

ХОЧУ ПОМОЧЬ  
ПРОЕКТУ

Пакет logging, ведение журнала в Python



✓ mrqz.me

РЕКЛАМА

### Бесплатная стратегия продвижения от сервиса Rookee

Простота подключения • Одни из лидеров рейтингов • Бесплатный аудит SEO

Получить предложение

Статья

— Алиса, позвони на кухню

Узнать больше

Реклама, ООО «Лаборатория Алисы», ИНН 5005173079 0+

/ Пакет logging, ведение журнала в Python

Типы событий для приложений

Функции и классы, которые реализуют гибкую систему регистраций событий для приложений и

Ведение журнала отслеживания событий, которые происходят при запуске какого-либо программного обеспечения добавляет в свой код протоколирование вызовов, которые указывают на определенные события. Событие описывается сообщением, которое может содержать переменные и данные, которые потенциально различны для каждого случая события. Важность также можно назвать уровнем или серьезностью.

Преимущество API ведения журнала, предоставляемого этим пакетом, заключается в том, что в ведении журнала могут участвовать все модули, следовательно журнал приложения может включать собственные сообщения, интегрированные с сообщениями от сторонних модулей.

Если нужно срочно добавить ведение журнала логов в какой-то существующий проект или стоит цель - просто начать работу с простыми и разумными настройками, то обратите внимание на продуманную "обертку" модуля logging - [log2d](#).

Также обратите внимание на [модуль loguru](#), который предназначен для того, чтобы сделать ведение логов менее болезненным и плюс ко всему добавляет ряд полезных функций. Модуль loguru полностью совместим со [стандартным пакетом logging](#).

[Функции ведения журнала](#) именуются в соответствии с уровнем логирования или серьезностью событий, которые они используют для отслеживания.

## Стандартные уровни логирования пакета logging.

Стандартные уровни и их применимость описаны в порядке возрастания серьезности:

Уровень	Числовое значение	Когда используется
logging.NOTSET	0	Сообщения отключены.
logging.DEBUG	10	Подробная информация, как правило, интересна только при диагностике проблем.
logging.INFO	20	Подтверждение того, что все работает как положено
logging.WARNING	30	Указание на то, что произошло что-то неожиданное или указание на проблему в ближайшем будущем, например недостаточно места на диске. Программное обеспечение все еще работает как ожидалось.
logging.ERROR	40	Из-за более серьезной проблемы программное обеспечение не смогло выполнить какую-либо функцию.
logging.CRITICAL	50	Серьезная ошибка, указывающая на то, что сама программа не может продолжить работу.

Уровень по умолчанию - WARNING, это означает, что будут отслеживаться только события этого уровня и выше, если [пакет logging](#) не настроен на обратное.

Самый простой способ обработки отслеживаемых событий - вывести сообщения на консоль.

### Примеры:

Самое простое использование с выводом сообщений на консоль.

Вверх

```
>>> import logging
>>> logging.debug('debug message')
>>> logging.info('info message')
>>> logging.warning('warning message')
# <module 'logging' from '/usr/lib/python3.7/logging/__init__.py'>
>>> logging.error('error message')
# <module 'logging' from '/usr/lib/python3.7/logging/__init__.py'>
>>> logging.critical('critical message')
# <module 'logging' from '/usr/lib/python3.7/logging/__init__.py'>
```

Сообщения не выводятся потому, что уровень регистрации событий по умолчанию - WARNING. Сообщение не содержит ни уровня, ни описания события, приведенного в журнале вызовов.

Для вывода сообщений необходимо в настройки регистратора передать уровень выводимых ошибок.

Обратите внимание, что функции `logging.basicConfig()` должны передаваться до первого вызова функций ведения журнала. Если интерпретатор уже была вызвана хотя бы одна функция, то необходимо [перезагрузить пакет logging](#) в консоль интерпретатора. С версии Python-3.8 для этой цели можно использовать `importlib.reload(logging)`.

```
# <module 'logging' from '/usr/lib/python3.7/logging/__init__.py'>
>>> import importlib
>>> importlib.reload(logging)
# <module 'logging' from '/usr/lib/python3.7/logging/__init__.py'>
>>> logging.basicConfig(level=logging.DEBUG)
>>> logging.debug('debug message')
# DEBUG:root:debug message
>>> logging.info('info message')
# INFO:root:info message
>>> logging.warning('warning message')
# WARNING:root:warning message
...

```

Так же в конфигурации регистратора `logging.basicConfig()` можно задать любое форматирование выводимой строки с отслеживаемым событием.

```
# перезагружаем модуль 'logging'
>>> import importlib
>>> logging.basicConfig(
...     level=logging.DEBUG, force=True
...     format='[%asctime)s] [%levelname)s] %(message)s',
...     datefmt='%Y-%m-%d %H:%M:%S')
>>> logging.debug('debug message')
# [2020-06-07 15:37:15] [DEBUG] debug message
>>> logging.info('info message')
# [2020-06-07 15:37:23] [INFO] info message
>>> logging.warning('warning message')
# [2020-06-07 15:37:27] [WARNING] warning message
...

```

Содержание раздела:

Вверх

- [КРАТКИЙ ОБЗОР МАТЕРИАЛА.](#)
- [Простое использование модуля logging](#)
- [Продвинутое использование модуля logging](#)
- [Принцип работы пакета logging](#)
- [Функция getLogger\(\) модуля logging](#)
- [Функции регистрации сообщений модуля logging](#)
- [Функция exception\(\) модуля logging](#)
- [Функция log\(\) модуля logging](#)
- [Функция disable\(\) модуля logging](#)
- [Функция addLevelName\(\) модуля logging](#)

- РЕКЛАМА

**Я Станция**

— Алиса, позвони  
на кухню

Узнать больше

Реклама, ООО «Лаборатория Алиса»,  
ИНН 9705173070

0+

[@docs\\_python\\_ru](#)

3/3