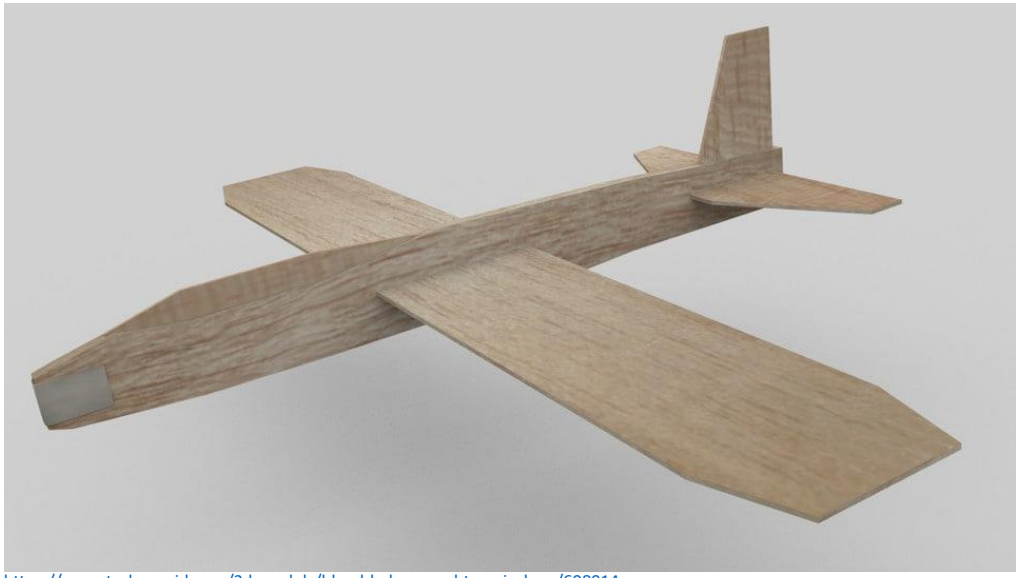
Balsa

Lightweight Python Logging



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

James Abel

PyBay - Sept 10, 2022

j@abel.co

www.abel.co @jamesabel www.github.com/jamesabel/balsa

The `logging` module

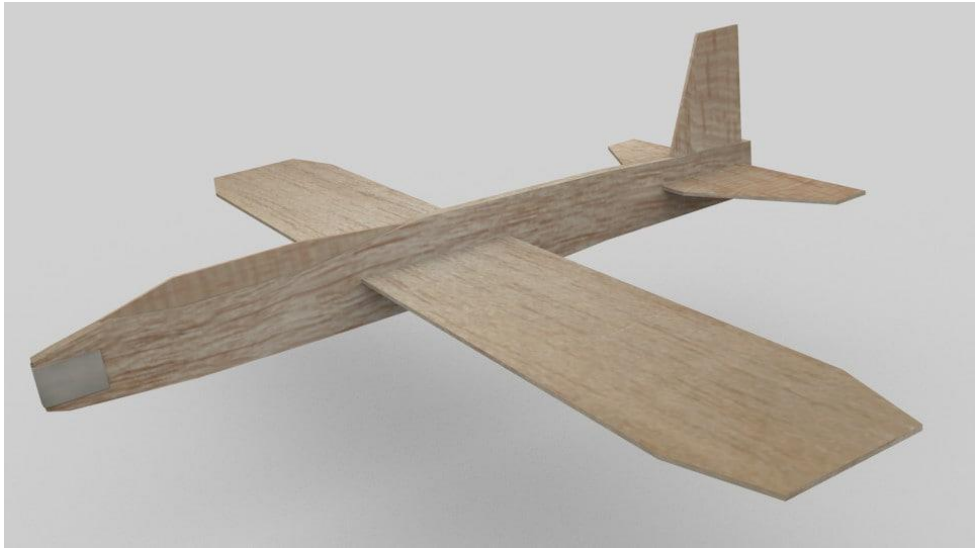
- The `logging` module is awesome!
  - Handlers – stream (console), files, sockets, HTTP, custom, … many more!
  - Filters
  - Formatters
  - Hierarchal
  - Log Levels – debug, info, warning, error, critical
- `logging.getLogger(name)` provides the logger associated with `name`
  - Can directly access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

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

- Log message format
- Handers
  - UI, File, Services
- Where to write log files?
- Log levels
  - Often different for each handler
- GUI
- Multiprocessing
- Tracebacks
- Interface into cloud services

Setting up `logging` Can Be Significant Effort

Balsa – Lightweight Logging

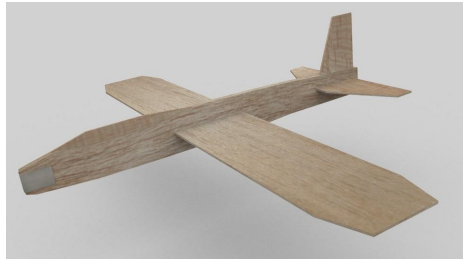


- Provide useful logging with a few lines of code
- Consistent formatting and interface
  - `appdirs` for log file directory
- CLI and GUI
  - GUI: `tkinter` or `PyQt5` dialog box (avoid writing to stdout/stderr)
- Structured Logging (via `yasf`)
- Supports Multiprocessing
  - Balsa → config as a dict → Balsa clone in separate process
- Object Oriented
  - Inherit from `Balsa` to select options, set alternative log levels, provide service credentials, etc.
- Optional Services
  - AWS CloudWatch (via `boto3`) – query-able structured logging
  - Sentry exception service - e.g., for >= ERROR log level (`sentry-sdk`)
- Verbosity expressed by intent rather than level (e.g., `verbose` for development)
- Cross-platform
- Available on PyPI

`pip install balsa`

www.abel.co @jamesabel www.github.com/jamesabel/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')
```

Output:

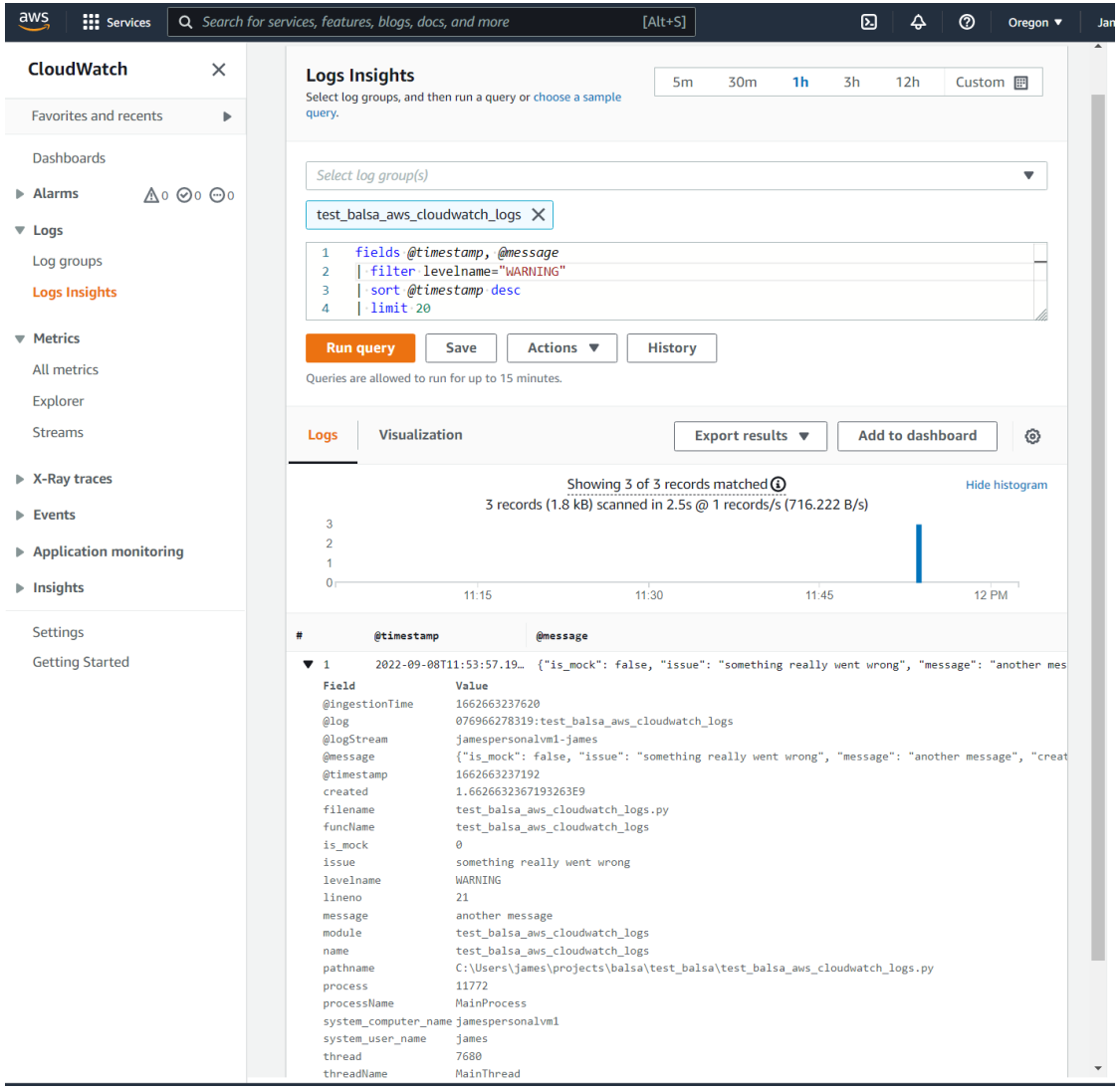
app name ↓ | function name ↓ | message ↓

```
2018-08-18 20:43:33,756 - example - balsa_simple_example.py - 12 - main - ERROR - my error example
```

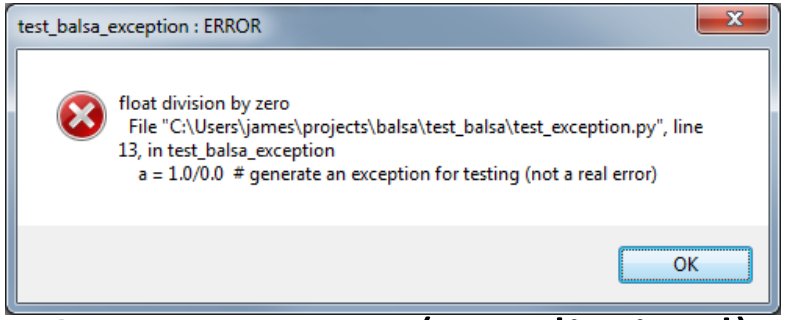
↑ timestamp | ↑ source file name | ↑ line number | ↑ level

Also writes out a file (e.g., Windows):

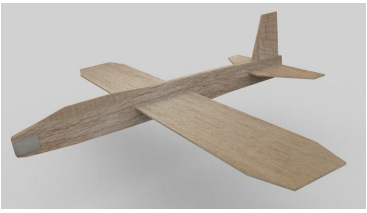
```
C:\Users\<user>\AppData\Local\james abel\example\Logs\example.log
```

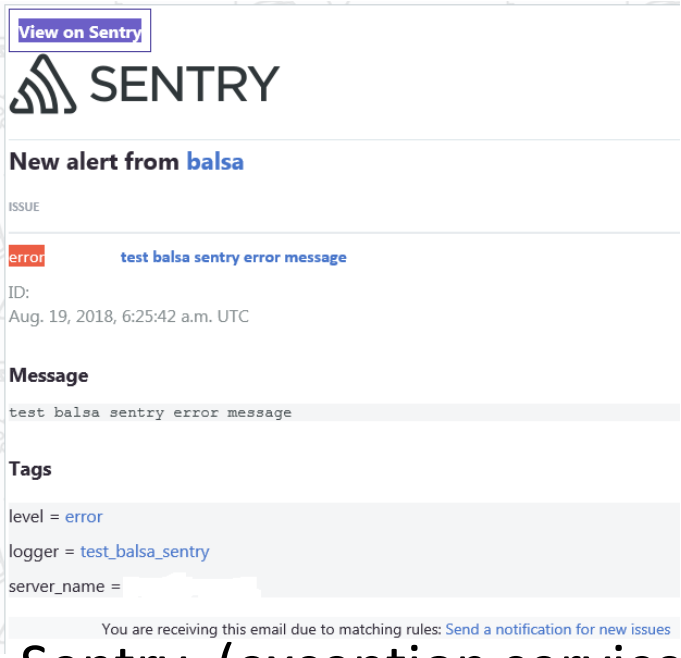


AWS CloudWatch



GUI messages (rate limited)

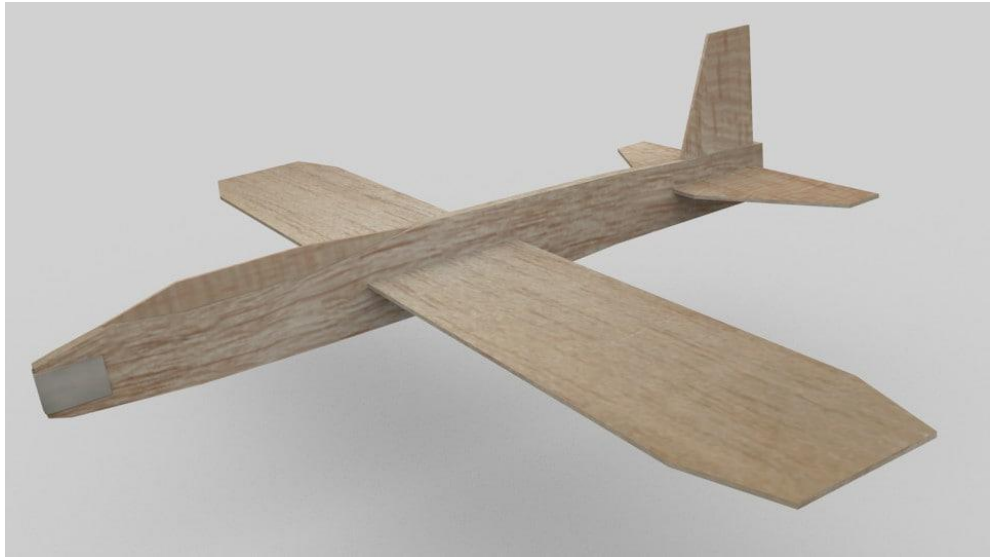




Sentry (exception service)

www.abel.co @jamesabel www.github.com/jamesabel/balsa

Summary and Thank You



- Try balsa!

  `pip install balsa`

  https://github.com/jamesabel/balsa

  http://balsa.readthedocs.io/

- Please provide feedback, issues, PRs, …

www.abel.co @jamesabel www.github.com/jamesabel/balsa