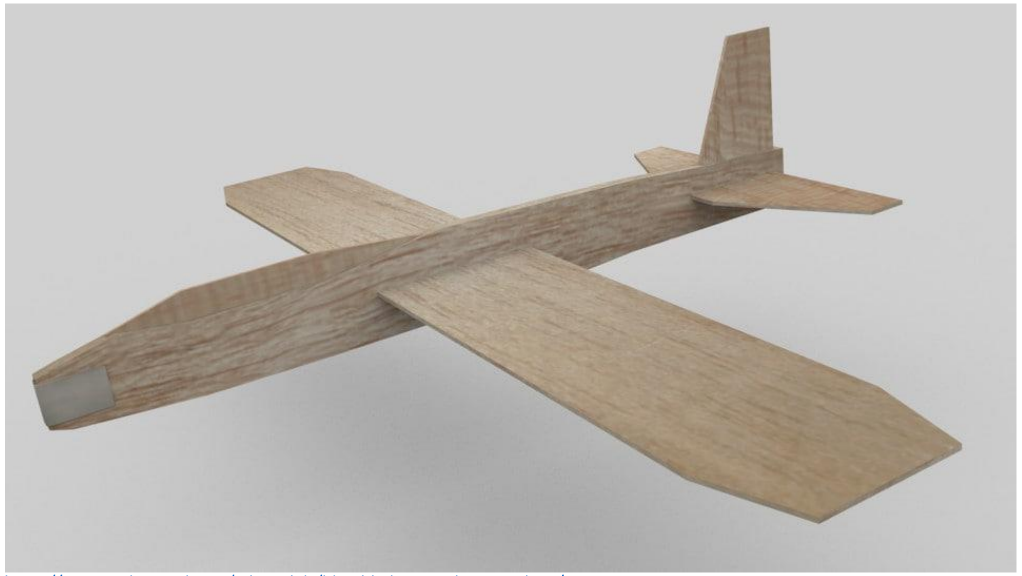
Balsa

Lightweight Python Logging


https://www.turbosquid.com/3d-models/blend-balsa-wood-toy-airplane/608014

James Abel

Dec 18, 2018

j@abel.co

www.abel.co   @jamesabel   www.github.com/jamesabel/balsa   Pyninsula Dec 2018

The `logging` module

- The `logging` module is awesome!
  - Handlers – stream (console), files, sockets, HTTP, custom, … many more!
  - Filters
  - Formatters
  - Hierarchal
  - Log Levels – debug, info, warning, error, critical
- `logging.getLogger(name)` provides the logger associated with `name`
  - Can directly access a logger from anywhere in your program with just the `name` string
- https://docs.python.org/library/logging.html

`logging` levels

Level	When it's used
DEBUG	Detailed information, typically of interest only when diagnosing problems.
INFO	Confirmation that things are working as expected.
WARNING	An indication that something unexpected happened, or indicative of some problem in the near future (e.g. 'disk space low'). The software is still working as expected.
ERROR	Due to a more serious problem, the software has not been able to perform some function.
CRITICAL	A serious error, indicating that the program itself may be unable to continue running.

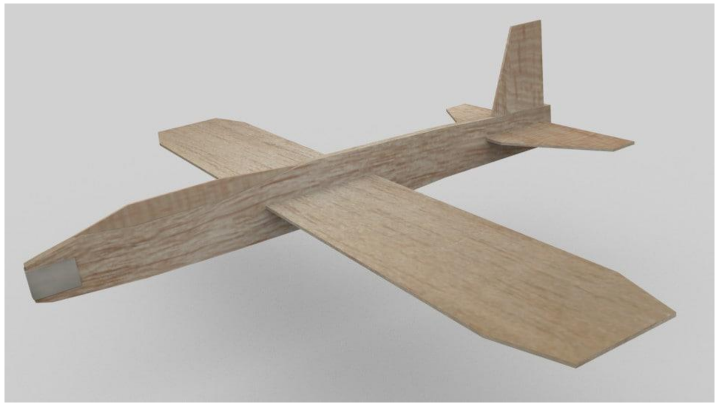
https://docs.python.org/howto/logging.html

However, logging options and configurations can get rather involved for relatively simple apps

- Log message format
- Handers
- Where to write log files?
- Log levels for each handler
- CLI vs. GUI
- Tracebacks
- Exception Services

Setting up `logging` Can Be A Significant Amount of Code and Complexity
Use Balsa to avoid writing the same code over and over

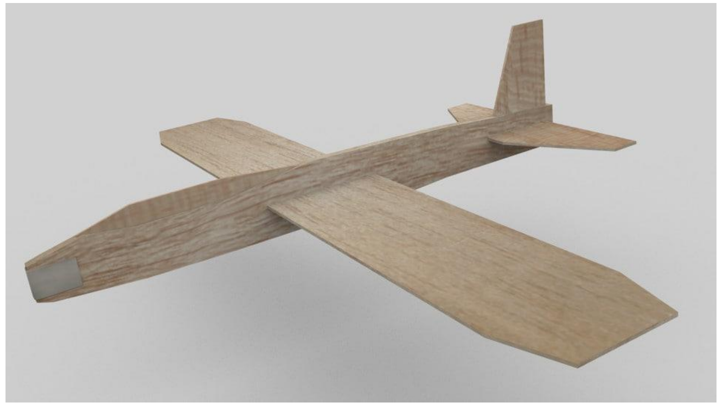
Balsa – Lightweight Logging



- Provide useful logging with just a few lines of code
- Consistent formatting and interface
  - `appdirs` for log file directory
- Console, GUI, files, exception services built-in
  - tkinter dialog box
  - Sentry (raven)
- Verbosity expressed by intent rather than explicit level (e.g. `verbose` for development)
- `Error` level callback
- Available on PyPI

```
pip install balsa
```

Simple Example

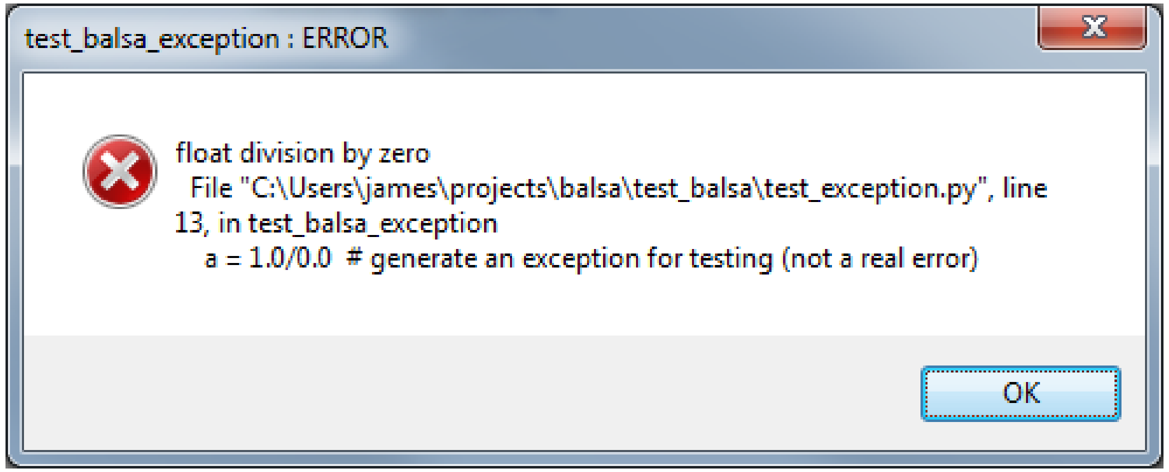


```
from balsa import get_logger, Balsa

application_name = 'example'
log = get_logger(application_name)

def main():
    balsa = Balsa(name=application_name , author='james abel')
    balsa.init_logger()

    log.error('my error example')
```

```
                         app name                              function name            message
                             ↓                                       ↓                     ↓
2018-08-18 20:43:33,756 - example - balsa_simple_example.py - 12 - main - ERROR - my error example
             ↑                                ↑                ↑            ↑
         timestamp                    source file name    line number    level
```

Also writes out a file (e.g. Windows):
`C:\Users\<user>\AppData\Local\james abel\example\Logs\example.log`

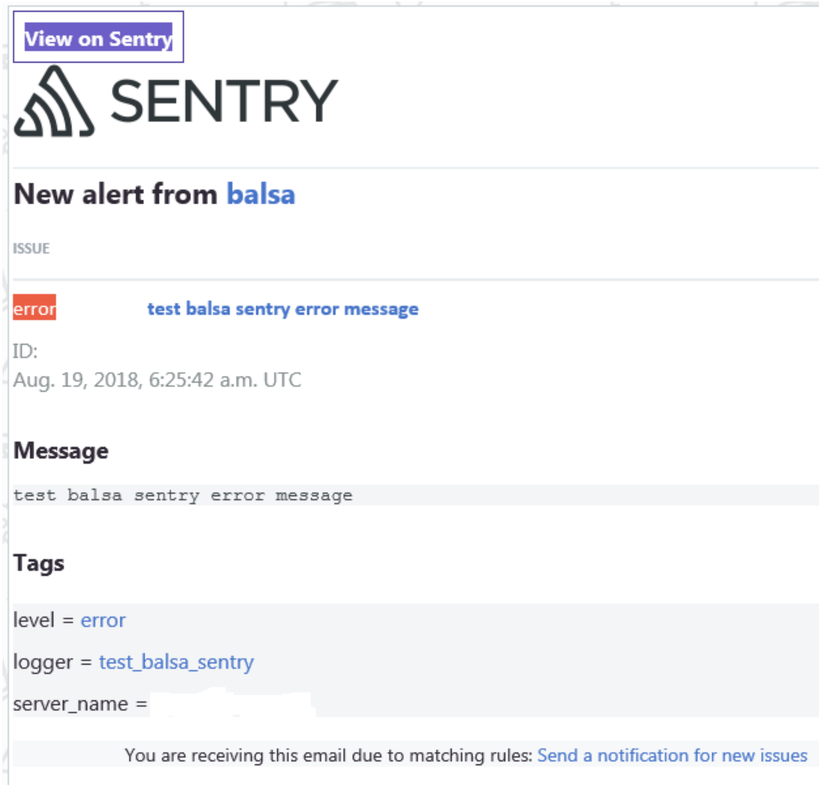
- GUI messages



test_balsa_exception : ERROR

```
float division by zero
  File "C:\Users\james\projects\balsa\test_balsa\test_exception.py", line
13, in test_balsa_exception
    a = 1.0/0.0  # generate an exception for testing (not a real error)
```

- Sentry support (exception service)



New alert from balsa

ISSUE

error test balsa sentry error message

ID:
Aug. 19, 2018, 6:25:42 a.m. UTC

Message

```
test balsa sentry error message
```

Tags

level = error

logger = test_balsa_sentry

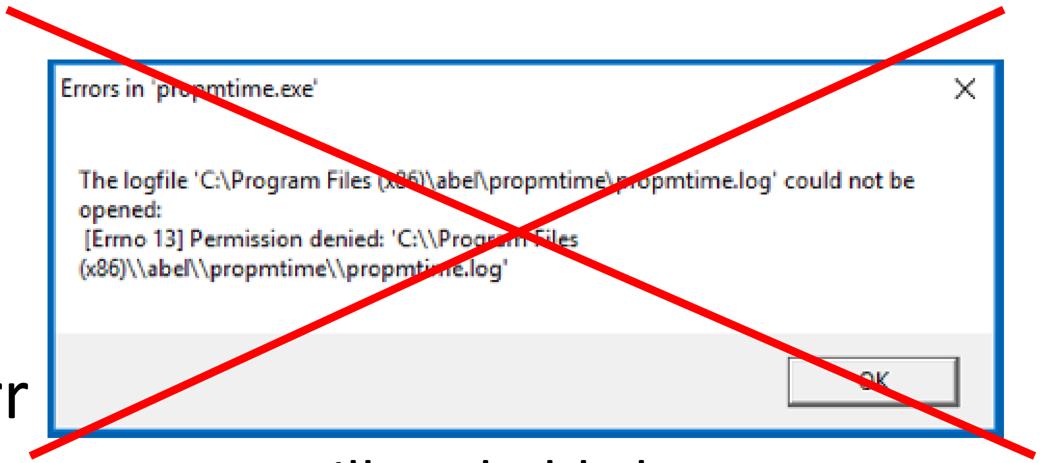
server_name =

You are receiving this email due to matching rules: Send a notification for new issues

Verbosity

- default Levels
  - `info`: file, internal string buffer
  - `warning` : stdout or GUI
  - `error` : cloud services, error callback
- `verbose=True`
  - `debug` : file
  - `info` : stdout or GUI, internal string buffer
  - `error` : cloud services, error callback

Balsa handles GUI apps



```
Errors in 'propmtime.exe'

The logfile 'C:\Program Files (x86)\abel\propmtime\propmtime.log' could not be
opened:
 [Errno 13] Permission denied: 'C:\\Program Files
(x86)\\abel\\propmtime\\propmtime.log'
```

- A GUI app must not write to stdout or stderr
  - e.g. causes an error popup in Windows and your program will probably be forcefully killed
  - Messages will be lost anyway
- Balsa handles GUI aspects when `gui=True`
- Uses tkinter (which is generally built-in to Python) or PyQt5
- Use the Error callback to gracefully handle the issue
  - e.g. for Exceptions gracefully exit

Catch-All `try/except`

```
try:
    main()  # your program
except Exception as e:
    # catch all exceptions
    log.error(e, exc_info=True, stack_info=True)
```

Useful for all apps
Critical for GUI apps

Balsa uses attrs

```
balsa = Balsa(application_name, author, verbose=True)
balsa.backup_count = 20  # lots of logging!
balsa.init_logger()
```

Options!

```
name = attrib(default=None)
author = attrib(default=None)
verbose = attrib(default=False)
gui = attrib(default=False)
delete_existing_log_files = attrib(default=False)
max_bytes = attrib(default=100 * 1e6)  # max size per log file
backup_count = attrib(default=3)  # max number of log files
error_callback = attrib(default=None)  # called on error or above
max_string_list_entries = attrib(default=100)  # string buffer internal to Balsa
log_directory = attrib(default=None)
log_extension = attrib(default=".log")
log_formatter = attrib(default=logging.Formatter("%(asctime)s - %(name)s - %(filename)s - 
%(lineno)s - %(funcName)s - %(levelname)s - %(message)s"))
is_root = attrib(default=True)
propagate = attrib(default=True)  # False for this logger to be independent of parent(s)
inhibit_cloud_services = attrib(default=False)
use_sentry = attrib(default=False)
sentry_dsn = attrib(default=None)  # get your project DSN at https://sentry.io
```

propagate

https://docs.python.org/3/library/logging.html

If this attribute evaluates to true, events logged to this logger will be passed to the handlers of higher level (ancestor) loggers, in addition to any handlers attached to this logger. Messages are passed directly to the ancestor loggers' handlers - neither the level nor filters of the ancestor loggers in question are considered.

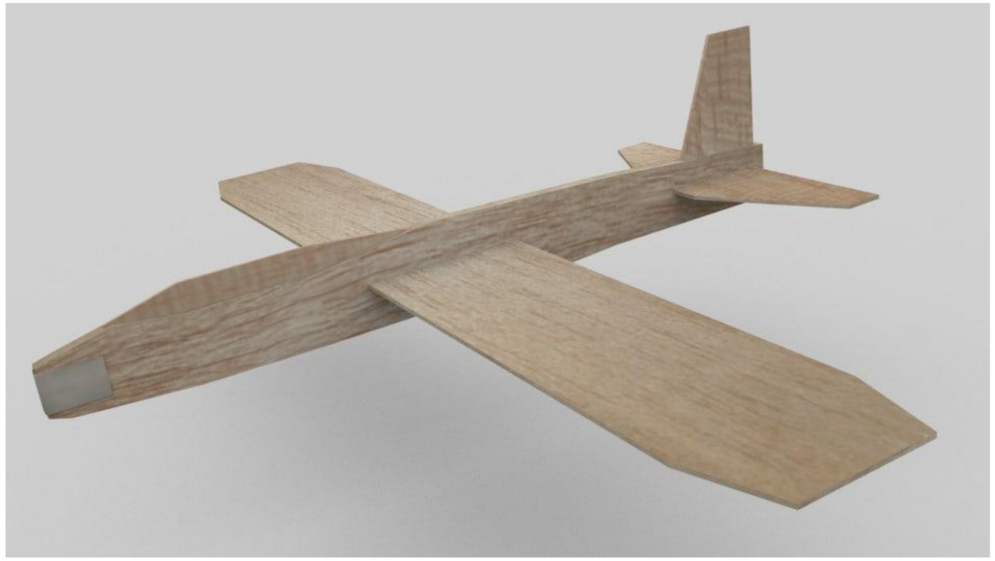
If this evaluates to false, logging messages are not passed to the handlers of ancestor loggers.

Set `propagate` to `False` for independent loggers such as a log window

Built-in CLI support with argparse

```
from balsa import delete_existing_arg_string, log_dir_arg_string, verbose_arg_string

parser = argparse.ArgumentParser()
parser.add_argument("-v", f"--{verbose_arg_string}", action="store_true", help="verbose")
parser.add_argument("-d", f"--{delete_existing_arg_string}", action="store_true", help="delete log")
parser.add_argument("-l", f"--{log_dir_arg_string}", help="log directory")
args = parser.parse_args()

balsa = Balsa(application_name, author)
balsa.init_logger_from_args(args)
```

Summary and Thank You



- Balsa is lightweight logging!
- Try balsa!

  ```
  pip install balsa
  ```

  https://github.com/jamesabel/balsa
  http://balsa.readthedocs.io/
- Please provide feedback, issues, PRs, …
- Thanks to Mark Rice (@MRice88) for testing and feedback

BACKUP

Verbose

	Default	Verbose
debug		file
info	file, string buffer	UI (stdout/stderr/GUI), string buffer
warning	UI (stdout/stderr/GUI)	
error	Exception services	Exception services