Unittest for `report`

January 28, 2020

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	python2, python3
Version	05cf9f57aa192a511c9cbe3ee2ec43fc
Dependencies	

1.2 Unittest Information

Unittest Information	
Version	2c03d3eba161a9fb0dbf0594fbda3965
Testruns with	python 2.7.17 (final), python 3.6.9 (final)

1.3 Test System Information

System Information	
Architecture	64bit
Distribution	LinuxMint 19.3 tricia
Hostname	ahorn
Kernel	5.3.0-26-generic (#28 18.04.1-Ubuntu SMP Wed Dec 18 16:40:14 UTC 2019)
Machine	x86_64
Path	/user_data/data/dirk/prj/unittest/report/unittest
System	Linux
Username	dirk

2 Statistic

2.1 Test-Statistic for testrun with python 2.7.17 (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	0.015s

2.2 Test-Statistic for testrun with python 3.6.9 (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

2.3 Coverage Statistic

Module- or Filename	Line-Coverage	Branch-Coverage
report	82.3%	51.2%
report.__init__.py	82.3%	

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 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/unittest/report/unittest/src/tests/__init__.py (19)
Start-Time:	2020-01-28 20:37:38,676
Finished-Time:	2020-01-28 20:37:38,680
Time-Consumption	0.005s

Testsummary:

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

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/unittest/report/unittest/src/tests/__init__.py (19)

Start-Time:	2020-01-28 20:37:39,125
Finished-Time:	2020-01-28 20:37:39,128
Time-Consumption	0.002s

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 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/unittest/report/unittest/src/tests/__init__.py (20)
Start-Time:	2020-01-28 20:37:38,681
Finished-Time:	2020-01-28 20:37:38,684
Time-Consumption	0.004s

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.

Testresult

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/unittest/report/unittest/src/tests/__init__.py (20)
Start-Time:	2020-01-28 20:37:39,128
Finished-Time:	2020-01-28 20:37:39,132
Time-Consumption	0.004s

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 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/unittest/report/unittest/src/tests/__init__.py (21)
Start-Time:	2020-01-28 20:37:38,685
Finished-Time:	2020-01-28 20:37:38,687
Time-Consumption	0.002s

Testsummary:

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

Testresult

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/unittest/report/unittest/src/tests/__init__.py (21)
Start-Time:	2020-01-28 20:37:39,132
Finished-Time:	2020-01-28 20:37:39,135
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)

Testresult

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

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/unittest/report/unittest/src/tests/__init__.py (22)
Start-Time:	2020-01-28 20:37:38,687
Finished-Time:	2020-01-28 20:37:38,689
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.

Testresult

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/unittest/report/unittest/src/tests/__init__.py (22)
Start-Time:	2020-01-28 20:37:39,135
Finished-Time:	2020-01-28 20:37:39,136
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 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/unittest/report/unittest/src/tests/__init__.py (23)
Start-Time:	2020-01-28 20:37:38,689

Finished-Time:	2020-01-28 20:37:38,692
Time-Consumption	0.002s

Testsummary:

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

Testresult

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/unittest/report/unittest/src/tests/__init__.py (23)
Start-Time:	2020-01-28 20:37:39,136
Finished-Time:	2020-01-28 20:37:39,138
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 2.7.17 (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.
Passing "Log entry number 2 with level INFO." to collectingHandler.
Passing "Log entry number 3 with level WARNING." to collectingHandler.
Passing "Log entry number 4 with level ERROR." to collectingHandler.
Passing "Log entry number 5 with level CRITICAL." to collectingHandler.
Passing "Log entry number 6 with level INFO." to collectingHandler.
Passing "Log entry number 7 with level ERROR." to collectingHandler.
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
Expectation (Logged information): result = [ 'Log entry number %d with level %s.', 40,
↪  'test_helpers.py', 7 ] (<type '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.
Passing "Log entry number 2 with level INFO." to collectingHandler.
Passing "Log entry number 3 with level WARNING." to collectingHandler.
Passing "Log entry number 4 with level ERROR." to collectingHandler.
Passing "Log entry number 5 with level CRITICAL." to collectingHandler.
Passing "Log entry number 6 with level INFO." to collectingHandler.
Passing "Log entry number 7 with level ERROR." to collectingHandler.
```

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
↪  ] (<type 'list'>)
Expectation (Indexlist of log entries in stringrepresentation): result = [ 49, 134, 221, 309,
↪  398, 486, 571 ] (<type 'list'>)
Result (Submitted value number 1): 49 (<type 'int'>)
Expectation (Submitted value number 1): result = 49 (<type 'int'>)
Submitted value number 1 is correct (Content 49 and Type is <type 'int'>).
Result (Submitted value number 2): 134 (<type 'int'>)
Expectation (Submitted value number 2): result = 134 (<type 'int'>)
Submitted value number 2 is correct (Content 134 and Type is <type 'int'>).
Result (Submitted value number 3): 221 (<type 'int'>)
Expectation (Submitted value number 3): result = 221 (<type 'int'>)
Submitted value number 3 is correct (Content 221 and Type is <type 'int'>).
Result (Submitted value number 4): 309 (<type 'int'>)
Expectation (Submitted value number 4): result = 309 (<type 'int'>)
Submitted value number 4 is correct (Content 309 and Type is <type 'int'>).
Result (Submitted value number 5): 398 (<type 'int'>)
Expectation (Submitted value number 5): result = 398 (<type 'int'>)
Submitted value number 5 is correct (Content 398 and Type is <type 'int'>).
Result (Submitted value number 6): 486 (<type 'int'>)
Expectation (Submitted value number 6): result = 486 (<type 'int'>)
Submitted value number 6 is correct (Content 486 and Type is <type 'int'>).
Result (Submitted value number 7): 571 (<type 'int'>)
Expectation (Submitted value number 7): result = 571 (<type 'int'>)
Submitted value number 7 is correct (Content 571 and Type is <type '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.
Passing "Log entry number 2 with level INFO." to collectingRingHandler.
Passing "Log entry number 3 with level WARNING." to collectingRingHandler.
Passing "Log entry number 4 with level ERROR." to collectingRingHandler.
Passing "Log entry number 5 with level CRITICAL." to collectingRingHandler.
Passing "Log entry number 6 with level INFO." to collectingRingHandler.
Passing "Log entry number 7 with level ERROR." to collectingRingHandler.
```

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

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

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

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

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

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

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

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

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

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

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

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

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

```
Expectation (Logged information): result = [ 'Log entry number %d with level %s.', 40,
↪  'test_helpers.py', 7 ] (<type '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.
Passing "Log entry number 2 with level INFO." to collectingRingHandler.
Passing "Log entry number 3 with level WARNING." to collectingRingHandler.
Passing "Log entry number 4 with level ERROR." to collectingRingHandler.
Passing "Log entry number 5 with level CRITICAL." to collectingRingHandler.
Passing "Log entry number 6 with level INFO." to collectingRingHandler.
Passing "Log entry number 7 with level ERROR." to collectingRingHandler.
```

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 ] (<type
↪  'list'>)
Expectation (Indexlist of log entries in stringrepresentation): result = [ 51, 139, 228, 316,
↪  401 ] (<type 'list'>)
Result (Submitted value number 1): 51 (<type 'int'>)
Expectation (Submitted value number 1): result = 51 (<type 'int'>)
Submitted value number 1 is correct (Content 51 and Type is <type 'int'>).
Result (Submitted value number 2): 139 (<type 'int'>)
Expectation (Submitted value number 2): result = 139 (<type 'int'>)
Submitted value number 2 is correct (Content 139 and Type is <type 'int'>).
Result (Submitted value number 3): 228 (<type 'int'>)
Expectation (Submitted value number 3): result = 228 (<type 'int'>)
Submitted value number 3 is correct (Content 228 and Type is <type 'int'>).
Result (Submitted value number 4): 316 (<type 'int'>)
Expectation (Submitted value number 4): result = 316 (<type 'int'>)
Submitted value number 4 is correct (Content 316 and Type is <type 'int'>).
Result (Submitted value number 5): 401 (<type 'int'>)
Expectation (Submitted value number 5): result = 401 (<type 'int'>)
Submitted value number 5 is correct (Content 401 and Type is <type '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.
Passing "Log entry number 2 with level INFO." to collectingRingHandler.
Passing "Log entry number 3 with level WARNING." to collectingRingHandler.
Passing "Log entry number 4 with level ERROR." to collectingRingHandler.
Passing "Log entry number 5 with level CRITICAL." to collectingRingHandler.
Passing "Log entry number 6 with level INFO." to collectingRingHandler.
Passing "Log entry number 7 with level ERROR." to collectingRingHandler.
```

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

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

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

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

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

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

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

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

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

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

B Trace for testrun with python 3.6.9 (final)

B.1 Tests with status Info (5)

B.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.
Passing "Log entry number 2 with level INFO." to collectingHandler.
Passing "Log entry number 3 with level WARNING." to collectingHandler.
Passing "Log entry number 4 with level ERROR." to collectingHandler.
Passing "Log entry number 5 with level CRITICAL." to collectingHandler.
Passing "Log entry number 6 with level INFO." to collectingHandler.
Passing "Log entry number 7 with level ERROR." to collectingHandler.
```

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'>)
```

B.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.
Passing "Log entry number 2 with level INFO." to collectingHandler.
Passing "Log entry number 3 with level WARNING." to collectingHandler.
Passing "Log entry number 4 with level ERROR." to collectingHandler.
Passing "Log entry number 5 with level CRITICAL." to collectingHandler.
Passing "Log entry number 6 with level INFO." to collectingHandler.
Passing "Log entry number 7 with level ERROR." to collectingHandler.
```

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'>).
```

B.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.
Passing "Log entry number 2 with level INFO." to collectingRingHandler.
Passing "Log entry number 3 with level WARNING." to collectingRingHandler.
Passing "Log entry number 4 with level ERROR." to collectingRingHandler.
Passing "Log entry number 5 with level CRITICAL." to collectingRingHandler.
Passing "Log entry number 6 with level INFO." to collectingRingHandler.
Passing "Log entry number 7 with level ERROR." to collectingRingHandler.
```

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'>)
```

B.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.
Passing "Log entry number 2 with level INFO." to collectingRingHandler.
Passing "Log entry number 3 with level WARNING." to collectingRingHandler.
Passing "Log entry number 4 with level ERROR." to collectingRingHandler.
Passing "Log entry number 5 with level CRITICAL." to collectingRingHandler.
Passing "Log entry number 6 with level INFO." to collectingRingHandler.
Passing "Log entry number 7 with level ERROR." to collectingRingHandler.
```

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'>).
```

B.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.
Passing "Log entry number 2 with level INFO." to collectingRingHandler.
Passing "Log entry number 3 with level WARNING." to collectingRingHandler.
Passing "Log entry number 4 with level ERROR." to collectingRingHandler.
Passing "Log entry number 5 with level CRITICAL." to collectingRingHandler.
Passing "Log entry number 6 with level INFO." to collectingRingHandler.
Passing "Log entry number 7 with level ERROR." to collectingRingHandler.
```

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'>)
```

C Test-Coverage

C.1 report

The line coverage for report was 82.3%
The branch coverage for report was 51.2%

C.1.1 report.__init__.py

The line coverage for report.__init__.py was 82.3%
The branch coverage for report.__init__.py was 51.2%

```
#!/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 logging
from logging.config import dictConfig
import os
import sys

logger_name = 'REPORT'
logger = logging.getLogger(logger_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__ = (2, 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])

    def __len__(self):
        return len(self.MY_LOGS)

    def __str__(self):
        return self.get_str(self.MY_LOGS)


class collectingRingHandler(collectingHandler):
    MY_LOGS = collections.deque([], 10)

    def __init__(self, max_logs=None):
        collectingHandler.__init__(self)
        if max_logs is not None and max_logs != self.MY_LOGS.maxlen:
            self.MY_LOGS.__init__(list(self.MY_LOGS), max_logs)

    def make_independent(self):
        self.MY_LOGS = collections.deque([], self.MY_LOGS.maxlen)
```

```
116
117     def get_logs(self):
118         return list(self.MY_LOGS)
119
120
121 TCEL_SINGLE = 0
122 """Testcase level (smoke), this is just a rough test for the main functionality"""
123 TCEL_SMOKE = 10
124 """Testcase level (smoke), this is just a rough test for the main functionality"""
125 TCEL_SHORT = 50
126 """Testcase level (short), this is a short test for an extended functionality"""
127 TCEL_FULL = 90
128 """Testcase level (full), this is a complete test for the full functionality"""
129 TCEL_NAMES = {
130     TCEL_SINGLE: 'Single Test',
131     TCEL_SMOKE: 'Smoke Test (Minumum subset)',
132     TCEL_SHORT: 'Short Test (Subset)',
133     TCEL_FULL: 'Full Test (all defined tests)'
134 }
135 """Dictionary for resolving the test case levels (TCL) to a (human readable) name"""
136
137 TCEL_REVERSE_NAMED = {
138     'short': TCEL_SHORT,
139     'smoke': TCEL_SMOKE,
140     'single': TCEL_SINGLE,
141     'full': TCEL_FULL,
142 }
143 """Dictionary for resolving named test case levels (TCL) to test case level number"""
144
145
146 class collectingTestcaseHandler(collectingHandler):
147     MY_LOGS = []
148
149     def emit(self, record):
150         self.format(record)
151         self.MY_LOGS.append(record.__dict__)
152         self.MY_LOGS[-1]['moduleLogger'] = collectingHandler().get_logs()
153
154 def appLoggingConfigure(basepath, target, log_name_lvl=[], fmt=SHORT_FMT, ring_logs=None):
155     target_handlers = ['main', ]
156     if basepath is not None:
157         target_handlers.append('logwarn')
158     # define handler
159     #
160     if target == 'stdout':
161         handler = dict(main={
162             'level': 'DEBUG',
163             'formatter': 'format',
164             'class': 'logging.StreamHandler',
165             'stream': 'ext://sys.stdout',
166         })
167     elif target == 'logfile':
168         handler = dict(main={
169             'level': 'DEBUG',
170             'formatter': 'format',
171             'class': 'logging.handlers.RotatingFileHandler',
172             'filename': os.path.join(basepath, 'messages.log'),
173             'mode': 'a',
174             'maxBytes': 10485760,
175             'backupCount': 7
176         })
177     else:
178         handler = dict(my_handler={
```

```
179             'level': 'DEBUG',
180             'formatter': 'my_format',
181             'class': 'logging.NullHandler',
182         })
183     if ring_logs is not None:
184         target_handlers.append('ring')
185         handler['ring'] = {
186             'class': 'report.collectingRingHandler',
187             'max_logs': ring_logs,
188         }
189     if basepath is not None:
190         handler['logwarn'] = {
191             'level': 'WARNING',
192             'formatter': 'long',
193             'class': 'logging.handlers.RotatingFileHandler',
194             'filename': os.path.join(basepath, 'messages.warn'),
195             'mode': 'a',
196             'maxBytes': 10485760,
197              'backupCount': 2
198         }
199     # define loggers
200     #
201     loggers = {}
202     for name, lvl in log_name_lvl:
203         loggers[name] = {
204             'handlers': target_handlers,
205             'level': lvl,
206             'propagate': False
207         }
208     # configure logging
209     #
210     dictConfig(dict(
211         version=1,
212         formatters={
213             'long': {
214                 'format': LONG_FMT
215             },
216             'format': {
217                 'format': fmt,
218             },
219         },
220         handlers=handler,
221         loggers=loggers,
222     ))
223
224
225 def stdoutLoggingConfigure(log_name_lvl=[], fmt=SHORT_FMT):
226     appLoggingConfigure(None, 'stdout', log_name_lvl=log_name_lvl, fmt=fmt)
227
228
229 class testSession(dict):
230     KEY_NAME = 'name'
231     KEY_FAILED_TESTS = 'number_of_failed_tests'
232     KEY_POSSIBLY_FAILED_TESTS = 'number_of_possibly_failed_tests'
233     KEY_SUCCESS_TESTS = 'number_of_successfull_tests'
234     KEY_ALL_TESTS = 'number_of_tests'
235     KEY_EXEC_LVL = 'testcase_execution_level'
236     KEY_EXEC_NAMES = 'testcase_names'
237     KEY_LVL_NAMES = 'level_names'
238     KEY_TESTCASELIST = 'testcases'
239     KEY_UID_LIST = 'uid_list_sorted'
240     #
```

```
241     DEFAULT_BASE_DATA = {
242         KEY_NAME: 'Default Testsession name',
243         KEY_FAILED_TESTS: 0,
244         KEY_POSSIBLY_FAILED_TESTS: 0,
245         KEY_FAILED_TESTS: 0,
246         KEY_SUCCESS_TESTS: 0,
247         KEY_ALL_TESTS: 0,
248         KEY_EXEC_LVL: TCEL_FULL,
249         KEY_EXEC_NAMES: TCEL_NAMES,
250     }
251 
252     def __init__(self, module_names=[], **kwargs):
253         dict.__init__(self, time_consumption=0.)
254         self.__testcase__ = None
255         self.__set_base_data__(**kwargs)
256         self.__configure_logging__(module_names)
257 
258     def __set_base_data__(self, **kwargs):
259         for key in set([key for key in self.DEFAULT_BASE_DATA.keys()] + [key for key in kwargs.
    keys()]):
260             self[key] = kwargs.get(key, self.DEFAULT_BASE_DATA.get(key))
261         self[self.KEY_TESTCASELIST] = {}
262         self[self.KEY_UID_LIST] = []
263 
264     def __configure_logging__(self, module_names):
265         #
266         # Configure logging for testSession
267         #
268         logging_config = dict(
269             version=1,
270             formatters={
271                 'short': {
272                     'format': SHORT_FMT,
273                 },
274                 'long': {
275                     'format': LONG_FMT,
276                 },
277             },
278             handlers={
279                 'console': {
280                     'level': 'DEBUG',
281                     'class': 'logging.NullHandler',
282                     'formatter': 'short',
283                 },
284                 'module_logs': {
285                     'level': 'DEBUG',
286                     'class': 'report.collectingHandler',
287                     'formatter': 'short',
288                 },
289                 'testcase_logs': {
290                     'level': 'DEBUG',
291                     'class': 'report.collectingTestcaseHandler',
292                     'formatter': 'short',
293                 },
294             },
295             loggers=self.__module_loggers__(module_names),
296         )
297         dictConfig(logging_config)
298 
```

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