Unittest for `task`

January 7, 2021

Contents

1 Test Information

1.1 Test Candidate Information

The Module task is designed to help with task issues like periodic tasks, delayed tasks, queues, threaded queues and crontabs. For more Information read the documentation.

Library Information	
Name	task
State	Released
Supported Interpreters	python2, python3
Version	85fa349c0c871c15abf76947efad0d1b
Dependencies	

1.2 Unittest Information

Unittest Information	
Version	bf12903e8541ad442a6d670b0e5f89b9
Testruns with	python 2.7.18 (final), python 3.8.5 (final)

1.3 Test System Information

System Information	
Architecture	64bit
Distribution	Linux Mint 20 ulyana
Hostname	ahorn
Kernel	5.4.0-59-generic (#65-Ubuntu SMP Thu Dec 10 12:01:51 UTC 2020)
Machine	x86_64
Path	/user_data/data/dirk/prj/unittest/task/unittest
System	Linux
Username	dirk

2 Statistic

2.1 Test-Statistic for testrun with python 2.7.18 (final)

Number of tests	**9**
Number of successfull tests	**9**
Number of possibly failed tests	**0**
Number of failed tests	**0**
Executionlevel	Full Test (all defined tests)
Time consumption	216.932s

2.2 Test-Statistic for testrun with python 3.8.5 (final)

Number of tests	**9**
Number of successfull tests	**9**
Number of possibly failed tests	**0**
Number of failed tests	**0**
Executionlevel	Full Test (all defined tests)
Time consumption	216.910s

2.3 Coverage Statistic

Module- or Filename	Line-Coverage	Branch-Coverage
task	98.9%	98.0%
task.__init__.py	98.9%	

3 Testcases with no corresponding Requirement

3.1 Summary for testrun with python 2.7.18 (final)

3.1.1 pylibs.task.crontab: Test cronjob

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.8!

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (28)
Start-Time:	2021-01-07 17:51:32,810
Finished-Time:	2021-01-07 17:51:32,825
Time-Consumption	0.015s

Testsummary:

Info	Initialising cronjob with minute: [23, 45]; hour: [12, 17]; day: 25; month: any; day_of_week: any.
Success	Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content True and Type is <type 'bool'>).
Success	Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5 is correct (Content True and Type is <type 'bool'>).
Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).
Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 3 is correct (Content False and Type is <type 'bool'>).
Success	Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).
Success	Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).
Info	Storing reminder for execution (minute: 23, hour: 17, day: 25, month: 2, day_of_week: 1).
Success	Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).
Success	Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5 is correct (Content True and Type is <type 'bool'>).
Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).
Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 3 is correct (Content False and Type is <type 'bool'>).
Success	Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).
Success	Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).
Info	Resetting trigger condition with minute: 22; hour: any; day: [12, 17, 25], month: 2.
Success	Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).
Success	Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5 is correct (Content False and Type is <type 'bool'>).
Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content True and Type is <type 'bool'>).

Success	Return value for minute: 22; hour: 17; day: 25; month: 05, day_of_week: 3 is correct (Content False and Type is <type 'bool'>).
Success	Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).
Success	Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).
Info	Resetting trigger condition (again).
Success	1st run - execution not needed is correct (Content False and Type is <type 'bool'>).
Success	2nd run - execution not needed is correct (Content False and Type is <type 'bool'>).
Success	3rd run - execution needed is correct (Content True and Type is <type 'bool'>).
Success	4th run - execution needed is correct (Content True and Type is <type 'bool'>).
Success	5th run - execution not needed is correct (Content False and Type is <type 'bool'>).
Success	6th run - execution not needed is correct (Content False and Type is <type 'bool'>).

3.1.2 pylibs.task.crontab: Test crontab

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.9!

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (29)
Start-Time:	2021-01-07 17:51:32,825
Finished-Time:	2021-01-07 17:55:02,932
Time-Consumption	210.107s

Testsummary:

Info	Creating Crontab with callback execution in +1 and +3 minutes.
Success	Number of submitted values is correct (Content 2 and Type is <type 'int'>).
Success	Timing of crontasks: Valueaccuracy and number of submitted values is correct. See detailed log for more information.

3.1.3 pylibs.task.delayed: Test parallel processing and timing for a delayed execution

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (21)
Start-Time:	2021-01-07 17:51:25,698
Finished-Time:	2021-01-07 17:51:26,210
Time-Consumption	0.512s

Testsummary:

Info	Added a delayed task for execution in 0.250s.
Success	Execution of task and delayed task (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.
Success	Time consumption is correct (Content 0.250194787979126 in [0.2465 ... 0.2545] and Type is <type 'float'>).

Info	Added a delayed task for execution in 0.010s.
Success	Execution of task and delayed task (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.
Success	Time consumption is correct (Content 0.010162115097045898 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).
Info	Added a delayed task for execution in 0.005s.
Success	Execution of task and delayed task (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.
Success	Time consumption is correct (Content 0.0057218074798583984 in [0.00395 ... 0.00705] and Type is <type 'float'>).

3.1.4 pylibs.task.periodic: Test periodic execution

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (22)
Start-Time:	2021-01-07 17:51:26,211
Finished-Time:	2021-01-07 17:51:28,756
Time-Consumption	2.545s

Testsummary:

Info	Running a periodic task for 10 cycles with a cycletime of 0.25s
Success	Minimum cycle time is correct (Content 0.2503688335418701 in [0.2465 ... 0.2545] and Type is <type 'float'>).
Success	Mean cycle time is correct (Content 0.2510094377729628 in [0.2465 ... 0.2545] and Type is <type 'float'>).
Success	Maximum cycle time is correct (Content 0.2516500949859619 in [0.2465 ... 0.2565] and Type is <type 'float'>).
Info	Running a periodic task for 10 cycles with a cycletime of 0.01s
Success	Minimum cycle time is correct (Content 0.010493040084838867 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).
Success	Mean cycle time is correct (Content 0.010910113652547201 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).
Success	Maximum cycle time is correct (Content 0.01141214370727539 in [0.008900000000000002 ... 0.0141] and Type is <type 'float'>).
Info	Running a periodic task for 10 cycles with a cycletime of 0.005s
Success	Minimum cycle time is correct (Content 0.005506992340087891 in [0.00395 ... 0.00705] and Type is <type 'float'>).
Success	Mean cycle time is correct (Content 0.0057862069871690534 in [0.00395 ... 0.00705] and Type is <type 'float'>).
Success	Maximum cycle time is correct (Content 0.006266117095947266 in [0.00395 ... 0.009049999999999999] and Type is <type 'float'>).

3.1.5 pylibs.task.queue: Test clean_queue method

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (25)
Start-Time:	2021-01-07 17:51:28,974
Finished-Time:	2021-01-07 17:51:28,981
Time-Consumption	0.008s

Testsummary:

Info	Enqueued 6 tasks (stop request within 3rd task).
Success	Size of Queue before execution is correct (Content 6 and Type is <type 'int'>).
Success	Size of Queue after execution is correct (Content 3 and Type is <type 'int'>).
Success	Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.
Info	Cleaning Queue.
Success	Size of Queue after cleaning queue is correct (Content 0 and Type is <type 'int'>).

3.1.6 pylibs.task.queue: Test qsize and queue execution order by priority

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (23)
Start-Time:	2021-01-07 17:51:28,757
Finished-Time:	2021-01-07 17:51:28,863
Time-Consumption	0.106s

Testsummary:

Info	Enqueued 6 unordered tasks.
Success	Size of Queue before execution is correct (Content 6 and Type is <type 'int'>).
Success	Size of Queue after execution is correct (Content 0 and Type is <type 'int'>).
Success	Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

3.1.7 pylibs.task.queue: Test stop method

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (24)
Start-Time:	2021-01-07 17:51:28,863
Finished-Time:	2021-01-07 17:51:28,973
Time-Consumption	0.109s

Testsummary:

Info	Enqueued 6 tasks (stop request within 4th task).
Success	Size of Queue before 1st execution is correct (Content 6 and Type is <type 'int'>).

Success	Size of Queue after 1st execution is correct (Content 2 and Type is <type 'int'>).
Success	Queue execution (1st part; identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.
Success	Size of Queue after 2nd execution is correct (Content 0 and Type is <type 'int'>).
Success	Queue execution (2nd part; identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

3.1.8 pylibs.task.threaded_queue: Test enqueue while queue is running

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.7!

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (27)
Start-Time:	2021-01-07 17:51:31,903
Finished-Time:	2021-01-07 17:51:32,512
Time-Consumption	0.608s

Testsummary:

Success	Size of Queue before execution is correct (Content 0 and Type is <type 'int'>).
Info	Enqueued 2 tasks.
Success	Size of Queue after execution is correct (Content 0 and Type is <type 'int'>).
Success	Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

3.1.9 pylibs.task.threaded_queue: Test qsize and queue execution order by priority

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.6!

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (26)
Start-Time:	2021-01-07 17:51:28,982
Finished-Time:	2021-01-07 17:51:31,903
Time-Consumption	2.921s

Testsummary:

Info	Enqueued 6 unordered tasks.
Success	Size of Queue before execution is correct (Content 6 and Type is <type 'int'>).
Info	Executing Queue, till Queue is empty..
Success	Size of Queue after execution is correct (Content 0 and Type is <type 'int'>).
Success	Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.
Info	Setting expire flag and enqueued again 2 tasks.
Success	Size of Queue before restarting queue is correct (Content 2 and Type is <type 'int'>).
Info	Executing Queue, till Queue is empty..
Success	Queue execution (rerun; identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

3.2 Summary for testrun with python 3.8.5 (final)

3.2.1 pylibs.task.crontab: Test cronjob

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.8!

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (28)
Start-Time:	2021-01-07 17:55:11,105
Finished-Time:	2021-01-07 17:55:11,116
Time-Consumption	0.011s

Testsummary:

Info	Initialising cronjob with minute: [23, 45]; hour: [12, 17]; day: 25; month: any; day_of_week: any.
Success	Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content True and Type is <class 'bool'>).
Success	Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5 is correct (Content True and Type is <class 'bool'>).
Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).
Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 3 is correct (Content False and Type is <class 'bool'>).
Success	Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).
Success	Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).
Info	Storing reminder for execution (minute: 23, hour: 17, day: 25, month: 2, day_of_week: 1).
Success	Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).
Success	Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5 is correct (Content True and Type is <class 'bool'>).
Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).
Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 3 is correct (Content False and Type is <class 'bool'>).
Success	Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).
Success	Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).
Info	Resetting trigger condition with minute: 22; hour: any; day: [12, 17, 25], month: 2.
Success	Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).
Success	Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5 is correct (Content False and Type is <class 'bool'>).
Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content True and Type is <class 'bool'>).
Success	Return value for minute: 22; hour: 17; day: 25; month: 05, day_of_week: 3 is correct (Content False and Type is <class 'bool'>).

Success	Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).
Success	Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).
Info	Resetting trigger condition (again).
Success	1st run - execution not needed is correct (Content False and Type is <class 'bool'>).
Success	2nd run - execution not needed is correct (Content False and Type is <class 'bool'>).
Success	3rd run - execution needed is correct (Content True and Type is <class 'bool'>).
Success	4th run - execution needed is correct (Content True and Type is <class 'bool'>).
Success	5th run - execution not needed is correct (Content False and Type is <class 'bool'>).
Success	6th run - execution not needed is correct (Content False and Type is <class 'bool'>).

3.2.2 pylibs.task.crontab: Test crontab

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.9!

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (29)
Start-Time:	2021-01-07 17:55:11,117
Finished-Time:	2021-01-07 17:58:41,223
Time-Consumption	210.107s

Testsummary:

Info	Creating Crontab with callback execution in +1 and +3 minutes.
Success	Number of submitted values is correct (Content 2 and Type is <class 'int'>).
Success	Timing of crontasks: Valueaccuracy and number of submitted values is correct. See detailed log for more information.

3.2.3 pylibs.task.delayed: Test parallel processing and timing for a delayed execution

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (21)
Start-Time:	2021-01-07 17:55:04,015
Finished-Time:	2021-01-07 17:55:04,525
Time-Consumption	0.510s

Testsummary:

Info	Added a delayed task for execution in 0.250s.
Success	Execution of task and delayed task (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.
Success	Time consumption is correct (Content 0.2502169609069824 in [0.2465 ... 0.2545] and Type is <class 'float'>).
Info	Added a delayed task for execution in 0.010s.
Success	Execution of task and delayed task (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

Success	Time consumption is correct (Content 0.010170221328735352 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).
Info	Added a delayed task for execution in 0.005s.
Success	Execution of task and delayed task (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.
Success	Time consumption is correct (Content 0.005202054977416992 in [0.00395 ... 0.00705] and Type is <class 'float'>).

3.2.4 pylibs.task.periodic: Test periodic execution

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (22)
Start-Time:	2021-01-07 17:55:04,526
Finished-Time:	2021-01-07 17:55:07,069
Time-Consumption	2.543s

Testsummary:

Info	Running a periodic task for 10 cycles with a cycletime of 0.25s
Success	Minimum cycle time is correct (Content 0.25056958198547363 in [0.2465 ... 0.2545] and Type is <class 'float'>).
Success	Mean cycle time is correct (Content 0.2507186730702718 in [0.2465 ... 0.2545] and Type is <class 'float'>).
Success	Maximum cycle time is correct (Content 0.25083422660827637 in [0.2465 ... 0.2565] and Type is <class 'float'>).
Info	Running a periodic task for 10 cycles with a cycletime of 0.01s
Success	Minimum cycle time is correct (Content 0.010314226150512695 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).
Success	Mean cycle time is correct (Content 0.01074263784620497 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).
Success	Maximum cycle time is correct (Content 0.01118922233581543 in [0.008900000000000002 ... 0.0141] and Type is <class 'float'>).
Info	Running a periodic task for 10 cycles with a cycletime of 0.005s
Success	Minimum cycle time is correct (Content 0.005311727523803711 in [0.00395 ... 0.00705] and Type is <class 'float'>).
Success	Mean cycle time is correct (Content 0.005622969733344184 in [0.00395 ... 0.00705] and Type is <class 'float'>).
Success	Maximum cycle time is correct (Content 0.006051540374755859 in [0.00395 ... 0.009049999999999999] and Type is <class 'float'>).

3.2.5 pylibs.task.queue: Test clean_queue method

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (25)

Start-Time:	2021-01-07 17:55:07,281
Finished-Time:	2021-01-07 17:55:07,285
Time-Consumption	0.004s

Testsummary:

Info	Enqueued 6 tasks (stop request within 3rd task).
Success	Size of Queue before execution is correct (Content 6 and Type is <class 'int'>).
Success	Size of Queue after execution is correct (Content 3 and Type is <class 'int'>).
Success	Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.
Info	Cleaning Queue.
Success	Size of Queue after cleaning queue is correct (Content 0 and Type is <class 'int'>).

3.2.6 pylibs.task.queue: Test qsize and queue execution order by priority

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (23)
Start-Time:	2021-01-07 17:55:07,069
Finished-Time:	2021-01-07 17:55:07,172
Time-Consumption	0.103s

Testsummary:

Info	Enqueued 6 unordered tasks.
Success	Size of Queue before execution is correct (Content 6 and Type is <class 'int'>).
Success	Size of Queue after execution is correct (Content 0 and Type is <class 'int'>).
Success	Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

3.2.7 pylibs.task.queue: Test stop method

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (24)
Start-Time:	2021-01-07 17:55:07,173
Finished-Time:	2021-01-07 17:55:07,281
Time-Consumption	0.108s

Testsummary:

Info	Enqueued 6 tasks (stop request within 4th task).
Success	Size of Queue before 1st execution is correct (Content 6 and Type is <class 'int'>).
Success	Size of Queue after 1st execution is correct (Content 2 and Type is <class 'int'>).
Success	Queue execution (1st part; identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

Success	Size of Queue after 2nd execution is correct (Content 0 and Type is <class 'int'>).
Success	Queue execution (2nd part; identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

3.2.8 pylibs.task.threaded_queue: Test enqueue while queue is running

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.7!

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (27)
Start-Time:	2021-01-07 17:55:10,201
Finished-Time:	2021-01-07 17:55:10,809
Time-Consumption	0.608s

Testsummary:

Success	Size of Queue before execution is correct (Content 0 and Type is <class 'int'>).
Info	Enqueued 2 tasks.
Success	Size of Queue after execution is correct (Content 0 and Type is <class 'int'>).
Success	Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

3.2.9 pylibs.task.threaded_queue: Test qsize and queue execution order by priority

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.6!

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (26)
Start-Time:	2021-01-07 17:55:07,285
Finished-Time:	2021-01-07 17:55:10,201
Time-Consumption	2.916s

Testsummary:

Info	Enqueued 6 unordered tasks.
Success	Size of Queue before execution is correct (Content 6 and Type is <class 'int'>).
Info	Executing Queue, till Queue is empty..
Success	Size of Queue after execution is correct (Content 0 and Type is <class 'int'>).
Success	Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.
Info	Setting expire flag and enqueued again 2 tasks.
Success	Size of Queue before restarting queue is correct (Content 2 and Type is <class 'int'>).
Info	Executing Queue, till Queue is empty..
Success	Queue execution (rerun; identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

A Trace for testrun with python 2.7.18 (final)

A.1 Tests with status Info (9)

A.1.1 pylibs.task.delayed: Test parallel processing and timing for a delayed execution

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

Info	Added a delayed task for execution in 0.250s.

Success	Execution of task and delayed task (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Execution of task and delayed task (identified by a submitted sequence number)): [ 1,
↪  2 ] (<type 'list'>)
Expectation (Execution of task and delayed task (identified by a submitted sequence number)):
↪  result = [ 1, 2 ] (<type 'list'>)
Result (Submitted value number 1): 1 (<type 'int'>)
Expectation (Submitted value number 1): result = 1 (<type 'int'>)
Submitted value number 1 is correct (Content 1 and Type is <type 'int'>).
Result (Submitted value number 2): 2 (<type 'int'>)
Expectation (Submitted value number 2): result = 2 (<type 'int'>)
Submitted value number 2 is correct (Content 2 and Type is <type 'int'>).
```

Success	Time consumption is correct (Content 0.250194787979126 in [0.2465 ... 0.2545] and Type is <type 'float'>).

```
Result (Time consumption): 0.250194787979126 (<type 'float'>)
Expectation (Time consumption): 0.2465 <= result <= 0.2545
```

Info	Added a delayed task for execution in 0.010s.

Success	Execution of task and delayed task (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Execution of task and delayed task (identified by a submitted sequence number)): [ 1,
↪  2 ] (<type 'list'>)
Expectation (Execution of task and delayed task (identified by a submitted sequence number)):
↪  result = [ 1, 2 ] (<type 'list'>)
Result (Submitted value number 1): 1 (<type 'int'>)
Expectation (Submitted value number 1): result = 1 (<type 'int'>)
Submitted value number 1 is correct (Content 1 and Type is <type 'int'>).
Result (Submitted value number 2): 2 (<type 'int'>)
Expectation (Submitted value number 2): result = 2 (<type 'int'>)
Submitted value number 2 is correct (Content 2 and Type is <type 'int'>).
```

Success Time consumption is correct (Content 0.010162115097045898 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).

```
Result (Time consumption): 0.010162115097045898 (<type 'float'>)
Expectation (Time consumption): 0.008900000000000002 <= result <= 0.0121
```

Info Added a delayed task for execution in 0.005s.

Success Execution of task and delayed task (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Execution of task and delayed task (identified by a submitted sequence number)): [ 1,
↪  2 ] (<type 'list'>)
Expectation (Execution of task and delayed task (identified by a submitted sequence number)):
↪  result = [ 1, 2 ] (<type 'list'>)
Result (Submitted value number 1): 1 (<type 'int'>)
Expectation (Submitted value number 1): result = 1 (<type 'int'>)
Submitted value number 1 is correct (Content 1 and Type is <type 'int'>).
Result (Submitted value number 2): 2 (<type 'int'>)
Expectation (Submitted value number 2): result = 2 (<type 'int'>)
Submitted value number 2 is correct (Content 2 and Type is <type 'int'>).
```

Success Time consumption is correct (Content 0.0057218074798583984 in [0.00395 ... 0.00705] and Type is <type 'float'>).

```
Result (Time consumption): 0.0057218074798583984 (<type 'float'>)
Expectation (Time consumption): 0.00395 <= result <= 0.00705
```

A.1.2 pylibs.task.periodic: Test periodic execution

Testresult

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

Info	Running a periodic task for 10 cycles with a cycletime of 0.25s

```
Task execution number 1 at 1610038286.212425
Task execution number 2 at 1610038286.463316
Task execution number 3 at 1610038286.714088
Task execution number 4 at 1610038286.965607
Task execution number 5 at 1610038287.217146
Task execution number 6 at 1610038287.468796
Task execution number 7 at 1610038287.719545
Task execution number 8 at 1610038287.970385
Task execution number 9 at 1610038288.220754
Task execution number 10 at 1610038288.471510
```

Success	Minimum cycle time is correct (Content 0.2503688335418701 in [0.2465 ... 0.2545] and Type is <type 'float'>).

```
Result (Minimum cycle time): 0.2503688335418701 (<type 'float'>)
Expectation (Minimum cycle time): 0.2465 <= result <= 0.2545
```

Success	Mean cycle time is correct (Content 0.2510094377729628 in [0.2465 ... 0.2545] and Type is <type 'float'>).

```
Result (Mean cycle time): 0.2510094377729628 (<type 'float'>)
Expectation (Mean cycle time): 0.2465 <= result <= 0.2545
```

Success	Maximum cycle time is correct (Content 0.2516500949859619 in [0.2465 ... 0.2565] and Type is <type 'float'>).

```
Result (Maximum cycle time): 0.2516500949859619 (<type 'float'>)
Expectation (Maximum cycle time): 0.2465 <= result <= 0.2565
```

Info	Running a periodic task for 10 cycles with a cycletime of 0.01s

```
Task execution number 1 at 1610038288.520468
Task execution number 2 at 1610038288.530961
Task execution number 3 at 1610038288.542178
Task execution number 4 at 1610038288.553036
Task execution number 5 at 1610038288.563922
Task execution number 6 at 1610038288.575334
Task execution number 7 at 1610038288.586156
Task execution number 8 at 1610038288.597121
Task execution number 9 at 1610038288.607879
Task execution number 10 at 1610038288.618659
```

Success	Minimum cycle time is correct (Content 0.010493040084838867 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).

```
Result (Minimum cycle time): 0.010493040084838867 (<type 'float'>)
Expectation (Minimum cycle time): 0.008900000000000002 <= result <= 0.0121
```

Success	Mean cycle time is correct (Content 0.010910113652547201 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).

```
Result (Mean cycle time): 0.010910113652547201 (<type 'float'>)
Expectation (Mean cycle time): 0.008900000000000002 <= result <= 0.0121
```

Success	Maximum cycle time is correct (Content 0.011412143707275390 in [0.008900000000000002 ... 0.0141] and Type is <type 'float'>).

```
Result (Maximum cycle time): 0.011412143707275390 (<type 'float'>)
Expectation (Maximum cycle time): 0.008900000000000002 <= result <= 0.0141
```

Info	Running a periodic task for 10 cycles with a cycletime of 0.005s

```
Task execution number 1 at 1610038288.644148
Task execution number 2 at 1610038288.649655
Task execution number 3 at 1610038288.655508
Task execution number 4 at 1610038288.661398
Task execution number 5 at 1610038288.667199
Task execution number 6 at 1610038288.673465
Task execution number 7 at 1610038288.679131
Task execution number 8 at 1610038288.684951
Task execution number 9 at 1610038288.690567
Task execution number 10 at 1610038288.696224
```

Success	Minimum cycle time is correct (Content 0.005506992340087891 in [0.00395 ... 0.00705] and Type is <type 'float'>).

```
Result (Minimum cycle time): 0.005506992340087891 (<type 'float'>)
Expectation (Minimum cycle time): 0.00395 <= result <= 0.00705
```

Success	Mean cycle time is correct (Content 0.0057862069871690534 in [0.00395 ... 0.00705] and Type is <type 'float'>).

```
Result (Mean cycle time): 0.0057862069871690534 (<type 'float'>)
Expectation (Mean cycle time): 0.00395 <= result <= 0.00705
```

Success	Maximum cycle time is correct (Content 0.006266117095947266 in [0.00395 ... 0.009049999999999999] and Type is <type 'float'>).

```
Result (Maximum cycle time): 0.006266117095947266 (<type 'float'>)
Expectation (Maximum cycle time): 0.00395 <= result <= 0.009049999999999999
```

A.1.3 pylibs.task.queue: Test qsize and queue execution order by priority

Testresult

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

Info	Enqueued 6 unordered tasks.

Success	Size of Queue before execution is correct (Content 6 and Type is <type 'int'>).

```
Result (Size of Queue before execution): 6 (<type 'int'>)
Expectation (Size of Queue before execution): result = 6 (<type 'int'>)
```

Success	Size of Queue after execution is correct (Content 0 and Type is <type 'int'>).

```
Result (Size of Queue after execution): 0 (<type 'int'>)
Expectation (Size of Queue after execution): result = 0 (<type 'int'>)
```

Success	Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Queue execution (identified by a submitted sequence number)): [ 1, 2, 3, 5, 6, 7 ]
↪  (<type 'list'>)
Expectation (Queue execution (identified by a submitted sequence number)): result = [ 1, 2,
↪  3, 5, 6, 7 ] (<type 'list'>)
Result (Submitted value number 1): 1 (<type 'int'>)
Expectation (Submitted value number 1): result = 1 (<type 'int'>)
Submitted value number 1 is correct (Content 1 and Type is <type 'int'>).
Result (Submitted value number 2): 2 (<type 'int'>)
Expectation (Submitted value number 2): result = 2 (<type 'int'>)
Submitted value number 2 is correct (Content 2 and Type is <type 'int'>).
Result (Submitted value number 3): 3 (<type 'int'>)
Expectation (Submitted value number 3): result = 3 (<type 'int'>)
Submitted value number 3 is correct (Content 3 and Type is <type 'int'>).
Result (Submitted value number 4): 5 (<type 'int'>)
Expectation (Submitted value number 4): result = 5 (<type 'int'>)
Submitted value number 4 is correct (Content 5 and Type is <type 'int'>).
Result (Submitted value number 5): 6 (<type 'int'>)
Expectation (Submitted value number 5): result = 6 (<type 'int'>)
Submitted value number 5 is correct (Content 6 and Type is <type 'int'>).
Result (Submitted value number 6): 7 (<type 'int'>)
Expectation (Submitted value number 6): result = 7 (<type 'int'>)
Submitted value number 6 is correct (Content 7 and Type is <type 'int'>).
```

A.1.4 pylibs.task.queue: Test stop method

Testresult

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

Info	Enqueued 6 tasks (stop request within 4th task).

Success	Size of Queue before 1st execution is correct (Content 6 and Type is <type 'int'>).

```
Result (Size of Queue before 1st execution): 6 (<type 'int'>)
Expectation (Size of Queue before 1st execution): result = 6 (<type 'int'>)
```

Success	Size of Queue after 1st execution is correct (Content 2 and Type is <type 'int'>).

```
Result (Size of Queue after 1st execution): 2 (<type 'int'>)
Expectation (Size of Queue after 1st execution): result = 2 (<type 'int'>)
```

Success	Queue execution (1st part; identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Queue execution (1st part; identified by a submitted sequence number)): [ 1, 2, 3, 5 ] (<type 'list'>)
Expectation (Queue execution (1st part; identified by a submitted sequence number)): result = [ 1, 2, 3, 5 ] (<type 'list'>)
Result (Submitted value number 1): 1 (<type 'int'>)
Expectation (Submitted value number 1): result = 1 (<type 'int'>)
Submitted value number 1 is correct (Content 1 and Type is <type 'int'>).
Result (Submitted value number 2): 2 (<type 'int'>)
Expectation (Submitted value number 2): result = 2 (<type 'int'>)
Submitted value number 2 is correct (Content 2 and Type is <type 'int'>).
Result (Submitted value number 3): 3 (<type 'int'>)
Expectation (Submitted value number 3): result = 3 (<type 'int'>)
Submitted value number 3 is correct (Content 3 and Type is <type 'int'>).
Result (Submitted value number 4): 5 (<type 'int'>)
Expectation (Submitted value number 4): result = 5 (<type 'int'>)
Submitted value number 4 is correct (Content 5 and Type is <type 'int'>).
```

Success	Size of Queue after 2nd execution is correct (Content 0 and Type is <type 'int'>).

```
Result (Size of Queue after 2nd execution): 0 (<type 'int'>)
Expectation (Size of Queue after 2nd execution): result = 0 (<type 'int'>)
```

Success	Queue execution (2nd part; identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Queue execution (2nd part; identified by a submitted sequence number)): [ 6, 7 ] (<type 'list'>)
Expectation (Queue execution (2nd part; identified by a submitted sequence number)): result = [ 6, 7 ] (<type 'list'>)
Result (Submitted value number 1): 6 (<type 'int'>)
Expectation (Submitted value number 1): result = 6 (<type 'int'>)
Submitted value number 1 is correct (Content 6 and Type is <type 'int'>).
Result (Submitted value number 2): 7 (<type 'int'>)
Expectation (Submitted value number 2): result = 7 (<type 'int'>)
Submitted value number 2 is correct (Content 7 and Type is <type 'int'>).
```

A.1.5 pylibs.task.queue: Test clean_queue method

Testresult

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

Info	Enqueued 6 tasks (stop request within 3rd task).

Success	Size of Queue before execution is correct (Content 6 and Type is <type 'int'>).

```
Result (Size of Queue before execution): 6 (<type 'int'>)
Expectation (Size of Queue before execution): result = 6 (<type 'int'>)
```

Success	Size of Queue after execution is correct (Content 3 and Type is <type 'int'>).

```
Result (Size of Queue after execution): 3 (<type 'int'>)
Expectation (Size of Queue after execution): result = 3 (<type 'int'>)
```

Success	Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Queue execution (identified by a submitted sequence number)): [ 1, 2, 3 ] (<type
↪  'list'>)
Expectation (Queue execution (identified by a submitted sequence number)): result = [ 1, 2, 3
↪  ] (<type 'list'>)
Result (Submitted value number 1): 1 (<type 'int'>)
Expectation (Submitted value number 1): result = 1 (<type 'int'>)
Submitted value number 1 is correct (Content 1 and Type is <type 'int'>).
Result (Submitted value number 2): 2 (<type 'int'>)
Expectation (Submitted value number 2): result = 2 (<type 'int'>)
Submitted value number 2 is correct (Content 2 and Type is <type 'int'>).
Result (Submitted value number 3): 3 (<type 'int'>)
Expectation (Submitted value number 3): result = 3 (<type 'int'>)
Submitted value number 3 is correct (Content 3 and Type is <type 'int'>).
```

Info	Cleaning Queue.

Success	Size of Queue after cleaning queue is correct (Content 0 and Type is <type 'int'>).

```
Result (Size of Queue after cleaning queue): 0 (<type 'int'>)
Expectation (Size of Queue after cleaning queue): result = 0 (<type 'int'>)
```

A.1.6 pylibs.task.threaded_queue: Test qsize and queue execution order by priority

Testresult

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

Info	Enqueued 6 unordered tasks.

```
Adding Task 5 with Priority 5
Adding Task 3 with Priority 3
Adding Task 7 with Priority 7
Adding Task 2 with Priority 2
Adding Task 6 with Priority 6
Adding Task 1 with Priority 1
```

Success	Size of Queue before execution is correct (Content 6 and Type is <type 'int'>).

```
Result (Size of Queue before execution): 6 (<type 'int'>)
Expectation (Size of Queue before execution): result = 6 (<type 'int'>)
```

Info Executing Queue, till Queue is empty..

```
Starting Queue execution (run)
Queue is empty.
```

Success Size of Queue after execution is correct (Content 0 and Type is <type 'int'>).

```
Result (Size of Queue after execution): 0 (<type 'int'>)
Expectation (Size of Queue after execution): result = 0 (<type 'int'>)
```

Success Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Queue execution (identified by a submitted sequence number)): [ 1, 2, 3, 5, 6, 7 ]
↪  (<type 'list'>)
Expectation (Queue execution (identified by a submitted sequence number)): result = [ 1, 2,
↪  3, 5, 6, 7 ] (<type 'list'>)
Result (Submitted value number 1): 1 (<type 'int'>)
Expectation (Submitted value number 1): result = 1 (<type 'int'>)
Submitted value number 1 is correct (Content 1 and Type is <type 'int'>).
Result (Submitted value number 2): 2 (<type 'int'>)
Expectation (Submitted value number 2): result = 2 (<type 'int'>)
Submitted value number 2 is correct (Content 2 and Type is <type 'int'>).
Result (Submitted value number 3): 3 (<type 'int'>)
Expectation (Submitted value number 3): result = 3 (<type 'int'>)
Submitted value number 3 is correct (Content 3 and Type is <type 'int'>).
Result (Submitted value number 4): 5 (<type 'int'>)
Expectation (Submitted value number 4): result = 5 (<type 'int'>)
Submitted value number 4 is correct (Content 5 and Type is <type 'int'>).
Result (Submitted value number 5): 6 (<type 'int'>)
Expectation (Submitted value number 5): result = 6 (<type 'int'>)
Submitted value number 5 is correct (Content 6 and Type is <type 'int'>).
Result (Submitted value number 6): 7 (<type 'int'>)
Expectation (Submitted value number 6): result = 7 (<type 'int'>)
Submitted value number 6 is correct (Content 7 and Type is <type 'int'>).
```

Info Setting expire flag and enqueued again 2 tasks.

```
Expire executed
Adding Task 6 with Priority 6
Adding Task 1 with Priority 1
```

Success Size of Queue before restarting queue is correct (Content 2 and Type is <type 'int'>).

```
Result (Size of Queue before restarting queue): 2 (<type 'int'>)
Expectation (Size of Queue before restarting queue): result = 2 (<type 'int'>)
```

Info	Executing Queue, till Queue is empty..

```
Starting Queue execution (run)
Queue joined and stopped.
```

Success	Queue execution (rerun; identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Queue execution (rerun; identified by a submitted sequence number)): [ 1, 6 ] (<type
↪  'list'>)
Expectation (Queue execution (rerun; identified by a submitted sequence number)): result = [
↪  1, 6 ] (<type 'list'>)
Result (Submitted value number 1): 1 (<type 'int'>)
Expectation (Submitted value number 1): result = 1 (<type 'int'>)
Submitted value number 1 is correct (Content 1 and Type is <type 'int'>).
Result (Submitted value number 2): 6 (<type 'int'>)
Expectation (Submitted value number 2): result = 6 (<type 'int'>)
Submitted value number 2 is correct (Content 6 and Type is <type 'int'>).
```

A.1.7 pylibs.task.threaded_queue: Test enqueue while queue is running

Testresult

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

Success	Size of Queue before execution is correct (Content 0 and Type is <type 'int'>).

```
Result (Size of Queue before execution): 0 (<type 'int'>)
Expectation (Size of Queue before execution): result = 0 (<type 'int'>)
```

Info	Enqueued 2 tasks.

```
Starting Queue execution (run)
Adding Task 6 with Priority 6 and waiting for 0.1s (half of the queue task delay time)
Adding Task 3 with Priority 3
Adding Task 2 with Priority 2
Adding Task 1 with Priority 1
```

Success	Size of Queue after execution is correct (Content 0 and Type is <type 'int'>).

```
Result (Size of Queue after execution): 0 (<type 'int'>)
Expectation (Size of Queue after execution): result = 0 (<type 'int'>)
```

Success	Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Queue execution (identified by a submitted sequence number)): [ 6, 1, 2, 3 ] (<type
↪  'list'>)
Expectation (Queue execution (identified by a submitted sequence number)): result = [ 6, 1,
↪  2, 3 ] (<type 'list'>)
Result (Submitted value number 1): 6 (<type 'int'>)
Expectation (Submitted value number 1): result = 6 (<type 'int'>)
Submitted value number 1 is correct (Content 6 and Type is <type 'int'>).
Result (Submitted value number 2): 1 (<type 'int'>)
Expectation (Submitted value number 2): result = 1 (<type 'int'>)
Submitted value number 2 is correct (Content 1 and Type is <type 'int'>).
Result (Submitted value number 3): 2 (<type 'int'>)
Expectation (Submitted value number 3): result = 2 (<type 'int'>)
Submitted value number 3 is correct (Content 2 and Type is <type 'int'>).
Result (Submitted value number 4): 3 (<type 'int'>)
Expectation (Submitted value number 4): result = 3 (<type 'int'>)
Submitted value number 4 is correct (Content 3 and Type is <type 'int'>).
```

A.1.8 pylibs.task.crontab: Test cronjob

Testresult

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

Info	Initialising cronjob with minute: [23, 45]; hour: [12, 17]; day: 25; month: any; day_of_week: any.

Success	Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content True and Type is <type 'bool'>).

```
Result (Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1): True
↪  (<type 'bool'>)
Expectation (Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1):
↪  result = True (<type 'bool'>)
```

Success	Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5 is correct (Content True and Type is <type 'bool'>).

```
Result (Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5): True
↪  (<type 'bool'>)
Expectation (Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5):
↪  result = True (<type 'bool'>)
```

Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).

```
Result (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1): False
↪  (<type 'bool'>)
Expectation (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1):
↪  result = False (<type 'bool'>)
```

Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 3 is correct (Content False and Type is <type 'bool'>).

```
Result (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 3): False
↪  (<type 'bool'>)
Expectation (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 3):
↪  result = False (<type 'bool'>)
```

Success	Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).

```
Result (Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1): False
↪  (<type 'bool'>)
Expectation (Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1):
↪  result = False (<type 'bool'>)
```

Success	Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).

```
Result (Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1): False
↪  (<type 'bool'>)
Expectation (Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1):
↪  result = False (<type 'bool'>)
```

Info	Storing reminder for execution (minute: 23, hour: 17, day: 25, month: 2, day_of_week: 1).

Success	Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).

```
Result (Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1): False
↪  (<type 'bool'>)
Expectation (Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1):
↪  result = False (<type 'bool'>)
```

Success	Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5 is correct (Content True and Type is <type 'bool'>).

```
Result (Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5): True
↪  (<type 'bool'>)
Expectation (Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5):
↪  result = True (<type 'bool'>)
```

Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).

```
Result (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1): False
↪  (<type 'bool'>)
Expectation (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1):
↪  result = False (<type 'bool'>)
```

Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 3 is correct (Content False and Type is <type 'bool'>).

```
Result (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 3): False
↪  (<type 'bool'>)
Expectation (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 3):
↪  result = False (<type 'bool'>)
```

Success	Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).

```
Result (Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1): False
↪  (<type 'bool'>)
Expectation (Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1):
↪  result = False (<type 'bool'>)
```

Success	Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).

```
Result (Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1): False
↪  (<type 'bool'>)
Expectation (Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1):
↪  result = False (<type 'bool'>)
```

Info	Resetting trigger condition with minute: 22; hour: any; day: [12, 17, 25], month: 2.

Success	Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).

```
Result (Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1): False
↪  (<type 'bool'>)
Expectation (Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1):
↪  result = False (<type 'bool'>)
```

Success	Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5 is correct (Content False and Type is <type 'bool'>).

```
Result (Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5): False
↪  (<type 'bool'>)
Expectation (Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5):
↪  result = False (<type 'bool'>)
```

Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content True and Type is <type 'bool'>).

```
Result (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1): True
↪  (<type 'bool'>)
Expectation (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1):
↪  result = True (<type 'bool'>)
```

Success	Return value for minute: 22; hour: 17; day: 25; month: 05, day_of_week: 3 is correct (Content False and Type is <type 'bool'>).

```
Result (Return value for minute: 22; hour: 17; day: 25; month: 05, day_of_week: 3): False
↪  (<type 'bool'>)
Expectation (Return value for minute: 22; hour: 17; day: 25; month: 05, day_of_week: 3):
↪  result = False (<type 'bool'>)
```

Success	Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).

```
Result (Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1): False
↪  (<type 'bool'>)
Expectation (Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1):
↪  result = False (<type 'bool'>)
```

Success	Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1 is correct (Content False and Type is <type 'bool'>).

```
Result (Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1): False
↪  (<type 'bool'>)
Expectation (Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1):
↪  result = False (<type 'bool'>)
```

Info	Resetting trigger condition (again).

Success	1st run - execution not needed is correct (Content False and Type is <type 'bool'>).

```
Result (1st run - execution not needed): False (<type 'bool'>)
Expectation (1st run - execution not needed): result = False (<type 'bool'>)
```

Success	2nd run - execution not needed is correct (Content False and Type is <type 'bool'>).

```
Result (2nd run - execution not needed): False (<type 'bool'>)
Expectation (2nd run - execution not needed): result = False (<type 'bool'>)
```

Success	3rd run - execution needed is correct (Content True and Type is <type 'bool'>).

```
Result (3rd run - execution needed): True (<type 'bool'>)
Expectation (3rd run - execution needed): result = True (<type 'bool'>)
```

Success	4th run - execution needed is correct (Content True and Type is <type 'bool'>).

```
Result (4th run - execution needed): True (<type 'bool'>)
Expectation (4th run - execution needed): result = True (<type 'bool'>)
```

Success	5th run - execution not needed is correct (Content False and Type is <type 'bool'>).

```
Result (5th run - execution not needed): False (<type 'bool'>)
Expectation (5th run - execution not needed): result = False (<type 'bool'>)
```

Success	6th run - execution not needed is correct (Content False and Type is <type 'bool'>).

```
Result (6th run - execution not needed): False (<type 'bool'>)
Expectation (6th run - execution not needed): result = False (<type 'bool'>)
```

A.1.9 pylibs.task.crontab: Test crontab

Testresult

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

Info	Creating Crontab with callback execution in +1 and +3 minutes.

Success	Number of submitted values is correct (Content 2 and Type is <type 'int'>).

```
Crontab accuracy is 30s
Crontab execution number 1 at 1610038322s, requested for 1610038320s
Crontab execution number 2 at 1610038442s, requested for 1610038440s
Result (Timing of crontasks): [ 1610038322, 1610038442 ] (<type 'list'>)
Result (Number of submitted values): 2 (<type 'int'>)
Expectation (Number of submitted values): result = 2 (<type 'int'>)
```

Success	Timing of crontasks: Valueaccuracy and number of submitted values is correct. See detailed log for more information.

```
Result (Submitted value number 1): 1610038322 (<type 'int'>)
Expectation (Submitted value number 1): 1610038320 <= result <= 1610038351
Submitted value number 1 is correct (Content 1610038322 in [1610038320 ... 1610038351] and
↪  Type is <type 'int'>).
Result (Submitted value number 2): 1610038442 (<type 'int'>)
Expectation (Submitted value number 2): 1610038440 <= result <= 1610038471
Submitted value number 2 is correct (Content 1610038442 in [1610038440 ... 1610038471] and
↪  Type is <type 'int'>).
```

B Trace for testrun with python 3.8.5 (final)

B.1 Tests with status Info (9)

B.1.1 pylibs.task.delayed: Test parallel processing and timing for a delayed execution

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

Info	Added a delayed task for execution in 0.250s.

Success	Execution of task and delayed task (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Execution of task and delayed task (identified by a submitted sequence number)): [ 1,
↪  2 ] (<class 'list'>)
Expectation (Execution of task and delayed task (identified by a submitted sequence number)):
↪  result = [ 1, 2 ] (<class 'list'>)
Result (Submitted value number 1): 1 (<class 'int'>)
Expectation (Submitted value number 1): result = 1 (<class 'int'>)
Submitted value number 1 is correct (Content 1 and Type is <class 'int'>).
Result (Submitted value number 2): 2 (<class 'int'>)
Expectation (Submitted value number 2): result = 2 (<class 'int'>)
Submitted value number 2 is correct (Content 2 and Type is <class 'int'>).
```

Success	Time consumption is correct (Content 0.2502169609069824 in [0.2465 ... 0.2545] and Type is <class 'float'>).

```
Result (Time consumption): 0.2502169609069824 (<class 'float'>)
Expectation (Time consumption): 0.2465 <= result <= 0.2545
```

Info	Added a delayed task for execution in 0.010s.

Success	Execution of task and delayed task (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Execution of task and delayed task (identified by a submitted sequence number)): [ 1,
↪  2 ] (<class 'list'>)
Expectation (Execution of task and delayed task (identified by a submitted sequence number)):
↪  result = [ 1, 2 ] (<class 'list'>)
Result (Submitted value number 1): 1 (<class 'int'>)
Expectation (Submitted value number 1): result = 1 (<class 'int'>)
Submitted value number 1 is correct (Content 1 and Type is <class 'int'>).
Result (Submitted value number 2): 2 (<class 'int'>)
Expectation (Submitted value number 2): result = 2 (<class 'int'>)
Submitted value number 2 is correct (Content 2 and Type is <class 'int'>).
```

Success Time consumption is correct (Content 0.010170221328735352 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).

```
Result (Time consumption): 0.010170221328735352 (<class 'float'>)
Expectation (Time consumption): 0.008900000000000002 <= result <= 0.0121
```

Info Added a delayed task for execution in 0.005s.

Success Execution of task and delayed task (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Execution of task and delayed task (identified by a submitted sequence number)): [ 1,
↪  2 ] (<class 'list'>)
Expectation (Execution of task and delayed task (identified by a submitted sequence number)):
↪  result = [ 1, 2 ] (<class 'list'>)
Result (Submitted value number 1): 1 (<class 'int'>)
Expectation (Submitted value number 1): result = 1 (<class 'int'>)
Submitted value number 1 is correct (Content 1 and Type is <class 'int'>).
Result (Submitted value number 2): 2 (<class 'int'>)
Expectation (Submitted value number 2): result = 2 (<class 'int'>)
Submitted value number 2 is correct (Content 2 and Type is <class 'int'>).
```

Success Time consumption is correct (Content 0.005202054977416992 in [0.00395 ... 0.00705] and Type is <class 'float'>).

```
Result (Time consumption): 0.005202054977416992 (<class 'float'>)
Expectation (Time consumption): 0.00395 <= result <= 0.00705
```

B.1.2 pylibs.task.periodic: Test periodic execution

Testresult

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

Info	Running a periodic task for 10 cycles with a cycletime of 0.25s

```
Task execution number 1 at 1610038504.527136
Task execution number 2 at 1610038504.777705
Task execution number 3 at 1610038505.028444
Task execution number 4 at 1610038505.279238
Task execution number 5 at 1610038505.529990
Task execution number 6 at 1610038505.780824
Task execution number 7 at 1610038506.031572
Task execution number 8 at 1610038506.282295
Task execution number 9 at 1610038506.533014
Task execution number 10 at 1610038506.783604
```

Success	Minimum cycle time is correct (Content 0.25056958198547363 in [0.2465 ... 0.2545] and Type is <class 'float'>).

```
Result (Minimum cycle time): 0.25056958198547363 (<class 'float'>)
Expectation (Minimum cycle time): 0.2465 <= result <= 0.2545
```

Success	Mean cycle time is correct (Content 0.2507186730702718 in [0.2465 ... 0.2545] and Type is <class 'float'>).

```
Result (Mean cycle time): 0.2507186730702718 (<class 'float'>)
Expectation (Mean cycle time): 0.2465 <= result <= 0.2545
```

Success	Maximum cycle time is correct (Content 0.25083422660827637 in [0.2465 ... 0.2565] and Type is <class 'float'>).

```
Result (Maximum cycle time): 0.25083422660827637 (<class 'float'>)
Expectation (Maximum cycle time): 0.2465 <= result <= 0.2565
```

Info	Running a periodic task for 10 cycles with a cycletime of 0.01s

```
Task execution number 1 at 1610038506.835191
Task execution number 2 at 1610038506.846380
Task execution number 3 at 1610038506.856695
Task execution number 4 at 1610038506.867366
Task execution number 5 at 1610038506.878433
Task execution number 6 at 1610038506.889082
Task execution number 7 at 1610038506.899747
Task execution number 8 at 1610038506.910884
Task execution number 9 at 1610038506.921244
Task execution number 10 at 1610038506.931875
```

Success Minimum cycle time is correct (Content 0.010314226150512695 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).

```
Result (Minimum cycle time): 0.010314226150512695 (<class 'float'>)
Expectation (Minimum cycle time): 0.008900000000000002 <= result <= 0.0121
```

Success Mean cycle time is correct (Content 0.01074263784620497 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).

```
Result (Mean cycle time): 0.01074263784620497 (<class 'float'>)
Expectation (Mean cycle time): 0.008900000000000002 <= result <= 0.0121
```

Success Maximum cycle time is correct (Content 0.011189222335815 43 in [0.008900000000000002 ... 0.0141] and Type is <class 'float'>).

```
Result (Maximum cycle time): 0.011189222335815 43 (<class 'float'>)
Expectation (Maximum cycle time): 0.008900000000000002 <= result <= 0.0141
```

Info Running a periodic task for 10 cycles with a cycletime of 0.005s

```
Task execution number 1 at 1610038506.957644
Task execution number 2 at 1610038506.963672
Task execution number 3 at 1610038506.969070
Task execution number 4 at 1610038506.974635
Task execution number 5 at 1610038506.980686
Task execution number 6 at 1610038506.985998
Task execution number 7 at 1610038506.991982
Task execution number 8 at 1610038506.997376
Task execution number 9 at 1610038507.002700
Task execution number 10 at 1610038507.008251
```

Success Minimum cycle time is correct (Content 0.005311727523803711 in [0.00395 ... 0.00705] and Type is <class 'float'>).

```
Result (Minimum cycle time): 0.005311727523803711 (<class 'float'>)
Expectation (Minimum cycle time): 0.00395 <= result <= 0.00705
```

Success	Mean cycle time is correct (Content 0.005622969733344184 in [0.00395 ... 0.00705] and Type is <class 'float'>).

```
Result (Mean cycle time): 0.005622969733344184 (<class 'float'>)
Expectation (Mean cycle time): 0.00395 <= result <= 0.00705
```

Success	Maximum cycle time is correct (Content 0.006051540374755859 in [0.00395 ... 0.009049999999999999] and Type is <class 'float'>).

```
Result (Maximum cycle time): 0.006051540374755859 (<class 'float'>)
Expectation (Maximum cycle time): 0.00395 <= result <= 0.009049999999999999
```

B.1.3 pylibs.task.queue: Test qsize and queue execution order by priority

Testresult

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

Info	Enqueued 6 unordered tasks.

Success	Size of Queue before execution is correct (Content 6 and Type is <class 'int'>).

```
Result (Size of Queue before execution): 6 (<class 'int'>)
Expectation (Size of Queue before execution): result = 6 (<class 'int'>)
```

Success	Size of Queue after execution is correct (Content 0 and Type is <class 'int'>).

```
Result (Size of Queue after execution): 0 (<class 'int'>)
Expectation (Size of Queue after execution): result = 0 (<class 'int'>)
```

Success	Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Queue execution (identified by a submitted sequence number)): [ 1, 2, 3, 5, 6, 7 ]
↪  (<class 'list'>)
Expectation (Queue execution (identified by a submitted sequence number)): result = [ 1, 2,
↪  3, 5, 6, 7 ] (<class 'list'>)
Result (Submitted value number 1): 1 (<class 'int'>)
Expectation (Submitted value number 1): result = 1 (<class 'int'>)
Submitted value number 1 is correct (Content 1 and Type is <class 'int'>).
Result (Submitted value number 2): 2 (<class 'int'>)
Expectation (Submitted value number 2): result = 2 (<class 'int'>)
Submitted value number 2 is correct (Content 2 and Type is <class 'int'>).
Result (Submitted value number 3): 3 (<class 'int'>)
Expectation (Submitted value number 3): result = 3 (<class 'int'>)
Submitted value number 3 is correct (Content 3 and Type is <class 'int'>).
Result (Submitted value number 4): 5 (<class 'int'>)
Expectation (Submitted value number 4): result = 5 (<class 'int'>)
Submitted value number 4 is correct (Content 5 and Type is <class 'int'>).
Result (Submitted value number 5): 6 (<class 'int'>)
Expectation (Submitted value number 5): result = 6 (<class 'int'>)
Submitted value number 5 is correct (Content 6 and Type is <class 'int'>).
Result (Submitted value number 6): 7 (<class 'int'>)
Expectation (Submitted value number 6): result = 7 (<class 'int'>)
Submitted value number 6 is correct (Content 7 and Type is <class 'int'>).
```

B.1.4 pylibs.task.queue: Test stop method

Testresult

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

Info	Enqueued 6 tasks (stop request within 4th task).

Success	Size of Queue before 1st execution is correct (Content 6 and Type is <class 'int'>).

```
Result (Size of Queue before 1st execution): 6 (<class 'int'>)
Expectation (Size of Queue before 1st execution): result = 6 (<class 'int'>)
```

Success	Size of Queue after 1st execution is correct (Content 2 and Type is <class 'int'>).

```
Result (Size of Queue after 1st execution): 2 (<class 'int'>)
Expectation (Size of Queue after 1st execution): result = 2 (<class 'int'>)
```

Success	Queue execution (1st part; identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Queue execution (1st part; identified by a submitted sequence number)): [ 1, 2, 3, 5 ] (<class 'list'>)
Expectation (Queue execution (1st part; identified by a submitted sequence number)): result = [ 1, 2, 3, 5 ] (<class 'list'>)
Result (Submitted value number 1): 1 (<class 'int'>)
Expectation (Submitted value number 1): result = 1 (<class 'int'>)
Submitted value number 1 is correct (Content 1 and Type is <class 'int'>).
Result (Submitted value number 2): 2 (<class 'int'>)
Expectation (Submitted value number 2): result = 2 (<class 'int'>)
Submitted value number 2 is correct (Content 2 and Type is <class 'int'>).
Result (Submitted value number 3): 3 (<class 'int'>)
Expectation (Submitted value number 3): result = 3 (<class 'int'>)
Submitted value number 3 is correct (Content 3 and Type is <class 'int'>).
Result (Submitted value number 4): 5 (<class 'int'>)
Expectation (Submitted value number 4): result = 5 (<class 'int'>)
Submitted value number 4 is correct (Content 5 and Type is <class 'int'>).
```

Success	Size of Queue after 2nd execution is correct (Content 0 and Type is <class 'int'>).

```
Result (Size of Queue after 2nd execution): 0 (<class 'int'>)
Expectation (Size of Queue after 2nd execution): result = 0 (<class 'int'>)
```

Success	Queue execution (2nd part; identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Queue execution (2nd part; identified by a submitted sequence number)): [ 6, 7 ] (<class 'list'>)
Expectation (Queue execution (2nd part; identified by a submitted sequence number)): result = [ 6, 7 ] (<class 'list'>)
Result (Submitted value number 1): 6 (<class 'int'>)
Expectation (Submitted value number 1): result = 6 (<class 'int'>)
Submitted value number 1 is correct (Content 6 and Type is <class 'int'>).
Result (Submitted value number 2): 7 (<class 'int'>)
Expectation (Submitted value number 2): result = 7 (<class 'int'>)
Submitted value number 2 is correct (Content 7 and Type is <class 'int'>).
```

B.1.5 pylibs.task.queue: Test clean_queue method

Testresult

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

Info	Enqueued 6 tasks (stop request within 3rd task).

Success	Size of Queue before execution is correct (Content 6 and Type is <class 'int'>).

```
Result (Size of Queue before execution): 6 (<class 'int'>)
Expectation (Size of Queue before execution): result = 6 (<class 'int'>)
```

Success	Size of Queue after execution is correct (Content 3 and Type is <class 'int'>).

```
Result (Size of Queue after execution): 3 (<class 'int'>)
Expectation (Size of Queue after execution): result = 3 (<class 'int'>)
```

Success	Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Queue execution (identified by a submitted sequence number)): [ 1, 2, 3 ] (<class
↪  'list'>)
Expectation (Queue execution (identified by a submitted sequence number)): result = [ 1, 2, 3
↪  ] (<class 'list'>)
Result (Submitted value number 1): 1 (<class 'int'>)
Expectation (Submitted value number 1): result = 1 (<class 'int'>)
Submitted value number 1 is correct (Content 1 and Type is <class 'int'>).
Result (Submitted value number 2): 2 (<class 'int'>)
Expectation (Submitted value number 2): result = 2 (<class 'int'>)
Submitted value number 2 is correct (Content 2 and Type is <class 'int'>).
Result (Submitted value number 3): 3 (<class 'int'>)
Expectation (Submitted value number 3): result = 3 (<class 'int'>)
Submitted value number 3 is correct (Content 3 and Type is <class 'int'>).
```

Info	Cleaning Queue.

Success	Size of Queue after cleaning queue is correct (Content 0 and Type is <class 'int'>).

```
Result (Size of Queue after cleaning queue): 0 (<class 'int'>)
Expectation (Size of Queue after cleaning queue): result = 0 (<class 'int'>)
```

B.1.6 pylibs.task.threaded_queue: Test qsize and queue execution order by priority

Testresult

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

Info	Enqueued 6 unordered tasks.

```
Adding Task 5 with Priority 5
Adding Task 3 with Priority 3
Adding Task 7 with Priority 7
Adding Task 2 with Priority 2
Adding Task 6 with Priority 6
Adding Task 1 with Priority 1
```

Success	Size of Queue before execution is correct (Content 6 and Type is <class 'int'>).

```
Result (Size of Queue before execution): 6 (<class 'int'>)
Expectation (Size of Queue before execution): result = 6 (<class 'int'>)
```

Info Executing Queue, till Queue is empty..

```
Starting Queue execution (run)
Queue is empty.
```

Success Size of Queue after execution is correct (Content 0 and Type is <class 'int'>).

```
Result (Size of Queue after execution): 0 (<class 'int'>)
Expectation (Size of Queue after execution): result = 0 (<class 'int'>)
```

Success Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Queue execution (identified by a submitted sequence number)): [ 1, 2, 3, 5, 6, 7 ]
↪  (<class 'list'>)
Expectation (Queue execution (identified by a submitted sequence number)): result = [ 1, 2,
↪  3, 5, 6, 7 ] (<class 'list'>)
Result (Submitted value number 1): 1 (<class 'int'>)
Expectation (Submitted value number 1): result = 1 (<class 'int'>)
Submitted value number 1 is correct (Content 1 and Type is <class 'int'>).
Result (Submitted value number 2): 2 (<class 'int'>)
Expectation (Submitted value number 2): result = 2 (<class 'int'>)
Submitted value number 2 is correct (Content 2 and Type is <class 'int'>).
Result (Submitted value number 3): 3 (<class 'int'>)
Expectation (Submitted value number 3): result = 3 (<class 'int'>)
Submitted value number 3 is correct (Content 3 and Type is <class 'int'>).
Result (Submitted value number 4): 5 (<class 'int'>)
Expectation (Submitted value number 4): result = 5 (<class 'int'>)
Submitted value number 4 is correct (Content 5 and Type is <class 'int'>).
Result (Submitted value number 5): 6 (<class 'int'>)
Expectation (Submitted value number 5): result = 6 (<class 'int'>)
Submitted value number 5 is correct (Content 6 and Type is <class 'int'>).
Result (Submitted value number 6): 7 (<class 'int'>)
Expectation (Submitted value number 6): result = 7 (<class 'int'>)
Submitted value number 6 is correct (Content 7 and Type is <class 'int'>).
```

Info Setting expire flag and enqueued again 2 tasks.

```
Expire executed
Adding Task 6 with Priority 6
Adding Task 1 with Priority 1
```

Success Size of Queue before restarting queue is correct (Content 2 and Type is <class 'int'>).

```
Result (Size of Queue before restarting queue): 2 (<class 'int'>)
Expectation (Size of Queue before restarting queue): result = 2 (<class 'int'>)
```

Info	Executing Queue, till Queue is empty..

```
Starting Queue execution (run)
Queue joined and stopped.
```

Success	Queue execution (rerun; identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Queue execution (rerun; identified by a submitted sequence number)): [ 1, 6 ] (<class
↪  'list'>)
Expectation (Queue execution (rerun; identified by a submitted sequence number)): result = [
↪  1, 6 ] (<class 'list'>)
Result (Submitted value number 1): 1 (<class 'int'>)
Expectation (Submitted value number 1): result = 1 (<class 'int'>)
Submitted value number 1 is correct (Content 1 and Type is <class 'int'>).
Result (Submitted value number 2): 6 (<class 'int'>)
Expectation (Submitted value number 2): result = 6 (<class 'int'>)
Submitted value number 2 is correct (Content 6 and Type is <class 'int'>).
```

B.1.7 pylibs.task.threaded_queue: Test enqueue while queue is running

Testresult

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

Success	Size of Queue before execution is correct (Content 0 and Type is <class 'int'>).

```
Result (Size of Queue before execution): 0 (<class 'int'>)
Expectation (Size of Queue before execution): result = 0 (<class 'int'>)
```

Info	Enqueued 2 tasks.

```
Starting Queue execution (run)
Adding Task 6 with Priority 6 and waiting for 0.1s (half of the queue task delay time)
Adding Task 3 with Priority 3
Adding Task 2 with Priority 2
Adding Task 1 with Priority 1
```

Success	Size of Queue after execution is correct (Content 0 and Type is <class 'int'>).

```
Result (Size of Queue after execution): 0 (<class 'int'>)
Expectation (Size of Queue after execution): result = 0 (<class 'int'>)
```

Success	Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

```
Result (Queue execution (identified by a submitted sequence number)): [ 6, 1, 2, 3 ] (<class
↪  'list'>)
Expectation (Queue execution (identified by a submitted sequence number)): result = [ 6, 1,
↪  2, 3 ] (<class 'list'>)
Result (Submitted value number 1): 6 (<class 'int'>)
Expectation (Submitted value number 1): result = 6 (<class 'int'>)
Submitted value number 1 is correct (Content 6 and Type is <class 'int'>).
Result (Submitted value number 2): 1 (<class 'int'>)
Expectation (Submitted value number 2): result = 1 (<class 'int'>)
Submitted value number 2 is correct (Content 1 and Type is <class 'int'>).
Result (Submitted value number 3): 2 (<class 'int'>)
Expectation (Submitted value number 3): result = 2 (<class 'int'>)
Submitted value number 3 is correct (Content 2 and Type is <class 'int'>).
Result (Submitted value number 4): 3 (<class 'int'>)
Expectation (Submitted value number 4): result = 3 (<class 'int'>)
Submitted value number 4 is correct (Content 3 and Type is <class 'int'>).
```

B.1.8 pylibs.task.crontab: Test cronjob

Testresult

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

Info	Initialising cronjob with minute: [23, 45]; hour: [12, 17]; day: 25; month: any; day_of_week: any.

Success	Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content True and Type is <class 'bool'>).

```
Result (Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1): True
↪  (<class 'bool'>)
Expectation (Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1):
↪  result = True (<class 'bool'>)
```

Success	Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5 is correct (Content True and Type is <class 'bool'>).

```
Result (Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5): True
↪  (<class 'bool'>)
Expectation (Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5):
↪  result = True (<class 'bool'>)
```

Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).

```
Result (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1): False
↪  (<class 'bool'>)
Expectation (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1):
↪  result = False (<class 'bool'>)
```

Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 3 is correct (Content False and Type is <class 'bool'>).

```
Result (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 3): False
↪  (<class 'bool'>)
Expectation (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 3):
↪  result = False (<class 'bool'>)
```

Success	Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).

```
Result (Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1): False
↪  (<class 'bool'>)
Expectation (Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1):
↪  result = False (<class 'bool'>)
```

Success	Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).

```
Result (Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1): False
↪  (<class 'bool'>)
Expectation (Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1):
↪  result = False (<class 'bool'>)
```

Info	Storing reminder for execution (minute: 23, hour: 17, day: 25, month: 2, day_of_week: 1).

Success	Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).

```
Result (Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1): False
↪  (<class 'bool'>)
Expectation (Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1):
↪  result = False (<class 'bool'>)
```

Success	Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5 is correct (Content True and Type is <class 'bool'>).

```
Result (Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5): True
↪  (<class 'bool'>)
Expectation (Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5):
↪  result = True (<class 'bool'>)
```

Success	Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).

```
Result (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1): False
↪  (<class 'bool'>)
Expectation (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1):
↪  result = False (<class 'bool'>)
```

Success Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 3 is correct (Content False and Type is <class 'bool'>).

```
Result (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 3): False
↪  (<class 'bool'>)
Expectation (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 3):
↪  result = False (<class 'bool'>)
```

Success Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).

```
Result (Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1): False
↪  (<class 'bool'>)
Expectation (Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1):
↪  result = False (<class 'bool'>)
```

Success Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).

```
Result (Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1): False
↪  (<class 'bool'>)
Expectation (Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1):
↪  result = False (<class 'bool'>)
```

Info Resetting trigger condition with minute: 22; hour: any; day: [12, 17, 25], month: 2.

Success Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).

```
Result (Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1): False
↪  (<class 'bool'>)
Expectation (Return value for minute: 23; hour: 17; day: 25; month: 02, day_of_week: 1):
↪  result = False (<class 'bool'>)
```

Success Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5 is correct (Content False and Type is <class 'bool'>).

```
Result (Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5): False
↪  (<class 'bool'>)
Expectation (Return value for minute: 45; hour: 12; day: 25; month: 03, day_of_week: 5):
↪  result = False (<class 'bool'>)
```

Success Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1 is correct (Content True and Type is <class 'bool'>).

```
Result (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1): True
↪  (<class 'bool'>)
Expectation (Return value for minute: 22; hour: 17; day: 25; month: 02, day_of_week: 1):
↪  result = True (<class 'bool'>)
```

Success	Return value for minute: 22; hour: 17; day: 25; month: 05, day_of_week: 3 is correct (Content False and Type is <class 'bool'>).

```
Result (Return value for minute: 22; hour: 17; day: 25; month: 05, day_of_week: 3): False
↪  (<class 'bool'>)
Expectation (Return value for minute: 22; hour: 17; day: 25; month: 05, day_of_week: 3):
↪  result = False (<class 'bool'>)
```

Success	Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).

```
Result (Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1): False
↪  (<class 'bool'>)
Expectation (Return value for minute: 45; hour: 14; day: 25; month: 02, day_of_week: 1):
↪  result = False (<class 'bool'>)
```

Success	Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1 is correct (Content False and Type is <class 'bool'>).

```
Result (Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1): False
↪  (<class 'bool'>)
Expectation (Return value for minute: 23; hour: 17; day: 24; month: 02, day_of_week: 1):
↪  result = False (<class 'bool'>)
```

Info	Resetting trigger condition (again).

Success	1st run - execution not needed is correct (Content False and Type is <class 'bool'>).

```
Result (1st run - execution not needed): False (<class 'bool'>)
Expectation (1st run - execution not needed): result = False (<class 'bool'>)
```

Success	2nd run - execution not needed is correct (Content False and Type is <class 'bool'>).

```
Result (2nd run - execution not needed): False (<class 'bool'>)
Expectation (2nd run - execution not needed): result = False (<class 'bool'>)
```

Success	3rd run - execution needed is correct (Content True and Type is <class 'bool'>).

```
Result (3rd run - execution needed): True (<class 'bool'>)
Expectation (3rd run - execution needed): result = True (<class 'bool'>)
```

Success	4th run - execution needed is correct (Content True and Type is <class 'bool'>).

```
Result (4th run - execution needed): True (<class 'bool'>)
Expectation (4th run - execution needed): result = True (<class 'bool'>)
```

Success	5th run - execution not needed is correct (Content False and Type is <class 'bool'>).

```
Result (5th run - execution not needed): False (<class 'bool'>)
Expectation (5th run - execution not needed): result = False (<class 'bool'>)
```

Success	6th run - execution not needed is correct (Content False and Type is <class 'bool'>).

```
Result (6th run - execution not needed): False (<class 'bool'>)
Expectation (6th run - execution not needed): result = False (<class 'bool'>)
```

B.1.9 pylibs.task.crontab: Test crontab

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

Info	Creating Crontab with callback execution in +1 and +3 minutes.

Success	Number of submitted values is correct (Content 2 and Type is <class 'int'>).

```
Crontab accuracy is 30s
Crontab execution number 1 at 1610038571s, requested for 1610038560s
Crontab execution number 2 at 1610038691s, requested for 1610038680s
Result (Timing of crontasks): [ 1610038571, 1610038691 ] (<class 'list'>)
Result (Number of submitted values): 2 (<class 'int'>)
Expectation (Number of submitted values): result = 2 (<class 'int'>)
```

Success	Timing of crontasks: Valueaccuracy and number of submitted values is correct. See detailed log for more information.

```
Result (Submitted value number 1): 1610038571 (<class 'int'>)
Expectation (Submitted value number 1): 1610038560 <= result <= 1610038591
Submitted value number 1 is correct (Content 1610038571 in [1610038560 ... 1610038591] and
↪  Type is <class 'int'>).
Result (Submitted value number 2): 1610038691 (<class 'int'>)
Expectation (Submitted value number 2): 1610038680 <= result <= 1610038711
Submitted value number 2 is correct (Content 1610038691 in [1610038680 ... 1610038711] and
↪  Type is <class 'int'>).
```

C Test-Coverage

C.1 task

The line coverage for task was 98.9%
The branch coverage for task was 98.0%

C.1.1 task.__init__.py

The line coverage for task.__init__.py was 98.9%
The branch coverage for task.__init__.py was 98.0%

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-

"""
task (Task Module)
==================

**Author:**

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

**Description:**

    This Module supports helpfull classes for queues, tasks, ...

**Submodules:**

* :class:`task.crontab`
* :class:`task.delayed`
* :class:`task.periodic`
* :class:`task.queue`
* :class:`task.threaded_queue`

**Unittest:**

        See also the :download:`unittest <task/_testresults_/unittest.pdf>` documentation.

**Module Documentation:**

"""
__DEPENDENCIES__ = []

import logging
import sys
import threading
import time
if sys.version_info >= (3, 0):
    from queue import PriorityQueue
    from queue import Empty
else:
    from Queue import PriorityQueue
    from Queue import Empty

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 task issues like periodic
    tasks, delayed tasks, queues, threaded queues and crontabs.
For more Information read the documentation.""" % __name__.replace('_', '\_')
"""The Module Description"""
__INTERPRETER__ = (2, 3)
"""The Tested Interpreter-Versions"""


```

```
57  class queue(object):
58      """
59      Class to execute queued callbacks.
60
61      :param bool expire: The default value for expire. See also :py:func:`expire`.
62
63      **Example:**
64
65      .. literalinclude:: task/_examples_/tqueue.py
66
67      Will result to the following output:
68
69      .. literalinclude:: task/_examples_/tqueue.log
70      """
71      class job(object):
72          def __init__(self, priority, callback, *args, **kwargs):
73              self.priority = priority
74              self.callback = callback
75              self.args = args
76              self.kwargs = kwargs
77
78          def run(self, queue):
79              self.callback(queue, *self.args, **self.kwargs)
80
81          def __lt__(self, other):
82              return self.priority < other.priority
83
84      def __init__(self, expire=True):
85          self.__expire = expire
86          self.__stop = False
87          self.queue = PriorityQueue()
88
89      def clean_queue(self):
90          """
91          This Methods removes all jobs from the queue.
92
93          .. note:: Be aware that already running jobs will not be terminated.
94          """
95          while not self.queue.empty():
96              try:
97                  self.queue.get(False)
98              except Empty:         # This block is hard to reach for a testcase, but is
99                  continue          # needed, if the thread runs dry while cleaning the queue.
100             self.queue.task_done()
101
102     def enqueue(self, priority, callback, *args, **kwargs):
103         """
104         This enqueues a given callback.
105
106         :param number priority: The priority indication number of this task. The lowest value
    will be queued first.
107         :param callback callback: Callback to be executed
108         :param args args: Arguments to be given to callback
109         :param kwargs kwargs: Keword Arguments to be given to callback
110
111         .. note:: Callback will get this instance as first argument, followed by :py:data:`args`
    und :py:data:`kwargs`.
112         """
113         self.queue.put(self.job(priority, callback, *args, **kwargs))
114
115     def qsize(self):
116         return self.queue.qsize()
```

```
117
118     def run(self):
119         """
120         This starts the execution of the queued callbacks.
121         """
122         self.__stop = False
123         while not self.__stop:
124             try:
125                 self.queue.get(timeout=0.1).run(self)
126             except Empty:
127                 if self.__expire:
128                     break
129         if type(self) is threaded_queue:
130             self.thread = None
131
132     def expire(self):
133         """
134         This sets the expire flag. That means that the process will stop after queue gets empty.
135         """
136         self.__expire = True
137
138     def stop(self):
139         """
140         This sets the stop flag. That means that the process will stop after finishing the active
      task.
141         """
142         self.__stop = True
143
144
145 class threaded_queue(queue):
146     """Class to execute queued callbacks in a background thread (See also parent :py:class:`queue
    `).
147
148     :param bool expire: The default value for expire. See also :py:func:`queue.expire`.
149
150     **Example:**
151
152     .. literalinclude:: task/_examples_/threaded_queue.py
153
154     Will result to the following output:
155
156     .. literalinclude:: task/_examples_/threaded_queue.log
157     """
158     def __init__(self, expire=False):
159         queue.__init__(self, expire=expire)
160         self.thread = None
161
162     def run(self):
163         if self.thread is None:
164             self.thread = threading.Thread(target=self._start, args=())
165             self.thread.daemon = True    # Daemonize thread
166             self.thread.start()          # Start the execution
167
168     def join(self):
169         """
170         This blocks till the queue is empty.
171
172         .. note:: If the queue does not run dry, join will block till the end of the days.
173         """
174         self.expire()
175         if self.thread is not None:
176             self.thread.join()
```

```
177
178     def stop(self):
179         queue.stop(self)
180         self.join()
181
182     def _start(self):
183         queue.run(self)
184
185
186 class periodic(object):
187     """
188     Class to execute a callback cyclicly.
189
190     :param float cycle_time: Cycle time in seconds -- callback will be executed every *cycle_time
    * seconds
191     :param callback callback: Callback to be executed
192     :param args args: Arguments to be given to the callback
193     :param kwargs kwargs: Keword Arguments to be given to callback
194
195     .. note:: The Callback will get this instance as first argument, followed by :py:data:`args`
    und :py:data:`kwargs`.
196
197     **Example:**
198
199     .. literalinclude:: task/_examples_/periodic.py
200
201     Will result to the following output:
202
203     .. literalinclude:: task/_examples_/periodic.log
204     """
205     def __init__(self, cycle_time, callback, *args, **kwargs):
206         self._lock = threading.Lock()
207         self._timer = None
208         self.callback = callback
209         self.cycle_time = cycle_time
210         self.args = args
211         self.kwargs = kwargs
212         self._stopped = True
213         self._last_tm = None
214         self.dt = None
215
216     def join(self):
217         """
218         This blocks till the cyclic task is terminated.
219
220         .. note:: Using join means that somewhere has to be a condition calling :py:func:`stop`
    to terminate. Otherwise :func:`task.join` will never return.
221         """
222         while not self._stopped:
223             time.sleep(.1)
224
225     def run(self):
226         """
227         This starts the cyclic execution of the given callback.
228         """
229         if self._stopped:
230             self._set_timer(force_now=True)
231
232     def stop(self):
233         """
234         This stops the execution of any further task.
235         """
```

```
236         self._lock.acquire()
237         self._stopped = True
238         if self._timer is not None:
239             self._timer.cancel()
240         self._lock.release()
241 
242     def _set_timer(self, force_now=False):
243         """
244         This sets the timer for the execution of the next task.
245         """
246         self._lock.acquire()
247         self._stopped = False
248         if force_now:
249             self._timer = threading.Timer(0, self._start)
250         else:
251             self._timer = threading.Timer(self.cycle_time, self._start)
252         self._timer.start()
253         self._lock.release()
254 
255     def _start(self):
256         tm = time.time()
257         if self._last_tm is not None:
258             self.dt = tm - self._last_tm
259         self._set_timer(force_now=False)
260         self.callback(self, *self.args, **self.kwargs)
261         self._last_tm = tm
262 
263 
264 class delayed(periodic):
265     """Class to execute a callback a given time in the future. See also parent :py:class:`
    periodic`.
266 
267     :param float time: Delay time for execution of the given callback
268     :param callback callback: Callback to be executed
269     :param args args: Arguments to be given to callback
270     :param kwargs kwargs: Keword Arguments to be given to callback
271 
272     **Example:**
273 
274     .. literalinclude:: task/_examples_/delayed.py
275 
276     Will result to the following output:
277 
278     .. literalinclude:: task/_examples_/delayed.log
279     """
280     def run(self):
281         """
282         This starts the timer for the delayed execution.
283         """
284         self._set_timer(force_now=False)
285 
286     def _start(self):
287         self.callback(*self.args, **self.kwargs)
288         self.stop()
289 
290 
291 class crontab(periodic):
```

```
292     """Class to execute a callback at the specified time conditions. See also parent :py:class:`
        periodic`.
293
294     :param accuracy: Repeat time in seconds for background task checking event triggering. This
        time is the maximum delay between specified time condition and the execution.
295     :type accuracy: float
296
297     **Example:**
298
299     .. literalinclude:: task/_examples_/crontab.py
300
301     Will result to the following output:
302
303     .. literalinclude:: task/_examples_/crontab.log
304     """
305     ANY = '*'
306     """Constant for matching every condition."""
307
308     class cronjob(object):
309         """Class to handle cronjob parameters and cronjob changes.
310
311         :param minute: Minute for execution. Either 0...59, [0...59, 0...59, ...] or :py:const:`
        crontab.ANY` for every Minute.
312         :type minute: int, list, str
313         :param hour: Hour for execution. Either 0...23, [0...23, 0...23, ...] or :py:const:`
        crontab.ANY` for every Hour.
314         :type hour: int, list, str
315         :param day_of_month: Day of Month for execution. Either 0...31, [0...31, 0...31, ...] or
        :py:const:`crontab.ANY` for every Day of Month.
316         :type day_of_month: int, list, str
317         :param month: Month for execution. Either 0...12, [0...12, 0...12, ...] or :py:const:`
        crontab.ANY` for every Month.
318         :type month: int, list, str
319         :param day_of_week: Day of Week for execution. Either 0...6, [0...6, 0...6, ...] or :py:
        const:`crontab.ANY` for every Day of Week.
320         :type day_of_week: int, list, str
321         :param callback: The callback to be executed. The instance of :py:class:`cronjob` will be
         given as the first, args and kwargs as the following parameters.
322         :type callback: func
323
324         .. note:: This class should not be used stand alone. An instance will be created by
        adding a cronjob by using :py:func:`crontab.add_cronjob()`.
325         """
326         class all_match(set):
327             """Universal set - match everything"""
328             def __contains__(self, item):
329                 (item)
330                 return True
331
332         def __init__(self, minute, hour, day_of_month, month, day_of_week, callback, *args, **
        kwargs):
333             self.set_trigger_conditions(minute or crontab.ANY, hour or crontab.ANY, day_of_month
        or crontab.ANY, month or crontab.ANY, day_of_week or crontab.ANY)
334             self.callback = callback
335             self.args = args
336             self.kwargs = kwargs
337             self.__last_cron_check_time__ = None
338             self.__last_execution__ = None
339
340         def set_trigger_conditions(self, minute=None, hour=None, day_of_month=None, month=None,
        day_of_week=None):
```

```
341             """This Method changes the execution parameters.
342 
343             :param minute: Minute for execution. Either 0...59, [0...59, 0...59, ...] or :py:
    const:`crontab.ANY` for every Minute.
344             :type minute: int, list, str
345             :param hour: Hour for execution. Either 0...23, [0...23, 0...23, ...] or :py:const:`
    crontab.ANY` for every Hour.
346             :type hour: int, list, str
347             :param day_of_month: Day of Month for execution. Either 0...31, [0...31, 0...31, ...]
     or :py:const:`crontab.ANY` for every Day of Month.
348             :type day_of_month: int, list, str
349             :param month: Month for execution. Either 0...12, [0...12, 0...12, ...] or :py:const
    :`crontab.ANY` for every Month.
350             :type month: int, list, str
351             :param day_of_week: Day of Week for execution. Either 0...6, [0...6, 0...6, ...] or :
    py:const:`crontab.ANY` for every Day of Week.
352             :type day_of_week: int, list, str
353             """
354             if minute is not None:
355                 self.minute = self.__conv_to_set__(minute)
356             if hour is not None:
357                 self.hour = self.__conv_to_set__(hour)
358             if day_of_month is not None:
359                 self.day_of_month = self.__conv_to_set__(day_of_month)
360             if month is not None:
361                 self.month = self.__conv_to_set__(month)
362             if day_of_week is not None:
363                 self.day_of_week = self.__conv_to_set__(day_of_week)
364 
365         def __conv_to_set__(self, obj):
366             if obj is crontab.ANY:
367                 return self.all_match()
368             elif isinstance(obj, (int, long) if sys.version_info < (3,0) else (int)):
369                 return set([obj])
370             else:
371                 return set(obj)
372 
373         def __execution_needed_for__(self, minute, hour, day_of_month, month, day_of_week):
374             if self.__last_execution__ != [minute, hour, day_of_month, month, day_of_week]:
375                 if minute in self.minute and hour in self.hour and day_of_month in self.
    day_of_month and month in self.month and day_of_week in self.day_of_week:
376                     return True
377             return False
378 
379         def __store_execution_reminder__(self, minute, hour, day_of_month, month, day_of_week):
380             self.__last_execution__ = [minute, hour, day_of_month, month, day_of_week]
381 
382         def cron_execution(self, tm):
383             """This Methods executes the Cron-Callback, if a execution is needed for the given
    time (depending on the parameters on initialisation)
384 
385             :param tm: (Current) Time Value to be checked. The time needs to be given in seconds
    since 1970 (e.g. generated by int(time.time())).
386             :type tm: int
387             """
388             if self.__last_cron_check_time__ is None:
389                 self.__last_cron_check_time__ = tm - 1
390             #
391             for t in range(self.__last_cron_check_time__ + 1, tm + 1):
392                 lt = time.localtime(t)
393                 if self.__execution_needed_for__(lt[4], lt[3], lt[2], lt[1], lt[6]):
394                     self.callback(self, *self.args, **self.kwargs)
395                     self.__store_execution_reminder__(lt[4], lt[3], lt[2], lt[1], lt[6])
396                     break
397             self.__last_cron_check_time__ = tm
```

```
398
399     def __init__(self, accuracy=30):
400         periodic.__init__(self, accuracy, self.__periodic__)
401         self.__crontab__ = []
402
403     def __periodic__(self, rt):
404         (rt)
405         tm = int(time.time())
406         for cronjob in self.__crontab__:
407             cronjob.cron_execution(tm)
408
409     def add_cronjob(self, minute, hour, day_of_month, month, day_of_week, callback, *args, **
    kwargs):
410         """This Method adds a cronjob to be executed.
411
412         :param minute: Minute for execution. Either 0...59, [0...59, 0...59, ...] or :py:const:`
    crontab.ANY` for every Minute.
413         :type minute: int, list, str
414         :param hour: Hour for execution. Either 0...23, [0...23, 0...23, ...] or :py:const:`
    crontab.ANY` for every Hour.
415         :type hour: int, list, str
416         :param day_of_month: Day of Month for execution. Either 0...31, [0...31, 0...31, ...] or
    :py:const:`crontab.ANY` for every Day of Month.
417         :type day_of_month: int, list, str
418         :param month: Month for execution. Either 0...12, [0...12, 0...12, ...] or :py:const:`
    crontab.ANY` for every Month.
419         :type month: int, list, str
420         :param day_of_week: Day of Week for execution. Either 0...6, [0...6, 0...6, ...] or :py:
    const:`crontab.ANY` for every Day of Week.
421         :type day_of_week: int, list, str
422         :param callback: The callback to be executed. The instance of :py:class:`cronjob` will be
     given as the first, args and kwargs as the following parameters.
423         :type callback: func
424
425         .. note:: The ``callback`` will be executed with it's instance of :py:class:`cronjob` as
    the first parameter.
426             The given Arguments (:data:`args`) and keyword Arguments (:data:`kwargs`) will be
    stored in that object.
427         """
428         self.__crontab__.append(self.cronjob(minute, hour, day_of_month, month, day_of_week,
    callback, *args, **kwargs))
```