Unittest for `report`

August 15, 2025

Contents

1 Test Information

1.1 Test Candidate Information

The Module report is designed to help with python logging and to support some handlers for logging to memory. For more Information read the sphinx documentation.

Library Information	
Name	report
State	Released
Supported Interpreters	python3
Version	6d1216d2ba09e60ad320112430818420
Dependencies	

1.2 Unittest Information

Unittest Information	
Version	2c03d3eba161a9fb0dbf0594fbda3965
Testruns with	python 3.13.5 (final)

1.3 Test System Information

System Information	
Architecture	64bit
Distribution	Debian GNU/Linux 13 trixie
Hostname	ahorn
Kernel	6.12.38+deb13-amd64 (#1 SMP PREEMPT_DYNAMIC Debian 6.12.38-1 (2025-07-16))
Machine	x86_64
Path	/home/dirk/work/unittest_collection/report
System	Linux
Username	dirk

2 Statistic

2.1 Test-Statistic for testrun with python 3.13.5 (final)

Number of tests	**5**
Number of successfull tests	**5**
Number of possibly failed tests	**0**
Number of failed tests	**0**
Executionlevel	Full Test (all defined tests)
Time consumption	0.017s

2.2 Coverage Statistic

Module- or Filename	Line-Coverage	Branch-Coverage
report	79.3%	55.6%
report.__init__.py	79.3%	

3 Tested Requirements

3.1 General Information

3.2 collectingHandler

3.3 collectingRingHandler

4 Requirements with no corresponding Testcase

4.1 Store log records (collectingHandler)

Description
Description 1.

4.2 String representation (collectingHandler)

4.3 Store log records (collectingRingHandler)

4.4 String representation (collectingRingHandler)

4.5 Number of stored logs in the RingHandler

Description
The number of stored log-records shall be given on initialisation or reinitialisation.

5 Testcases with no corresponding Requirement

5.1 Summary for testrun with python 3.13.5 (final)

5.1.1 _ErFPoCvVEeqssZLMJF_fcg

Testresult
This test was passed with the state: **Success**. See also full trace in section A.1.2!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/report/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 21:50:51,965
Finished-Time:	2025-08-15 21:50:51,969
Time-Consumption	0.003s

Testsummary:

Info	Running logger test sequence.
Success	Indexlist of log entries in stringrepresentation: Values and number of submitted values is correct. See detailed log for more information.

5.1.2 _PVhZECvVEeqssZLMJF_fcg

Testresult
This test was passed with the state: **Success**. See also full trace in section A.1.3!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/report/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 21:50:51,969
Finished-Time:	2025-08-15 21:50:51,973
Time-Consumption	0.004s

Testsummary:

Info	Running logger test sequence.
Success	Length of collected logs is correct (Content 5 and Type is <class 'int'>).
Success	Logged information is correct (Content ['Log entry number %d with level %s.', 30, 'test_helpers.py', 3] and Type is <class 'list'>).
Success	Logged information is correct (Content ['Log entry number %d with level %s.', 40, 'test_helpers.py', 4] and Type is <class 'list'>).
Success	Logged information is correct (Content ['Log entry number %d with level %s.', 50, 'test_helpers.py', 5] and Type is <class 'list'>).
Success	Logged information is correct (Content ['Log entry number %d with level %s.', 20, 'test_helpers.py', 6] and Type is <class 'list'>).
Success	Logged information is correct (Content ['Log entry number %d with level %s.', 40, 'test_helpers.py', 7] and Type is <class 'list'>).

5.1.3 _QBmb8CvVEeqssZLMJF_fcg

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.4!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/report/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 21:50:51,973
Finished-Time:	2025-08-15 21:50:51,976
Time-Consumption	0.003s

Testsummary:

Info	Running logger test sequence.
Success	Indexlist of log entries in stringrepresentation: Values and number of submitted values is correct. See detailed log for more information.

5.1.4 _XzMFcHYZEem_kd-7nxt1sg

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.1!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/report/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 21:50:51,962
Finished-Time:	2025-08-15 21:50:51,965
Time-Consumption	0.003s

Testsummary:

Info	Running logger test sequence.
Success	Length of collected logs is correct (Content 7 and Type is <class 'int'>).
Success	Logged information is correct (Content ['Log entry number %d with level %s.', 10, 'test_helpers.py', 1] and Type is <class 'list'>).
Success	Logged information is correct (Content ['Log entry number %d with level %s.', 20, 'test_helpers.py', 2] and Type is <class 'list'>).
Success	Logged information is correct (Content ['Log entry number %d with level %s.', 30, 'test_helpers.py', 3] and Type is <class 'list'>).
Success	Logged information is correct (Content ['Log entry number %d with level %s.', 40, 'test_helpers.py', 4] and Type is <class 'list'>).
Success	Logged information is correct (Content ['Log entry number %d with level %s.', 50, 'test_helpers.py', 5] and Type is <class 'list'>).
Success	Logged information is correct (Content ['Log entry number %d with level %s.', 20, 'test_helpers.py', 6] and Type is <class 'list'>).
Success	Logged information is correct (Content ['Log entry number %d with level %s.', 40, 'test_helpers.py', 7] and Type is <class 'list'>).

5.1.5 _fW5s8CzQEeqYsdFh5Cd6ng

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.5!

Testrun:	python 3.13.5 (final)
Caller:	/home/dirk/work/unittest_collection/report/unittest/src/report/__init__.py (331)
Start-Time:	2025-08-15 21:50:51,977
Finished-Time:	2025-08-15 21:50:51,981
Time-Consumption	0.004s

Testsummary:

Info	Running logger test sequence.
Success	Length of collectingRingHandler is correct (Content 5 and Type is <class 'int'>).
Success	Length of collectingRingHandler after reinitialisation is correct (Content 3 and Type is <class 'int'>).
Success	Log text is correct (Content 'Log entry number 5 with level CRITICAL.' and Type is <class 'str'>).
Success	Log text is correct (Content 'Log entry number 6 with level INFO.' and Type is <class 'str'>).
Success	Log text is correct (Content 'Log entry number 7 with level ERROR.' and Type is <class 'str'>).

A Trace for testrun with python 3.13.5 (final)

A.1 Tests with status Info (5)

A.1.1 _XzMFcHYZEem_kd-7nxt1sg

Testresult
This test was passed with the state: **Success**.

Info	Running logger test sequence.

```
Configuring collecting logger
Passing "Log entry number 1 with level DEBUG." to collectingHandler.
Log entry number 1 with level DEBUG.
Passing "Log entry number 2 with level INFO." to collectingHandler.
Log entry number 2 with level INFO.
Passing "Log entry number 3 with level WARNING." to collectingHandler.
Log entry number 3 with level WARNING.
Passing "Log entry number 4 with level ERROR." to collectingHandler.
Log entry number 4 with level ERROR.
Passing "Log entry number 5 with level CRITICAL." to collectingHandler.
Log entry number 5 with level CRITICAL.
Passing "Log entry number 6 with level INFO." to collectingHandler.
Log entry number 6 with level INFO.
Passing "Log entry number 7 with level ERROR." to collectingHandler.
Log entry number 7 with level ERROR.
```

Success	Length of collected logs is correct (Content 7 and Type is <class 'int'>).

```
Result (Length of collected logs): 7 (<class 'int'>)
Expectation (Length of collected logs): result = 7 (<class 'int'>)
```

Success	Logged information is correct (Content ['Log entry number %d with level %s.', 10, 'test_helpers.py', 1] and Type is <class 'list'>).

```
Result (Logged information): [ 'Log entry number %d with level %s.', 10, 'test_helpers.py', 1
↪  ] (<class 'list'>)
Expectation (Logged information): result = [ 'Log entry number %d with level %s.', 10,
↪  'test_helpers.py', 1 ] (<class 'list'>)
```

Success	Logged information is correct (Content ['Log entry number %d with level %s.', 20, 'test_helpers.py', 2] and Type is <class 'list'>).

```
Result (Logged information): [ 'Log entry number %d with level %s.', 20, 'test_helpers.py', 2
↪  ] (<class 'list'>)
```

```
Expectation (Logged information): result = [ 'Log entry number %d with level %s.', 20,
↪  'test_helpers.py', 2 ] (<class 'list'>)
```

Success	Logged information is correct (Content ['Log entry number %d with level %s.', 30, 'test_helpers.py', 3] and Type is <class 'list'>).

```
Result (Logged information): [ 'Log entry number %d with level %s.', 30, 'test_helpers.py', 3
↪  ] (<class 'list'>)
Expectation (Logged information): result = [ 'Log entry number %d with level %s.', 30,
↪  'test_helpers.py', 3 ] (<class 'list'>)
```

Success	Logged information is correct (Content ['Log entry number %d with level %s.', 40, 'test_helpers.py', 4] and Type is <class 'list'>).

```
Result (Logged information): [ 'Log entry number %d with level %s.', 40, 'test_helpers.py', 4
↪  ] (<class 'list'>)
Expectation (Logged information): result = [ 'Log entry number %d with level %s.', 40,
↪  'test_helpers.py', 4 ] (<class 'list'>)
```

Success	Logged information is correct (Content ['Log entry number %d with level %s.', 50, 'test_helpers.py', 5] and Type is <class 'list'>).

```
Result (Logged information): [ 'Log entry number %d with level %s.', 50, 'test_helpers.py', 5
↪  ] (<class 'list'>)
Expectation (Logged information): result = [ 'Log entry number %d with level %s.', 50,
↪  'test_helpers.py', 5 ] (<class 'list'>)
```

Success	Logged information is correct (Content ['Log entry number %d with level %s.', 20, 'test_helpers.py', 6] and Type is <class 'list'>).

```
Result (Logged information): [ 'Log entry number %d with level %s.', 20, 'test_helpers.py', 6
↪  ] (<class 'list'>)
Expectation (Logged information): result = [ 'Log entry number %d with level %s.', 20,
↪  'test_helpers.py', 6 ] (<class 'list'>)
```

Success	Logged information is correct (Content ['Log entry number %d with level %s.', 40, 'test_helpers.py', 7] and Type is <class 'list'>).

```
Result (Logged information): [ 'Log entry number %d with level %s.', 40, 'test_helpers.py', 7
↪  ] (<class 'list'>)
Expectation (Logged information): result = [ 'Log entry number %d with level %s.', 40,
↪  'test_helpers.py', 7 ] (<class 'list'>)
```

A.1.2 _ErFPoCvVEeqssZLMJF_fcg

Testresult

This test was passed with the state: **Success**.

Info	Running logger test sequence.

```
Configuring collecting logger
Passing "Log entry number 1 with level DEBUG." to collectingHandler.
Log entry number 1 with level DEBUG.
Passing "Log entry number 2 with level INFO." to collectingHandler.
Log entry number 2 with level INFO.
Passing "Log entry number 3 with level WARNING." to collectingHandler.
Log entry number 3 with level WARNING.
Passing "Log entry number 4 with level ERROR." to collectingHandler.
Log entry number 4 with level ERROR.
Passing "Log entry number 5 with level CRITICAL." to collectingHandler.
Log entry number 5 with level CRITICAL.
Passing "Log entry number 6 with level INFO." to collectingHandler.
Log entry number 6 with level INFO.
Passing "Log entry number 7 with level ERROR." to collectingHandler.
Log entry number 7 with level ERROR.
```

Success Indexlist of log entries in stringrepresentation: Values and number of submitted values is correct. See detailed log for more information.

```
Result (Indexlist of log entries in stringrepresentation): [ 52, 141, 229, 320, 409, 501, 589
↪  ] (<class 'list'>)
Expectation (Indexlist of log entries in stringrepresentation): result = [ 52, 141, 229, 320,
↪  409, 501, 589 ] (<class 'list'>)
Result (Submitted value number 1): 52 (<class 'int'>)
Expectation (Submitted value number 1): result = 52 (<class 'int'>)
Submitted value number 1 is correct (Content 52 and Type is <class 'int'>).
Result (Submitted value number 2): 141 (<class 'int'>)
Expectation (Submitted value number 2): result = 141 (<class 'int'>)
Submitted value number 2 is correct (Content 141 and Type is <class 'int'>).
Result (Submitted value number 3): 229 (<class 'int'>)
Expectation (Submitted value number 3): result = 229 (<class 'int'>)
Submitted value number 3 is correct (Content 229 and Type is <class 'int'>).
Result (Submitted value number 4): 320 (<class 'int'>)
Expectation (Submitted value number 4): result = 320 (<class 'int'>)
Submitted value number 4 is correct (Content 320 and Type is <class 'int'>).
Result (Submitted value number 5): 409 (<class 'int'>)
Expectation (Submitted value number 5): result = 409 (<class 'int'>)
Submitted value number 5 is correct (Content 409 and Type is <class 'int'>).
Result (Submitted value number 6): 501 (<class 'int'>)
Expectation (Submitted value number 6): result = 501 (<class 'int'>)
Submitted value number 6 is correct (Content 501 and Type is <class 'int'>).
Result (Submitted value number 7): 589 (<class 'int'>)
Expectation (Submitted value number 7): result = 589 (<class 'int'>)
Submitted value number 7 is correct (Content 589 and Type is <class 'int'>).
```

A.1.3 _PVhZECvVEeqssZLMJF_fcg

Testresult

This test was passed with the state: **Success**.

Info	Running logger test sequence.

```
Configuring collecting logger
Passing "Log entry number 1 with level DEBUG." to collectingRingHandler.
Log entry number 1 with level DEBUG.
Passing "Log entry number 2 with level INFO." to collectingRingHandler.
Log entry number 2 with level INFO.
Passing "Log entry number 3 with level WARNING." to collectingRingHandler.
Log entry number 3 with level WARNING.
Passing "Log entry number 4 with level ERROR." to collectingRingHandler.
Log entry number 4 with level ERROR.
Passing "Log entry number 5 with level CRITICAL." to collectingRingHandler.
Log entry number 5 with level CRITICAL.
Passing "Log entry number 6 with level INFO." to collectingRingHandler.
Log entry number 6 with level INFO.
Passing "Log entry number 7 with level ERROR." to collectingRingHandler.
Log entry number 7 with level ERROR.
```

Success	Length of collected logs is correct (Content 5 and Type is <class 'int'>).

```
Result (Length of collected logs): 5 (<class 'int'>)
Expectation (Length of collected logs): result = 5 (<class 'int'>)
```

Success	Logged information is correct (Content ['Log entry number %d with level %s.', 30, 'test_helpers.py', 3] and Type is <class 'list'>).

```
Result (Logged information): [ 'Log entry number %d with level %s.', 30, 'test_helpers.py', 3
↪  ] (<class 'list'>)
Expectation (Logged information): result = [ 'Log entry number %d with level %s.', 30,
↪  'test_helpers.py', 3 ] (<class 'list'>)
```

Success	Logged information is correct (Content ['Log entry number %d with level %s.', 40, 'test_helpers.py', 4] and Type is <class 'list'>).

```
Result (Logged information): [ 'Log entry number %d with level %s.', 40, 'test_helpers.py', 4
↪  ] (<class 'list'>)
Expectation (Logged information): result = [ 'Log entry number %d with level %s.', 40,
↪  'test_helpers.py', 4 ] (<class 'list'>)
```

Success	Logged information is correct (Content ['Log entry number %d with level %s.', 50, 'test_helpers.py', 5] and Type is <class 'list'>).

```
Result (Logged information): [ 'Log entry number %d with level %s.', 50, 'test_helpers.py', 5
↪  ] (<class 'list'>)
Expectation (Logged information): result = [ 'Log entry number %d with level %s.', 50,
↪  'test_helpers.py', 5 ] (<class 'list'>)
```

Success	Logged information is correct (Content ['Log entry number %d with level %s.', 20, 'test_helpers.py', 6] and Type is <class 'list'>).

```
Result (Logged information): [ 'Log entry number %d with level %s.', 20, 'test_helpers.py', 6
↪  ] (<class 'list'>)
Expectation (Logged information): result = [ 'Log entry number %d with level %s.', 20,
↪  'test_helpers.py', 6 ] (<class 'list'>)
```

Success	Logged information is correct (Content ['Log entry number %d with level %s.', 40, 'test_helpers.py', 7] and Type is <class 'list'>).

```
Result (Logged information): [ 'Log entry number %d with level %s.', 40, 'test_helpers.py', 7
↪  ] (<class 'list'>)
Expectation (Logged information): result = [ 'Log entry number %d with level %s.', 40,
↪  'test_helpers.py', 7 ] (<class 'list'>)
```

A.1.4 _QBmb8CvVEeqssZLMJF_fcg

Testresult

This test was passed with the state: **Success**.

Info	Running logger test sequence.

```
Configuring collecting logger
Passing "Log entry number 1 with level DEBUG." to collectingRingHandler.
Log entry number 1 with level DEBUG.
Passing "Log entry number 2 with level INFO." to collectingRingHandler.
Log entry number 2 with level INFO.
Passing "Log entry number 3 with level WARNING." to collectingRingHandler.
Log entry number 3 with level WARNING.
Passing "Log entry number 4 with level ERROR." to collectingRingHandler.
Log entry number 4 with level ERROR.
Passing "Log entry number 5 with level CRITICAL." to collectingRingHandler.
Log entry number 5 with level CRITICAL.
Passing "Log entry number 6 with level INFO." to collectingRingHandler.
Log entry number 6 with level INFO.
```

```
Passing "Log entry number 7 with level ERROR." to collectingRingHandler.
Log entry number 7 with level ERROR.
```

Success	Indexlist of log entries in stringrepresentation: Values and number of submitted values is correct. See detailed log for more information.

```
Result (Indexlist of log entries in stringrepresentation): [ 52, 143, 232, 324, 412 ] (<class
↪  'list'>)
Expectation (Indexlist of log entries in stringrepresentation): result = [ 52, 143, 232, 324,
↪  412 ] (<class 'list'>)
Result (Submitted value number 1): 52 (<class 'int'>)
Expectation (Submitted value number 1): result = 52 (<class 'int'>)
Submitted value number 1 is correct (Content 52 and Type is <class 'int'>).
Result (Submitted value number 2): 143 (<class 'int'>)
Expectation (Submitted value number 2): result = 143 (<class 'int'>)
Submitted value number 2 is correct (Content 143 and Type is <class 'int'>).
Result (Submitted value number 3): 232 (<class 'int'>)
Expectation (Submitted value number 3): result = 232 (<class 'int'>)
Submitted value number 3 is correct (Content 232 and Type is <class 'int'>).
Result (Submitted value number 4): 324 (<class 'int'>)
Expectation (Submitted value number 4): result = 324 (<class 'int'>)
Submitted value number 4 is correct (Content 324 and Type is <class 'int'>).
Result (Submitted value number 5): 412 (<class 'int'>)
Expectation (Submitted value number 5): result = 412 (<class 'int'>)
Submitted value number 5 is correct (Content 412 and Type is <class 'int'>).
```

A.1.5 _fW5s8CzQEeqYsdFh5Cd6ng

Testresult

This test was passed with the state: **Success**.

Info	Running logger test sequence.

```
Configuring collecting logger
Passing "Log entry number 1 with level DEBUG." to collectingRingHandler.
Log entry number 1 with level DEBUG.
Passing "Log entry number 2 with level INFO." to collectingRingHandler.
Log entry number 2 with level INFO.
Passing "Log entry number 3 with level WARNING." to collectingRingHandler.
Log entry number 3 with level WARNING.
Passing "Log entry number 4 with level ERROR." to collectingRingHandler.
Log entry number 4 with level ERROR.
Passing "Log entry number 5 with level CRITICAL." to collectingRingHandler.
Log entry number 5 with level CRITICAL.
```

```
Passing "Log entry number 6 with level INFO." to collectingRingHandler.
Log entry number 6 with level INFO.
Passing "Log entry number 7 with level ERROR." to collectingRingHandler.
Log entry number 7 with level ERROR.
```

Success Length of collectingRingHandler is correct (Content 5 and Type is <class 'int'>).

```
Result (Length of collectingRingHandler): 5 (<class 'int'>)
Expectation (Length of collectingRingHandler): result = 5 (<class 'int'>)
```

Success Length of collectingRingHandler after reinitialisation is correct (Content 3 and Type is <class 'int'>).

```
Result (Length of collectingRingHandler after reinitialisation): 3 (<class 'int'>)
Expectation (Length of collectingRingHandler after reinitialisation): result = 3 (<class
↪  'int'>)
```

Success Log text is correct (Content 'Log entry number 5 with level CRITICAL.' and Type is <class 'str'>).

```
Result (Log text): 'Log entry number 5 with level CRITICAL.' (<class 'str'>)
Expectation (Log text): result = 'Log entry number 5 with level CRITICAL.' (<class 'str'>)
```

Success Log text is correct (Content 'Log entry number 6 with level INFO.' and Type is <class 'str'>).

```
Result (Log text): 'Log entry number 6 with level INFO.' (<class 'str'>)
Expectation (Log text): result = 'Log entry number 6 with level INFO.' (<class 'str'>)
```

Success Log text is correct (Content 'Log entry number 7 with level ERROR.' and Type is <class 'str'>).

```
Result (Log text): 'Log entry number 7 with level ERROR.' (<class 'str'>)
Expectation (Log text): result = 'Log entry number 7 with level ERROR.' (<class 'str'>)
```

B Test-Coverage

B.1 report

The line coverage for report was 79.3%
The branch coverage for report was 55.6%

B.1.1 report.__init__.py

The line coverage for report.__init__.py was 79.3%
The branch coverage for report.__init__.py was 55.6%

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
"""
report (Report Module)
======================

**Author:**

* Dirk Alders <sudo-dirk@mount-mockery.de>

**Description:**

    The Module is designed to help with python logging and to support some handlers for logging
    to memory.

**Submodules:**

* :class:`report.collectingHandler`
* :class:`report.collectingRingHandler`
* :class:`report.collectingTestcaseHandler`
* :func:`report.consoleLoggingConfigure`
* :class:`report.testSession`


**Unittest:**

        See also the :download:`unittest <../../report/_testresults_/unittest.pdf>` documentation.
"""
__DEPENDENCIES__ = []

import collections
import json
import logging
from logging.config import dictConfig
import logging.handlers
import os
import sys

try:
    from config import DEBUG
except ImportError:
    DEBUG = True
try:
    from config import LOG_LEVEL
except ImportError:
    LOG_LEVEL = logging.WARNING
try:
    from config import LOG_HOSTNAME
except ImportError:
    LOG_HOSTNAME = 'localhost'
try:
    from config import APP_NAME as ROOT_LOGGER_NAME
except ImportError:
    ROOT_LOGGER_NAME = 'root'
logger = logging.getLogger(ROOT_LOGGER_NAME).getChild(__name__)

__DESCRIPTION__ = """The Module {\\tt %s} is designed to help with python logging and to support some handlers for logging to memory.
For more Information read the sphinx documentation.""" % __name__.replace('_', '\\_')
```

```
"""The Module Description"""
__INTERPRETER__ = (3, )
"""The Tested Interpreter-Versions"""

JOURNAL_FMT = "%(levelname)8s - %(name)s - %(message)s"
""" A short journal formatter including the most important information"""
SHORT_FMT = "%(asctime)s: %(levelname)8s - %(name)s - %(message)s"
""" A short formatter including the most important information"""
LONG_FMT = SHORT_FMT + "\n                                        File \"%(pathname)s\", line %(
    lineno)d, in %(funcName)s"
""" A long formatter which results in links to the source code inside Eclipse"""
MAX_FMT = """
%(name)s
%(levelno)s
%(levelname)s
%(pathname)s
%(filename)s
%(module)s
%(lineno)d
%(funcName)s
%(created)f
%(asctime)s
%(msecs)d
%(relativeCreated)d
%(thread)d
%(threadName)s
%(process)d
%(message)s"""
DEFAULT_FMT = LONG_FMT
"""The default formatstring"""


class collectingHandler(logging.Handler):
    MY_LOGS = []

    def __init__(self):
        logging.Handler.__init__(self)
        self.setFormatter(logging.Formatter(MAX_FMT))
        self.setLevel(logging.DEBUG)

    def emit(self, record):
        self.format(record)
        self.MY_LOGS.append(record.__dict__)

    def make_independent(self):
        self.MY_LOGS = []

    def get_logs(self):
        rv = []
        while len(self.MY_LOGS) > 0:
            entry = self.MY_LOGS.pop(0)
            # remove exception info (not json iterasable)
            if 'exc_info' in entry:
                entry.pop('exc_info')
            rv.append(entry)
        return rv

    def get_str(self, logs=None, fmt=SHORT_FMT):
        logs = logs or self.MY_LOGS
        return '\n'.join([fmt % log for log in logs])

```

```
119     def __len__(self):
120         return len(self.MY_LOGS)
121 
122     def    str   (self):
123         return self.get  str(self.MY_LOGS)
124 
125 
126 class collectingRingHandler(collectingHandler):
127     MY_LOGS = collections.deque([], 10)
128 
129     def __init__(self, max_logs=None):
130         collectingHandler.__init__(self)
131         if max_logs is not None and max_logs != self.MY_LOGS.maxlen:
132             self.MY_LOGS.   init   (list(self.MY_LOGS), max_logs)
133 
134     def make_independent(self):
135         self.MY_LOGS = collections.deque([], self.MY_LOGS.maxlen)
136 
137     def get_logs(self):
138         return list(self.MY_LOGS)
139 
140 
141 TCEL_SINGLE = 0
142 """Testcase level (smoke), this is just a rough test for the main functionality"""
143 TCEL_SMOKE = 10
144 """Testcase level (smoke), this is just a rough test for the main functionality"""
145 TCEL_SHORT = 50
146 """Testcase level (short), this is a short test for an extended functionality"""
147 TCEL_FULL = 90
148 """Testcase level (full), this is a complete test for the full functionality"""
149 TCEL_NAMES = {
150     TCEL_SINGLE: 'Single Test',
151     TCEL_SMOKE: 'Smoke Test (Minumum subset)',
152     TCEL_SHORT: 'Short Test (Subset)',
153     TCEL_FULL: 'Full Test (all defined tests)'
154 }
155 """Dictionary for resolving the test case levels (TCL) to a (human readable) name"""
156 
157 TCEL_REVERSE_NAMED = {
158     'short': TCEL_SHORT,
159     'smoke': TCEL_SMOKE,
160     'single': TCEL_SINGLE,
161     'full': TCEL_FULL,
162 }
163 """Dictionary for resolving named test case levels (TCL) to test case level number"""
164 
165 
166 class collectingTestcaseHandler(collectingHandler):
167     MY_LOGS = []
168 
169     def emit(self, record):
170         self.format(record)
171         self.MY_LOGS.append(record.__dict__)
172         self.MY_LOGS[-1]['moduleLogger'] = collectingHandler().get_logs()
173 
174 
175 class JsonFormatter(logging.Formatter):
176     def format(self, record):
177         obj = {}
178         for key in ["name", "levelno", "levelname", "pathname", "filename", "module", "lineno", "
    funcName", "created", "msecs", "relativeCreated", "thread", "threadName", "process", "
    processName", "msg", "args", "exc_info", "exc_text"]:
179             obj[key] = getattr(record, key)
180         obj["msg"] = obj["msg"] % obj["args"]
181         return json.dumps(obj)
```

```
182
183
184 def add_handler_stdout(logger: logging.Logger, level: int = logging.WARNING, fmt: str =
    DEFAULT_FMT) -> logging.StreamHandler:
185     logger.setLevel(logging.DEBUG)
186     #
187     handler = logging.StreamHandler(sys.stdout)
188     handler.setFormatter(logging.Formatter(fmt))
189     handler.setLevel(level)
190     logger.addHandler(handler)
191     return handler
192
193
194 def add_handler_file(logger: logging.Logger, filename: str, maxMbytes: int, backupCount: int,
    level: int = logging.DEBUG, fmt: str = DEFAULT_FMT) -> logging.handlers.RotatingFileHandler:
195     if maxMbytes < 1:
196         raise ValueError(f"The parameter maxMbytes needs to be >= 1. maxMbytes={maxMbytes}")
197     if backupCount < 1:
198         raise ValueError(f"The parameter backupCount needs to be >= 1. backupCount={backupCount}"
    )
199     #
200     logger.setLevel(logging.DEBUG)
201     #
202     handler = logging.handlers.RotatingFileHandler(filename, maxBytes=maxMbytes*1048576,
    backupCount=backupCount)
203     handler.setFormatter(logging.Formatter(fmt))
204     handler.setLevel(level)
205     logger.addHandler(handler)
206     return handler
207
208
209 def add_handler_socket(logger: logging.Logger, level: int = logging.DEBUG, host: str = "localhost
    ", port: int = 19996) -> logging.handlers.SocketHandler:
210     logger.setLevel(logging.DEBUG)
211     #
212     handler = logging.handlers.SocketHandler(host, port)
213     handler.setLevel(level)
214     logger.addHandler(handler)
215     return handler
216
217
218 def default_logging_config(fmt=JOURNAL_FMT):
219     """You are able to configure logging by a config file including these line:
220
221     import logging
222
223
224     DEBUG = False
225     #
226     # Logging
227     #
228     APP_NAME = "<YOUR_APP_NAME>"
229     LOG_HOSTNAME = "loggy"          # When DEBUG is True
230     LOG_LEVEL = logging.INFO        # STDOUT logging
231     """
232     logger = logging.getLogger(ROOT_LOGGER_NAME)
233
234     add_handler_stdout(logger, LOG_LEVEL, fmt=fmt)
235     if DEBUG:
236         add_handler_socket(logger, host=LOG_HOSTNAME)
237     return logger.getChild('main')
```

```
238
239
240 def app_logging_config():
241     """
242     For details see comment in default_logging_config.
243     """
244     return default_logging_config(LONG_FMT)
245
246
247 class testSession(dict):
248     KEY_NAME = 'name'
249     KEY_FAILED_TESTS = 'number_of_failed_tests'
250     KEY_POSSIBLY_FAILED_TESTS = 'number_of_possibly_failed_tests'
251     KEY_SUCCESS_TESTS = 'number_of_successfull_tests'
252     KEY_ALL_TESTS = 'number_of_tests'
253     KEY_EXEC_LVL = 'testcase_execution_level'
254     KEY_EXEC_NAMES = 'testcase_names'
255     KEY_LVL_NAMES = 'level_names'
256     KEY_TESTCASELIST = 'testcases'
257     KEY_UID_LIST = 'uid_list_sorted'
258     #
259     DEFAULT_BASE_DATA = {
260         KEY_NAME: 'Default Testsession name',
261         KEY_FAILED_TESTS: 0,
262         KEY_POSSIBLY_FAILED_TESTS: 0,
263         KEY_FAILED_TESTS: 0,
264         KEY_SUCCESS_TESTS: 0,
265         KEY_ALL_TESTS: 0,
266         KEY_EXEC_LVL: TCEL_FULL,
267         KEY_EXEC_NAMES: TCEL_NAMES,
268     }
269
270     def __init__(self, module_names=[], **kwargs):
271         dict.__init__(self, time_consumption=0.)
272         self.__testcase__ = None
273         self.__set_base_data__(**kwargs)
274         self.__configure_logging__(module_names)
275
276     def __set_base_data__(self, **kwargs):
277         for key in set([key for key in self.DEFAULT_BASE_DATA.keys()] + [key for key in kwargs.
    keys()]):
278             self[key] = kwargs.get(key, self.DEFAULT_BASE_DATA.get(key))
279         self[self.KEY_TESTCASELIST] = {}
280         self[self.KEY_UID_LIST] = []
281
282     def __configure_logging__(self, module_names):
283         #
284         # Configure logging for testSession
285         #
286         logging_config = dict(
287             version=1,
288             formatters={
289                 'short': {
290                     'format': SHORT_FMT,
291                 },
292                 'long': {
293                     'format': LONG_FMT,
294                 },
295             },
296             handlers={
297                 'console': {
298                     'level': 'DEBUG',
```

```
299                     'class': 'logging.NullHandler',
300                     'formatter': 'short',
301                 },
302                 'module_logs': {
303                     'level': 'DEBUG',
304                     'class': 'report.collectingHandler',
305                     'formatter': 'short',
306                 },
307                 'testcase_logs': {
308                     'level': 'DEBUG',
309                     'class': 'report.collectingTestcaseHandler',
310                     'formatter': 'short',
311                 },
312             },
313             loggers=self.__module_loggers__(module_names),
314         )
315         dictConfig(logging_config)
316 
317     def __module_loggers__(self, module_names):
318         rv = {}
319         rv['__tLogger__'] = dict(handlers=['console', 'testcase_logs'], level='DEBUG', propagate=
    False)
320         for name in module_names + ['__mLogger__']:
321             rv[name] = dict(handlers=['console', 'module_logs'], level='DEBUG', propagate=False)
322         return rv
323 
324     def testCase(self, name, testcase_execution_level, test_method, *args, **kwargs):
325         if testcase_execution_level <= self[self.KEY_EXEC_LVL]:
326             sys.stdout.write('     %s...' % name[:75])
327             tLogger = logging.getLogger('__tLogger__')
328             tHandler = collectingTestcaseHandler()
329             if len(tHandler.MY_LOGS) > 0:
330                 raise AttributeError("Testcaselogger shall be empty after closing testcase!")
331             tLogger._log(logging.DEBUG, name, None)
332             if len(tHandler.MY_LOGS) != 1:
333                 raise AttributeError("Testcaselogger shall have only one entry for the main
    testcase (temporary)!")
334             self.__testcase__ = tHandler.get_logs()[0]
335             test_method(logging.getLogger('__tLogger__'), *args, **kwargs)
336             self.__close_active_testcase__()
337 
338     def __close_active_testcase__(self):
339         if self.__testcase__ is not None:
340             name = self.__testcase__.get('message')
341             #
342             # Add testcase
343             #
344             tch = collectingTestcaseHandler()
345             self.__testcase__['testcaseLogger'] = tch.get_logs()
346             if name in self[self.KEY_TESTCASELIST]:
347                 raise AttributeError("Testcase named %s already exists" % name)
348             self[self.KEY_TESTCASELIST][name] = self.__testcase__
349             self[self.KEY_UID_LIST].append(name)
350             #
351             # Adapt testcase data
352             #
353             self[self.KEY_TESTCASELIST][name]['levelno'] = 0
354             self[self.KEY_TESTCASELIST][name]['time_consumption'] = 0.
355             for teststep in self[self.KEY_TESTCASELIST][name]['testcaseLogger']:
356                 # store maximum level to testcase
357                 if teststep.get('levelno') > self[self.KEY_TESTCASELIST][name]['levelno']:
358                     self[self.KEY_TESTCASELIST][name]['levelno'] = teststep.get('levelno')
359                     self[self.KEY_TESTCASELIST][name]['levelname'] = teststep.get('levelname')
```

```
360                 # store time_consumption for teststep
361                 try:
362                     teststep['time_consumption'] = teststep['created'] - teststep['moduleLogger'
    ][-1]['created']
363                 except IndexError:
364                     teststep['time_consumption'] = 0.
365                 # Increment testcase time_comsumption
366             # Increment testcase counters
367             #
368             self[self.KEY_ALL_TESTS] += 1
369             if self[self.KEY_TESTCASELIST][name]['levelno'] <= logging.INFO:
370                 self[self.KEY_SUCCESS_TESTS] += 1
371                 sys.stdout.write('\033[92mSUCCESS\033[0m\n')
372             elif self[self.KEY_TESTCASELIST][name]['levelno'] >= logging.ERROR:
373                 self[self.KEY_FAILED_TESTS] += 1
374                 sys.stdout.write('\033[91mFAILED\033[0m\n')
375             else:
376                 self[self.KEY_POSSIBLY_FAILED_TESTS] += 1
377                 sys.stdout.write('\033[93mPOSSIBLY FAILED\033[0m\n')
378             # Set testcase time and time_consumption
379             self[self.KEY_TESTCASELIST][name]['time_start'] = self.__testcase__['asctime']
380             self[self.KEY_TESTCASELIST][name]['time_finished'] = teststep['asctime']
381             self[self.KEY_TESTCASELIST][name]['time_consumption'] = teststep['created'] - self.
    __testcase__['created']
382             # Set testcase time consumption
383             self['time_consumption'] += self[self.KEY_TESTCASELIST][name]['time_consumption']
384         self.__testcase__ = None
```