Logging for Forest Modules

Greyson Liu & Josh Barback

August 12, 2020

Overview

1. Goals
2. Python's `logging` module
3. Recommendations for Forest
4. Generating log records
5. Delivering log records
6. Example output
7. Resources

1. Goals

It's often useful to have human-readable, comprehensive records of data processing tasks.

- For end users:
  - Identify anomalies,
  - Verify task completion.
- For developers:
  - Locate errors or bugs,
  - Collect performance benchmarks.

2. Python's `logging` module

- Framework:
  - Events are associated with a `LogRecord` that includes a timestamp, traceback information, and a brief message.
  - `Logger` objects *emit* records.
  - `Handler` objects *format and deliver* records.

- Out-of-the-box features:
  - Easy to implement,
  - Flexible human-readable output.

- Some limitations:
  - Less flexible input options,
  - Less suitable for machine-readable output.

3. Recommendations for Forest

- Implement `logging` for any Forest module that will be imported.
  - Focus on human clients (end users & developers).
  - Deliver log records to CSV files.

- What should be logged?
  - Failure of a task,
  - Checkpoints during a time-consuming process,
  - Any unusual situations that the end user should know about,
  - Events of interest for development and testing.

- For details and code snippets:

  `logging_guidelines_for_developers.py`

4. Generating log records with a Logger

- To associate a `Logger` object with a module:

```
import logging
logger = logging.getLogger(__name__)
```

- To log events:

```
logger.info('Here is some information.')
logger.warning('This is a warning message!')
```

- Event logging works well with flow control:

```
if type(x) is int:  logger.info('This is an integer.')
else:  logger.warning('Not an integer!')

try:  assert(type(x) is int)
except:  logger.warning('Not an integer!')
```

5. Delivering log records with a `Handler`

- At runtime, we want a `Handler` object to collect log records and write them to a CSV.

- `log.py` provides some basic tools for formatting and delivering log records.

- Basic approach:

```
from forest_module import f
from log import log_to_csv
log_to_csv(<path/to/output/directory>)
f(x)
```

- We can also write wrappers that automatically handle log records, see code snippets.

6. Example output

Sample CSV containing log records:

asctime	msecs	levelname	module	funcName	message
2020-07-28 00:48:54	908	INFO	log	log_to_csv	Writing log messages to log.csv...
2020-07-28 00:48:54	910	INFO	summary	setup_output	Created output directories.
2020-07-28 00:48:54	910	INFO	templates	do	Setting up for user 1 of 53...
2020-07-28 00:48:54	910	INFO	templates	do	Processing data for user 1 of 53...
2020-07-28 00:48:55	95	WARNING	functions	summarize_file	Unknown device.
2020-07-28 00:48:55	95	INFO	summary	summarize_user	Summarized data for user 1.
2020-07-28 00:48:55	95	INFO	templates	do	Writing records for user 1 of 53...
2020-07-28 00:48:55	96	INFO	summary	write_user_records	Updated records for user 1.
2020-07-28 00:48:55	97	INFO	templates	do	Setting up for user 2 of 53...
⋮	⋮	⋮	⋮	⋮	⋮

7. Resources

- **Documentation for the logging package:**

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

- **Logging tutorial:**

  `https://docs.python.org/3/howto/logging.html`

- **Log record attributes:**

  `https://docs.python.org/3.8/library/logging.html?highlight=logging#logrecord-attributes`