Unittest for `report`

May 8, 2022

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	6d66b6f5e942506500a4c4a1e2e6f8e2
Dependencies	

1.2 Unittest Information

Unittest Information	
Version	2c03d3eba161a9fb0dbf0594fbda3965
Testruns with	python 3.10.4 (final)

1.3 Test System Information

System Information	
Architecture	64bit
Distribution	Ubuntu 22.04 Jammy Jellyfish
Hostname	ahorn
Kernel	5.15.0-27-lowlatency (#28-Ubuntu SMP PREEMPT Tue Apr 19 15:27:08 UTC 2022)
Machine	x86_64
Path	/home/dirk/my_repositories/unittest/report
System	Linux
Username	dirk

2 Statistic

2.1 Test-Statistic for testrun with python 3.10.4 (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	0.012s

2.2 Coverage Statistic

Module- or Filename	Line-Coverage	Branch-Coverage
report	81.5%	60.8%
report.__init__.py	81.5%	

3 Tested Requirements

3.1 General Information

Many Methods and Classes in this Module are used for the unittest itself. Others are configuring python logging, which is also used for the unittest itself. Therefore, the unittest for this Module is limited. Also the coverage information is not only reached by the testcases, cause the Module is used by the unittest itself.

3.2 collectingHandler

3.2.1 Store log records (collectingHandler)

Description
Description 1.

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

Testrun:	python 3.10.4 (final)
Caller:	/home/dirk/my_repositories/unittest/report/unittest/src/tests/__init__.py (19)
Start-Time:	2022-05-08 21:03:46,971
Finished-Time:	2022-05-08 21:03:46,973
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)

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

Testrun:	python 3.10.4 (final)
Caller:	/home/dirk/my_repositories/unittest/report/unittest/src/tests/__init__.py (20)
Start-Time:	2022-05-08 21:03:46,974
Finished-Time:	2022-05-08 21:03:46,976
Time-Consumption	0.002s

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)

Testresult

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

Testrun:	python 3.10.4 (final)
Caller:	/home/dirk/my_repositories/unittest/report/unittest/src/tests/__init__.py (21)
Start-Time:	2022-05-08 21:03:46,976
Finished-Time:	2022-05-08 21:03:46,978
Time-Consumption	0.002s

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)

Testresult

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

Testrun:	python 3.10.4 (final)
Caller:	/home/dirk/my_repositories/unittest/report/unittest/src/tests/__init__.py (22)
Start-Time:	2022-05-08 21:03:46,978
Finished-Time:	2022-05-08 21:03:46,981

Time-Consumption	0.002s

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

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:	python 3.10.4 (final)
Caller:	/home/dirk/my_repositories/unittest/report/unittest/src/tests/__init__.py (23)
Start-Time:	2022-05-08 21:03:46,981
Finished-Time:	2022-05-08 21:03:46,983
Time-Consumption	0.002s

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 3.10.4 (final)

A.1 Tests with status Info (5)

A.1.1 Store log records (collectingHandler)

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)

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): [ 49, 134, 221, 309, 398, 486, 571
↪  ] (<class 'list'>)
Expectation (Indexlist of log entries in stringrepresentation): result = [ 49, 134, 221, 309,
↪  398, 486, 571 ] (<class 'list'>)
Result (Submitted value number 1): 49 (<class 'int'>)
Expectation (Submitted value number 1): result = 49 (<class 'int'>)
Submitted value number 1 is correct (Content 49 and Type is <class 'int'>).
Result (Submitted value number 2): 134 (<class 'int'>)
Expectation (Submitted value number 2): result = 134 (<class 'int'>)
Submitted value number 2 is correct (Content 134 and Type is <class 'int'>).
Result (Submitted value number 3): 221 (<class 'int'>)
Expectation (Submitted value number 3): result = 221 (<class 'int'>)
Submitted value number 3 is correct (Content 221 and Type is <class 'int'>).
Result (Submitted value number 4): 309 (<class 'int'>)
Expectation (Submitted value number 4): result = 309 (<class 'int'>)
Submitted value number 4 is correct (Content 309 and Type is <class 'int'>).
Result (Submitted value number 5): 398 (<class 'int'>)
Expectation (Submitted value number 5): result = 398 (<class 'int'>)
```

```
Submitted value number 5 is correct (Content 398 and Type is <class 'int'>).
Result (Submitted value number 6): 486 (<class 'int'>)
Expectation (Submitted value number 6): result = 486 (<class 'int'>)
Submitted value number 6 is correct (Content 486 and Type is <class 'int'>).
Result (Submitted value number 7): 571 (<class 'int'>)
Expectation (Submitted value number 7): result = 571 (<class 'int'>)
Submitted value number 7 is correct (Content 571 and Type is <class 'int'>).
```

A.1.3 Store log records (collectingRingHandler)

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)

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): [ 51, 139, 228, 316, 401 ] (<class
↪  'list'>)
Expectation (Indexlist of log entries in stringrepresentation): result = [ 51, 139, 228, 316,
↪  401 ] (<class 'list'>)
Result (Submitted value number 1): 51 (<class 'int'>)
Expectation (Submitted value number 1): result = 51 (<class 'int'>)
Submitted value number 1 is correct (Content 51 and Type is <class 'int'>).
Result (Submitted value number 2): 139 (<class 'int'>)
Expectation (Submitted value number 2): result = 139 (<class 'int'>)
Submitted value number 2 is correct (Content 139 and Type is <class 'int'>).
Result (Submitted value number 3): 228 (<class 'int'>)
Expectation (Submitted value number 3): result = 228 (<class 'int'>)
Submitted value number 3 is correct (Content 228 and Type is <class 'int'>).
Result (Submitted value number 4): 316 (<class 'int'>)
Expectation (Submitted value number 4): result = 316 (<class 'int'>)
Submitted value number 4 is correct (Content 316 and Type is <class 'int'>).
Result (Submitted value number 5): 401 (<class 'int'>)
Expectation (Submitted value number 5): result = 401 (<class 'int'>)
Submitted value number 5 is correct (Content 401 and Type is <class 'int'>).
```

A.1.5 Number of stored logs in the RingHandler

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 81.5%
The branch coverage for report was 60.8%

B.1.1 report.__init__.py

The line coverage for report.__init__.py was 81.5%
The branch coverage for report.__init__.py was 60.8%

```
#!/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 os
import sys

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"""

SHORT_FMT = "%(asctime)s: %(name)s - %(levelname)s - %(message)s"
""" A short formatter including the most important information"""
LONG_FMT = """~~~~(%(levelname)-10s)
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

File "%(pathname)s", line %(lineno)d, in %(funcName)s
%(asctime)s: %(name)s- %(message)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:
            rv.append(self.MY_LOGS.pop(0))
        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])
```

```
102
103     def __len__(self):
104         return len(self.MY_LOGS)
105
106     def __str__(self):
107         return self.get_str(self.MY_LOGS)
108
109
110 class collectingRingHandler(collectingHandler):
111     MY_LOGS = collections.deque([], 10)
112
113     def __init__(self, max_logs=None):
114         collectingHandler.__init__(self)
115         if max_logs is not None and max_logs != self.MY_LOGS.maxlen:
116             self.MY_LOGS.__init__(list(self.MY_LOGS), max_logs)
117
118     def make_independent(self):
119         self.MY_LOGS = collections.deque([], self.MY_LOGS.maxlen)
120
121     def get_logs(self):
122         return list(self.MY_LOGS)
123
124
125 TCEL_SINGLE = 0
126 """Testcase level (smoke), this is just a rough test for the main functionality"""
127 TCEL_SMOKE = 10
128 """Testcase level (smoke), this is just a rough test for the main functionality"""
129 TCEL_SHORT = 50
130 """Testcase level (short), this is a short test for an extended functionality"""
131 TCEL_FULL = 90
132 """Testcase level (full), this is a complete test for the full functionality"""
133 TCEL_NAMES = {
134     TCEL_SINGLE: 'Single Test',
135     TCEL_SMOKE: 'Smoke Test (Minumum subset)',
136     TCEL_SHORT: 'Short Test (Subset)',
137     TCEL_FULL: 'Full Test (all defined tests)'
138 }
139 """Dictionary for resolving the test case levels (TCL) to a (human readable) name"""
140
141 TCEL_REVERSE_NAMED = {
142     'short': TCEL_SHORT,
143     'smoke': TCEL_SMOKE,
144     'single': TCEL_SINGLE,
145     'full': TCEL_FULL,
146 }
147 """Dictionary for resolving named test case levels (TCL) to test case level number"""
148
149
150 class collectingTestcaseHandler(collectingHandler):
151     MY_LOGS = []
152
153     def emit(self, record):
154         self.format(record)
155         self.MY_LOGS.append(record.__dict__)
156         self.MY_LOGS[-1]['moduleLogger'] = collectingHandler().get_logs()
157
158
159 class JsonFormatter(logging.Formatter):
160     def format(self, record):
```

```
161         obj = {}
162         for key in ["name", "levelno", "levelname", "pathname", "filename", "module", "lineno", "
    funcName", "created", "msecs", "relativeCreated", "thread", "threadName", "process", "
    processName", "msg", "args", "exc_info", "exc_text"]:
163             obj[key] = getattr(record, key)
164         obj["msg"] = obj["msg"] % obj["args"]
165         return json.dumps(obj)
166 
167 
168 def appLoggingConfigure(basepath, target, log_name_lvl=[], fmt=SHORT_FMT, ring_logs=None, host=
    None, port=None):
169     target_handlers = ['main', ]
170     # define handler
171     #
172     if target == 'stdout':
173         handler = dict(main={
174             'level': 'DEBUG',
175             'formatter': 'format',
176             'class': 'logging.StreamHandler',
177             'stream': 'ext://sys.stdout',
178         })
179     elif target == 'logfile':
180         handler = dict(main={
181             'level': 'DEBUG',
182             'formatter': 'json',
183             'class': 'logging.handlers.RotatingFileHandler',
184             'filename': os.path.join(basepath, 'messages.log'),
185             'mode': 'a',
186             'maxBytes': 10485760,
187             'backupCount': 7
188         })
189     else:
190         handler = dict(main={
191             'level': 'DEBUG',
192             'formatter': 'json',
193             'class': 'logging.NullHandler',
194         })
195     if host is not None and port is not None:
196         target_handlers.append('socket')
197         handler['socket']={
198             'level': 'DEBUG',
199             'class': 'logging.handlers.SocketHandler',
200             'host': host,
201             'port': port
202         }
203     if ring_logs is not None:
204         target_handlers.append('ring')
205         handler['ring'] = {
206             'class': 'report.collectingRingHandler',
207             'max_logs': ring_logs,
208         }
209     # define loggers
210     #
211     loggers = {}
212     for name, lvl in log_name_lvl:
213         loggers[name] = {
214             'handlers': target_handlers,
215             'level': lvl,
216             'propagate': False
217         }
218     # configure logging
219     #
```

```
220     dictConfig(dict(
221         version=1,
222         formatters={
223             'json': {
224                 '()': JsonFormatter
225             },
226             'long': {
227                 'format': LONG_FMT
228             },
229             'format': {
230                 'format': fmt,
231             },
232         },
233         handlers=handler,
234         loggers=loggers,
235     ))
236 
237 
238 def stdoutLoggingConfigure(log_name_lvl=[], fmt=SHORT_FMT):
239     appLoggingConfigure(None, 'stdout', log_name_lvl=log_name_lvl, fmt=fmt)
240 
241 
242 class testSession(dict):
243     KEY_NAME = 'name'
244     KEY_FAILED_TESTS = 'number_of_failed_tests'
245     KEY_POSSIBLY_FAILED_TESTS = 'number_of_possibly_failed_tests'
246     KEY_SUCCESS_TESTS = 'number_of_successfull_tests'
247     KEY_ALL_TESTS = 'number_of_tests'
248     KEY_EXEC_LVL = 'testcase_execution_level'
249     KEY_EXEC_NAMES = 'testcase_names'
250     KEY_LVL_NAMES = 'level_names'
251     KEY_TESTCASELIST = 'testcases'
252     KEY_UID_LIST = 'uid_list_sorted'
253     #
254     DEFAULT_BASE_DATA = {
255         KEY_NAME: 'Default Testsession name',
256         KEY_FAILED_TESTS: 0,
257         KEY_POSSIBLY_FAILED_TESTS: 0,
258         KEY_FAILED_TESTS: 0,
259         KEY_SUCCESS_TESTS: 0,
260         KEY_ALL_TESTS: 0,
261         KEY_EXEC_LVL: TCEL_FULL,
262         KEY_EXEC_NAMES: TCEL_NAMES,
263     }
264 
265     def __init__(self, module_names=[], **kwargs):
266         dict.__init__(self, time_consumption=0.)
267         self.__testcase__ = None
268         self.__set_base_data__(**kwargs)
269         self.__configure_logging__(module_names)
270 
271     def __set_base_data__(self, **kwargs):
272         for key in set([key for key in self.DEFAULT_BASE_DATA.keys()] + [key for key in kwargs.
    keys()]):
273             self[key] = kwargs.get(key, self.DEFAULT_BASE_DATA.get(key))
274         self[self.KEY_TESTCASELIST] = {}
275         self[self.KEY_UID_LIST] = []
276 
277     def __configure_logging__(self, module_names):
278         #
279         # Configure logging for testSession
280         #
```

```
281         logging_config = dict(
282             version=1,
283             formatters={
284                 'short': {
285                     'format': SHORT_FMT,
286                 },
287                 'long': {
288                     'format': LONG_FMT,
289                 },
290             },
291             handlers={
292                 'console': {
293                     'level': 'DEBUG',
294                     'class': 'logging.NullHandler',
295                     'formatter': 'short',
296                 },
297                 'module_logs': {
298                     'level': 'DEBUG',
299                     'class': 'report.collectingHandler',
300                     'formatter': 'short',
301                 },
302                 'testcase_logs': {
303                     'level': 'DEBUG',
304                     'class': 'report.collectingTestcaseHandler',
305                     'formatter': 'short',
306                 },
307             },
308             loggers=self.__module_loggers__(module_names),
309         )
310         dictConfig(logging_config)
311 
312     def __module_loggers__(self, module_names):
313         rv = {}
314         rv['__tLogger__'] = dict(handlers=['console', 'testcase_logs'], level='DEBUG', propagate=
    False)
315         for name in module_names + ['__mLogger__']:
316             rv[name] = dict(handlers=['console', 'module_logs'], level='DEBUG', propagate=False)
317         return rv
318 
319     def testCase(self, name, testcase_execution_level, test_method, *args, **kwargs):
320         if testcase_execution_level <= self[self.KEY_EXEC_LVL]:
321             sys.stdout.write('     %s ... ' % name[:75])
322             tLogger = logging.getLogger('__tLogger__')
323             tHandler = collectingTestcaseHandler()
324             if len(tHandler.MY_LOGS) > 0:
325                 raise AttributeError("Testcaselogger shall be empty after closing testcase!")
326             tLogger._log(logging.DEBUG, name, None)
327             if len(tHandler.MY_LOGS) != 1:
328                 raise AttributeError("Testcaselogger shall have only one entry for the main
    testcase (temporary)!")
329             self.__testcase__ = tHandler.get_logs()[0]
330             test_method(logging.getLogger('__tLogger__'), *args, **kwargs)
331             self.__close_active_testcase__()
332 
333     def __close_active_testcase__(self):
334         if self.__testcase__ is not None:
335             name = self.__testcase__.get('message')
336             #
337             # Add testcase
338             #
339             tch = collectingTestcaseHandler()
340             self.__testcase__['testcaseLogger'] = tch.get_logs()
```

```
341             if name in self[self.KEY_TESTCASELIST]:
342                 raise AttributeError("Testcase named %s already exists" % name)
343             self[self.KEY_TESTCASELIST][name] = self.__testcase__
344             self[self.KEY_UID_LIST].append(name)
345             #
346             # Adapt testcase data
347             #
348             self[self.KEY_TESTCASELIST][name]['levelno'] = 0
349             self[self.KEY_TESTCASELIST][name]['time_consumption'] = 0.
350             for teststep in self[self.KEY_TESTCASELIST][name]['testcaseLogger']:
351                 # store maximum level to testcase
352                 if teststep.get('levelno') > self[self.KEY_TESTCASELIST][name]['levelno']:
353                     self[self.KEY_TESTCASELIST][name]['levelno'] = teststep.get('levelno')
354                     self[self.KEY_TESTCASELIST][name]['levelname'] = teststep.get('levelname')
355                 # store time_consumption for teststep
356                 try:
357                     teststep['time_consumption'] = teststep['created'] - teststep['moduleLogger'
    ][-1]['created']
358                 except IndexError:
359                     teststep['time_consumption'] = 0.
360                 # Increment testcase time_comsumption
361             # Increment testcase counters
362             #
363             self[self.KEY_ALL_TESTS] += 1
364             if self[self.KEY_TESTCASELIST][name]['levelno'] <= logging.INFO:
365                 self[self.KEY_SUCCESS_TESTS] += 1
366                 sys.stdout.write('\033[92mSUCCESS\033[0m\n')
367             elif self[self.KEY_TESTCASELIST][name]['levelno'] >= logging.ERROR:
368                 self[self.KEY_FAILED_TESTS] += 1
369                 sys.stdout.write('\033[91mFAILED\033[0m\n')
370             else:
371                 self[self.KEY_POSSIBLY_FAILED_TESTS] += 1
372                 sys.stdout.write('\033[93mPOSSIBLY FAILED\033[0m\n')
373             # Set testcase time and time_consumption
374             self[self.KEY_TESTCASELIST][name]['time_start'] = self.__testcase__['asctime']
375             self[self.KEY_TESTCASELIST][name]['time_finished'] = teststep['asctime']
376             self[self.KEY_TESTCASELIST][name]['time_consumption'] = teststep['created'] - self.
    __testcase__['created']
377             # Set testcase time consumption
378             self['time_consumption'] += self[self.KEY_TESTCASELIST][name]['time_consumption']
379         self.__testcase__ = None
```