# logging

**默认情况下，logging将日志打印到屏幕，日志级别为WARNING；**
**日志级别大小关系为：CRITICAL > ERROR > WARNING > INFO > DEBUG > NOTSET。**
**打印和输出都是大于等于定义的错误级别**

--logging.notest(str)/debug(str)/info(str)/warning(str)/error(str)/critical(str)

--logging.NOTEST/DEBUG/INFO/WARNING/ERROR/CRITICAL

--logging.basicConfig(level = logging.WARNING[, format=None][,datefmt=None][,filename=None][filemode=None])

logging.basicConfig函数各参数:
filename: 指定日志文件名
filemode: 和file函数意义相同，指定日志文件的打开模式，'w'或'a'
format: 指定输出的格式和内容，format可以输出很多有用信息，如上例所示:
 %(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，其实就是sys.argv[0]
 %(filename)s: 打印当前执行程序名
 %(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
datefmt: 指定时间格式，同time.strftime()
level: 设置日志级别，默认为logging.WARNING

FE:

```
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='myapp.log',
                filemode='w')

logging.debug('This is debug message')
logging.info('This is info message')
logging.warning('This is warning message')
```

./myapp.log文件中内容为:

```
Sun, 24 May 2009 21:48:54 demo2.py[line:11] DEBUG This is debug message
Sun, 24 May 2009 21:48:54 demo2.py[line:12] INFO This is info message
Sun, 24 May 2009 21:48:54 demo2.py[line:13] WARNING This is warning message
```

```
============================================================
__author__='VicDong'
import logging

logging.basicConfig(level=logging.WARNING,
                    format='%(asctime)s - %(filename)s[line:%(lineno)d] - %(levelname)s: %(message)s')
# use logging
logging.info('this is a loggging info message')
logging.debug('this is a loggging debug message')
logging.warning('this is loggging a warning message')
logging.error('this is an loggging error message')
logging.critical('this is a loggging critical message')
```

consule中输出内容是:

```
>>>
============== RESTART: C:\Users\lenovo\Desktop\TemporaryPy.py ==============
2017-02-02 21:22:22,780 - TemporaryPy.py[line:9] - WARNING: this is loggging a warning message
2017-02-02 21:22:22,795 - TemporaryPy.py[line:10] - ERROR: this is an loggging error message
2017-02-02 21:22:22,811 - TemporaryPy.py[line:11] - CRITICAL: this is a loggging critical message
```

--class logger = logging.getLogger()

--class fh=logging.FileHandler(filename,mode='w')
设置handler，用于写入文件

--class ch = logging.StreamHandler()
设置handler，用于屏幕输出

--class formatter = logging.Formatter(format)
设置输出格式

--oneclass.setLevel(logging.INFO)
设置等级

--oneclass.setFormatter(formatter)
设置输出格式

--logger.addHandler(oneclass)
将控制文件/输出的class绑定到logger中

FE：

```
__author__ = 'liu.chunming'
import logging

# 第一步，创建一个logger
logger = logging.getLogger()
logger.setLevel(logging.INFO)    # Log等级总开关

# 第二步，创建一个handler，用于写入日志文件
logfile = './log/logger.txt'
fh = logging.FileHandler(logfile, mode='w')
fh.setLevel(logging.DEBUG)   # 输出到file的log等级的开关

# 第三步，再创建一个handler，用于输出到控制台
ch = logging.StreamHandler()
ch.setLevel(logging.WARNING)   # 输出到console的log等级的开关
formatter = logging.Formatter("%(asctime)s - %(filename)s[line:%(lineno)d] - %(levelname)s: %(message)s")
fh.setFormatter(formatter)
ch.setFormatter(formatter)

# 第五步，将logger添加到handler里面
logger.addHandler(fh)
logger.addHandler(ch)

# 日志
logger.debug('this is a logger debug message')
logger.info('this is a logger info message')
logger.warning('this is a logger warning message')
logger.error('this is a logger error message')
logger.critical('this is a logger critical message')
```