Unittest for `task`

December 27, 2019

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	138e2db63e5416bcfc110e775fb54e4c
Dependencies	

1.2 Unittest Information

Unittest Information	
Version	bf12903e8541ad442a6d670b0e5f89b9
Testruns with	python 2.7.17 (final), python 3.6.9 (final)

1.3 Test System Information

System Information	
Architecture	64bit
Distribution	LinuxMint 19.3 tricia
Hostname	ahorn
Kernel	5.0.0-37-generic (#40 18.04.1-Ubuntu SMP Thu Nov 14 12:06:39 UTC 2019)
Machine	x86_64
Path	/user_data/data/dirk/prj/modules/task/unittest
System	Linux
Username	dirk

2 Statistic

2.1 Test-Statistic for testrun with python 2.7.17 (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	216.908s

2.2 Test-Statistic for testrun with python 3.6.9 (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	216.877s

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 2.7.17 (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 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (28)
Start-Time:	2019-12-27 08:21:07,418
Finished-Time:	2019-12-27 08:21:07,427
Time-Consumption	0.009s

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 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (29)
Start-Time:	2019-12-27 08:21:07,428
Finished-Time:	2019-12-27 08:24:37,527
Time-Consumption	210.100s

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 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (21)
Start-Time:	2019-12-27 08:21:00,318
Finished-Time:	2019-12-27 08:21:00,828
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 0.25037693977355957 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 0.010622024536132812 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 0.005093097686767578 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 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (22)
Start-Time:	2019-12-27 08:21:00,829
Finished-Time:	2019-12-27 08:21:03,371
Time-Consumption	2.542s

Testsummary:

Info	Running a periodic task for 10 cycles with a cycletime of 0.25s
Success	Minimum cycle time is correct (Content 0.2503390312194824 in [0.2465 ... 0.2545] and Type is <type 'float'>).
Success	Mean cycle time is correct (Content 0.25071009000142414 in [0.2465 ... 0.2545] and Type is <type 'float'>).
Success	Maximum cycle time is correct (Content 0.2511889934539795 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 0.010482072830200195 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).
Success	Mean cycle time is correct (Content 0.01073855823940701 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).
Success	Maximum cycle time is correct (Content 0.011135101318359375 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 0.0053789615631103516 in [0.00395 ... 0.00705] and Type is <type 'float'>).
Success	Mean cycle time is correct (Content 0.0056752363840738935 in [0.00395 ... 0.00705] and Type is <type 'float'>).
Success	Maximum cycle time is correct (Content 0.006085872650146484 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 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (25)
Start-Time:	2019-12-27 08:21:03,585
Finished-Time:	2019-12-27 08:21:03,589
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 <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 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (23)
Start-Time:	2019-12-27 08:21:03,371
Finished-Time:	2019-12-27 08:21:03,477
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 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (24)
Start-Time:	2019-12-27 08:21:03,477
Finished-Time:	2019-12-27 08:21:03,584
Time-Consumption	0.107s

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 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (27)
Start-Time:	2019-12-27 08:21:06,512
Finished-Time:	2019-12-27 08:21:07,120
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 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (26)
Start-Time:	2019-12-27 08:21:03,589
Finished-Time:	2019-12-27 08:21:06,511
Time-Consumption	2.922s

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 3.6.9 (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 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (28)
Start-Time:	2019-12-27 08:24:45,074
Finished-Time:	2019-12-27 08:24:45,086
Time-Consumption	0.012s

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 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (29)
Start-Time:	2019-12-27 08:24:45,086
Finished-Time:	2019-12-27 08:28:15,166
Time-Consumption	210.081s

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 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (21)
Start-Time:	2019-12-27 08:24:37,990
Finished-Time:	2019-12-27 08:24:38,499
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 0.25007009506225586 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 0.010076522827148438 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 0.00506138801574707 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 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (22)
Start-Time:	2019-12-27 08:24:38,500
Finished-Time:	2019-12-27 08:24:41,042
Time-Consumption	2.542s

Testsummary:

Info	Running a periodic task for 10 cycles with a cycletime of 0.25s
Success	Minimum cycle time is correct (Content 0.2501676082611084 in [0.2465 ... 0.2545] and Type is <class 'float'>).
Success	Mean cycle time is correct (Content 0.2504596445295546 in [0.2465 ... 0.2545] and Type is <class 'float'>).
Success	Maximum cycle time is correct (Content 0.2506415843963623 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 0.01018214225769043 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).
Success	Mean cycle time is correct (Content 0.010425064298841689 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).
Success	Maximum cycle time is correct (Content 0.010566234588623047 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 0.005247592926025391 in [0.00395 ... 0.00705] and Type is <class 'float'>).
Success	Mean cycle time is correct (Content 0.00538325309753418 in [0.00395 ... 0.00705] and Type is <class 'float'>).
Success	Maximum cycle time is correct (Content 0.005558967590332031 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 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (25)

Start-Time:	2019-12-27 08:24:41,253
Finished-Time:	2019-12-27 08:24:41,257
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 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (23)
Start-Time:	2019-12-27 08:24:41,042
Finished-Time:	2019-12-27 08:24:41,148
Time-Consumption	0.105s

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 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (24)
Start-Time:	2019-12-27 08:24:41,148
Finished-Time:	2019-12-27 08:24:41,253
Time-Consumption	0.105s

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 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (27)
Start-Time:	2019-12-27 08:24:44,171
Finished-Time:	2019-12-27 08:24:44,776
Time-Consumption	0.605s

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 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/task/unittest/src/tests/__init__.py (26)
Start-Time:	2019-12-27 08:24:41,257
Finished-Time:	2019-12-27 08:24:44,171
Time-Consumption	2.914s

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 2.7.17 (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 0.25037693977355957 in [0.2465 ... 0.2545] and Type is <type 'float'>).

```
Result (Time consumption): 0.25037693977355957 (<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 0.010622024536132812 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).

```
Result (Time consumption): 0.010622024536132812 (<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 0.005093097686767578 in [0.00395 ... 0.00705] and Type is <type 'float'>).

```
Result (Time consumption): 0.005093097686767578 (<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 1577431260.830670
Task execution number 2 at 1577431261.081857
Task execution number 3 at 1577431261.332465
Task execution number 4 at 1577431261.583169
Task execution number 5 at 1577431261.833508
Task execution number 6 at 1577431262.083997
Task execution number 7 at 1577431262.334639
Task execution number 8 at 1577431262.585828
Task execution number 9 at 1577431262.836671
Task execution number 10 at 1577431263.087061
```

Success	Minimum cycle time is correct (Content 0.2503390312194824 in [0.2465 ... 0.2545] and Type is <type 'float'>).

```
Result (Minimum cycle time): 0.2503390312194824 (<type 'float'>)
Expectation (Minimum cycle time): 0.2465 <= result <= 0.2545
```

Success	Mean cycle time is correct (Content 0.25071009000142414 in [0.2465 ... 0.2545] and Type is <type 'float'>).

```
Result (Mean cycle time): 0.25071009000142414 (<type 'float'>)
Expectation (Mean cycle time): 0.2465 <= result <= 0.2545
```

Success	Maximum cycle time is correct (Content 0.2511889934539795 in [0.2465 ... 0.2565] and Type is <type 'float'>).

```
Result (Maximum cycle time): 0.2511889934539795 (<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 1577431263.136190
Task execution number 2 at 1577431263.146672
Task execution number 3 at 1577431263.157228
Task execution number 4 at 1577431263.168363
Task execution number 5 at 1577431263.178933
Task execution number 6 at 1577431263.189998
Task execution number 7 at 1577431263.200579
Task execution number 8 at 1577431263.211154
Task execution number 9 at 1577431263.221770
Task execution number 10 at 1577431263.232837
```

Success	Minimum cycle time is correct (Content 0.010482072830200195 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).

```
Result (Minimum cycle time): 0.010482072830200195 (<type 'float'>)
Expectation (Minimum cycle time): 0.008900000000000002 <= result <= 0.0121
```

Success	Mean cycle time is correct (Content 0.01073855823940701 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).

```
Result (Mean cycle time): 0.01073855823940701 (<type 'float'>)
Expectation (Mean cycle time): 0.008900000000000002 <= result <= 0.0121
```

Success	Maximum cycle time is correct (Content 0.011135101318359375 in [0.008900000000000002 ... 0.0141] and Type is <type 'float'>).

```
Result (Maximum cycle time): 0.011135101318359375 (<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 1577431263.259307
Task execution number 2 at 1577431263.264840
Task execution number 3 at 1577431263.270685
Task execution number 4 at 1577431263.276064
Task execution number 5 at 1577431263.281601
Task execution number 6 at 1577431263.287635
Task execution number 7 at 1577431263.293169
Task execution number 8 at 1577431263.299255
Task execution number 9 at 1577431263.304808
Task execution number 10 at 1577431263.310384
```

Success	Minimum cycle time is correct (Content 0.0053789615631103516 in [0.00395 ... 0.00705] and Type is <type 'float'>).

```
Result (Minimum cycle time): 0.0053789615631103516 (<type 'float'>)
Expectation (Minimum cycle time): 0.00395 <= result <= 0.00705
```

Success	Mean cycle time is correct (Content 0.0056752363840738935 in [0.00395 ... 0.00705] and Type is <type 'float'>).

```
Result (Mean cycle time): 0.0056752363840738935 (<type 'float'>)
Expectation (Mean cycle time): 0.00395 <= result <= 0.00705
```

Success	Maximum cycle time is correct (Content 0.006085872650146484 in [0.00395 ... 0.009049999999999999] and Type is <type 'float'>).

```
Result (Maximum cycle time): 0.006085872650146484 (<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 1577431327s, requested for 1577431320s
Crontab execution number 2 at 1577431447s, requested for 1577431440s
Result (Timing of crontasks): [ 1577431327, 1577431447 ] (<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): 1577431327 (<type 'int'>)
Expectation (Submitted value number 1): 1577431320 <= result <= 1577431351
Submitted value number 1 is correct (Content 1577431327 in [1577431320 ... 1577431351] and
↪  Type is <type 'int'>).
Result (Submitted value number 2): 1577431447 (<type 'int'>)
Expectation (Submitted value number 2): 1577431440 <= result <= 1577431471
Submitted value number 2 is correct (Content 1577431447 in [1577431440 ... 1577431471] and
↪  Type is <type 'int'>).
```

B Trace for testrun with python 3.6.9 (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 0.25007009506225586 in [0.2465 ... 0.2545] and Type is <class 'float'>).

```
Result (Time consumption): 0.25007009506225586 (<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 0.010076522827148438 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).

```
Result (Time consumption): 0.010076522827148438 (<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 0.00506138801574707 in [0.00395 ... 0.00705] and Type is <class 'float'>).

```
Result (Time consumption): 0.00506138801574707 (<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 1577431478.501697
Task execution number 2 at 1577431478.752308
Task execution number 3 at 1577431479.002476
Task execution number 4 at 1577431479.252823
Task execution number 5 at 1577431479.503223
Task execution number 6 at 1577431479.753506
Task execution number 7 at 1577431480.004085
Task execution number 8 at 1577431480.254681
Task execution number 9 at 1577431480.505192
Task execution number 10 at 1577431480.755834
```

Success	Minimum cycle time is correct (Content 0.2501676082611084 in [0.2465 ... 0.2545] and Type is <class 'float'>).

```
Result (Minimum cycle time): 0.2501676082611084 (<class 'float'>)
Expectation (Minimum cycle time): 0.2465 <= result <= 0.2545
```

Success	Mean cycle time is correct (Content 0.2504596445295546 in [0.2465 ... 0.2545] and Type is <class 'float'>).

```
Result (Mean cycle time): 0.2504596445295546 (<class 'float'>)
Expectation (Mean cycle time): 0.2465 <= result <= 0.2545
```

Success	Maximum cycle time is correct (Content 0.2506415843963623 in [0.2465 ... 0.2565] and Type is <class 'float'>).

```
Result (Maximum cycle time): 0.2506415843963623 (<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 1577431480.807815
Task execution number 2 at 1577431480.818227
Task execution number 3 at 1577431480.828409
Task execution number 4 at 1577431480.838654
Task execution number 5 at 1577431480.849118
Task execution number 6 at 1577431480.859474
Task execution number 7 at 1577431480.870033
Task execution number 8 at 1577431480.880509
Task execution number 9 at 1577431480.891075
Task execution number 10 at 1577431480.901641
```

Success	Minimum cycle time is correct (Content 0.01018214225769043 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).

```
Result (Minimum cycle time): 0.01018214225769043 (<class 'float'>)
Expectation (Minimum cycle time): 0.008900000000000002 <= result <= 0.0121
```

Success	Mean cycle time is correct (Content 0.010425064298841689 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).

```
Result (Mean cycle time): 0.010425064298841689 (<class 'float'>)
Expectation (Mean cycle time): 0.008900000000000002 <= result <= 0.0121
```

Success	Maximum cycle time is correct (Content 0.010566234588623047 in [0.008900000000000002 ... 0.0141] and Type is <class 'float'>).

```
Result (Maximum cycle time): 0.010566234588623047 (<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 1577431480.930800
Task execution number 2 at 1577431480.936359
Task execution number 3 at 1577431480.941607
Task execution number 4 at 1577431480.946866
Task execution number 5 at 1577431480.952171
Task execution number 6 at 1577431480.957479
Task execution number 7 at 1577431480.962825
Task execution number 8 at 1577431480.968287
Task execution number 9 at 1577431480.973785
Task execution number 10 at 1577431480.979250
```

Success	Minimum cycle time is correct (Content 0.005247592926025391 in [0.00395 ... 0.00705] and Type is <class 'float'>).

```
Result (Minimum cycle time): 0.005247592926025391 (<class 'float'>)
Expectation (Minimum cycle time): 0.00395 <= result <= 0.00705
```

Success	Mean cycle time is correct (Content 0.00538325309753418 in [0.00395 ... 0.00705] and Type is <class 'float'>).

```
Result (Mean cycle time): 0.00538325309753418 (<class 'float'>)
Expectation (Mean cycle time): 0.00395 <= result <= 0.00705
```

Success	Maximum cycle time is correct (Content 0.005558967590332031 in [0.00395 ... 0.009049999999999999] and Type is <class 'float'>).

```
Result (Maximum cycle time): 0.005558967590332031 (<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 1577431515s, requested for 1577431500s
Crontab execution number 2 at 1577431635s, requested for 1577431620s
Result (Timing of crontasks): [ 1577431515, 1577431635 ] (<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): 1577431515 (<class 'int'>)
Expectation (Submitted value number 1): 1577431500 <= result <= 1577431531
Submitted value number 1 is correct (Content 1577431515 in [1577431500 ... 1577431531] and
↪  Type is <class 'int'>).
Result (Submitted value number 2): 1577431635 (<class 'int'>)
Expectation (Submitted value number 2): 1577431620 <= result <= 1577431651
Submitted value number 2 is correct (Content 1577431635 in [1577431620 ... 1577431651] 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.
"""
__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

logger_name = 'TASK'
logger = logging.getLogger(logger_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"""


class queue(object):
```

```
52      """Class to execute queued methods.
53
54      :param bool expire: The default value for expire. See also :py:func:`expire`.
55
56      **Example:**
57
58      .. literalinclude:: ../../task/_examples_/queue.py
59
60      Will result to the following output:
61
62      .. literalinclude:: ../../task/_examples_/queue.log
63      """
64      class job(object):
65          def __init__(self, priority, callback, *args, **kwargs):
66              self.priority = priority
67              self.callback = callback
68              self.args = args
69              self.kwargs = kwargs
70
71          def run(self, queue):
72              self.callback(queue, *self.args, **self.kwargs)
73
74          def __lt__(self, other):
75              return self.priority < other.priority
76
77      def __init__(self, expire=True):
78          self.__expire = expire
79          self.__stop = False
80          self.queue = PriorityQueue()
81
82      def clean_queue(self):
83          """
84          This Methods removes all jobs from the queue.
85
86          .. note:: Be aware that already runnung jobs will not be terminated.
87          """
88          while not self.queue.empty():
89              try:
90                  self.queue.get(False)
91              except Empty:         # This block is hard to reach for a testcase, but is
92                  continue          # needed, if the thread runs dry while cleaning the queue.
93              self.queue.task_done()
94
95      def enqueue(self, priority, method, *args, **kwargs):
96          """
97          This enqueues a given callback.
98
99          :param number priority: The priority indication number of this task. The lowest value
    will be queued first.
100         :param method method: Method to be executed
101         :param args args: Arguments to be given to method
102         :param kwargs kwargs: Kewordsarguments to be given to method
103
104         .. note:: Called method will get this instance as first argument, followed by :py:data:`
    args` und :py:data:`kwargs`.
105         """
106         self.queue.put(self.job(priority, method, *args, **kwargs))
107
108     def qsize(self):
109         return self.queue.qsize()
110
111     def run(self):
```

```
112         """
113         This starts the execution of the queued methods.
114         """
115         self.__stop = False
116         while not self.__stop:
117             try:
118                 self.queue.get(timeout=0.1).run(self)
119             except Empty:
120                 if self.__expire:
121                     break
122         if type(self) is threaded_queue:
123             self.thread = None
124 
125     def expire(self):
126         """
127         This sets the expire flag. That means that the process will stop after queue gets empty.
128         """
129         self.__expire = True
130 
131     def stop(self):
132         """
133         This sets the stop flag. That means that the process will stop after finishing the active
      task.
134         """
135         self.__stop = True
136 
137 
138 class threaded_queue(queue):
139     """Class to execute queued methods in a background thread (See also parent :py:class:`queue`)
    .
140 
141     :param bool expire: The default value for expire. See also :py:func:`queue.expire`.
142 
143     **Example:**
144 
145     .. literalinclude:: ../../task/_examples_/threaded_queue.py
146 
147     Will result to the following output:
148 
149     .. literalinclude:: ../../task/_examples_/threaded_queue.log
150     """
151     def __init__(self, expire=False):
152         queue.__init__(self, expire=expire)
153         self.thread = None
154 
155     def run(self):
156         if self.thread is None:
157             self.thread = threading.Thread(target=self._start, args=())
158             self.thread.daemon = True    # Daemonize thread
159             self.thread.start()          # Start the execution
160 
161     def join(self):
162         """
163         This blocks till the queue is empty.
164 
165         .. note:: If the queue does not run dry, join will block till the end of the days.
166         """
167         self.expire()
168         if self.thread is not None:
169             self.thread.join()
170 
```

```
171     def stop(self):
172         queue.stop(self)
173         self.join()
174 
175     def _start(self):
176         queue.run(self)
177 
178 
179 class periodic(object):
180     """
181     :param float cycle_time: Cycle time in seconds -- method will be executed every *cycle_time* seconds
182     :param method method: Method to be executed
183     :param args args: Arguments to be given to method
184     :param kwargs kwargs: Kewordsarguments to be given to method
185 
186     Class to execute a method cyclicly.
187 
188     .. note:: Called method will get this instance as first argument, followed by :py:data:`args` und :py:data:`kwargs`.
189 
190     **Example:**
191 
192     .. literalinclude:: ../../task/_examples_/periodic.py
193 
194     Will result to the following output:
195 
196     .. literalinclude:: ../../task/_examples_/periodic.log
197     """
198     def __init__(self, cycle_time, method, *args, **kwargs):
199         self._lock = threading.Lock()
200         self._timer = None
201         self.method = method
202         self.cycle_time = cycle_time
203         self.args = args
204         self.kwargs = kwargs
205         self._stopped = True
206         self._last_tm = None
207         self.dt = None
208 
209     def join(self, timeout=0.1):
210         """
211         This blocks till the cyclic task is terminated.
212 
213         :param float timeout: Cycle time for checking if task is stopped
214 
215         .. note:: Using join means that somewhere has to be a condition calling :py:func:`stop` to terminate.
216         """
217         while not self._stopped:
218             time.sleep(timeout)
219 
220     def run(self):
221         """
222         This starts the cyclic execution of the given method.
223         """
224         if self._stopped:
225             self._set_timer(force_now=True)
226 
227     def stop(self):
228         """
229         This stops the execution of any following task.
230         """
```

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

```
287     """Class to execute a callback at the specified time conditions. See also parent :py:class:`
        periodic`.
288
289     :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.
290     :type accuracy: float
291
292     **Example:**
293
294     .. literalinclude:: ../../task/_examples_/crontab.py
295
296     Will result to the following output:
297
298     .. literalinclude:: ../../task/_examples_/crontab.log
299     """
300     ANY = '*'
301     """Constant for matching every condition."""
302
303     class cronjob(object):
304         """Class to handle cronjob parameters and cronjob changes.
305
306         :param minute: Minute for execution. Either 0...59, [0...59, 0...59, ...] or :py:const:`
        crontab.ANY` for every Minute.
307         :type minute: int, list, str
308         :param hour: Hour for execution. Either 0...23, [0...23, 0...23, ...] or :py:const:`
        crontab.ANY` for every Hour.
309         :type hour: int, list, str
310         :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.
311         :type day_of_month: int, list, str
312         :param month: Month for execution. Either 0...12, [0...12, 0...12, ...] or :py:const:`
        crontab.ANY` for every Month.
313         :type month: int, list, str
314         :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.
315         :type day_of_week: int, list, str
316         :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.
317         :type callback: func
318
319         .. 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()`.
320         """
321         class all_match(set):
322             """Universal set - match everything"""
323             def __contains__(self, item):
324                 (item)
325                 return True
326
327         def __init__(self, minute, hour, day_of_month, month, day_of_week, callback, *args, **
        kwargs):
328             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)
329             self.callback = callback
330             self.args = args
331             self.kwargs = kwargs
332             self.__last_cron_check_time__ = None
333             self.__last_execution__ = None
334
335         def set_trigger_conditions(self, minute=None, hour=None, day_of_month=None, month=None,
        day_of_week=None):
```

```
336             """This Method changes the execution parameters.
337 
338             :param minute: Minute for execution. Either 0...59, [0...59, 0...59, ...] or :py:
    const:`crontab.ANY` for every Minute.
339             :type minute: int, list, str
340             :param hour: Hour for execution. Either 0...23, [0...23, 0...23, ...] or :py:const:`
    crontab.ANY` for every Hour.
341             :type hour: int, list, str
342             :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.
343             :type day_of_month: int, list, str
344             :param month: Month for execution. Either 0...12, [0...12, 0...12, ...] or :py:const
    :`crontab.ANY` for every Month.
345             :type month: int, list, str
346             :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.
347             :type day_of_week: int, list, str
348             """
349             if minute is not None:
350                 self.minute = self.__conv_to_set__(minute)
351             if hour is not None:
352                 self.hour = self.__conv_to_set__(hour)
353             if day_of_month is not None:
354                 self.day_of_month = self.__conv_to_set__(day_of_month)
355             if month is not None:
356                 self.month = self.__conv_to_set__(month)
357             if day_of_week is not None:
358                 self.day_of_week = self.__conv_to_set__(day_of_week)
359 
360         def __conv_to_set__(self, obj):
361             if obj is crontab.ANY:
362                 return self.all_match()
363             elif isinstance(obj, (int, long) if sys.version_info < (3,0) else (int)):
364                 return set([obj])
365             else:
366                 return set(obj)
367 
368         def __execution_needed_for__(self, minute, hour, day_of_month, month, day_of_week):
369             if self.__last_execution__ != [minute, hour, day_of_month, month, day_of_week]:
370                 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:
371                     return True
372             return False
373 
374         def __store_execution_reminder__(self, minute, hour, day_of_month, month, day_of_week):
375             self.__last_execution__ = [minute, hour, day_of_month, month, day_of_week]
376 
377         def cron_execution(self, tm):
378             """This Methods executes the Cron-Callback, if a execution is needed for the given
    time (depending on the parameters on initialisation)
379 
380             :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())).
381             :type tm: int
382             """
383             if self.__last_cron_check_time__ is None:
384                 self.__last_cron_check_time__ = tm - 1
385             #
386             for t in range(self.__last_cron_check_time__ + 1, tm + 1):
387                 lt = time.localtime(t)
388                 if self.__execution_needed_for__(lt[4], lt[3], lt[2], lt[1], lt[6]):
389                     self.callback(self, *self.args, **self.kwargs)
390                     self.__store_execution_reminder__(lt[4], lt[3], lt[2], lt[1], lt[6])
391                     break
392             self.__last_cron_check_time__ = tm
```

```
393
394     def __init__(self, accuracy=30):
395         periodic.__init__(self, accuracy, self.__periodic__)
396         self.__crontab__ = []
397
398     def __periodic__(self, rt):
399         (rt)
400         tm = int(time.time())
401         for cronjob in self.__crontab__:
402             cronjob.cron_execution(tm)
403
404     def add_cronjob(self, minute, hour, day_of_month, month, day_of_week, callback, *args, **
    kwargs):
405         """This Method adds a cronjob to be executed.
406
407         :param minute: Minute for execution. Either 0...59, [0...59, 0...59, ...] or :py:const:`
    crontab.ANY` for every Minute.
408         :type minute: int, list, str
409         :param hour: Hour for execution. Either 0...23, [0...23, 0...23, ...] or :py:const:`
    crontab.ANY` for every Hour.
410         :type hour: int, list, str
411         :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.
412         :type day_of_month: int, list, str
413         :param month: Month for execution. Either 0...12, [0...12, 0...12, ...] or :py:const:`
    crontab.ANY` for every Month.
414         :type month: int, list, str
415         :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.
416         :type day_of_week: int, list, str
417         :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.
418         :type callback: func
419
420         .. note:: The ``callback`` will be executed with it's instance of :py:class:`cronjob` as
    the first parameter.
421         """
422         self.__crontab__.append(self.cronjob(minute, hour, day_of_month, month, day_of_week,
    callback, *args, **kwargs))
```