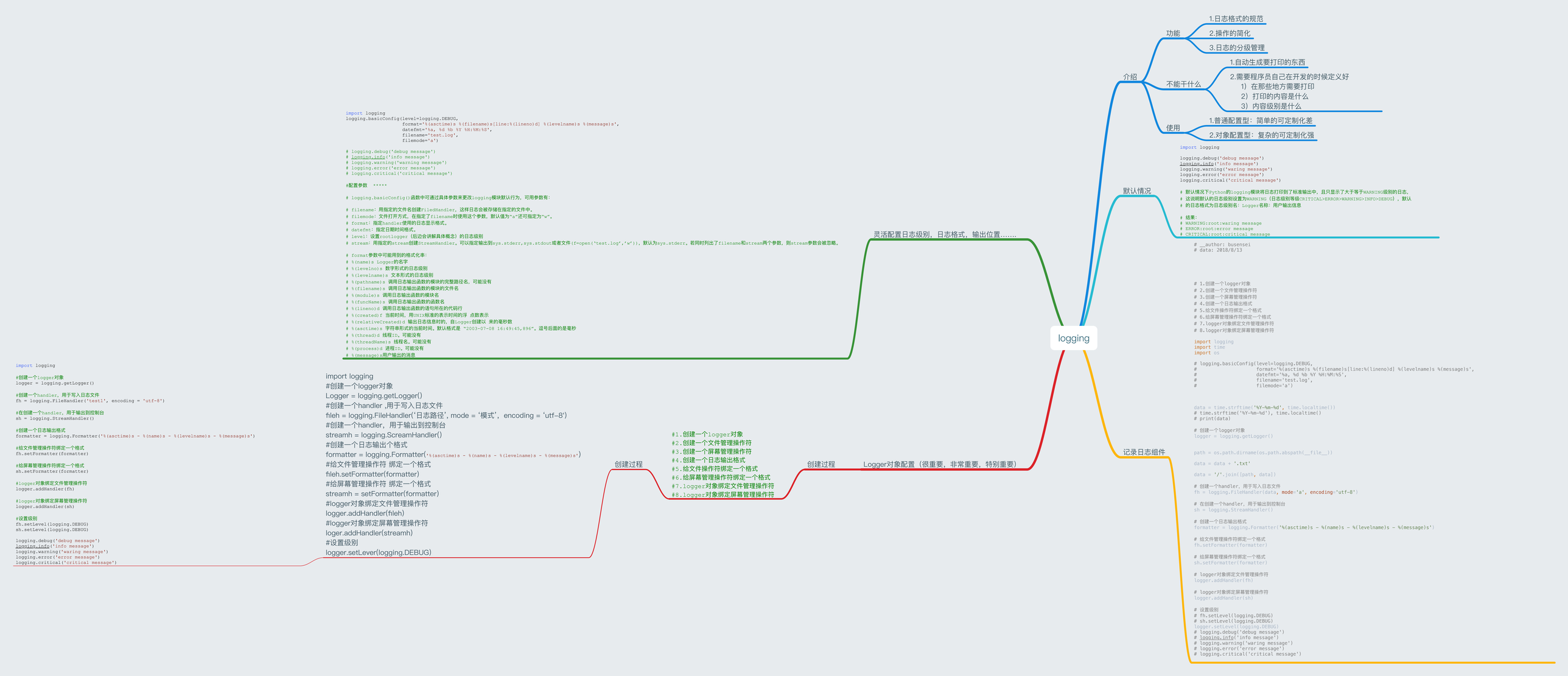
logging
介绍
功能
1.日志格式的规范
2.操作的简化
3.日志的分级管理
不能干什么
1.自动生成要打印的东西
2.需要程序员自己在开发的时候定义好
1）在那些地方需要打印
2）打印的内容是什么
3）内容级别是什么
使用
1.普通配置型：简单的可定制化差
2.对象配置型：复杂的可定制化强
默认情况
import logging
logging.debug('debug message')
logging.info('info message')
logging.warning('waring message')
logging.error('error message')
logging.critical('critical message')
# 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），默认
# 的日志格式为日志级别名：Logger名称：用户输出信息
# 结果:
# WARNING:root:waring message
# ERROR:root:error message
# CRITICAL:root:critical message
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='test.log',
filemode='a')
# logging.debug('debug message')
# logging.info('info message')
# logging.warning('warning message')
# logging.error('error message')
# logging.critical('critical message')
#配置参数 *****
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，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(f=open('test.log','w'))，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
# __author: busensei
# data: 2018/8/13
# 1,创建一个logger对象
# 2,创建一个文件管理操作符
# 3,创建一个屏幕管理操作符
# 4,创建一个日志输出格式
# 5,给文件操作符绑定一个格式
# 6,给屏幕管理操作符绑定一个格式
# 7,logger对象绑定文件管理操作符
# 8,logger对象绑定屏幕管理操作符
import logging
import time
import os
# 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='test.log',
# filemode='a')
data = time.strftime('%Y-%m-%d', time.localtime())
# time.strftime('%Y-%m-%d'), time.localtime()
# print(data)
# 创建一个logger对象
logger = logging.getLogger()
path = os.path.dirname(os.path.abspath(__file__))
data = data + '.txt'
data = '/'.join([path, data])
# 创建一个handler，用于写入日志文件
fh = logging.FileHandler(data, mode='a', encoding='utf-8')
# 在创建一个handler，用于输出到控制台
sh = logging.StreamHandler()
# 创建一个日志输出格式
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
# 给文件管理操作符绑定一个格式
fh.setFormatter(formatter)
# 给屏幕管理操作符绑定一个格式
sh.setFormatter(formatter)
# logger对象绑定文件管理操作符
logger.addHandler(fh)
# logger对象绑定屏幕管理操作符
logger.addHandler(sh)
# 设置级别
# fh.setLevel(logging.DEBUG)
# sh.setLevel(logging.DEBUG)
logger.setLevel(logging.DEBUG)
# logging.debug('debug message')
# logging.info('info message')
# logging.warning('waring message')
# logging.error('error message')
# logging.critical('critical message')
记录日志组件
Logger对象配置（很重要，非常重要，特别重要）
创建过程
#1.创建一个logger对象
#2.创建一个文件管理操作符
#3.创建一个屏幕管理操作符
#4.创建一个日志输出格式
#5.给文件操作符绑定一个格式
#6.给屏幕管理操作符绑定一个格式
#7.logger对象绑定文件管理操作符
#8.logger对象绑定屏幕管理操作符
创建过程
import logging
#创建一个logger对象
Logger = logging.getLogger()
#创建一个handler ,用于写入日志文件
fileh = logging.FileHandler('日志路径', mode = '模式', encoding = 'utf-8')
#创建一个handler，用于输出到控制台
streamh = logging.ScreamHandler()
#创建一个日志输出个格式
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
#给文件管理操作符 绑定一个格式
fileh.setFormatter(formatter)
#给屏幕管理操作符 绑定一个格式
streamh = setFormatter(formatter)
#logger对象绑定文件管理操作符
logger.addHandler(fileh)
#logger对象绑定屏幕管理操作符
loger.addHandler(streamh)
#设置级别
logger.setLever(logging.DEBUG)
import logging
#创建一个logger对象
logger = logging.getLogger()
#创建一个handler，用于写入日志文件
fh = logging.FileHandler('test1', encoding = 'utf-8')
#在创建一个handler，用于输出到控制台
sh = logging.StreamHandler()
#创建一个日志输出格式
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
#给文件管理操作符绑定一个格式
fh.setFormatter(formatter)
#给屏幕管理操作符绑定一个格式
sh.setFormatter(formatter)
#logger对象绑定文件管理操作符
logger.addHandler(fh)
#logger对象绑定屏幕管理操作符
logger.addHandler(sh)
#设置级别
fh.setLevel(logging.DEBUG)
sh.setLevel(logging.DEBUG)
logging.debug('debug message')
logging.info('info message')
logging.warning('waring message')
logging.error('error message')
logging.critical('critical message')