logzero Documentation

Release 1.5.0

Chris Hager

Mar 07, 2018

Contents

Robust and effective logging for Python 2 and 3.

Features

- Easy logging to console and/or (rotating) file.
- Provides a fully configured standard Python logger object.
- Pretty formatting, including level-specific colors in the console.
- Windows color output supported by colorama
- Robust against str/bytes encoding problems, works with all kinds of character encodings and special characters.
- Multiple loggers can write to the same logfile (also works across multiple Python files).
- Global default logger with *logzero.logger* and custom loggers with *logzero.setup_logger(..)*.
- Compatible with Python 2 and 3.
- All contained in a single file.
- Licensed under the MIT license.
- Heavily inspired by the Tornado web framework.
- Hosted on GitHub: https://github.com/metachris/logzero

```
$ python demo.py
[D 170628 09:30:53 demo:4] hello
[I 170628 09:30:53 demo:5] info
[W 170628 09:30:53 demo:6] warn
[E 170628 09:30:53 demo:7] error
```

CHAPTER 1

Installation

Install *logzero* with pip:

```
$ pip install -U logzero
```

If you don't have pip installed, this Python installation guide can guide you through the process.

You can also install *logzero* from the public Github repo:

```
$ git clone https://github.com/metachris/logzero.git
$ cd logzero
$ python setup.py install
```

On openSUSE you can install the current version from repos: python2-logzero, python3-logzero. In the newest openSUSE release you can install it with zypper: `sudo zypper in python2-logzero`.

CHAPTER 2

Example Usage

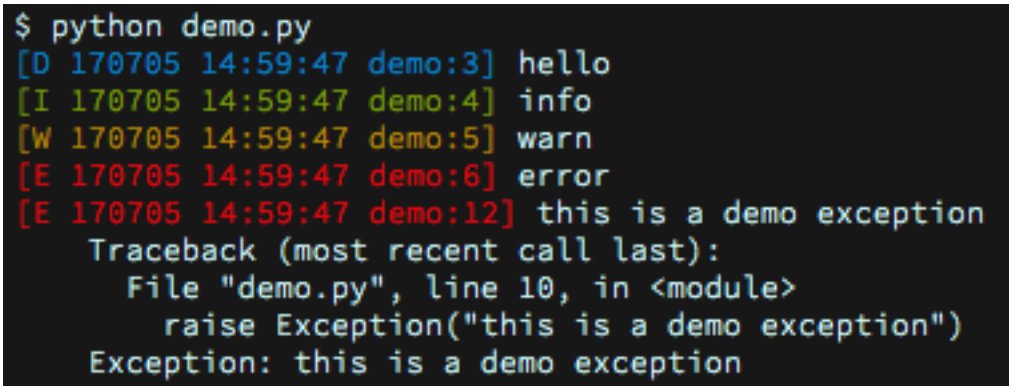
You can use *logzero* like this (logs only to the console by default):

```
from logzero import logger

# These log messages are sent to the console
logger.debug("hello")
logger.info("info")
logger.warning("warning")
logger.error("error")

# This is how you'd log an exception
try:
    raise Exception("this is a demo exception")
except Exception as e:
    logger.exception(e)
```

If this was a file called `demo.py`, the output will look like this:




```
[D 170705 14:59:47 demo:3] hello
[I 170705 14:59:47 demo:4] info
[W 170705 14:59:47 demo:5] warn
```

```
[E 170705 14:59:47 demo:6] error
[E 170705 14:59:47 demo:12] this is a demo exception
    Traceback (most recent call last):
    File "demo.py", line 10, in <module>
        raise Exception("this is a demo exception")
    Exception: this is a demo exception
```

2.1 Rotating Logfile

Adding a rotating logfile is that easy:

```
import logzero
from logzero import logger

# Setup rotating logfile with 3 rotations, each with a maximum filesize of 1MB:
logzero.logfile("/tmp/rotating-logfile.log", maxBytes=1e6, backupCount=3)

# Log messages
logger.info("This log message goes to the console and the logfile")
```

2.2 Advanced Usage Examples

Here are more examples which show how to use logfiles, custom formatters and setting a minimum loglevel.

Outcome	Method
Set a minimum log level	*logzero.loglevel(..)*
Add logging to a logfile	*logzero.logfile(..)*
Setup a rotating logfile	*logzero.logfile(..)*
Disable logging to a logfile	*logzero.logfile(None)*
Log to syslog	*logzero.syslog(...)*
Use a custom formatter	*logzero.formatter(..)*

```
import logging
import logzero
from logzero import logger

# This log message goes to the console
logger.debug("hello")

# Set a minimum log level
logzero.loglevel(logging.INFO)

# Set a logfile (all future log messages are also saved there)
logzero.logfile("/tmp/logfile.log")

# Set a logfile (all future log messages are also saved there), but disable the␣
↪default stderr logging
logzero.logfile("/tmp/logfile.log", disableStderrLogger=True)

# You can also set a different loglevel for the file handler
logzero.logfile("/tmp/logfile.log", loglevel=logging.ERROR)
```

```
# Set a rotating logfile (replaces the previous logfile handler)
logzero.logfile("/tmp/rotating-logfile.log", maxBytes=1000000, backupCount=3)

# Disable logging to a file
logzero.logfile(None)

# Log to syslog, using default logzero logger and 'user' syslog facility
logzero.syslog()

# Log to syslog, using default logzero logger and 'local0' syslog facility
logzero.syslog(facility=SysLogHandler.LOG_LOCAL0)

# Set a custom formatter
formatter = logging.Formatter('%(name)s - %(asctime)-15s - %(levelname)s: %(message)s
↪');
logzero.formatter(formatter)

# Log some variables
logger.info("var1: %s, var2: %s", var1, var2)
```

2.3 Custom Logger Instances

Instead of using the default logger you can also setup specific logger instances with *logzero.setup_logger(..)*:

```
from logzero import setup_logger
logger1 = setup_logger(name="mylogger1", logfile="/tmp/test-logger1.log",␣
↪level=logging.INFO)
logger2 = setup_logger(name="mylogger2", logfile="/tmp/test-logger2.log",␣
↪level=logging.INFO)
logger3 = setup_logger(name="mylogger3", logfile="/tmp/test-logger3.log",␣
↪level=logging.INFO, disableStderrLogger=True)

# By default, logging
logger1.info("info for logger 1")
logger2.info("info for logger 2")

# log to a file only, excluding the default stderr logger
logger3.info("info for logger 3")
```

2.4 Adding custom handlers (eg. SocketHandler)

Since *logzero* uses the standard Python logger object, you can attach any Python logging handlers you can imagine!

This is how you add a SocketHandler:

```
import logzero
import logging
from logging.handlers import SocketHandler

# Setup the SocketHandler
socket_handler = SocketHandler(address=('localhost', logging.DEFAULT_TCP_LOGGING_
↪PORT))
```

```
socket_handler.setLevel(logging.DEBUG)
socket_handler.setFormatter(logzero.LogFormatter(color=False))

# Attach it to the logzero default logger
logzero.logger.addHandler(socket_handler)

# Log messages
logzero.logger.info("this is a test")
```

CHAPTER 3

Documentation

3.1 *logzero.logger*

logzero.logger is an already set up standard Python logger instance for your convenience. You can use it from all your files and modules directly like this:

```
from logzero import logger

logger.debug("hello")
logger.info("info")
logger.warning("warning")
logger.error("error")
```

You can reconfigure the default logger globally with *logzero.setup_default_logger(..)*.

See the documentation for the Python logger instance for more information about how you can use it.

3.2 *logzero.loglevel(..)*

logzero.**loglevel**(*level=10*, *update_custom_handlers=False*)

Set the minimum loglevel for the default logger (*logzero.logger*).

This reconfigures only the internal handlers of the default logger (eg. stream and logfile). You can also update the loglevel for custom handlers by using *update_custom_handlers=True*.

Parameters

- **level** (*int*) – Minimum logging-level to display (default: *logging.DEBUG*).
- **update_custom_handlers** (*bool*) – If you added custom handlers to this logger and want this to update them too, you need to set *update_custom_handlers* to *True*

3.3 *logzero.logfile(..)*

logzero.**logfile**(*filename*, *formatter=None*, *mode='a'*, *maxBytes=0*, *backupCount=0*, *encoding=None*, *loglevel=None*, *disableStderrLogger=False*)

Setup logging to file (using a RotatingFileHandler internally).

By default, the file grows indefinitely (no rotation). You can use the `maxBytes` and `backupCount` values to allow the file to rollover at a predetermined size. When the size is about to be exceeded, the file is closed and a new file is silently opened for output. Rollover occurs whenever the current log file is nearly `maxBytes` in length; if either of `maxBytes` or `backupCount` is zero, rollover never occurs.

If `backupCount` is non-zero, the system will save old log files by appending the extensions '.1', '.2' etc., to the filename. For example, with a `backupCount` of 5 and a base file name of app.log, you would get app.log, app.log.1, app.log.2, up to app.log.5. The file being written to is always app.log. When this file is filled, it is closed and renamed to app.log.1, and if files app.log.1, app.log.2, etc. exist, then they are renamed to app.log.2, app.log.3 etc. respectively.

Parameters

- **filename** (*string*) – Filename of the logfile. Set to *None* to disable logging to the logfile.
- **formatter** (*Formatter*) – Python logging Formatter object (by default uses the internal LogFormatter).
- **mode** (*string*) – mode to open the file with. Defaults to `a`
- **maxBytes** (*int*) – Size of the logfile when rollover should occur. Defaults to 0, rollover never occurs.
- **backupCount** (*int*) – Number of backups to keep. Defaults to 0, rollover never occurs.
- **encoding** (*string*) – Used to open the file with that encoding.
- **loglevel** (*int*) – Set a custom loglevel for the file logger, else uses the current global loglevel.
- **disableStderrLogger** (*bool*) – Should the default stderr logger be disabled. Defaults to False.

3.4 *logzero.formatter(..)*

logzero.**formatter**(*formatter*, *update_custom_handlers=False*)

Set the formatter for all handlers of the default logger (`logzero.logger`).

This reconfigures only the logzero internal handlers by default, but you can also reconfigure custom handlers by using `update_custom_handlers=True`.

Beware that setting a formatter which uses colors also may write the color codes to logfiles.

Parameters

- **formatter** (*Formatter*) – Python logging Formatter object (by default uses the internal LogFormatter).
- **update_custom_handlers** (*bool*) – If you added custom handlers to this logger and want this to update them too, you need to set `update_custom_handlers` to *True*

3.5 *logzero.setup_logger(..)*

logzero.**setup_logger**(*name=None*, *logfile=None*, *level=10*, *formatter=None*, *maxBytes=0*, *backupCount=0*, *fileLoglevel=None*, *disableStderrLogger=False*)

Configures and returns a fully configured logger instance, no hassles. If a logger with the specified name already exists, it returns the existing instance, else creates a new one.

If you set the `logfile` parameter with a filename, the logger will save the messages to the logfile, but does not rotate by default. If you want to enable log rotation, set both `maxBytes` and `backupCount`.

Usage:

```
from logzero import setup_logger
logger = setup_logger()
logger.info("hello")
```

Parameters

- **name** (*string*) – Name of the Logger object. Multiple calls to `setup_logger()` with the same name will always return a reference to the same Logger object. (default: `__name__`)
- **logfile** (*string*) – If set, also write logs to the specified filename.
- **level** (*int*) – Minimum logging-level to display (default: `logging.DEBUG`).
- **formatter** (*Formatter*) – Python logging Formatter object (by default uses the internal LogFormatter).
- **maxBytes** (*int*) – Size of the logfile when rollover should occur. Defaults to 0, rollover never occurs.
- **backupCount** (*int*) – Number of backups to keep. Defaults to 0, rollover never occurs.
- **fileLoglevel** (*int*) – Minimum logging-level for the file logger (is not set, it will use the loglevel from the `level` argument)
- **disableStderrLogger** (*bool*) – Should the default stderr logger be disabled. Defaults to False.

Returns

A fully configured Python logging Logger object you can use with `.debug("msg")`, etc.

3.6 Default Log Format

This is the default log format string:

```
DEFAULT_FORMAT = '%(color)s[%(levelname)1.1s %(asctime)s %(module)s:%(lineno)d]%(end_
↪color)s %(message)s'
```

See also the Python LogRecord attributes you can use.

3.7 Custom Formatting

It is easy to use a custom formatter / a custom log format string:

- Define your log format string (you can use any of the LogRecord attributes).
- Create a Formatter object (based on *logzero.LogFormatter* to get all the encoding helpers).
- Supply the formatter object to the *formatter* argument in the *setup_logger(..)* method.

This is a working example on how to setup logging with a custom format:

```
import logzero

log_format = '%(color)s[%(levelname)1.1s %(asctime)s %(module)s:%(lineno)d]%(end_
↪color)s %(message)s'
formatter = logzero.LogFormatter(fmt=log_format)
logzero.setup_default_logger(formatter=formatter)
```

CHAPTER 4

Issues, Feedback & Contributions

All kinds of feedback and contributions are welcome.

- Create an issue
- Create a pull request
- https://github.com/metachris/logzero
- chris@linuxuser.at // @metachris

Index