1.format：指定输出的格式和内容

%(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

stream: 指定将日志的输出流，可以指定输出到sys.stderr,sys.stdout或者文件，默认输出到sys.stderr，当stream和filename同时指定时，stream被忽略

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

datefmt=%a, %d %b %Y %H:%M:%S

2.logging的几种handle方式

logging.StreamHandler: 日志输出到流，可以是sys.stderr、sys.stdout或者文件

logging.FileHandler: 日志输出到文件

日志回滚方式，实际使用时用RotatingFileHandler和TimedRotatingFileHandler

logging.handlers.BaseRotatingHandler

logging.handlers.RotatingFileHandler

logging.handlers.TimedRotatingFileHandler

logging.handlers.SocketHandler: 远程输出日志到TCP/IP sockets

logging.handlers.DatagramHandler: 远程输出日志到UDP sockets

logging.handlers.SMTPHandler: 远程输出日志到邮件地址

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.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

TimedRotatingFileHandler用法:

```
#日志打印格式
log_fmt = '%(asctime)s\tFile \"%(filename)s\",line %(lineno)s\t%(levelname)s: %(message)s'
formatter = logging.Formatter(log_fmt)
#创建TimedRotatingFileHandler对象
log_file_handler = TimedRotatingFileHandler(filename="ds_update", when="M", interval=2, backupCount=2)
#log_file_handler.suffix = "%Y-%m-%d_%H-%M.log"
#log_file_handler.extMatch = re.compile(r"^\d{4}-\d{2}-\d{2}_\d{2}-\d{2}.log$") log_file_handler.setFormatter(formatter)
logging.basicConfig(level=logging.INFO)
log = logging.getLogger()
log.addHandler(log_file_handler)
Log.info('test!!')
```

- filename：日志文件名的prefix；
- when：是一个字符串，用于描述滚动周期的基本单位，字符串的值及意义如下：

“S”：Seconds

“M”：Minutes

“H”：Hours

“D”：Days

“W”：Week day (0=Monday)

“midnight”：Roll over at midnight

- interval: 滚动周期，单位有when指定，比如：when=’D’,interval=1，表示每天产生一个日志文件；
- backupCount: 表示日志文件的保留个数；

来自〈https://blog.csdn.net/ashi198866/article/details/46725813〉

关于日志level.

共有8个级别，按照从低到高为：All < Trace < Debug < Info < Warn < Error < Fatal < OFF.

All:最低等级的，用于打开所有日志记录.

Trace:是追踪，就是程序推进以下，你就可以写个trace输出，所以trace应该会特别多，不过没关系，我们可以设置最低日志级别不让他输出.

Debug:指出细粒度信息事件对调试应用程序是非常有帮助的.

Info:消息在粗粒度级别上突出强调应用程序的运行过程.

Warn:输出警告及warn以下级别的日志.

Error:输出错误信息日志.

Fatal:输出每个严重的错误事件将会导致应用程序的退出的日志.

OFF:最高等级的，用于关闭所有日志记录.

来自〈https://blog.csdn.net/Q176782/article/details/78288734〉