Unittest for `report`

May 16, 2026

Contents

1 Test Information

1.1 Testobject 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.

Information	
Name	report
State	In test
Version	1.8.1
Dependencies	
	Python standards

1.2 Testdata Information

Information	
Version	0.0.1
Testruns	Library test

1.3 Testsystem Information

System Information	
Architecture	64bit
Distribution	Debian GNU/Linux 13 trixie
Hostname	ahorn
Kernel	6.12.86+deb13-amd64 (#1 SMP PREEMPT_DYNAMIC Debian 6.12.86-1 (2026-05-08))
Machine	x86_64
Path	/home/dirk/work/unittest_collection/report/unittest/src
System	Linux
Username	dirk

2 Statistic

2.1 Test-Statistic for testrun: Library test

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	0.014s

2.2 Coverage Statistic

Module- or Filename	Line-Coverage	Branch-Coverage
report	78.5%	52.6%
report.__init__.py	78.5%	

3 Tested Requirements for testrun: Library test

3.1 General Information

3.2 collectingHandler

3.2.1 Store log records (collectingHandler) (REQ-0001)

Description
Description 1.

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

Testrun:	
Caller:	/home/dirk/work/unittest_collection/report/unittest/src/tests/__init__.py (12)
Start-Time:	2026-05-16 21:10:51,806
Finished-Time:	2026-05-16 21:10:51,809
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'>).

3.2.2 String representation (collectingHandler) (REQ-0002)

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

Testrun:	
Caller:	/home/dirk/work/unittest_collection/report/unittest/src/tests/__init__.py (13)
Start-Time:	2026-05-16 21:10:51,809
Finished-Time:	2026-05-16 21:10:51,812

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.

3.3 collectingRingHandler

3.3.1 Store log records (collectingRingHandler) (REQ-0003)

Testresult

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

Testrun:	
Caller:	/home/dirk/work/unittest_collection/report/unittest/src/tests/__init__.py (14)
Start-Time:	2026-05-16 21:10:51,813
Finished-Time:	2026-05-16 21:10:51,816
Time-Consumption	0.003s

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'>).

3.3.2 String representation (collectingRingHandler) (REQ-0004)

Testresult

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

Testrun:	
Caller:	/home/dirk/work/unittest_collection/report/unittest/src/tests/__init__.py (15)
Start-Time:	2026-05-16 21:10:51,816
Finished-Time:	2026-05-16 21:10:51,818
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.

3.3.3 Number of stored logs in the RingHandler (REQ-0005)

Description

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

Testresult

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

Testrun:	
Caller:	/home/dirk/work/unittest_collection/report/unittest/src/tests/__init__.py (16)
Start-Time:	2026-05-16 21:10:51,819
Finished-Time:	2026-05-16 21:10:51,821
Time-Consumption	0.003s

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: Library test

A.1 Tests with status Info (5)

A.1.1 Store log records (collectingHandler) (REQ-0001)

Description
Description 1.

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 String representation (collectingHandler) (REQ-0002)

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 Store log records (collectingRingHandler) (REQ-0003)

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 String representation (collectingRingHandler) (REQ-0004)

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 Number of stored logs in the RingHandler (REQ-0005)

Description

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

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 78.5%
The branch coverage for report was 52.6%

B.1.1 report.__init__.py

The line coverage for report.__init__.py was 78.5%
The branch coverage for report.__init__.py was 52.6%

```
"""
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.
"""

__VERSION__ = "1.8.1"
__DEPENDENCIES__ = ["Python standards"]

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:
```

```
45      LOG_LEVEL = logging.WARNING
46  try:
47      from config import LOG_HOSTNAME
48  except ImportError:
49      LOG_HOSTNAME = "localhost"
50  try:
51      from config import APP_NAME as ROOT_LOGGER_NAME
52  except ImportError:
53      ROOT_LOGGER_NAME = "root"
54  logger = logging.getLogger(ROOT_LOGGER_NAME).getChild(__name__)
55
56  __DESCRIPTION__ = """The Module {\\tt %s} is designed to help with python logging and to support
        some handlers for logging to memory.
57  For more Information read the sphinx documentation.""" % __name__.replace("_", "\\_")
58  """The Module Description"""
59
60  JOURNAL_FMT = "%(levelname)8s - %(name)s - %(message)s"
61  """ A short journal formatter including the most important information"""
62  SHORT_FMT = "%(asctime)s: %(levelname)8s - %(name)s - %(message)s"
63  """ A short formatter including the most important information"""
64  LONG_FMT = SHORT_FMT + '\n                                        File "%(pathname)s", line %(lineno)
        d, in %(funcName)s'
65  """ A long formatter which results in links to the source code inside Eclipse"""
66  MAX_FMT = """
67  %(name)s
68  %(levelno)s
69  %(levelname)s
70  %(pathname)s
71  %(filename)s
72  %(module)s
73  %(lineno)d
74  %(funcName)s
75  %(created)f
76  %(asctime)s
77  %(msecs)d
78  %(relativeCreated)d
79  %(thread)d
80  %(threadName)s
81  %(process)d
82  %(message)s"""
83  DEFAULT_FMT = LONG_FMT
84  """The default formatstring"""
85
86
87  class collectingHandler(logging.Handler):
88      MY_LOGS = []
89
90      def __init__(self):
91          logging.Handler.__init__(self)
92          self.setFormatter(logging.Formatter(MAX_FMT))
93          self.setLevel(logging.DEBUG)
94
95      def emit(self, record):
96          self.format(record)
97          self.MY_LOGS.append(record.__dict__)
98
99      def make_independent(self):
100         self.MY_LOGS = []
101
102     def get_logs(self):
103         rv = []
104         while len(self.MY_LOGS) > 0:
105             entry = self.MY_LOGS.pop(0)
```

```
106             # remove exception info (not json iterasable)
107             if "exc_info" in entry:
108                 entry.pop("exc_info")
109             rv.append(entry)
110         return rv
111 
112     def get_str(self, logs=None, fmt=SHORT_FMT):
113         logs = logs or self.MY_LOGS
114         return "\n".join([fmt % log for log in logs])
115 
116     def __len__(self):
117         return len(self.MY_LOGS)
118 
119     def __str__(self):
120         return self.get_str(self.MY_LOGS)
121 
122 
123 class collectingRingHandler(collectingHandler):
124     MY_LOGS = collections.deque([], 10)
125 
126     def __init__(self, max_logs=None):
127         collectingHandler.__init__(self)
128         if max_logs is not None and max_logs != self.MY_LOGS.maxlen:
129             self.MY_LOGS.__init__(list(self.MY_LOGS), max_logs)
130 
131     def make_independent(self):
132         self.MY_LOGS = collections.deque([], self.MY_LOGS.maxlen)
133 
134     def get_logs(self):
135         return list(self.MY_LOGS)
136 
137 
138 TCEL_SINGLE = 0
139 """Testcase level (smoke), this is just a rough test for the main functionality"""
140 TCEL_SMOKE = 10
141 """Testcase level (smoke), this is just a rough test for the main functionality"""
142 TCEL_SHORT = 50
143 """Testcase level (short), this is a short test for an extended functionality"""
144 TCEL_FULL = 90
145 """Testcase level (full), this is a complete test for the full functionality"""
146 TCEL_NAMES = {
147     TCEL_SINGLE: "Single Test",
148     TCEL_SMOKE: "Smoke Test (Minumum subset)",
149     TCEL_SHORT: "Short Test (Subset)",
150     TCEL_FULL: "Full Test (all defined tests)",
151 }
152 """Dictionary for resolving the test case levels (TCL) to a (human readable) name"""
153 
154 TCEL_REVERSE_NAMED = {
155     "short": TCEL_SHORT,
156     "smoke": TCEL_SMOKE,
157     "single": TCEL_SINGLE,
158     "full": TCEL_FULL,
159 }
160 """Dictionary for resolving named test case levels (TCL) to test case level number"""
161 
162 
163 class collectingTestcaseHandler(collectingHandler):
164     MY_LOGS = []
165 
```

```
166     def emit(self, record):
167         self.format(record)
168         self.MY_LOGS.append(record.__dict__)
169         self.MY_LOGS[-1]["moduleLogger"] = collectingHandler().get_logs()
170 
171 
172 class JsonFormatter(logging.Formatter):
173     def format(self, record):
174         obj = {}
175         for key in [
176             "name",
177             "levelno",
178             "levelname",
179             "pathname",
180             "filename",
181             "module",
182             "lineno",
183             "funcName",
184             "created",
185             "msecs",
186             "relativeCreated",
187             "thread",
188             "threadName",
189             "process",
190             "processName",
191             "msg",
192             "args",
193             "exc_info",
194             "exc_text",
195         ]:
196             obj[key] = getattr(record, key)
197         obj["msg"] = obj["msg"] % obj["args"]
198         return json.dumps(obj)
199 
200 
201 def add_handler_stdout(logger: logging.Logger, level: int = logging.WARNING, fmt: str =
    DEFAULT_FMT) -> logging.StreamHandler:
202     logger.setLevel(logging.DEBUG)
203     #
204     handler = logging.StreamHandler(sys.stdout)
205     handler.setFormatter(logging.Formatter(fmt))
206     handler.setLevel(level)
207     logger.addHandler(handler)
208     return handler
209 
210 
211 def add_handler_file(
212     logger: logging.Logger, filename: str, maxMbytes: int, backupCount: int, level: int = logging
    .DEBUG, fmt: str = DEFAULT_FMT
213 ) -> logging.handlers.RotatingFileHandler:
214     if maxMbytes < 1:
215         raise ValueError(f"The parameter maxMbytes needs to be >= 1. maxMbytes={maxMbytes}")
216     if backupCount < 1:
217         raise ValueError(f"The parameter backupCount needs to be >= 1. backupCount={backupCount}"
    )
218     #
219     logger.setLevel(logging.DEBUG)
220     #
221     handler = logging.handlers.RotatingFileHandler(filename, maxBytes=maxMbytes * 1048576,
    backupCount=backupCount)
222     handler.setFormatter(logging.Formatter(fmt))
223     handler.setLevel(level)
224     logger.addHandler(handler)
225     return handler
```

```
226
227
228 def add_handler_socket(
229     logger: logging.Logger, level: int = logging.DEBUG, host: str = "localhost", port: int =
    19996
230 ) -> logging.handlers.SocketHandler:
231     logger.setLevel(logging.DEBUG)
232     #
233     handler = logging.handlers.SocketHandler(host, port)
234     handler.setLevel(level)
235     logger.addHandler(handler)
236     return handler
237
238
239 def default_logging_config(fmt=JOURNAL_FMT):
240     """You are able to configure logging by a config file including these line:
241
242     import logging
243
244
245     DEBUG = False
246     #
247     # Logging
248     #
249     APP_NAME = "<YOUR_APP_NAME>"
250     LOG_HOSTNAME = "loggy"           # When DEBUG is True
251     LOG_LEVEL = logging.INFO         # STDOUT logging
252     """
253     if DEBUG:
254         fmt = LONG_FMT
255     logger = logging.getLogger(ROOT_LOGGER_NAME)
256
257     add_handler_stdout(logger, LOG_LEVEL, fmt=fmt)
258     if DEBUG:
259         add_handler_socket(logger, host=LOG_HOSTNAME)
260     return logger.getChild("main")
261
262
263 def app_logging_config():
264     """
265     For details see comment in default_logging_config.
266     """
267     return default_logging_config(LONG_FMT)
268
269
270 class testSession(dict):
271     KEY_NAME = "name"
272     KEY_FAILED_TESTS = "number_of_failed_tests"
273     KEY_POSSIBLY_FAILED_TESTS = "number_of_possibly_failed_tests"
274     KEY_SUCCESS_TESTS = "number_of_successfull_tests"
275     KEY_ALL_TESTS = "number_of_tests"
276     KEY_EXEC_LVL = "testcase_execution_level"
277     KEY_EXEC_NAMES = "testcase_names"
278     KEY_LVL_NAMES = "level_names"
279     KEY_TESTCASELIST = "testcases"
280     KEY_UID_LIST = "uid_list_sorted"
281     #
282     DEFAULT_BASE_DATA = {
```

```
283         KEY_NAME: "Default Testsession name",
284         KEY_FAILED_TESTS: 0,
285         KEY_POSSIBLY_FAILED_TESTS: 0,
286         KEY_FAILED_TESTS: 0,
287         KEY_SUCCESS_TESTS: 0,
288         KEY_ALL_TESTS: 0,
289         KEY_EXEC_LVL: TCEL_FULL,
290         KEY_EXEC_NAMES: {"%d" % key: TCEL_NAMES[key] for key in TCEL_NAMES},
291     }
292 
293     def __init__(self, module_names=[], **kwargs):
294         dict.__init__(self, time_consumption=0.0)
295         self.__testcase__ = None
296         self.__set_base_data__(**kwargs)
297         self.__configure_logging__(module_names)
298 
299     def __set_base_data__(self, **kwargs):
300         for key in set([key for key in self.DEFAULT_BASE_DATA.keys()] + [key for key in kwargs.
    keys()]):
301             self[key] = kwargs.get(key, self.DEFAULT_BASE_DATA.get(key))
302         self[self.KEY_TESTCASELIST] = {}
303         self[self.KEY_UID_LIST] = []
304 
305     def __configure_logging__(self, module_names):
306         #
307         # Configure logging for testSession
308         #
309         logging_config = dict(
310             version=1,
311             formatters={
312                 "short": {
313                     "format": SHORT_FMT,
314                 },
315                 "long": {
316                     "format": LONG_FMT,
317                 },
318             },
319             handlers={
320                 "console": {
321                     "level": "DEBUG",
322                     "class": "logging.NullHandler",
323                     "formatter": "short",
324                 },
325                 "module_logs": {
326                     "level": "DEBUG",
327                     "class": "report.collectingHandler",
328                     "formatter": "short",
329                 },
330                 "testcase_logs": {
331                     "level": "DEBUG",
332                     "class": "report.collectingTestcaseHandler",
333                     "formatter": "short",
334                 },
335             },
336             loggers=self.__module_loggers__(module_names),
337         )
338         dictConfig(logging_config)
339 
340     def __module_loggers__(self, module_names):
341         rv = {}
342         rv["__tLogger__"] = dict(handlers=["console", "testcase_logs"], level="DEBUG", propagate=
    False)
```

```
343         for name in module_names + ["__mLogger__"]:
344             rv[name] = dict(handlers=["console", "module_logs"], level="DEBUG", propagate=False)
345         return rv
346 
347     def testCase(self, name, testcase_execution_level, test_method, *args, **kwargs):
348         if testcase_execution_level <= self[self.KEY_EXEC_LVL]:
349             sys.stdout.write("     %s..." % name[:75])
350             tLogger = logging.getLogger("__tLogger__")
351             tHandler = collectingTestcaseHandler()
352             if len(tHandler.MY_LOGS) > 0:
353                 raise AttributeError("Testcaselogger shall be empty after closing testcase!")
354             tLogger._log(logging.DEBUG, name, None, stacklevel=2)
355             if len(tHandler.MY_LOGS) != 1:
356                 raise AttributeError("Testcaselogger shall have only one entry for the main
    testcase (temporary)!")
357             self.__testcase__ = tHandler.get_logs()[0]
358             test_method(logging.getLogger("__tLogger__"), *args, **kwargs)
359             self._close_active_testcase()
360 
361     def _close_active_testcase(self):
362         if self._testcase is not None:
363             name = self._testcase.get("message")
364             #
365             # Add testcase
366             #
367             tch = collectingTestcaseHandler()
368             self._testcase["testcaseLogger"] = tch.get_logs()
369             if name in self[self.KEY_TESTCASELIST]:
370                 raise AttributeError("Testcase named %s already exists" % name)
371             self[self.KEY_TESTCASELIST][name] = self.__testcase__
372             self[self.KEY_UID_LIST].append(name)
373             #
374             # Adapt testcase data
375             #
376             self[self.KEY_TESTCASELIST][name]["levelno"] = 0
377             self[self.KEY_TESTCASELIST][name]["time_consumption"] = 0.0
378             for teststep in self[self.KEY_TESTCASELIST][name]["testcaseLogger"]:
379                 # store maximum level to testcase
380                 if teststep.get("levelno") > self[self.KEY_TESTCASELIST][name]["levelno"]:
381                     self[self.KEY_TESTCASELIST][name]["levelno"] = teststep.get("levelno")
382                     self[self.KEY_TESTCASELIST][name]["levelname"] = teststep.get("levelname")
383                 # store time consumption for teststep
384                 try:
385                     teststep["time_consumption"] = teststep["created"] - teststep["moduleLogger"
    ][-1]["created"]
386                 except IndexError:
387                     teststep["time_consumption"] = 0.0
388                 # Increment testcase time_comsumption
389             # Increment testcase counters
390             #
391             self[self.KEY_ALL_TESTS] += 1
392             if self[self.KEY_TESTCASELIST][name]["levelno"] <= logging.INFO:
393                 self[self.KEY_SUCCESS_TESTS] += 1
394                 sys.stdout.write("\033[92mSUCCESS\033[0m\n")
395             elif self[self.KEY_TESTCASELIST][name]["levelno"] >= logging.ERROR:
396                 self[self.KEY_FAILED_TESTS] += 1
397                 sys.stdout.write("\033[91mFAILED\033[0m\n")
398             else:
399                 self[self.KEY_POSSIBLY_FAILED_TESTS] += 1
400                 sys.stdout.write("\033[93mPOSSIBLY FAILED\033[0m\n")
401             # Set testcase time and time consumption
402             self[self.KEY_TESTCASELIST][name]["time_start"] = self.__testcase__["asctime"]
403             self[self.KEY_TESTCASELIST][name]["time_finished"] = teststep["asctime"]
404             self[self.KEY_TESTCASELIST][name]["time_consumption"] = teststep["created"] - self.
    _testcase["created"]
405             # Set testcase time consumption
406             self["time_consumption"] += self[self.KEY_TESTCASELIST][name]["time_consumption"]
407         self.__testcase__ = None
```