

咸鱼也是有梦想的

博客园 首页 新随笔 联系 订阅 管理

python中logging日志模块详解

博客转自：<https://www.cnblogs.com/CJOKER/p/8295272.html>

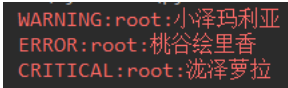
用Python写代码的时候，在想看的地方写个print xx 就能在控制台上显示打印信息，这样子就能知道它是什么了，但是当我需要看大量的地方或者在一个文件中查看的时候，这时候print就不大方便了，所以Python引入了logging模块来记录我想要的信息。

print也可以输入日志，logging相对print来说更好控制输出在哪个地方，怎么输出及控制消息级别来过滤掉那些不需要的信息。

1、日志级别

```
import logging # 引入logging模块
# 将信息打印到控制台上
logging.debug(u"苍井空")
logging.info(u"麻生希")
logging.warning(u"小泽玛利亚")
logging.error(u"桃谷绘里香")
logging.critical(u"泷泽萝拉")
```

回显：



上面可以看到只有后面三个能打印出来

默认生成的root logger的level是logging.WARNING,低于该级别的就不输出了

级别排序:CRITICAL > ERROR > WARNING > INFO > DEBUG

debug：打印全部的日志,详细的信息,通常只出现在诊断问题上

info：打印info,warning,error,critical级别的日志,确认一切按预期运行

warning：打印warning,error,critical级别的日志,一个迹象表明,一些意想不到的事情发生了,或表明一些问题在不久的将来(例如.磁盘空间低"),这个软件还能按预期工作

error：打印error,critical级别的日志,更严重的问题,软件没能执行一些功能

critical：打印critical级别,一个严重的错误,这表明程序本身可能无法继续运行

这时候，如果需要显示低于WARNING级别的内容，可以引入NOTSET级别来显示：

```
import logging # 引入logging模块
logging.basicConfig(level=logging.NOTSET) # 设置日志级别
logging.debug(u"如果设置了日志级别为NOTSET,那么这里可以采取debug、info的级别的内容也可以显示在控制台上了")
```

回显：



2、部分名词解释

Logging.Formatter：这个类配置了日志的格式，在里面自定义设置日期和时间，输出日志的时候将会按照设置的格式显示内容。

Logging.Logger：Logger是Logging模块的主体，进行以下三项工作：

- 1. 为程序提供记录日志的接口
- 2. 判断日志所处级别，并判断是否要过滤
- 3. 根据其日志级别将该条日志分发给不同handler

常用函数有：

Logger.setLevel() 设置日志级别
Logger.addHandler() 和 Logger.removeHandler() 添加和删除一个Handler
Logger.addFilter() 添加一个Filter,过滤作用

Logging.Handler：Handler基于日志级别对日志进行分发，如设置为WARNING级别的Handler只会处理WARNING及以上级别的日志。

常用函数有：

setLevel() 设置级别
setFormatter() 设置Formatter

3、日志输出-控制台

公告

昵称：咸鱼也是有梦想的
园龄：3年
粉丝：4
关注：3
+加关注

< 2020年11月				
日	一	二	三	
1	2	3	4	
8	9	10	11	
15	16	17	18	
22	23	24	25	
29	30	1	2	
6	7	8	9	

搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类

appium (2)
LoadRunner (2)
Python从入门到实践 (18)
常见问题集锦 (11)

随笔档案

2019年5月(2)
2019年4月(8)
2019年3月(8)
2019年2月(2)
2019年1月(7)
2018年12月(2)
2018年11月(5)
2018年9月(7)
2018年8月(4)
2018年7月(1)
2018年6月(6)
2018年5月(6)
2018年4月(9)
2018年3月(10)
2016年6月(1)

最新评论

```
import logging # 引入logging模块
logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s - %(filename)s[line:%(lineno)d] - %(levelname)s: %(message)s') # logging.basicConfig
# 由于日志基本配置中级别设置为DEBUG, 所以下打印信息将会全部显示在控制台上
logging.info('this is a logging info message')
logging.debug('this is a logging debug message')
logging.warning('this is logging a warning message')
logging.error('this is an logging error message')
logging.critical('this is a logging critical message')
```

上面代码通过logging.basicConfig函数进行配置了日志级别和日志内容输出格式；因为级别为DEBUG，所以会将DEBUG级别以上的信息都输出显示再控制台上。

回显：

```
2018-01-16 16:34:45,970 - widget.py[line:100] - INFO: this is a logging info message
2018-01-16 16:34:45,977 - widget.py[line:101] - DEBUG: this is a logging debug message
2018-01-16 16:34:45,977 - widget.py[line:102] - WARNING: this is logging a warning message
2018-01-16 16:34:45,977 - widget.py[line:103] - ERROR: this is an logging error message
2018-01-16 16:34:45,977 - widget.py[line:104] - CRITICAL: this is a logging critical message
```

4、日志输出文件

```
import logging # 引入logging模块
import os.path
import time
# 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
rq = time.strftime('%Y%m%d%H%M', time.localtime(time.time()))
log_path = os.path.dirname(os.getcwd()) + '/Logs/'
log_name = log_path + rq + '.log'
logfile = log_name
fh = logging.FileHandler(logfile, mode='w')
fh.setLevel(logging.DEBUG) # 输出到file的log等级的开关
# 第三步, 定义handler的输出格式
formatter = logging.Formatter("%(asctime)s - %(filename)s[line:%(lineno)d] - %(levelname)s: %(message)s")
fh.setFormatter(formatter)
# 第四步, 将logger添加到handler里面
logger.addHandler(fh)
# 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')
```

回显(打开同一目录下生成的文件)：

```
201801161658.log 2018/1/16 16:58 文本文档 1 KB
201801161658.log - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
2018-01-16 16:58:52,338 - logging_test.py[line:81] - INFO: this is a logger info message
2018-01-16 16:58:52,338 - logging_test.py[line:82] - WARNING: this is a logger warning message
2018-01-16 16:58:52,338 - logging_test.py[line:83] - ERROR: this is a logger error message
2018-01-16 16:58:52,338 - logging_test.py[line:84] - CRITICAL: this is a logger critical message
```

5、日志输出控制台和文件

只要在输入到日志中的第二步和第三步插入一个handler输出到控制台：
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
第四步和第五步分别加入以下代码即可
ch.setFormatter(formatter)
logger.addHandler(ch)

6、format常用格式说明
%(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

1. Re:python中logging日志模块博主，知识就是力量，能否借一
2. Re:python中logging日志模块博主老司机啦
3. Re:SQL基础教程（第2版）笔SQL基础教程.第2版pdf:
4. Re:python中获取文件后缀名f....(path)[1] 建议修正>> ...(path
5. Re:python中logging日志模块小泽玛利亚就很内个

阅读排行榜
1. python中logging日志模块详解
2. python中获取文件后缀名的方
3. CentOS7版本安装MariaDB出tart mariadb.service: Unit not fo
4. 使用Hyper-V安装Linux系统(
5. Jenkins地址无法访问 http://lo

评论排行榜
1. python中logging日志模块详解
2. python中获取文件后缀名的方
3. SQL基础教程（第2版）笔记

推荐排行榜
1. python中logging日志模块详解
2. python中获取文件后缀名的方
3. SQL练习题（1）(1)
4. Python2代码转换Python3脚本

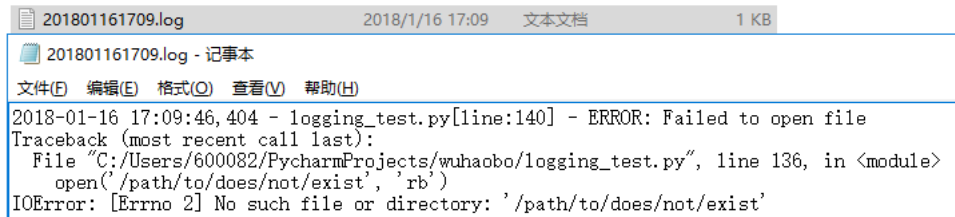
%(threadName)s: 打印线程名称
 %(process)d: 打印进程ID
 %(message)s: 打印日志信息
 7、捕捉异常,用traceback记录

```
import os.path
import time
import logging
# 创建一个logger
logger = logging.getLogger()
logger.setLevel(logging.INFO) # Log等级总开关

# 创建一个handler,用于写入日志文件
rq = time.strftime('%Y%m%d%H%M', time.localtime(time.time()))
log_path = os.path.dirname(os.getcwd()) + '/Logs/'
log_name = log_path + rq + '.log'
logfile = log_name
fh = logging.FileHandler(logfile, mode='w')
fh.setLevel(logging.DEBUG) # 输出到file的log等级的开关

# 定义handler的输出格式
formatter = logging.Formatter("%(asctime)s - %(filename)s[line:%(lineno)d] - %(levelname)s: %(message)s")
fh.setFormatter(formatter)
logger.addHandler(fh)
# 使用logger.XX来记录错误,这里的"error"可以根据所需要的级别进行修改
try:
    open('/path/to/does/not/exist', 'rb')
except (SystemExit, KeyboardInterrupt):
    raise
except Exception, e:
    logger.error('Failed to open file', exc_info=True)
```

回显(存储在文件中):



```
2018-01-16 17:09:46,404 - logging_test.py[line:140] - ERROR: Failed to open file
Traceback (most recent call last):
  File "C:/Users/600082/PycharmProjects/wuhaobo/logging_test.py", line 136, in <module>
    open('/path/to/does/not/exist', 'rb')
IOError: [Errno 2] No such file or directory: '/path/to/does/not/exist'
```

如果需要将日志不上报错误,仅记录,可以将exc_info=False,回显如下:

8、多模块调用logging,日志输出顺序

warning_output.py

```
import logging
```

```
def write_warning():
    logging.warning(u"记录文件warning_output.py的日志")
```

error_output.py

```
import logging
```

```
def write_error():
    logging.error(u"记录文件error_output.py的日志")
```

main.py

```
import logging
import warning_output
import error_output
```

```
def write_critical():
    logging.critical(u"记录文件main.py的日志")
```

```
warning_output.write_warning() # 调用warning_output文件中write_warning方法
write_critical()
error_output.write_error() # 调用error_output文件中write_error方法
```

回显：

```
WARNING:root:记录文件warning_output.py的日志
CRITICAL:root:记录文件main.py的日志
ERROR:root:记录文件error_output.py的日志
```

从上面来看，日志的输出顺序和模块执行顺序是一致的。

9、日志滚动和过期删除(按时间

```
# coding:utf-8
import logging
import time
import re
from logging.handlers import TimedRotatingFileHandler
from logging.handlers import RotatingFileHandler

def backroll():
    #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)
    #循环打印日志
    log_content = "test log"
    count = 0
    while count < 30:
        log.error(log_content)
        time.sleep(20)
        count = count + 1
    log.removeHandler(log_file_handler)

if __name__ == "__main__":
    backroll()
```

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

好文要顶

关注我

收藏该文



咸鱼也是有梦想的

关注 - 3

粉丝 - 4

5

0

+加关注

« 上一篇：[yaml.load\(\)时总是出现警告：YAMLLoadWarning: calling yaml.load\(\) without Loader=...](#)
» 下一篇：[接口自动化测试中解决所遇问题的博客链接](#)

posted @ 2019-04-08 15:54 咸鱼也是有梦想的 阅读(64173) 评论(3) 编辑 收藏

评论列表

#1楼 2020-07-23 23:35 giveyoualetter

小泽玛利亚就很内个

支持(3) 反对(0)

#2楼 2020-10-12 15:42 落雨止青

博主老司机啦

#3楼 2020-10-22 17:58 君子谦谦如玉

博主，知识就是力量，能否借一部说话

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

登录后才能发表评论，立即 [登录](#) 或 [注册](#)， [访问](#) 网站首页。

[首页](#) [新闻](#) [博问](#) [专区](#) [闪存](#) [班级](#)

- 【推荐】News: 大型组态、工控、仿真、CADGIS 50万行VC++源码免费下载
- 【推荐】博客园 & 陌上花开HIMMR 给单身的程序员小哥哥助力脱单啦~
- 【推荐】了不起的开发者，挡不住的华为，园子里的品牌专区
- 【推荐】未知数的距离，毫秒间的传递，声网与你实时互动
- 【福利】AWS携手博客园为开发者送免费套餐与抵扣券
- 【推荐】阿里云折扣价格返场，错过再等一年

相关博文：

- [Python日志处理--【logging】](#)
- [logging](#)
- [23.logging](#)
- [logging模块](#)
- [python-logging模块](#)
- » [更多推荐...](#)

最新 IT 新闻:

- [最终还是卖掉了，没了华为，荣耀还是荣耀吗？](#)
- [每秒442千万亿次计算，日本超算蝉联世界榜首之位](#)
- [补贴越大，保险越贵？买新能源车到底划不划算](#)
- [华为发布5G微波长距E-band创新解决方案](#)
- [联发科宣布8500万美元并购Intel旗下电源管理芯片业务](#)
- » [更多新闻...](#)