Unittest for `task`

December 21, 2020

Contents

1 Test Information **3**

1.1 Test Candidate Information . . . 3

1.2 Unittest Information . . . 3

1.3 Test System Information . . . 3

2 Statistic **3**

2.1 Test-Statistic for testrun with python 2.7.18 (final) . . . 3

2.2 Test-Statistic for testrun with python 3.8.5 (final) . . . 4

2.3 Coverage Statistic . . . 4

3 Testcases with no corresponding Requirement **5**

3.1 Summary for testrun with python 2.7.18 (final) . . . 5

3.1.1 pylibs.task.crontab: Test cronjob . . . 5

3.1.2 pylibs.task.crontab: Test crontab . . . 6

3.1.3 pylibs.task.delayed: Test parallel processing and timing for a delayed execution . . . 6

3.1.4 pylibs.task.periodic: Test periodic execution . . . 7

3.1.5 pylibs.task.queue: Test clean_queue method . . . 7

3.1.6 pylibs.task.queue: Test qsize and queue execution order by priority . . . 8

3.1.7 pylibs.task.queue: Test stop method . . . 8

3.1.8 pylibs.task.threaded_queue: Test enqueue while queue is running . . . 9

3.1.9 pylibs.task.threaded_queue: Test qsize and queue execution order by priority . . . 9

3.2 Summary for testrun with python 3.8.5 (final) . . . 10

3.2.1 pylibs.task.crontab: Test cronjob . . . 10

3.2.2 pylibs.task.crontab: Test crontab . . . 11

3.2.3 pylibs.task.delayed: Test parallel processing and timing for a delayed execution . . . 11

3.2.4 pylibs.task.periodic: Test periodic execution . . . 12

3.2.5 pylibs.task.queue: Test clean_queue method . . . 12

3.2.6 pylibs.task.queue: Test qsize and queue execution order by priority . . . 13

3.2.7 pylibs.task.queue: Test stop method . . . 13

3.2.8 pylibs.task.threaded_queue: Test enqueue while queue is running . . . 14

3.2.9 pylibs.task.threaded_queue: Test qsize and queue execution order by priority . . . 14

A Trace for testrun with python 2.7.18 (final) 15

A.1 Tests with status Failed (2) . . . 15

A.1.1 pylibs.task.delayed: Test parallel processing and timing for a delayed execution . . . 15

A.1.2 pylibs.task.periodic: Test periodic execution . . . 17

A.2 Tests with status Info (7) . . . 19

A.2.1 pylibs.task.queue: Test qsize and queue execution order by priority . . . 19

A.2.2 pylibs.task.queue: Test stop method . . . 20

A.2.3 pylibs.task.queue: Test clean_queue method . . . 21

A.2.4 pylibs.task.threaded_queue: Test qsize and queue execution order by priority . . . 22

A.2.5 pylibs.task.threaded_queue: Test enqueue while queue is running . . . 24

A.2.6 pylibs.task.crontab: Test cronjob . . . 25

A.2.7 pylibs.task.crontab: Test crontab . . . 29

B Trace for testrun with python 3.8.5 (final) 30

B.1 Tests with status Failed (1) . . . 30

B.1.1 pylibs.task.periodic: Test periodic execution . . . 30

B.2 Tests with status Info (8) . . . 32

B.2.1 pylibs.task.delayed: Test parallel processing and timing for a delayed execution . . . 32

B.2.2 pylibs.task.queue: Test qsize and queue execution order by priority . . . 34

B.2.3 pylibs.task.queue: Test stop method . . . 35

B.2.4 pylibs.task.queue: Test clean_queue method . . . 36

B.2.5 pylibs.task.threaded_queue: Test qsize and queue execution order by priority . . . 37

B.2.6 pylibs.task.threaded_queue: Test enqueue while queue is running . . . 39

B.2.7 pylibs.task.crontab: Test cronjob . . . 40

B.2.8 pylibs.task.crontab: Test crontab . . . 44

C Test-Coverage 44

C.1 task . . . 44

C.1.1 task.__init__.py . . . 45

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	0e9c2f2af925870ae005edce5afd48c0
Dependencies	

1.2 Unittest Information

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

1.3 Test System Information

System Information	
Architecture	64bit
Distribution	Linux Mint 20 ulyana
Hostname	ahorn
Kernel	5.4.0-58-generic (#64-Ubuntu SMP Wed Dec 9 08:16:25 UTC 2020)
Machine	x86_64
Path	/user_data/data/dirk/prj/unittest/task/unittest
System	Linux
Username	dirk

2 Statistic

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

Number of tests	**9**
Number of successfull tests	**7**
Number of possibly failed tests	**0**
Number of failed tests	**2**
Executionlevel	Full Test (all defined tests)
Time consumption	217.036s

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

Number of tests	**9**
Number of successfull tests	**8**
Number of possibly failed tests	**0**
Number of failed tests	**1**
Executionlevel	Full Test (all defined tests)
Time consumption	216.964s

2.3 Coverage Statistic

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

3 Testcases with no corresponding Requirement

3.1 Summary for testrun with python 2.7.18 (final)

3.1.1 pylibs.task.crontab: Test cronjob

Testresult

This test was passed with the state: **Success**. See also full trace in section A.2.6!

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (28)
Start-Time:	2020-12-21 01:33:11,410
Finished-Time:	2020-12-21 01:33:11,423
Time-Consumption	0.013s

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.2.7!

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (29)
Start-Time:	2020-12-21 01:33:11,424
Finished-Time:	2020-12-21 01:36:41,528
Time-Consumption	210.104s

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: **Failed**. 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:	2020-12-21 01:33:04,290
Finished-Time:	2020-12-21 01:33:04,824
Time-Consumption	0.534s

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 0.25006985664367676 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 0.010081052780151367 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.
Failed	Time consumption is NOT correct. See detailed log for more information.

3.1.4 pylibs.task.periodic: Test periodic execution

Testresult

This test was passed with the state: **Failed**. 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:	2020-12-21 01:33:04,825
Finished-Time:	2020-12-21 01:33:07,370
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 0.25013303756713867 in [0.2465 ... 0.2545] and Type is <type 'float'>).
Success	Mean cycle time is correct (Content 0.2512981096903483 in [0.2465 ... 0.2545] and Type is <type 'float'>).
Success	Maximum cycle time is correct (Content 0.25360703468322754 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 0.010352849960327148 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).
Success	Mean cycle time is correct (Content 0.011315557691786025 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).
Success	Maximum cycle time is correct (Content 0.012820959091186523 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 0.00450897216796875 in [0.00395 ... 0.00705] and Type is <type 'float'>).
Success	Mean cycle time is correct (Content 0.005825572543674045 in [0.00395 ... 0.00705] and Type is <type 'float'>).
Failed	Maximum cycle time is NOT correct. See detailed log for more information.

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.2.3!

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (25)
Start-Time:	2020-12-21 01:33:07,582
Finished-Time:	2020-12-21 01:33:07,585
Time-Consumption	0.003s

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.2.1!

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (23)
Start-Time:	2020-12-21 01:33:07,370
Finished-Time:	2020-12-21 01:33:07,474
Time-Consumption	0.103s

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.2.2!

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (24)
Start-Time:	2020-12-21 01:33:07,474
Finished-Time:	2020-12-21 01:33:07,581
Time-Consumption	0.107s

Testsummary:

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

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

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

Testresult

This test was passed with the state: **Success**. See also full trace in section A.2.5!

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (27)
Start-Time:	2020-12-21 01:33:10,505
Finished-Time:	2020-12-21 01:33:11,211
Time-Consumption	0.706s

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.2.4!

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (26)
Start-Time:	2020-12-21 01:33:07,585
Finished-Time:	2020-12-21 01:33:10,504
Time-Consumption	2.919s

Testsummary:

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

3.2 Summary for testrun with python 3.8.5 (final)

3.2.1 pylibs.task.crontab: Test cronjob

Testresult

This test was passed with the state: **Success**. See also full trace in section B.2.7!

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (28)
Start-Time:	2020-12-21 01:36:49,435
Finished-Time:	2020-12-21 01:36:49,470
Time-Consumption	0.035s

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.2.8!

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (29)
Start-Time:	2020-12-21 01:36:49,472
Finished-Time:	2020-12-21 01:40:19,576
Time-Consumption	210.104s

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.2.1!

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (21)
Start-Time:	2020-12-21 01:36:42,312
Finished-Time:	2020-12-21 01:36:42,822
Time-Consumption	0.510s

Testsummary:

Info	Added a delayed task for execution in 0.250s.
Success	Execution of task and delayed task (identified by a submitted sequence number): Values and number of submitted values is correct. See detailed log for more information.
Success	Time consumption is correct (Content 0.25013089179992676 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 0.010036706924438477 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 0.00513005256652832 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: **Failed**. 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 (22)
Start-Time:	2020-12-21 01:36:42,823
Finished-Time:	2020-12-21 01:36:45,384
Time-Consumption	2.561s

Testsummary:

Info	Running a periodic task for 10 cycles with a cycletime of 0.25s
Success	Minimum cycle time is correct (Content 0.25089335441589355 in [0.2465 ... 0.2545] and Type is <class 'float'>).
Success	Mean cycle time is correct (Content 0.2514987786610921 in [0.2465 ... 0.2545] and Type is <class 'float'>).
Success	Maximum cycle time is correct (Content 0.25246405601501465 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 0.010365486145019531 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).
Success	Mean cycle time is correct (Content 0.011408329010009766 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).
Success	Maximum cycle time is correct (Content 0.013473272323608398 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 0.005218505859375 in [0.00395 ... 0.00705] and Type is <class 'float'>).
Success	Mean cycle time is correct (Content 0.0062287913428412545 in [0.00395 ... 0.00705] and Type is <class 'float'>).
Failed	Maximum cycle time is NOT correct. See detailed log for more information.

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.2.4!

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

Start-Time:	2020-12-21 01:36:45,596
Finished-Time:	2020-12-21 01:36:45,610
Time-Consumption	0.015s

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.2.2!

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (23)
Start-Time:	2020-12-21 01:36:45,384
Finished-Time:	2020-12-21 01:36:45,489
Time-Consumption	0.104s

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.2.3!

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (24)
Start-Time:	2020-12-21 01:36:45,489
Finished-Time:	2020-12-21 01:36:45,595
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 <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.2.6!

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (27)
Start-Time:	2020-12-21 01:36:48,531
Finished-Time:	2020-12-21 01:36:49,141
Time-Consumption	0.610s

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.2.5!

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/task/unittest/src/tests/__init__.py (26)
Start-Time:	2020-12-21 01:36:45,611
Finished-Time:	2020-12-21 01:36:48,530
Time-Consumption	2.919s

Testsummary:

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

A Trace for testrun with python 2.7.18 (final)

A.1 Tests with status Failed (2)

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

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

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 0.25006985664367676 in [0.2465 ... 0.2545] and Type is <type 'float'>).

```
Result (Time consumption): 0.25006985664367676 (<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 0.010081052780151367 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).

```
Result (Time consumption): 0.010081052780151367 (<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'>).
```

Failed Time consumption is NOT correct. See detailed log for more information.

```
Result (Time consumption): 0.008383989334106445 (<type 'float'>)
Expectation (Time consumption): 0.00395 <= result <= 0.00705
Content 0.008383989334106445 is incorrect.
```

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

Testresult

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

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

```
Task execution number 1 at 1608510784.827596
Task execution number 2 at 1608510785.079052
Task execution number 3 at 1608510785.330798
Task execution number 4 at 1608510785.580931
Task execution number 5 at 1608510785.833529
Task execution number 6 at 1608510786.083797
Task execution number 7 at 1608510786.334294
Task execution number 8 at 1608510786.585219
Task execution number 9 at 1608510786.835672
Task execution number 10 at 1608510787.089279
```

Success	Minimum cycle time is correct (Content 0.25013303756713867 in [0.2465 ... 0.2545] and Type is <type 'float'>).

```
Result (Minimum cycle time): 0.25013303756713867 (<type 'float'>)
Expectation (Minimum cycle time): 0.2465 <= result <= 0.2545
```

Success	Mean cycle time is correct (Content 0.2512981096903483 in [0.2465 ... 0.2545] and Type is <type 'float'>).

```
Result (Mean cycle time): 0.2512981096903483 (<type 'float'>)
Expectation (Mean cycle time): 0.2465 <= result <= 0.2545
```

Success	Maximum cycle time is correct (Content 0.25360703468322754 in [0.2465 ... 0.2565] and Type is <type 'float'>).

```
Result (Maximum cycle time): 0.25360703468322754 (<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 1608510787.133625
Task execution number 2 at 1608510787.146178
Task execution number 3 at 1608510787.158999
Task execution number 4 at 1608510787.170375
Task execution number 5 at 1608510787.180770
Task execution number 6 at 1608510787.191777
Task execution number 7 at 1608510787.202972
Task execution number 8 at 1608510787.214167
Task execution number 9 at 1608510787.224520
Task execution number 10 at 1608510787.235465
```

Success	Minimum cycle time is correct (Content 0.010352849960327148 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).

```
Result (Minimum cycle time): 0.010352849960327148 (<type 'float'>)
Expectation (Minimum cycle time): 0.008900000000000002 <= result <= 0.0121
```

Success	Mean cycle time is correct (Content 0.011315557691786025 in [0.008900000000000002 ... 0.0121] and Type is <type 'float'>).

```
Result (Mean cycle time): 0.011315557691786025 (<type 'float'>)
Expectation (Mean cycle time): 0.008900000000000002 <= result <= 0.0121
```

Success	Maximum cycle time is correct (Content 0.012820959091186523 in [0.008900000000000002 ... 0.0141] and Type is <type 'float'>).

```
Result (Maximum cycle time): 0.012820959091186523 (<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 1608510787.256209
Task execution number 2 at 1608510787.261650
Task execution number 3 at 1608510787.267084
Task execution number 4 at 1608510787.272660
Task execution number 5 at 1608510787.281990
Task execution number 6 at 1608510787.286499
Task execution number 7 at 1608510787.291959
Task execution number 8 at 1608510787.297896
Task execution number 9 at 1608510787.303264
Task execution number 10 at 1608510787.308639
```

Success	Minimum cycle time is correct (Content 0.00450897216796875 in [0.00395 ... 0.00705] and Type is <type 'float'>).

```
Result (Minimum cycle time): 0.00450897216796875 (<type 'float'>)
Expectation (Minimum cycle time): 0.00395 <= result <= 0.00705
```

Success	Mean cycle time is correct (Content 0.005825572543674045 in [0.00395 ... 0.00705] and Type is <type 'float'>).

```
Result (Mean cycle time): 0.005825572543674045 (<type 'float'>)
Expectation (Mean cycle time): 0.00395 <= result <= 0.00705
```

Failed	Maximum cycle time is NOT correct. See detailed log for more information.

```
Result (Maximum cycle time): 0.009330034255981445 (<type 'float'>)
Expectation (Maximum cycle time): 0.00395 <= result <= 0.009049999999999999
Content 0.009330034255981445 is incorrect.
```

A.2 Tests with status Info (7)

A.2.1 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.2.2 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.2.3 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.2.4 pylibs.task.threaded_queue: Test qsize and queue execution order by priority

Testresult

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

Info	Enqueued 6 unordered tasks.

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

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

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

Info Executing Queue, till Queue is empty..

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

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

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

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

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

Info Setting expire flag and enqueued again 2 tasks.

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

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

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

Info	Executing Queue, till Queue is empty..

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

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

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

A.2.5 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.2.6 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.2.7 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 1608510851s, requested for 1608510840s
Crontab execution number 2 at 1608510971s, requested for 1608510960s
Result (Timing of crontasks): [ 1608510851, 1608510971 ] (<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): 1608510851 (<type 'int'>)
Expectation (Submitted value number 1): 1608510840 <= result <= 1608510871
Submitted value number 1 is correct (Content 1608510851 in [1608510840 ... 1608510871] and
↪  Type is <type 'int'>).
Result (Submitted value number 2): 1608510971 (<type 'int'>)
Expectation (Submitted value number 2): 1608510960 <= result <= 1608510991
Submitted value number 2 is correct (Content 1608510971 in [1608510960 ... 1608510991] and
↪  Type is <type 'int'>).
```

B Trace for testrun with python 3.8.5 (final)

B.1 Tests with status Failed (1)

B.1.1 pylibs.task.periodic: Test periodic execution

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

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

```
Task execution number 1 at 1608511002.825815
Task execution number 2 at 1608511003.078279
Task execution number 3 at 1608511003.330425
Task execution number 4 at 1608511003.582762
Task execution number 5 at 1608511003.833807
Task execution number 6 at 1608511004.084734
Task execution number 7 at 1608511004.335762
Task execution number 8 at 1608511004.587069
Task execution number 9 at 1608511004.838411
Task execution number 10 at 1608511005.089304
```

Success	Minimum cycle time is correct (Content 0.25089335441589355 in [0.2465 ... 0.2545] and Type is <class 'float'>).

```
Result (Minimum cycle time): 0.25089335441589355 (<class 'float'>)
Expectation (Minimum cycle time): 0.2465 <= result <= 0.2545
```

Success	Mean cycle time is correct (Content 0.2514987786610921 in [0.2465 ... 0.2545] and Type is <class 'float'>).

```
Result (Mean cycle time): 0.2514987786610921 (<class 'float'>)
Expectation (Mean cycle time): 0.2465 <= result <= 0.2545
```

Success	Maximum cycle time is correct (Content 0.25246405601501465 in [0.2465 ... 0.2565] and Type is <class 'float'>).

```
Result (Maximum cycle time): 0.25246405601501465 (<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 1608511005.131525
Task execution number 2 at 1608511005.141891
Task execution number 3 at 1608511005.153270
Task execution number 4 at 1608511005.166744
Task execution number 5 at 1608511005.177683
Task execution number 6 at 1608511005.189078
Task execution number 7 at 1608511005.199764
Task execution number 8 at 1608511005.212616
Task execution number 9 at 1608511005.223682
Task execution number 10 at 1608511005.234200
```

Success	Minimum cycle time is correct (Content 0.010365486145019531 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).

```
Result (Minimum cycle time): 0.010365486145019531 (<class 'float'>)
Expectation (Minimum cycle time): 0.008900000000000002 <= result <= 0.0121
```

Success	Mean cycle time is correct (Content 0.011408329010009766 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).

```
Result (Mean cycle time): 0.011408329010009766 (<class 'float'>)
Expectation (Mean cycle time): 0.008900000000000002 <= result <= 0.0121
```

Success	Maximum cycle time is correct (Content 0.013473272323608398 in [0.008900000000000002 ... 0.0141] and Type is <class 'float'>).

```
Result (Maximum cycle time): 0.013473272323608398 (<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 1608511005.268366
Task execution number 2 at 1608511005.274698
Task execution number 3 at 1608511005.280254
Task execution number 4 at 1608511005.285642
Task execution number 5 at 1608511005.290865
Task execution number 6 at 1608511005.300302
Task execution number 7 at 1608511005.307268
Task execution number 8 at 1608511005.313901
Task execution number 9 at 1608511005.319119
Task execution number 10 at 1608511005.324425
```

Success	Minimum cycle time is correct (Content 0.005218505859375 in [0.00395 ... 0.00705] and Type is <class 'float'>).

```
Result (Minimum cycle time): 0.005218505859375 (<class 'float'>)
Expectation (Minimum cycle time): 0.00395 <= result <= 0.00705
```

Success	Mean cycle time is correct (Content 0.0062287913428412545 in [0.00395 ... 0.00705] and Type is <class 'float'>).

```
Result (Mean cycle time): 0.0062287913428412545 (<class 'float'>)
Expectation (Mean cycle time): 0.00395 <= result <= 0.00705
```

Failed	Maximum cycle time is NOT correct. See detailed log for more information.

```
Result (Maximum cycle time): 0.009437084197998047 (<class 'float'>)
Expectation (Maximum cycle time): 0.00395 <= result <= 0.009049999999999999
Content 0.009437084197998047 is incorrect.
```

B.2 Tests with status Info (8)

B.2.1 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 0.25013089179992676 in [0.2465 ... 0.2545] and Type is <class 'float'>).

```
Result (Time consumption): 0.25013089179992676 (<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 0.010036706924438477 in [0.008900000000000002 ... 0.0121] and Type is <class 'float'>).

```
Result (Time consumption): 0.010036706924438477 (<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 0.00513005256652832 in [0.00395 ... 0.00705] and Type is <class 'float'>).

```
Result (Time consumption): 0.00513005256652832 (<class 'float'>)
Expectation (Time consumption): 0.00395 <= result <= 0.00705
```

B.2.2 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.2.3 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.2.4 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.2.5 pylibs.task.threaded_queue: Test qsize and queue execution order by priority

Testresult

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

Info	Enqueued 6 unordered tasks.

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

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

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

Info	Executing Queue, till Queue is empty..

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

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

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

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

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

Info Setting expire flag and enqueued again 2 tasks.

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

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

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

Info Executing Queue, till Queue is empty..

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

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

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

B.2.6 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.2.7 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.2.8 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 1608511039s, requested for 1608511020s
Crontab execution number 2 at 1608511159s, requested for 1608511140s
Result (Timing of crontasks): [ 1608511039, 1608511159 ] (<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): 1608511039 (<class 'int'>)
Expectation (Submitted value number 1): 1608511020 <= result <= 1608511051
Submitted value number 1 is correct (Content 1608511039 in [1608511020 ... 1608511051] and
↪  Type is <class 'int'>).
Result (Submitted value number 2): 1608511159 (<class 'int'>)
Expectation (Submitted value number 2): 1608511140 <= result <= 1608511171
Submitted value number 2 is correct (Content 1608511159 in [1608511140 ... 1608511171] and
↪  Type is <class 'int'>).
```

C Test-Coverage

C.1 task

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

C.1.1 task.__init__.py

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

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

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

**Author:**

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

**Description:**

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

**Submodules:**

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

**Unittest:**

        See also the :download:`unittest <../../task/_testresults_/unittest.pdf>` documentation.
"""
__DEPENDENCIES__ = []

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

try:
    from config import APP_NAME as ROOT_LOGGER_NAME
except ImportError:
    ROOT_LOGGER_NAME = 'root'
logger = logging.getLogger(ROOT_LOGGER_NAME).getChild(__name__)

__DESCRIPTION__ = """The Module {\\tt %s} is designed to help with task issues like periodic
    tasks, delayed tasks, queues, threaded queues and crontabs.
For more Information read the documentation.""" % __name__.replace('_', '\_')
"""The Module Description"""
__INTERPRETER__ = (2, 3)
"""The Tested Interpreter-Versions"""


class queue(object):
```

```
55      """Class to execute queued methods.
56
57      :param bool expire: The default value for expire. See also :py:func:`expire`.
58
59      **Example:**
60
61      .. literalinclude:: ../../task/_examples_/queue.py
62
63      Will result to the following output:
64
65      .. literalinclude:: ../../task/_examples_/queue.log
66      """
67      class job(object):
68          def __init__(self, priority, callback, *args, **kwargs):
69              self.priority = priority
70              self.callback = callback
71              self.args = args
72              self.kwargs = kwargs
73
74          def run(self, queue):
75              self.callback(queue, *self.args, **self.kwargs)
76
77          def __lt__(self, other):
78              return self.priority < other.priority
79
80      def __init__(self, expire=True):
81          self.__expire = expire
82          self.__stop = False
83          self.queue = PriorityQueue()
84
85      def clean_queue(self):
86          """
87          This Methods removes all jobs from the queue.
88
89          .. note:: Be aware that already runnung jobs will not be terminated.
90          """
91          while not self.queue.empty():
92              try:
93                  self.queue.get(False)
94              except Empty:         # This block is hard to reach for a testcase, but is
95                  continue          # needed, if the thread runs dry while cleaning the queue.
96              self.queue.task_done()
97
98      def enqueue(self, priority, method, *args, **kwargs):
99          """
100         This enqueues a given callback.
101
102         :param number priority: The priority indication number of this task. The lowest value
    will be queued first.
103         :param method method: Method to be executed
104         :param args args: Arguments to be given to method
105         :param kwargs kwargs: Kewordsarguments to be given to method
106
107         .. note:: Called method will get this instance as first argument, followed by :py:data:`
    args` und :py:data:`kwargs`.
108         """
109         self.queue.put(self.job(priority, method, *args, **kwargs))
110
111     def qsize(self):
112         return self.queue.qsize()
113
114     def run(self):
```

```
115         """
116         This starts the execution of the queued methods.
117         """
118         self.__stop = False
119         while not self.__stop:
120             try:
121                 self.queue.get(timeout=0.1).run(self)
122             except Empty:
123                 if self.__expire:
124                     break
125         if type(self) is threaded_queue:
126             self.thread = None
127 
128     def expire(self):
129         """
130         This sets the expire flag. That means that the process will stop after queue gets empty.
131         """
132         self.__expire = True
133 
134     def stop(self):
135         """
136         This sets the stop flag. That means that the process will stop after finishing the active
      task.
137         """
138         self.__stop = True
139 
140 
141 class threaded_queue(queue):
142     """Class to execute queued methods in a background thread (See also parent :py:class:`queue`)
    .
143 
144     :param bool expire: The default value for expire. See also :py:func:`queue.expire`.
145 
146     **Example:**
147 
148     .. literalinclude:: ../../task/_examples_/threaded_queue.py
149 
150     Will result to the following output:
151 
152     .. literalinclude:: ../../task/_examples_/threaded_queue.log
153     """
154     def __init__(self, expire=False):
155         queue.__init__(self, expire=expire)
156         self.thread = None
157 
158     def run(self):
159         if self.thread is None:
160             self.thread = threading.Thread(target=self._start, args=())
161             self.thread.daemon = True    # Daemonize thread
162             self.thread.start()          # Start the execution
163 
164     def join(self):
165         """
166         This blocks till the queue is empty.
167 
168         .. note:: If the queue does not run dry, join will block till the end of the days.
169         """
170         self.expire()
171         if self.thread is not None:
172             self.thread.join()
173 
```

```
174     def stop(self):
175         queue.stop(self)
176         self.join()
177 
178     def _start(self):
179         queue.run(self)
180 
181 
182 class periodic(object):
183     """
184     :param float cycle_time: Cycle time in seconds -- method will be executed every *cycle_time* seconds
185     :param method method: Method to be executed
186     :param args args: Arguments to be given to method
187     :param kwargs kwargs: Kewordsarguments to be given to method
188 
189     Class to execute a method cyclicly.
190 
191     .. note:: Called method will get this instance as first argument, followed by :py:data:`args` und :py:data:`kwargs`.
192 
193     **Example:**
194 
195     .. literalinclude:: ../../task/_examples_/periodic.py
196 
197     Will result to the following output:
198 
199     .. literalinclude:: ../../task/_examples_/periodic.log
200     """
201     def __init__(self, cycle_time, method, *args, **kwargs):
202         self._lock = threading.Lock()
203         self._timer = None
204         self.method = method
205         self.cycle_time = cycle_time
206         self.args = args
207         self.kwargs = kwargs
208         self._stopped = True
209         self._last_tm = None
210         self.dt = None
211 
212     def join(self, timeout=0.1):
213         """
214         This blocks till the cyclic task is terminated.
215 
216         :param float timeout: Cycle time for checking if task is stopped
217 
218         .. note:: Using join means that somewhere has to be a condition calling :py:func:`stop` to terminate.
219         """
220         while not self._stopped:
221             time.sleep(timeout)
222 
223     def run(self):
224         """
225         This starts the cyclic execution of the given method.
226         """
227         if self._stopped:
228             self._set_timer(force_now=True)
229 
230     def stop(self):
231         """
232         This stops the execution of any following task.
233         """
```

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

```
290     """Class to execute a callback at the specified time conditions. See also parent :py:class:`
        periodic`.
291
292     :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.
293     :type accuracy: float
294
295     **Example:**
296
297     .. literalinclude:: ../../task/_examples_/crontab.py
298
299     Will result to the following output:
300
301     .. literalinclude:: ../../task/_examples_/crontab.log
302     """
303     ANY = '*'
304     """Constant for matching every condition."""
305
306     class cronjob(object):
307         """Class to handle cronjob parameters and cronjob changes.
308
309         :param minute: Minute for execution. Either 0...59, [0...59, 0...59, ...] or :py:const:`
        crontab.ANY` for every Minute.
310         :type minute: int, list, str
311         :param hour: Hour for execution. Either 0...23, [0...23, 0...23, ...] or :py:const:`
        crontab.ANY` for every Hour.
312         :type hour: int, list, str
313         :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.
314         :type day_of_month: int, list, str
315         :param month: Month for execution. Either 0...12, [0...12, 0...12, ...] or :py:const:`
        crontab.ANY` for every Month.
316         :type month: int, list, str
317         :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.
318         :type day_of_week: int, list, str
319         :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.
320         :type callback: func
321
322         .. 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()`.
323         """
324         class all_match(set):
325             """Universal set - match everything"""
326             def __contains__(self, item):
327                 (item)
328                 return True
329
330         def __init__(self, minute, hour, day_of_month, month, day_of_week, callback, *args, **
        kwargs):
331             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)
332             self.callback = callback
333             self.args = args
334             self.kwargs = kwargs
335             self.__last_cron_check_time__ = None
336             self.__last_execution__ = None
337
338         def set_trigger_conditions(self, minute=None, hour=None, day_of_month=None, month=None,
        day_of_week=None):
```

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

```
396
397     def __init__(self, accuracy=30):
398         periodic.__init__(self, accuracy, self.__periodic__)
399         self.__crontab__ = []
400
401     def __periodic__(self, rt):
402         (rt)
403         tm = int(time.time())
404         for cronjob in self.__crontab__:
405             cronjob.cron_execution(tm)
406
407     def add_cronjob(self, minute, hour, day_of_month, month, day_of_week, callback, *args, **
    kwargs):
408         """This Method adds a cronjob to be executed.
409
410         :param minute: Minute for execution. Either 0...59, [0...59, 0...59, ...] or :py:const:`
    crontab.ANY` for every Minute.
411         :type minute: int, list, str
412         :param hour: Hour for execution. Either 0...23, [0...23, 0...23, ...] or :py:const:`
    crontab.ANY` for every Hour.
413         :type hour: int, list, str
414         :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.
415         :type day_of_month: int, list, str
416         :param month: Month for execution. Either 0...12, [0...12, 0...12, ...] or :py:const:`
    crontab.ANY` for every Month.
417         :type month: int, list, str
418         :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.
419         :type day_of_week: int, list, str
420         :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.
421         :type callback: func
422
423         .. note:: The ``callback`` will be executed with it's instance of :py:class:`cronjob` as
    the first parameter.
424         """
425         self.__crontab__.append(self.cronjob(minute, hour, day_of_month, month, day_of_week,
    callback, *args, **kwargs))
```