# logpp Documentation

*Release 0.0.8*

**Pat Daburu**

**Jun 02, 2019**

# CONTENTS:

python logging extensions

CHAPTER

ONE

# MANUAL

## 1.1 Why *logpp*?

*logpp* is a fairly simple module that contains some extensions for Python's built in logging module. It provides a few facilities that allow you to pass extended information with logging messages.

The three principle components are listed below.

- *logpp.logging.msg()*
- *logpp.logging.LogppMessage*
- *logpp.logging.LogppHandler*

The module also provides the *logpp.logging.LogppMixin* which you can use to provide standardized access to a logger via the *logpp.logging.LogppMixin.logger* method.

## 1.2 Logging a Message

The example below has been expanded to make the components easier to see, but it's actually a fairly simple one-liner. The *logpp.logging.msg()* function takes a summary *str* and a detail object (which in the example is just a dictionary).

The function returns a *logpp.logging.LogppMessage* which, when represented in *str* form is simply the summary.

```
logging.info(
    msg(
        'The weather is currently sunny with a temperature of 25°C.',
        {
            'conditions': 'sunny',
            'temperature': 25
        }
    )
)
```

Logging handlers that aren't aware of the detail information should simply see the *logpp* message as the summary.

## 1.3 Handling Messages

If you're using *logpp*, chances are you want to do something useful or clever with the detail information. To accomplish that you can create your own logging handler. If your custom handler is only interested in *logpp* messages,

you can extend the *logpp.logging.LogppHandler* and override the *logpp.logging.LogppHandler.emit_logpp()* method. The base class will perform checks to make sure that only logging messages that are instances of the *logpp.logging.LogppMessage* class are passed to this method.

## 1.4 Putting It All Together

The sample below briefly demonstrates the creation of a custom log handler and should give you an idea of what to expect from such a facility.

```
import logging
from logpp import msg, LogppMessage, LogppHandler


# Create a custom handler.
class CustomLogppHandler(LogppHandler):

    def emit_logpp(self, msg_: LogppMessage):
        print(f'SUMMARY: {msg_.summary}')
        print(f'DETAILS: {msg_.detail}')


logging.basicConfig(level=logging.INFO)
# Add the custom handler to the logger (just as you would with any handler).
logging.getLogger().addHandler(CustomLogppHandler())

# Log a message to be handled by the custom handler.
logging.info(
    msg(
        'The weather is currently sunny with a temperature of 25°C.',
        {
            'conditions': 'sunny',
            'temperature': 25
        }
    )
)

# Log a message that will be ignored by the custom handler.
logging.info('This message will be ignored by the custom handler.')
```

## 1.5 Using the `logpp.logging.LogppMixin`

Let's say you have a class that needs to log its activities. Often you'll want to use a named logger. This can involve a few lines of boiler plate which can be a bit tedious to produce in every class. By extending the *logpp.logging.LogppMixin* your class gains the *logpp.logging.LogppMixin.logger()* function which returns a logger with a name that reflects the name of the class (though you can override that behavior by adding a *__loggername__* attribute to the class).

```
import logging
from logpp import LogppMixin


# Just so we may demonstrate the use of the mixin, here's a base class
# that has nothing to do with logging from which we can inherit.
```

(continues on next page)

(continued from previous page)

```
class SampleBaseClass(object):
    pass


# Now let's create a class that extends the sample base class, but
# which also mixes in the logging facility.
class LoggableClass(SampleBaseClass, LogppMixin):

    def log_something(self):
        self.logger().info('Hello world!')


# Set up basic logging
logging.basicConfig(level=logging.INFO)

# Create a new instance of the mixed-in class...
loggable = LoggableClass()
# ...and ask it to log something.
loggable.log_something()
```

CHAPTER

TWO

# API DOCUMENTATION

python logging extensions

This module contains the basic logging extensions.

**class** logpp.logging.**LogppHandler**(*level=0*)
Bases: logging.Handler, abc.ABC

Extend this class to create handlers specific to *LogppMessage* messages.

**emit**(*record: logging.LogRecord*)
This is the standard logging handler method that will filter out any messages that aren't *LogppMessage* instances. When you extend this type of handler, override the *LogppHandler.emit_logpp()* method.

**Parameters** **record** – the logging record

**emit_logpp**(*msg_: logpp.logging.LogppMessage*)
Override this method to handle *LogppMessage* messages when they are logged.

**Parameters** **msg** – the logpp logging message

**class** logpp.logging.**LogppMessage**
Bases: tuple

This is a logging record, suitable to pass on to a logger as the primary logging message.

**detail**
the message detail object

**summary**
the message summary

**class** logpp.logging.**LogppMixin**
Bases: object

This is a mixin that provides standard access to a logger via the *LogppMixin.logger()* function.

The name of the logger reflect's the class name, though you can override that by providing your class with a *__loggername__* attribute.

**classmethod logger**()

logpp.logging.**msg**(*summary: str*, *detail: Any*) → logpp.logging.LogppMessage
Create a logging record.

**Parameters**

- **summary** – the principal summary of the logging event
- **detail** – the message detail data object

**Returns** a logging record

CHAPTER

THREE

# PYTHON MODULE DEPENDENCIES

The `requirements.txt` file contains this project's module dependencies. You can install these dependencies using `pip`.

```
pip install -r requirements.txt
```

## 3.1 requirements.txt

```
click>=6.7,<7
parameterized>=0.6.1,<1
pip-check-reqs==2.0.1
pylint>=1.8.4,<2
pytest>=3.4.0,<4
pytest-cov>=2.5.1,<3
pytest-pythonpath>=0.7.2,<1
setuptools>=38.4.0
Sphinx>=1.7.1,<2
sphinx-rtd-theme>=0.2.4,<1
tox>=3.0.0,<4
twine>=1.11.0,<2
```

CHAPTER

FOUR

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## l

# INDEX