redata-commons

Release v0.2.0

Chun Ly, UA Research Data Repository (ReDATA) Team

Jun 08, 2021

CONTENTS:

CHAPTER

ONE

OVERVIEW

This repository contains commonly used codes by ReDATA software

The GitHub repository is available here.

TL;DR: The primary sub-package is `commons`. It includes a number of modules, such as:

1. `git_info`
2. `logger`

1.1 Installation

From PyPI:

```
(venv) $ pip install redata
```

From source:

```
(venv) $ git clone git@github.com:UAL-ODIS/redata-commons.git
(venv) $ python setup.py install
```

1.2 Execution

1.2.1 Using `git_info`

To use, there are a number of ways to import it the main class, `GitInfo`.

```
import redata

code_path = "/path/to/repo"
gi = redata.commons.git_info.GitInfo(code_path)
```

or

```
from redata.commons.git_info import GitInfo

code_path = "/path/to/repo"
gi = GitInfo(code_path)
```

1.2.2 Using `logger`

There are a number of functions and classes available with `logger`:

1. `LogClass`: The main `Logger` object for stdout and file logging
2. `LogCommons`: Object that has methods to simplify repetitive logging
3. `log_stdout`: Function for stdout logging
4. `log_setup`: Function to set-up stdout and file logging. Call `LogClass.get_logger()`
5. `get_user_hostname`: Function to retrieve system information (user, host, IP, OS)
6. `get_log_file`: Function to retrieve filenames for file logging
7. `log_settings`: Function to log configuration settings. Only arguments specified through CLI arguments are shown
8. `pandas_write_buffer`: Function to write a prettified (i.e., Markdown) version of the table to log handler(s)

First you can either import `logger` via:

```
import redata
```

or

```
from redata.commons import logger
```

To construct a stdout and file logging object, the simplest approach is to use `log_setup()`:

```
from redata.commons import logger

log_dir = '/mnt/curation'
logfile_prefix = 'mylog'
log = logger.log_setup(log_dir, logfile_prefix)
log.info("print log message")
```

To only log to stdout, use `log_stdout()`:

```
from redata.commons import logger

log_std = logger.log_stdout()
log_std.info("print log message")
```

For simplicity, `LogCommons` simplifies many of the calls in various scripts and modules:

```
from redata.commons import logger, git_info

log_dir = '/mnt/curation'
logfile_prefix = 'mylog'
log = logger.log_setup(log_dir, logfile_prefix)

code_path = "/path/to/repo"
gi = git_info.GitInfo(code_path)

lc = LogCommons(log, 'script_run', gi)
lc.script_start()  # Starting log message
lc.script_sys_info()  # Retrieves user and hostname metadata and write to log
```

(continues on next page)

(continued from previous page)

```
lc.script_end()  # End of script
lc.log_permission()  # Change permission of log file to read and write for creator and␣
↪group
```

To retrieve the full path of the file log, use `get_log_file()`:

```
from redata.commons import logger

log_dir = '/mnt/curation'
logfile_prefix = 'mylog'
log = logger.log_setup(log_dir, logfile_prefix)
for handler in log.handlers:
    log_file = logger.get_log_file(handler)
```

To retrieve system (OS, IP) and user information, use `get_user_hostname()`:

```
from redata.commons import logger

sys_info_dict = logger.get_user_hostname()
```

The `log_settings` allows for explicit logging of input arguments to command-line scripts. The below example uses inputs specific to ReQUIAM.

```
from redata.commons import logger

log_dir = '/mnt/curation'
logfile_prefix = 'mylog'
log = logger.log_setup(log_dir, logfile_prefix)

config_dict = {
    'ldap_host': 'eds.iam.arizona.edu',
    'ldap_base_dn': 'dc=eds,dc=arizona,dc=edu',
    'ldap_user': 'figshare',
    'ldap_password': '***override***'
}
vargs = {'ldap_password': 'abcdef123456'}
protected_keys = ['ldap_password']
logger.log_settings(vargs, config_dict, protected_keys, log=log)
```

Finally, `pandas_write_buffer` is often used to provide `pandas` DataFrame in logs:

```
from redata.commons import logger
import pandas as pd

log_dir = '/mnt/curation'
logfile_prefix = 'mylog'
log = logger.log_setup(log_dir, logfile_prefix)
for handler in log.handlers:
    log_filename = logger.get_log_file(handler)

df = pd.read_csv('data.csv')  # This is a dummy filename
logger.pandas_write_buffer(df, log_filename)
```

1.3 Authors

- Chun Ly, Ph.D. (@astrochun) - University of Arizona Libraries, Office of Digital Innovation and Stewardship

See also the list of contributors who participated in this project.

1.4 License

This project is licensed under the MIT License - see the LICENSE file for details.

1.5 API Documentation

1.5.1 Subpackages

commons sub-package

Submodules

git_info module

class redata.commons.git_info.**GitInfo**(*input_path*)
Bases: object

Provides git repo information

Parameters **input_path** (str) – Full path containing the .git contents

Variables

- **input_path** – Full path containing the .git contents
- **head_path** – Full path of the .git HEAD
- **branch** – Active branch name
- **commit** – Full hash
- **short_commit** – short hash commit

get_active_branch_name()
Retrieve active branch name

Return type str

get_latest_commit()
Retrieve latest commit hash

Return type Tuple[str, str]

logger module

class redata.commons.logger.**LogClass**(*log_dir*, *logfile*)
Bases: object

Main class to log information to stdout and ASCII logfile

Parameters

- **log_dir** (str) – Relative path for exported logfile directory
- **logfile** (str) – Filename for exported log file

Variables

- **LOG_FILENAME** – Full path of log file
- **file_log_level** – File log level: DEBUG

To use: `log = LogClass(log_dir, logfile).get_logger()`

get_logger()
Primary method to retrieve stdout and ASCII file Logging object

class redata.commons.logger.**LogCommons**(*log*, *script_name*, *gi*, *code_name=''*, *version='0.4.1'*)
Bases: object

Common methods used when logging

Parameters

- **log** (Logger) – Logging object
- **script_name** (str) – Name of script for log messages
- **gi** (*GitInfo*) – Object containing git info
- **code_name** (str) – Name of codebase/software (e.g., ReQUIAM, LD-Cool-P)
- **version** (str) – Version of codebase/software. Default: Use redata's

Variables

- **log** – Logging object
- **script_name** – Name of script for log messages
- **gi** – Object containing git info
- **code_name** – Name of codebase/software (e.g., ReQUIAM, LD-Cool-P)
- **version** – Version of codebase/software.
- **start_text** – Text for script start
- **asterisk** – Parsing of start_text as asterisks
- **sys_info** – System info dict

log_permission()
Change permission for file logs

script_end()
Log end of script

script_start()
Log start of script

script_sys_info()
Log system info

redata.commons.logger.**get_log_file**(*log_handler*)
Get log file

Parameters **log_handler** – Logger object

Return log_file Full path of log file

Return type str

redata.commons.logger.**get_user_hostname**()
Retrieve user, hostname, IP, and OS configurations

Return type dict

Returns sys_info

redata.commons.logger.**log_settings**(*vargs*, *config_dict*, *protected_keys*, *log=<Logger stdout_logger (INFO)>*)
Log parsed arguments settings for scripts

Parameters

- **vargs** (dict) – Parsed arguments
- **config_dict** (dict) – Contains configuration settings. See commons.dict_load
- **protected_keys** (list) – list of private arguments to print unset or set status
- **log** (Logger) – LogClass

Return type int

Returns Number of errors with credentials

redata.commons.logger.**log_setup**(*log_dir*, *logfile_prefix*)
Create Logger object (log) for stdout and file logging

Parameters

- **log_dir** (str) – Directory for logs
- **logfile_prefix** (str) – Log file prefix

Return type Logger

Returns Logger object

redata.commons.logger.**log_stdout**()
Stdout logger

Return type Logger

Returns log

redata.commons.logger.**pandas_write_buffer**(*df*, *log_filename*)
Write pandas content via to_markdown() to log_filename

Parameters

- **df** (DataFrame) – DataFrame to write to buffer
- **log_filename** (str) – Full path for log file

Module contents

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

r

INDEX