Unittest for `task`

January 11, 2021

Contents

1 Test Information

1.1 Test Candidate Information

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

Library Information	
Name	task
State	Released
Supported Interpreters	python2, python3
Version	d03c7bd7995b3c967ec523eaddf376d5
Dependencies	

1.2 Unittest Information

Unittest Information	
Version	0de92de1eb874ac24955dd6f67631bee
Testruns with	python 2.7.18 (final), python 3.8.5 (final)

1.3 Test System Information

System Information	
Architecture	64bit
Distribution	Linux Mint 20.1 ulyssa
Hostname	ahorn
Kernel	5.4.0-60-generic (#67-Ubuntu SMP Tue Jan 5 18:31:36 UTC 2021)
Machine	x86_64
Path	/user_data/data/dirk/prj/unittest/task/unittest
System	Linux
Username	dirk

2 Statistic

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

Number of tests	**9**
Number of successfull tests	**9**
Number of possibly failed tests	**0**
Number of failed tests	**0**
Executionlevel	Full Test (all defined tests)
Time consumption	217.112s

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

Number of tests	**9**
Number of successfull tests	**9**
Number of possibly failed tests	**0**
Number of failed tests	**0**
Executionlevel	Full Test (all defined tests)
Time consumption	217.163s

2.3 Coverage Statistic

Module- or Filename	Line-Coverage	Branch-Coverage
task	98.9%	98.1%
task.__init__.py	98.9%	

3 Testcases with no corresponding Requirement

3.1 Summary for testrun with python 2.7.18 (final)

3.1.1 pylibs.task.crontab: Test cronjob

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (28)
Start-Time:	2021-01-11 01:01:42,776
Finished-Time:	2021-01-11 01:01:42,785
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 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (29)
Start-Time:	2021-01-11 01:01:42,786
Finished-Time:	2021-01-11 01:05:12,886
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 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (21)
Start-Time:	2021-01-11 01:01:35,473
Finished-Time:	2021-01-11 01:01:35,984
Time-Consumption	0.511s

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 0.2502110004425049 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 0.01007699966430664 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 0.005555868148803711 in [0.00395 ... 0.00705] and Type is <type 'float'>).

3.1.4 pylibs.task.periodic: Test periodic execution

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (22)
Start-Time:	2021-01-11 01:01:35,984
Finished-Time:	2021-01-11 01:01:38,529
Time-Consumption	2.545s

Testsummary:

Info	Running a periodic task for 10 cycles with a cycletime of 0.25s
Success	Minimum cycle time is correct (Content 0.2503471374511719 in [0.2465 ... 0.2545] and Type is <type 'float'>).
Success	Mean cycle time is correct (Content 0.2507365544637044 in [0.2465 ... 0.2545] and Type is <type 'float'>).
Success	Maximum cycle time is correct (Content 0.250917911529541 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 0.01040506362915039 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).
Success	Mean cycle time is correct (Content 0.010750558641221788 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).
Success	Maximum cycle time is correct (Content 0.01114797592163086 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 0.005441904067993164 in [0.00395 ... 0.00705] and Type is <type 'float'>).
Success	Mean cycle time is correct (Content 0.005706442726982964 in [0.00395 ... 0.00705] and Type is <type 'float'>).
Success	Maximum cycle time is correct (Content 0.005857944488525391 in [0.00395 ... 0.009049999999999999] and Type is <type 'float'>).

3.1.5 pylibs.task.queue: Test clean_queue method

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (25)
Start-Time:	2021-01-11 01:01:38,743
Finished-Time:	2021-01-11 01:01:38,748
Time-Consumption	0.005s

Testsummary:

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

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

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (23)
Start-Time:	2021-01-11 01:01:38,530
Finished-Time:	2021-01-11 01:01:38,636
Time-Consumption	0.106s

Testsummary:

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

3.1.7 pylibs.task.queue: Test stop method

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (24)
Start-Time:	2021-01-11 01:01:38,636
Finished-Time:	2021-01-11 01:01:38,742
Time-Consumption	0.106s

Testsummary:

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

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

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

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (27)
Start-Time:	2021-01-11 01:01:41,871
Finished-Time:	2021-01-11 01:01:42,479
Time-Consumption	0.608s

Testsummary:

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

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

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (26)
Start-Time:	2021-01-11 01:01:38,748
Finished-Time:	2021-01-11 01:01:41,870
Time-Consumption	3.122s

Testsummary:

Info	Enqueued 6 unordered tasks.
Success	Size of Queue before execution is correct (Content 7 and Type is <type 'int'>).
Info	Executing Queue, till Queue is empty..
Success	Size of Queue after execution is correct (Content 0 and Type is <type 'int'>).
Success	Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.
Info	Setting expire flag and enqueued again 2 tasks.
Success	Size of Queue before restarting queue is correct (Content 2 and Type is <type 'int'>).
Info	Executing Queue, till Queue is empty..
Success	Queue execution (rerun; identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

3.2 Summary for testrun with python 3.8.5 (final)

3.2.1 pylibs.task.crontab: Test cronjob

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (28)
Start-Time:	2021-01-11 01:05:21,205
Finished-Time:	2021-01-11 01:05:21,219
Time-Consumption	0.014s

Testsummary:

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

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

3.2.2 pylibs.task.crontab: Test crontab

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (29)
Start-Time:	2021-01-11 01:05:21,220
Finished-Time:	2021-01-11 01:08:51,323
Time-Consumption	210.103s

Testsummary:

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

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

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (21)
Start-Time:	2021-01-11 01:05:13,857
Finished-Time:	2021-01-11 01:05:14,392
Time-Consumption	0.535s

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 0.2502706050872803 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 0.010149478912353516 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 0.005141496658325195 in [0.00395 ... 0.00705] and Type is <class 'float'>).

3.2.4 pylibs.task.periodic: Test periodic execution

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (22)
Start-Time:	2021-01-11 01:05:14,393
Finished-Time:	2021-01-11 01:05:16,943
Time-Consumption	2.550s

Testsummary:

Info	Running a periodic task for 10 cycles with a cycletime of 0.25s
Success	Minimum cycle time is correct (Content 0.2502434253692627 in [0.2465 ... 0.2545] and Type is <class 'float'>).
Success	Mean cycle time is correct (Content 0.25084299511379665 in [0.2465 ... 0.2545] and Type is <class 'float'>).
Success	Maximum cycle time is correct (Content 0.2528243064880371 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 0.010353803634643555 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).
Success	Mean cycle time is correct (Content 0.01063484615749783 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).
Success	Maximum cycle time is correct (Content 0.010976314544677734 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 0.005337238311767578 in [0.00395 ... 0.00705] and Type is <class 'float'>).
Success	Mean cycle time is correct (Content 0.005993445714314778 in [0.00395 ... 0.00705] and Type is <class 'float'>).
Success	Maximum cycle time is correct (Content 0.0076105594635009766 in [0.00395 ... 0.009049999999999999] and Type is <class 'float'>).

3.2.5 pylibs.task.queue: Test clean_queue method

Testresult

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

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

Start-Time:	2021-01-11 01:05:17,157
Finished-Time:	2021-01-11 01:05:17,168
Time-Consumption	0.012s

Testsummary:

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

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

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (23)
Start-Time:	2021-01-11 01:05:16,943
Finished-Time:	2021-01-11 01:05:17,046
Time-Consumption	0.103s

Testsummary:

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

3.2.7 pylibs.task.queue: Test stop method

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (24)
Start-Time:	2021-01-11 01:05:17,046
Finished-Time:	2021-01-11 01:05:17,154
Time-Consumption	0.108s

Testsummary:

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

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

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

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (27)
Start-Time:	2021-01-11 01:05:20,301
Finished-Time:	2021-01-11 01:05:20,908
Time-Consumption	0.607s

Testsummary:

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

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

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (26)
Start-Time:	2021-01-11 01:05:17,169
Finished-Time:	2021-01-11 01:05:20,301
Time-Consumption	3.131s

Testsummary:

Info	Enqueued 6 unordered tasks.
Success	Size of Queue before execution is correct (Content 7 and Type is <class 'int'>).
Info	Executing Queue, till Queue is empty..
Success	Size of Queue after execution is correct (Content 0 and Type is <class 'int'>).
Success	Queue execution (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.
Info	Setting expire flag and enqueued again 2 tasks.
Success	Size of Queue before restarting queue is correct (Content 2 and Type is <class 'int'>).
Info	Executing Queue, till Queue is empty..
Success	Queue execution (rerun; identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.

A Trace for testrun with python 2.7.18 (final)

A.1 Tests with status Info (9)

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

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

Info	Added a delayed task for execution in 0.250s.

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

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

Success	Time consumption is correct (Content 0.2502110004425049 in [0.2465 ... 0.2545] and Type is <type 'float'>).

```
Result (Time consumption): 0.2502110004425049 (<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 0.01007699966430664 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).

```
Result (Time consumption): 0.01007699966430664 (<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 0.005555868148803711 in [0.00395 ... 0.00705] and Type is <type 'float'>).

```
Result (Time consumption): 0.005555868148803711 (<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 1610323295.985120
Task execution number 2 at 1610323296.236038
Task execution number 3 at 1610323296.486850
Task execution number 4 at 1610323296.737573
Task execution number 5 at 1610323296.987920
Task execution number 6 at 1610323297.238769
Task execution number 7 at 1610323297.489640
```

```
Task execution number 8 at 1610323297.740310
Task execution number 9 at 1610323297.991000
Task execution number 10 at 1610323298.241749
```

Success	Minimum cycle time is correct (Content 0.2503471374511719 in [0.2465 ... 0.2545] and Type is <type 'float'>).

```
Result (Minimum cycle time): 0.2503471374511719 (<type 'float'>)
Expectation (Minimum cycle time): 0.2465 <= result <= 0.2545
```

Success	Mean cycle time is correct (Content 0.2507365544637044 in [0.2465 ... 0.2545] and Type is <type 'float'>).

```
Result (Mean cycle time): 0.2507365544637044 (<type 'float'>)
Expectation (Mean cycle time): 0.2465 <= result <= 0.2545
```

Success	Maximum cycle time is correct (Content 0.250917911529541 in [0.2465 ... 0.2565] and Type is <type 'float'>).

```
Result (Maximum cycle time): 0.250917911529541 (<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 1610323298.293181
Task execution number 2 at 1610323298.303801
Task execution number 3 at 1610323298.314949
Task execution number 4 at 1610323298.325354
Task execution number 5 at 1610323298.335975
Task execution number 6 at 1610323298.346748
Task execution number 7 at 1610323298.357556
Task execution number 8 at 1610323298.368520
Task execution number 9 at 1610323298.379186
Task execution number 10 at 1610323298.389936
```

Success	Minimum cycle time is correct (Content 0.01040506362915039 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).

```
Result (Minimum cycle time): 0.01040506362915039 (<type 'float'>)
Expectation (Minimum cycle time): 0.008900000000000002 <= result <= 0.0121
```

Success	Mean cycle time is correct (Content 0.010750558641221788 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).

```
Result (Mean cycle time): 0.010750558641221788 (<type 'float'>)
Expectation (Mean cycle time): 0.008900000000000002 <= result <= 0.0121
```

Success	Maximum cycle time is correct (Content 0.01114797592163086 in [0.008900000000000002 ... 0.0141] and Type is <type 'float'>).

```
Result (Maximum cycle time): 0.01114797592163086 (<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 1610323298.416973
Task execution number 2 at 1610323298.422415
Task execution number 3 at 1610323298.428273
Task execution number 4 at 1610323298.433859
Task execution number 5 at 1610323298.439613
Task execution number 6 at 1610323298.445224
Task execution number 7 at 1610323298.451057
Task execution number 8 at 1610323298.456798
Task execution number 9 at 1610323298.462636
Task execution number 10 at 1610323298.468331
```

Success	Minimum cycle time is correct (Content 0.005441904067993164 in [0.00395 ... 0.00705] and Type is <type 'float'>).

```
Result (Minimum cycle time): 0.005441904067993164 (<type 'float'>)
Expectation (Minimum cycle time): 0.00395 <= result <= 0.00705
```

Success	Mean cycle time is correct (Content 0.005706442726982964 in [0.00395 ... 0.00705] and Type is <type 'float'>).

```
Result (Mean cycle time): 0.005706442726982964 (<type 'float'>)
Expectation (Mean cycle time): 0.00395 <= result <= 0.00705
```

Success	Maximum cycle time is correct (Content 0.005857944488525391 in [0.00395 ... 0.009049999999999999] and Type is <type 'float'>).

```
Result (Maximum cycle time): 0.005857944488525391 (<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 5.1 with Priority 5
Adding Task 3.0 with Priority 3
Adding Task 7.0 with Priority 7
Adding Task 5.2 with Priority 5
Adding Task 2.0 with Priority 2
Adding Task 6.0 with Priority 6
Adding Task 1.0 with Priority 1
```

Success	Size of Queue before execution is correct (Content 7 and Type is <type 'int'>).

```
Result (Size of Queue before execution): 7 (<type 'int'>)
Expectation (Size of Queue before execution): result = 7 (<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, 5.1, 5.2, 6,
↪  7 ] (<type 'list'>)
Expectation (Queue execution (identified by a submitted sequence number)): result = [ 1, 2,
↪  3, 5.1, 5.2, 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): 5.1 (<type 'float'>)
Expectation (Submitted value number 4): result = 5.1 (<type 'float'>)
Submitted value number 4 is correct (Content 5.1 and Type is <type 'float'>).
Result (Submitted value number 5): 5.2 (<type 'float'>)
Expectation (Submitted value number 5): result = 5.2 (<type 'float'>)
Submitted value number 5 is correct (Content 5.2 and Type is <type 'float'>).
Result (Submitted value number 6): 6 (<type 'int'>)
Expectation (Submitted value number 6): result = 6 (<type 'int'>)
Submitted value number 6 is correct (Content 6 and Type is <type 'int'>).
Result (Submitted value number 7): 7 (<type 'int'>)
Expectation (Submitted value number 7): result = 7 (<type 'int'>)
Submitted value number 7 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 1610323332s, requested for 1610323320s
Crontab execution number 2 at 1610323452s, requested for 1610323440s
Result (Timing of crontasks): [ 1610323332, 1610323452 ] (<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): 1610323332 (<type 'int'>)
Expectation (Submitted value number 1): 1610323320 <= result <= 1610323351
Submitted value number 1 is correct (Content 1610323332 in [1610323320 ... 1610323351] and
↪  Type is <type 'int'>).
Result (Submitted value number 2): 1610323452 (<type 'int'>)
Expectation (Submitted value number 2): 1610323440 <= result <= 1610323471
Submitted value number 2 is correct (Content 1610323452 in [1610323440 ... 1610323471] and
↪  Type is <type 'int'>).
```

B Trace for testrun with python 3.8.5 (final)

B.1 Tests with status Info (9)

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

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

Info	Added a delayed task for execution in 0.250s.

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

```
Result (Execution of task and delayed task (identified by a submitted sequence number)): [ 1,
↪  2 ] (<class 'list'>)
```

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

Success Time consumption is correct (Content 0.2502706050872803 in [0.2465 ... 0.2545] and Type is <class 'float'>).

```
Result (Time consumption): 0.2502706050872803 (<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 0.010149478912353516 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).

```
Result (Time consumption): 0.010149478912353516 (<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 0.005141496658325195 in [0.00395 ... 0.00705] and Type is <class 'float'>).

```
Result (Time consumption): 0.005141496658325195 (<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 1610323514.394252
Task execution number 2 at 1610323514.644655
Task execution number 3 at 1610323514.895618
Task execution number 4 at 1610323515.146284
Task execution number 5 at 1610323515.399108
Task execution number 6 at 1610323515.649352
Task execution number 7 at 1610323515.900382
Task execution number 8 at 1610323516.150792
Task execution number 9 at 1610323516.401243
Task execution number 10 at 1610323516.651839
```

Success	Minimum cycle time is correct (Content 0.2502434253692627 in [0.2465 ... 0.2545] and Type is <class 'float'>).

```
Result (Minimum cycle time): 0.2502434253692627 (<class 'float'>)
Expectation (Minimum cycle time): 0.2465 <= result <= 0.2545
```

Success	Mean cycle time is correct (Content 0.25084299511379665 in [0.2465 ... 0.2545] and Type is <class 'float'>).

```
Result (Mean cycle time): 0.25084299511379665 (<class 'float'>)
Expectation (Mean cycle time): 0.2465 <= result <= 0.2545
```

Success Maximum cycle time is correct (Content 0.2528243064880371 in [0.2465 ... 0.2565] and Type is <class 'float'>).

```
Result (Maximum cycle time): 0.2528243064880371 (<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 1610323516.708188
Task execution number 2 at 1610323516.718542
Task execution number 3 at 1610323516.729140
Task execution number 4 at 1610323516.739825
Task execution number 5 at 1610323516.750353
Task execution number 6 at 1610323516.760861
Task execution number 7 at 1610323516.771696
Task execution number 8 at 1610323516.782551
Task execution number 9 at 1610323516.793527
Task execution number 10 at 1610323516.803901
```

Success Minimum cycle time is correct (Content 0.010353803634643555 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).

```
Result (Minimum cycle time): 0.010353803634643555 (<class 'float'>)
Expectation (Minimum cycle time): 0.008900000000000002 <= result <= 0.0121
```

Success Mean cycle time is correct (Content 0.01063484615749783 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).

```
Result (Mean cycle time): 0.01063484615749783 (<class 'float'>)
Expectation (Mean cycle time): 0.008900000000000002 <= result <= 0.0121
```

Success Maximum cycle time is correct (Content 0.010976314544677734 in [0.008900000000000002 ... 0.0141] and Type is <class 'float'>).

```
Result (Maximum cycle time): 0.010976314544677734 (<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 1610323516.830745
Task execution number 2 at 1610323516.836256
Task execution number 3 at 1610323516.841707
Task execution number 4 at 1610323516.847104
Task execution number 5 at 1610323516.852681
```

```
Task execution number 6 at 1610323516.860291
Task execution number 7 at 1610323516.866224
Task execution number 8 at 1610323516.873495
Task execution number 9 at 1610323516.879349
Task execution number 10 at 1610323516.884686
```

Success	Minimum cycle time is correct (Content 0.005337238311767578 in [0.00395 ... 0.00705] and Type is <class 'float'>).

```
Result (Minimum cycle time): 0.005337238311767578 (<class 'float'>)
Expectation (Minimum cycle time): 0.00395 <= result <= 0.00705
```

Success	Mean cycle time is correct (Content 0.005993445714314778 in [0.00395 ... 0.00705] and Type is <class 'float'>).

```
Result (Mean cycle time): 0.005993445714314778 (<class 'float'>)
Expectation (Mean cycle time): 0.00395 <= result <= 0.00705
```

Success	Maximum cycle time is correct (Content 0.0076105594635009766 in [0.00395 ... 0.009049999999999999] and Type is <class 'float'>).

```
Result (Maximum cycle time): 0.0076105594635009766 (<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 5.1 with Priority 5
Adding Task 3.0 with Priority 3
Adding Task 7.0 with Priority 7
Adding Task 5.2 with Priority 5
Adding Task 2.0 with Priority 2
Adding Task 6.0 with Priority 6
```

```
Adding Task 1.0 with Priority 1
```

Success Size of Queue before execution is correct (Content 7 and Type is <class 'int'>).

```
Result (Size of Queue before execution): 7 (<class 'int'>)
Expectation (Size of Queue before execution): result = 7 (<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, 5.1, 5.2, 6,
↪  7 ] (<class 'list'>)
Expectation (Queue execution (identified by a submitted sequence number)): result = [ 1, 2,
↪  3, 5.1, 5.2, 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): 5.1 (<class 'float'>)
Expectation (Submitted value number 4): result = 5.1 (<class 'float'>)
Submitted value number 4 is correct (Content 5.1 and Type is <class 'float'>).
Result (Submitted value number 5): 5.2 (<class 'float'>)
Expectation (Submitted value number 5): result = 5.2 (<class 'float'>)
Submitted value number 5 is correct (Content 5.2 and Type is <class 'float'>).
Result (Submitted value number 6): 6 (<class 'int'>)
Expectation (Submitted value number 6): result = 6 (<class 'int'>)
Submitted value number 6 is correct (Content 6 and Type is <class 'int'>).
Result (Submitted value number 7): 7 (<class 'int'>)
Expectation (Submitted value number 7): result = 7 (<class 'int'>)
```

```
Submitted value number 7 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 1610323581s, requested for 1610323560s
Crontab execution number 2 at 1610323701s, requested for 1610323680s
Result (Timing of crontasks): [ 1610323581, 1610323701 ] (<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): 1610323581 (<class 'int'>)
Expectation (Submitted value number 1): 1610323560 <= result <= 1610323591
Submitted value number 1 is correct (Content 1610323581 in [1610323560 ... 1610323591] and
↪  Type is <class 'int'>).
Result (Submitted value number 2): 1610323701 (<class 'int'>)
Expectation (Submitted value number 2): 1610323680 <= result <= 1610323711
Submitted value number 2 is correct (Content 1610323701 in [1610323680 ... 1610323711] 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 98.1%

C.1.1 task.__init__.py

The line coverage for task.__init__.py was 98.9%
The branch coverage for task.__init__.py was 98.1%

```
#!/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.
```

```
27
28 **Module Documentation:**
29
30 """
31 __DEPENDENCIES__ = []
32
33 import logging
34 import sys
35 import threading
36 import time
37 if sys.version_info >= (3, 0):
38     from queue import PriorityQueue
39     from queue import Empty
40 else:
41     from Queue import PriorityQueue
42     from Queue import Empty
43
44 try:
45     from config import APP_NAME as ROOT_LOGGER_NAME
46 except ImportError:
47     ROOT_LOGGER_NAME = 'root'
48 logger = logging.getLogger(ROOT_LOGGER_NAME).getChild(__name__)
49
50 __DESCRIPTION__ = """The Module {\\tt %s} is designed to help with task issues like periodic
    tasks, delayed tasks, queues, threaded queues and crontabs.
51 For more Information read the documentation.""" % __name__.replace('_', '\_')
52 """The Module Description"""
53 __INTERPRETER__ = (2, 3)
54 """The Tested Interpreter-Versions"""
55
56
57 class queue(object):
58     """
59     Class to execute queued callbacks.
60
61     :param bool expire: The default value for expire. See also :py:func:`expire`.
62
63     **Example:**
64
65     .. literalinclude:: task/_examples_/tqueue.py
66
67     Will result to the following output:
68
69     .. literalinclude:: task/_examples_/tqueue.log
70     """
71     class job(object):
72         def __init__(self, priority, callback, *args, **kwargs):
73             self.time = time.time()
74             self.priority = priority
75             self.callback = callback
76             self.args = args
77             self.kwargs = kwargs
78
79         def run(self, queue):
80             self.callback(queue, *self.args, **self.kwargs)
81
82         def __lt__(self, other):
83             if self.priority != other.priority:
84                 return self.priority < other.priority
85             else:
86                 return self.time < other.time
87
```

```
88      def __init__(self, expire=True):
89          self.__expire = expire
90          self.__stop = False
91          self.queue = PriorityQueue()
92
93      def clean_queue(self):
94          """
95          This Methods removes all jobs from the queue.
96
97          .. note:: Be aware that already running jobs will not be terminated.
98          """
99          while not self.queue.empty():
100             try:
101                 self.queue.get(False)
102             except Empty:         # This block is hard to reach for a testcase, but is
103                 continue          # needed, if the thread runs dry while cleaning the queue.
104             self.queue.task_done()
105
106     def enqueue(self, priority, callback, *args, **kwargs):
107         """
108         This enqueues a given callback.
109
110         :param number priority: The priority indication number of this task. The lowest value
    will be queued first.
111         :param callback callback: Callback to be executed
112         :param args args: Arguments to be given to callback
113         :param kwargs kwargs: Keword Arguments to be given to callback
114
115         .. note:: Callback will get this instance as first argument, followed by :py:data:`args`
    und :py:data:`kwargs`.
116         """
117         self.queue.put(self.job(priority, callback, *args, **kwargs))
118
119     def qsize(self):
120         return self.queue.qsize()
121
122     def run(self):
123         """
124         This starts the execution of the queued callbacks.
125         """
126         self.__stop = False
127         while not self.__stop:
128             try:
129                 self.queue.get(timeout=0.1).run(self)
130             except Empty:
131                 if self.__expire:
132                     break
133         if type(self) is threaded_queue:
134             self.thread = None
135
136     def expire(self):
137         """
138         This sets the expire flag. That means that the process will stop after queue gets empty.
139         """
140         self.__expire = True
141
142     def stop(self):
143         """
144         This sets the stop flag. That means that the process will stop after finishing the active
     task.
145         """
146         self.__stop = True
```

```
147
148
149 class threaded_queue(queue):
150     """Class to execute queued callbacks in a background thread (See also parent :py:class:`queue
        `).
151
152     :param bool expire: The default value for expire. See also :py:func:`queue.expire`.
153
154     **Example:**
155
156     .. literalinclude:: task/_examples_/threaded_queue.py
157
158     Will result to the following output:
159
160     .. literalinclude:: task/_examples_/threaded_queue.log
161     """
162     def __init__(self, expire=False):
163         queue.__init__(self, expire=expire)
164         self.thread = None
165
166     def run(self):
167         if self.thread is None:
168             self.thread = threading.Thread(target=self._start, args=())
169             self.thread.daemon = True    # Daemonize thread
170             self.thread.start()          # Start the execution
171
172     def join(self):
173         """
174         This blocks till the queue is empty.
175
176         .. note:: If the queue does not run dry, join will block till the end of the days.
177         """
178         self.expire()
179         if self.thread is not None:
180             self.thread.join()
181
182     def stop(self):
183         queue.stop(self)
184         self.join()
185
186     def _start(self):
187         queue.run(self)
188
189
190 class periodic(object):
191     """
192     Class to execute a callback cyclicly.
193
194     :param float cycle_time: Cycle time in seconds -- callback will be executed every *cycle_time
        * seconds
195     :param callback callback: Callback to be executed
196     :param args args: Arguments to be given to the callback
197     :param kwargs kwargs: Keword Arguments to be given to callback
198
199     .. note:: The Callback will get this instance as first argument, followed by :py:data:`args`
        und :py:data:`kwargs`.
200
201     **Example:**
202
203     .. literalinclude:: task/_examples_/periodic.py
204
205     Will result to the following output:
206
207     .. literalinclude:: task/_examples_/periodic.log
208     """
```

```
209     def __init__(self, cycle_time, callback, *args, **kwargs):
210         self._lock = threading.Lock()
211         self._timer = None
212         self.callback = callback
213         self.cycle_time = cycle_time
214         self.args = args
215         self.kwargs = kwargs
216         self._stopped = True
217         self._last_tm = None
218         self.dt = None
219 
220     def join(self):
221         """
222         This blocks till the cyclic task is terminated.
223 
224         .. note:: Using join means that somewhere has to be a condition calling :py:func:`stop`
    to terminate. Otherwise :func:`task.join` will never return.
225         """
226         while not self._stopped:
227             time.sleep(.1)
228 
229     def run(self):
230         """
231         This starts the cyclic execution of the given callback.
232         """
233         if self._stopped:
234             self._set_timer(force_now=True)
235 
236     def stop(self):
237         """
238         This stops the execution of any further task.
239         """
240         self._lock.acquire()
241         self._stopped = True
242         if self._timer is not None:
243             self._timer.cancel()
244         self._lock.release()
245 
246     def _set_timer(self, force_now=False):
247         """
248         This sets the timer for the execution of the next task.
249         """
250         self._lock.acquire()
251         self._stopped = False
252         if force_now:
253             self._timer = threading.Timer(0, self._start)
254         else:
255             self._timer = threading.Timer(self.cycle_time, self._start)
256         self._timer.start()
257         self._lock.release()
258 
259     def _start(self):
260         tm = time.time()
261         if self._last_tm is not None:
262             self.dt = tm - self._last_tm
263         self._set_timer(force_now=False)
264         self.callback(self, *self.args, **self.kwargs)
265         self._last_tm = tm
266 
267 
268 class delayed(periodic):
```

```
269     """Class to execute a callback a given time in the future. See also parent :py:class:`
        periodic`.
270
271     :param float time: Delay time for execution of the given callback
272     :param callback callback: Callback to be executed
273     :param args args: Arguments to be given to callback
274     :param kwargs kwargs: Keword Arguments to be given to callback
275
276     **Example:**
277
278     .. literalinclude:: task/_examples_/delayed.py
279
280     Will result to the following output:
281
282     .. literalinclude:: task/_examples_/delayed.log
283     """
284     def run(self):
285         """
286         This starts the timer for the delayed execution.
287         """
288         self._set_timer(force_now=False)
289
290     def _start(self):
291         self.callback(*self.args, **self.kwargs)
292         self.stop()
293
294
295 class crontab(periodic):
296     """Class to execute a callback at the specified time conditions. See also parent :py:class:`
        periodic`.
297
298     :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.
299     :type accuracy: float
300
301     **Example:**
302
303     .. literalinclude:: task/_examples_/crontab.py
304
305     Will result to the following output:
306
307     .. literalinclude:: task/_examples_/crontab.log
308     """
309     ANY = '*'
310     """Constant for matching every condition."""
311
312     class cronjob(object):
313         """Class to handle cronjob parameters and cronjob changes.
314
315         :param minute: Minute for execution. Either 0...59, [0...59, 0...59, ...] or :py:const:`
        crontab.ANY` for every Minute.
316         :type minute: int, list, str
317         :param hour: Hour for execution. Either 0...23, [0...23, 0...23, ...] or :py:const:`
        crontab.ANY` for every Hour.
318         :type hour: int, list, str
319         :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.
320         :type day_of_month: int, list, str
321         :param month: Month for execution. Either 0...12, [0...12, 0...12, ...] or :py:const:`
        crontab.ANY` for every Month.
322         :type month: int, list, str
```

```
323         :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.
324         :type day_of_week: int, list, str
325         :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.
326         :type callback: func
327 
328         .. 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()`.
329         """
330         class all_match(set):
331             """Universal set - match everything"""
332             def __contains__(self, item):
333                 (item)
334                 return True
335 
336         def __init__(self, minute, hour, day_of_month, month, day_of_week, callback, *args, **
    kwargs):
337             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)
338             self.callback = callback
339             self.args = args
340             self.kwargs = kwargs
341             self.__last_cron_check_time__ = None
342             self.__last_execution__ = None
343 
344         def set_trigger_conditions(self, minute=None, hour=None, day_of_month=None, month=None,
    day_of_week=None):
345             """This Method changes the execution parameters.
346 
347             :param minute: Minute for execution. Either 0...59, [0...59, 0...59, ...] or :py:
    const:`crontab.ANY` for every Minute.
348             :type minute: int, list, str
349             :param hour: Hour for execution. Either 0...23, [0...23, 0...23, ...] or :py:const:`
    crontab.ANY` for every Hour.
350             :type hour: int, list, str
351             :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.
352             :type day_of_month: int, list, str
353             :param month: Month for execution. Either 0...12, [0...12, 0...12, ...] or :py:const
    :`crontab.ANY` for every Month.
354             :type month: int, list, str
355             :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.
356             :type day_of_week: int, list, str
357             """
358             if minute is not None:
359                 self.minute = self.__conv_to_set__(minute)
360             if hour is not None:
361                 self.hour = self.__conv_to_set__(hour)
362             if day_of_month is not None:
363                 self.day_of_month = self.__conv_to_set__(day_of_month)
364             if month is not None:
365                 self.month = self.__conv_to_set__(month)
366             if day_of_week is not None:
367                 self.day_of_week = self.__conv_to_set__(day_of_week)
368 
369         def __conv_to_set__(self, obj):
370             if obj is crontab.ANY:
371                 return self.all_match()
372             elif isinstance(obj, (int, long) if sys.version_info < (3,0) else (int)):
373                 return set([obj])
```

```
374             else:
375                 return set(obj)
376 
377         def __execution_needed_for__(self, minute, hour, day_of_month, month, day_of_week):
378             if self.__last_execution__ != [minute, hour, day_of_month, month, day_of_week]:
379                 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:
380                     return True
381             return False
382 
383         def __store_execution_reminder__(self, minute, hour, day_of_month, month, day_of_week):
384             self.__last_execution__ = [minute, hour, day_of_month, month, day_of_week]
385 
386         def cron_execution(self, tm):
387             """This Methods executes the Cron-Callback, if a execution is needed for the given
    time (depending on the parameters on initialisation)
388 
389             :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())).
390             :type tm: int
391             """
392             if self.__last_cron_check_time__ is None:
393                 self.__last_cron_check_time__ = tm - 1
394             #
395             for t in range(self.__last_cron_check_time__ + 1, tm + 1):
396                 lt = time.localtime(t)
397                 if self.__execution_needed_for__(lt[4], lt[3], lt[2], lt[1], lt[6]):
398                     self.callback(self, *self.args, **self.kwargs)
399                     self.__store_execution_reminder__(lt[4], lt[3], lt[2], lt[1], lt[6])
400                     break
401             self.__last_cron_check_time__ = tm
402 
403     def __init__(self, accuracy=30):
404         periodic.__init__(self, accuracy, self.__periodic__)
405         self.__crontab__ = []
406 
407     def __periodic__(self, rt):
408         (rt)
409         tm = int(time.time())
410         for cronjob in self.__crontab__:
411             cronjob.cron_execution(tm)
412 
413     def add_cronjob(self, minute, hour, day_of_month, month, day_of_week, callback, *args, **
    kwargs):
414         """This Method adds a cronjob to be executed.
415 
416         :param minute: Minute for execution. Either 0...59, [0...59, 0...59, ...] or :py:const:`
    crontab.ANY` for every Minute.
417         :type minute: int, list, str
418         :param hour: Hour for execution. Either 0...23, [0...23, 0...23, ...] or :py:const:`
    crontab.ANY` for every Hour.
419         :type hour: int, list, str
420         :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.
421         :type day_of_month: int, list, str
422         :param month: Month for execution. Either 0...12, [0...12, 0...12, ...] or :py:const:`
    crontab.ANY` for every Month.
423         :type month: int, list, str
424         :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.
425         :type day_of_week: int, list, str
```

```
426         :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.
427         :type callback: func
428 
429         .. note:: The ``callback`` will be executed with it's instance of :py:class:`cronjob` as the first parameter.
430             The given Arguments (:data:`args`) and keyword Arguments (:data:`kwargs`) will be stored in that object.
431         """
432         self.__crontab__.append(self.cronjob(minute, hour, day_of_month, month, day_of_week, callback, *args, **kwargs))
```