging模块保证在同一个python解释器内，多次调用logging.getLogger('log_name')都会返回同一个logger实例，即使是在多个模块的情况下。所以典型的多模块场景下使用logging的方式是在main模块中配置logging，这个配置会作用于多个的子模块，然后在其他模块中直接通过getLogger获取Logger对象即可。

main.py：

```
import logging
import logging.config

logging.config.fileConfig('logging.conf')
root_logger = logging.getLogger('root')
root_logger.debug('test root logger...')

logger = logging.getLogger('main')
logger.info('test main logger')
logger.info('start import module \'mod\'...')
import mod

logger.debug('let\'s test mod.testLogger()')
mod.testLogger()

root_logger.info('finish test...')
```

子模块mod.py：

```
import logging
import submod

logger = logging.getLogger('main.mod')
logger.info('logger of mod say something...')

def testLogger():
    logger.debug('this is mod.testLogger...')
    submod.tst()
```

子子模块submod.py：

```
import logging

logger = logging.getLogger('main.mod.submod')
logger.info('logger of submod say something...')

def tst():
    logger.info('this is submod.tst()...')
```