Unittest for `stringtools`

May 10, 2019

Contents

1 Test Information

1.1 Test Candidate Information

The Module stringtools is designed to support functionality for strings (e.g. transfer strings via a bytestream, compressing, extracting, ...). For more Information read the sphinx documentation.

Library Information	
Name	stringtools
State	Released
Supported Interpreters	python2, python3
Version	9ebad0f7e7cb358cf530ff95dff9675b
Dependencies	
report	a7e82f164db67e701435d4b95395bbc0
stringtools	9ebad0f7e7cb358cf530ff95dff9675b

1.2 Unittest Information

Unittest Information	
Version	b713fb069a516102306982e0b42f8bb8
Testruns with	python 2.7.15 (candidate), python 3.6.7 (final)

1.3 Test System Information

System Information	
Architecture	64bit
Distribution	LinuxMint 19.1 tessa
Hostname	ahorn
Kernel	4.15.0-48-generic (#51-Ubuntu SMP Wed Apr 3 08:28:49 UTC 2019)
Machine	x86_64
Path	/user_data/data/dirk/prj/modules/stringtools/unittest
System	Linux
Username	dirk

2 Statistic

2.1 Test-Statistic for testrun with python 2.7.15 (candidate)

Number of tests	**14**
Number of successfull tests	**14**
Number of possibly failed tests	**0**
Number of failed tests	**0**
Executionlevel	Full Test (all defined tests)
Time consumption	0.018s

2.2 Test-Statistic for testrun with python 3.6.7 (final)

Number of tests	**14**
Number of successfull tests	**14**
Number of possibly failed tests	**0**
Number of failed tests	**0**
Executionlevel	Full Test (all defined tests)
Time consumption	0.017s

2.3 Coverage Statistic

Module- or Filename	Line-Coverage	Branch-Coverage
stringtools	100.0%	100.0%
stringtools.__init__.py	100.0%	
stringtools.csp.py	100.0%	
stringtools.stp.py	100.0%	

3 Summary for testrun with python 2.7.15 (candidate)

3.1 Tests with status Success (14)

3.1.1 stringtools.hexlify: Test length information of the output of hexlify method

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (21)
Start-Time:	2019-05-10 01:23:56,315
Finished-Time:	2019-05-10 01:23:56,316
Time-Consumption	0.000s
Testsummary:	
Info	Checking test pattern with length 11.
Success	Length pattern of hexlify method is correct (Content '(11):' and Type is <type 'str'>).

3.1.2 stringtools.hexlify: Test data information of the output of hexlify method

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (22)
Start-Time:	2019-05-10 01:23:56,316
Finished-Time:	2019-05-10 01:23:56,316
Time-Consumption	0.000s
Testsummary:	
Info	Checking test pattern with length 11.

Success	Data pattern of hexlify method is correct (Content '11 27 10 74 65 73 74 64 61 74 61' and Type is <type 'str'>).

3.1.3 stringtools.linefeed_filter: Test output of filter

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (23)
Start-Time:	2019-05-10 01:23:56,316
Finished-Time:	2019-05-10 01:23:56,317
Time-Consumption	0.000s
Testsummary:	
Info	Checking test pattern with length 11.
Success	Returnvalue of linefeed_filter is correct (Content 'test//r//n123//r//n' and Type is <type 'str'>).

3.1.4 stringtools.gzip*: Test compress and extract method

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (24)
Start-Time:	2019-05-10 01:23:56,317
Finished-Time:	2019-05-10 01:23:56,319
Time-Consumption	0.002s
Testsummary:	
Info	Compressing Testdata
Success	Zipped data with and without logger as argument is correct (Content (28): 1f 8b 08 00 8c b6 d4 5c 02 ff 63 60 20 15 08 fc 27 19 00 00 4e ba ce c0 65 00 00 00 and Type is <type 'str'>).
Info	Extracting Testdata
Success	Unzipped data with and without logger as argument is correct (Content (101): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff and Type is <type 'str'>).
Success	Result after compressing and extracting Testdata is correct (Content (101): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff and Type is <type 'str'>).

3.1.5 stringtools.stp.build_frame: Test STP Frame creation

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (25)
Start-Time:	2019-05-10 01:23:56,319
Finished-Time:	2019-05-10 01:23:56,320

Time-Consumption	0.000s

Testsummary:

Info	Creating testframe for "testframe: for stp"
Success	STP-Frame is correct (Content ':<testframe:= for stp:>' and Type is <type 'str'>).

3.1.6 stringtools.stp.stp: Test STP Frame processing

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (26)
Start-Time:	2019-05-10 01:23:56,320
Finished-Time:	2019-05-10 01:23:56,320
Time-Consumption	0.001s

Testsummary:

Info	Processing testframe: ":<testframe:= for stp:>"
Success	Processed STP-Frame is correct (Content 'testframe: for stp' and Type is <type 'str'>).

3.1.7 stringtools.stp.stp: Test processing wrong data type and wrong length

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (27)
Start-Time:	2019-05-10 01:23:56,321
Finished-Time:	2019-05-10 01:23:56,321
Time-Consumption	0.001s

Testsummary:

Info	Processing wrong data
Success	Return value if processing wrong data is correct (Content None and Type is <type 'NoneType'>).
Success	State after processing wrong data is correct (Content 0 and Type is <type 'int'>).

3.1.8 stringtools.stp.stp: Test processing data before start of frame

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (28)
Start-Time:	2019-05-10 01:23:56,321
Finished-Time:	2019-05-10 01:23:56,322
Time-Consumption	0.001s

Testsummary:

Info	Processing data which is not an stp frame.
Success	Return value list if processing 3 bytes of data before start of frame is correct (Content [None, None, None] and Type is <type 'list'>).

Success	State after processing data before start of frame is correct (Content 0 and Type is <type 'int'>).

3.1.9 stringtools.stp.stp: Test processing no clear_buffer after first data_sync

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (29)
Start-Time:	2019-05-10 01:23:56,322
Finished-Time:	2019-05-10 01:23:56,324
Time-Consumption	0.001s

Testsummary:

Info	Processing data with an unsufficient start pattern.
Success	Return value list if processing incorrect start of frame is correct (Content [None, None] and Type is <type 'list'>).
Success	State after processing incorrect start of frame is correct (Content 0 and Type is <type 'int'>).
Info	Processing data with an unsufficient start pattern (two times sync).
Success	Return value list if processing data_sync twice is correct (Content [None, None] and Type is <type 'list'>).
Success	State after processing data_sync twice is correct (Content 1 and Type is <type 'int'>).

3.1.10 stringtools.stp.stp: Test processing incorrect end frame pattern

This test was passed with the state: Success. See also full trace in section A.1.10!

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (30)
Start-Time:	2019-05-10 01:23:56,324
Finished-Time:	2019-05-10 01:23:56,327
Time-Consumption	0.003s

Testsummary:

Info	Processing data with an unsufficient end pattern.
Success	Return value list if processing data_sync and data again after start of frame is correct (Content [None, None, None, None, None, None] and Type is <type 'list'>).
Success	State after processing data_sync and data again after start of frame is correct (Content 0 and Type is <type 'int'>).
Info	Processing data with an unsufficient end pattern (start pattern instead of end pattern).
Success	Return value list if processing 2nd start of frame is correct (Content [None, None, None, None, None, None] and Type is <type 'list'>).
Success	State after processing 2nd start of frame is correct (Content 3 and Type is <type 'int'>).
Success	Buffer size after processing 2nd start of frame is correct (Content 0 and Type is <type 'int'>).
Info	Processing data with an unsufficient end pattern (two times sync instead of end pattern).
Success	Return value list if processing data_sync twice after start of frame is correct (Content [None, None, None, None, None, None] and Type is <type 'list'>).
Success	State after processing data_sync twice after start of frame is correct (Content 1 and Type is <type 'int'>).

3.1.11 stringtools.stp.stp: Test processing data after state corruption

This test was passed with the state: Success. See also full trace in section A.1.11!

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (31)
Start-Time:	2019-05-10 01:23:56,327
Finished-Time:	2019-05-10 01:23:56,329
Time-Consumption	0.002s

Testsummary:

Info	Corrupting stp state and processing data.
Success	Return value list if processing start of a frame after state had been corrupted is correct (Content [None, None, None, None] and Type is <type 'list'>).
Success	State after processing start of a frame after state had been corrupted is correct (Content 3 and Type is <type 'int'>).

3.1.12 stringtools.csp.build_frame: Test CSP Frame creation

This test was passed with the state: Success. See also full trace in section A.1.12!

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (32)
Start-Time:	2019-05-10 01:23:56,330
Finished-Time:	2019-05-10 01:23:56,331
Time-Consumption	0.001s

Testsummary:

Info	Creating testframe for ":testframe: for csp"
Success	STP-Frame is correct (Content ':testframe: for csp/n' and Type is <type 'str'>).

3.1.13 stringtools.csp.csp: Test CSP Frame processing

This test was passed with the state: Success. See also full trace in section A.1.13!

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (33)
Start-Time:	2019-05-10 01:23:56,332
Finished-Time:	2019-05-10 01:23:56,336
Time-Consumption	0.005s

Testsummary:

Info	Processing testframe: ":testframe: for csp/n"
Success	Processed STP-Frame is correct (Content ':testframe: for csp' and Type is <type 'str'>).

3.1.14 stringtools.csp.csp: Test processing wrong data type and wrong length

This test was passed with the state: Success. See also full trace in section A.1.14!

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (34)

Start-Time:	2019-05-10 01:23:56,337
Finished-Time:	2019-05-10 01:23:56,338
Time-Consumption	0.001s

Testsummary:

Info	Processing wrong data
Success	Return value if processing wrong data is correct (Content None and Type is <type 'NoneType'>).
Success	Buffer length after processing wrong data is correct (Content 0 and Type is <type 'int'>).

4 Summary for testrun with python 3.6.7 (final)

4.1 Tests with status Success (14)

4.1.1 stringtools.hexlify: Test length information of the output of hexlify method

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (21)
Start-Time:	2019-05-10 01:23:56,716
Finished-Time:	2019-05-10 01:23:56,717
Time-Consumption	0.001s

Testsummary:

Info	Checking test pattern with length 11.
Success	Length pattern of hexlify method is correct (Content '(11):' and Type is <class 'str'>).

4.1.2 stringtools.hexlify: Test data information of the output of hexlify method

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (22)
Start-Time:	2019-05-10 01:23:56,717
Finished-Time:	2019-05-10 01:23:56,717
Time-Consumption	0.000s

Testsummary:

Info	Checking test pattern with length 11.
Success	Data pattern of hexlify method is correct (Content '11 27 10 74 65 73 74 64 61 74 61' and Type is <class 'str'>).

4.1.3 stringtools.linefeed_filter: Test output of filter

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (23)

Start-Time:	2019-05-10 01:23:56,718
Finished-Time:	2019-05-10 01:23:56,718
Time-Consumption	0.000s

Testsummary:

Info	Checking test pattern with length 11.
Success	Returnvalue of linefeed_filter is correct (Content b'test//r//n123//r//n' and Type is <class 'bytes'>).

4.1.4 stringtools.gzip*: Test compress and extract method

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (24)
Start-Time:	2019-05-10 01:23:56,718
Finished-Time:	2019-05-10 01:23:56,720
Time-Consumption	0.002s

Testsummary:

Info	Compressing Testdata
Success	Zipped data with and without logger as argument is correct (Content (28): 1f 8b 08 00 8c b6 d4 5c 02 ff 63 60 20 15 08 fc 27 19 00 00 4e ba ce c0 65 00 00 00 and Type is <class 'bytes'>).
Info	Extracting Testdata
Success	Unzipped data with and without logger as argument is correct (Content (101): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff and Type is <class 'bytes'>).
Success	Result after compressing and extracting Testdata is correct (Content (101): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff and Type is <class 'bytes'>).

4.1.5 stringtools.stp.build_frame: Test STP Frame creation

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (25)
Start-Time:	2019-05-10 01:23:56,720
Finished-Time:	2019-05-10 01:23:56,720
Time-Consumption	0.000s

Testsummary:

Info	Creating testframe for 'b'testframe: for stp"
Success	STP-Frame is correct (Content b':<testframe:= for stp:>' and Type is <class 'bytes'>).

4.1.6 stringtools.stp.stp: Test STP Frame processing

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (26)
Start-Time:	2019-05-10 01:23:56,721
Finished-Time:	2019-05-10 01:23:56,721
Time-Consumption	0.001s

Testsummary:

Info	Processing testframe: 'b':<testframe:= for stp:>"
Success	Processed STP-Frame is correct (Content b'testframe: for stp' and Type is <class 'bytes'>).

4.1.7 stringtools.stp.stp: Test processing wrong data type and wrong length

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (27)
Start-Time:	2019-05-10 01:23:56,721
Finished-Time:	2019-05-10 01:23:56,722
Time-Consumption	0.000s

Testsummary:

Info	Processing wrong data
Success	Return value if processing wrong data is correct (Content None and Type is <class 'NoneType'>).
Success	State after processing wrong data is correct (Content 0 and Type is <class 'int'>).

4.1.8 stringtools.stp.stp: Test processing data before start of frame

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (28)
Start-Time:	2019-05-10 01:23:56,722
Finished-Time:	2019-05-10 01:23:56,723
Time-Consumption	0.001s

Testsummary:

Info	Processing data which is not an stp frame.
Success	Return value list if processing 3 bytes of data before start of frame is correct (Content [None, None, None] and Type is <class 'list'>).
Success	State after processing data before start of frame is correct (Content 0 and Type is <class 'int'>).

4.1.9 stringtools.stp.stp: Test processing no clear_buffer after first data_sync

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

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (29)
Start-Time:	2019-05-10 01:23:56,723
Finished-Time:	2019-05-10 01:23:56,725
Time-Consumption	0.001s

Testsummary:

Info	Processing data with an unsufficient start pattern.
Success	Return value list if processing incorrect start of frame is correct (Content [None, None] and Type is <class 'list'>).
Success	State after processing incorrect start of frame is correct (Content 0 and Type is <class 'int'>).
Info	Processing data with an unsufficient start pattern (two times sync).
Success	Return value list if processing data_sync twice is correct (Content [None, None] and Type is <class 'list'>).
Success	State after processing data_sync twice is correct (Content 1 and Type is <class 'int'>).

4.1.10 stringtools.stp.stp: Test processing incorrect end frame pattern

This test was passed with the state: Success. See also full trace in section B.1.10!

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (30)
Start-Time:	2019-05-10 01:23:56,725
Finished-Time:	2019-05-10 01:23:56,727
Time-Consumption	0.002s

Testsummary:

Info	Processing data with an unsufficient end pattern.
Success	Return value list if processing data_sync and data again after start of frame is correct (Content [None, None, None, None, None, None] and Type is <class 'list'>).
Success	State after processing data_sync and data again after start of frame is correct (Content 0 and Type is <class 'int'>).
Info	Processing data with an unsufficient end pattern (start pattern instead of end pattern).
Success	Return value list if processing 2nd start of frame is correct (Content [None, None, None, None, None, None] and Type is <class 'list'>).
Success	State after processing 2nd start of frame is correct (Content 3 and Type is <class 'int'>).
Success	Buffer size after processing 2nd start of frame is correct (Content 0 and Type is <class 'int'>).
Info	Processing data with an unsufficient end pattern (two times sync instead of end pattern).
Success	Return value list if processing data_sync twice after start of frame is correct (Content [None, None, None, None, None, None] and Type is <class 'list'>).
Success	State after processing data_sync twice after start of frame is correct (Content 1 and Type is <class 'int'>).

4.1.11 stringtools.stp.stp: Test processing data after state corruption

This test was passed with the state: Success. See also full trace in section B.1.11!

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (31)
Start-Time:	2019-05-10 01:23:56,727

Finished-Time:	2019-05-10 01:23:56,728
Time-Consumption	0.001s

Testsummary:	
Info	Corrupting stp state and processing data.
Success	Return value list if processing start of a frame after state had been corrupted is correct (Content [None, None, None, None] and Type is <class 'list'>).
Success	State after processing start of a frame after state had been corrupted is correct (Content 3 and Type is <class 'int'>).

4.1.12 stringtools.csp.build_frame: Test CSP Frame creation

This test was passed with the state: Success. See also full trace in section B.1.12!

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (32)
Start-Time:	2019-05-10 01:23:56,728
Finished-Time:	2019-05-10 01:23:56,729
Time-Consumption	0.001s

Testsummary:	
Info	Creating testframe for 'b':testframe: for csp"
Success	STP-Frame is correct (Content b':testframe: for csp/n' and Type is <class 'bytes'>).

4.1.13 stringtools.csp.csp: Test CSP Frame processing

This test was passed with the state: Success. See also full trace in section B.1.13!

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (33)
Start-Time:	2019-05-10 01:23:56,729
Finished-Time:	2019-05-10 01:23:56,734
Time-Consumption	0.005s

Testsummary:	
Info	Processing testframe: 'b':testframe: for csp/n"
Success	Processed STP-Frame is correct (Content b':testframe: for csp' and Type is <class 'bytes'>).

4.1.14 stringtools.csp.csp: Test processing wrong data type and wrong length

This test was passed with the state: Success. See also full trace in section B.1.14!

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (34)
Start-Time:	2019-05-10 01:23:56,735
Finished-Time:	2019-05-10 01:23:56,737
Time-Consumption	0.002s

Testsummary:	
Info	Processing wrong data

Success	Return value if processing wrong data is correct (Content None and Type is <class 'NoneType'>).
Success	Buffer length after processing wrong data is correct (Content 0 and Type is <class 'int'>).

A Trace for testrun with python 2.7.15 (candidate)

A.1 Tests with status Success (14)

A.1.1 stringtools.hexlify: Test length information of the output of hexlify method

This test was passed with the state: Success.

Info	Checking test pattern with length 11.

Success	Length pattern of hexlify method is correct (Content '(11):' and Type is <type 'str'>).

```
Result: '(11):' (<type 'str'>)
Expectation: result = '(11):' (<type 'str'>)
```

A.1.2 stringtools.hexlify: Test data information of the output of hexlify method

This test was passed with the state: Success.

Info	Checking test pattern with length 11.

Success	Data pattern of hexlify method is correct (Content '11 27 10 74 65 73 74 64 61 74 61' and Type is <type 'str'>).

```
Result: '11 27 10 74 65 73 74 64 61 74 61' (<type 'str'>)
Expectation: result = '11 27 10 74 65 73 74 64 61 74 61' (<type 'str'>)
```

A.1.3 stringtools.linefeed_filter: Test output of filter

This test was passed with the state: Success.

Info	Checking test pattern with length 11.

Success	Returnvalue of linefeed_filter is correct (Content 'test//r//n123//r//n' and Type is <type 'str'>).

```
Result: 'test\\r\\n123\\r\\n' (<type 'str'>)
Expectation: result = 'test\\r\\n123\\r\\n' (<type 'str'>)
```

A.1.4 stringtools.gzip*: Test compress and extract method

This test was passed with the state: Success.

Info	Compressing Testdata

```
Finished to compress a string (compression_rate=0.277, consumed_time=0.0s).
```

Success	Zipped data with and without logger as argument is correct (Content (28): 1f 8b 08 00 8c b6 d4 5c 02 ff 63 60 20 15 08 fc 27 19 00 00 4e ba ce c0 65 00 00 00 and Type is <type 'str'>).

```
Result: (28): 1f 8b 08 00 8c b6 d4 5c 02 ff 63 60 20 15 08 fc 27 19 00 00 4e ba ce c0 65 00
↪  00 00 (<type 'str'>)
```

```
Expectation: result = (28): 1f 8b 08 00 8c b6 d4 5c 02 ff 63 60 20 15 08 fc 27 19 00 00 4e ba
↪  ce c0 65 00 00 00 (<type 'str'>)
```

Info	Extracting Testdata

```
Finished to extract a string (compression_rate=0.277, consumed_time=0.0s).
```

Success	Unzipped data with and without logger as argument is correct (Content (101): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff and Type is <type 'str'>).

```
Result: (101): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
↪  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff (<type 'str'>)
```

```
Expectation: result = (101): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
↪  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10
↪  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff (<type 'str'>)
```

Success	Result after compressing and extracting Testdata is correct (Content (101): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff and Type is <type 'str'>).

```
Result: (101): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
↪  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff (<type 'str'>)
```

```
Expectation: result = (101): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
↪  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10
↪  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff (<type 'str'>)
```

A.1.5 stringtools.stp.build_frame: Test STP Frame creation

This test was passed with the state: Success.

Info	Creating testframe for "testframe: for stp"

Success	STP-Frame is correct (Content ':<testframe:= for stp:>' and Type is <type 'str'>).

```
Result: ':<testframe:= for stp:>' (<type 'str'>)
Expectation: result = ':<testframe:= for stp:>' (<type 'str'>)
```

A.1.6 stringtools.stp.stp: Test STP Frame processing

This test was passed with the state: Success.

Info	Processing testframe: ":<testframe:= for stp:>"

```
STP:        data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP:        clear buffer (3c) received => changing state STP_STATE_ESCAPE_1 ->
↪  STP_STATE_STORE_DATA
STP:        data sync (3a) received => changing state STP_STATE_STORE_DATA ->
↪  STP_STATE_ESCAPE_2
STP:        store sync value (3d) received => changing state STP_STATE_ESCAPE_2 ->
↪  STP_STATE_STORE_DATA
STP:        data sync (3a) received => changing state STP_STATE_STORE_DATA ->
↪  STP_STATE_ESCAPE_2
STP:        message completed (3e) => changing state STP_STATE_ESCAPE_2 -> STP_STATE_IDLE
```

Success	Processed STP-Frame is correct (Content 'testframe: for stp' and Type is <type 'str'>).

```
Result: 'testframe: for stp' (<type 'str'>)
Expectation: result = 'testframe: for stp' (<type 'str'>)
```

A.1.7 stringtools.stp.stp: Test processing wrong data type and wrong length

This test was passed with the state: Success.

Info	Processing wrong data

```
STP:        got wrong data [1, 2, 3]. Expecting a string or unicode with length 1.
```

Success	Return value if processing wrong data is correct (Content None and Type is <type 'NoneType'>).

```
Result: None (<type 'NoneType'>)
Expectation: result = None (<type 'NoneType'>)
```

Success	State after processing wrong data is correct (Content 0 and Type is <type 'int'>).

```
Result: 0 (<type 'int'>)
Expectation: result = 0 (<type 'int'>)
```

A.1.8 stringtools.stp.stp: Test processing data before start of frame

This test was passed with the state: Success.

Info	Processing data which is not an stp frame.

```
STP:        no data sync (31) received => ignoring byte
STP:        no data sync (32) received => ignoring byte
STP:        no data sync (33) received => ignoring byte
```

Success	Return value list if processing 3 bytes of data before start of frame is correct (Content [None, None, None] and Type is <type 'list'>).

```
Result: [ None, None, None ] (<type 'list'>)
Expectation: result = [ None, None, None ] (<type 'list'>)
```

Success	State after processing data before start of frame is correct (Content 0 and Type is <type 'int'>).

```
Result: 0 (<type 'int'>)
Expectation: result = 0 (<type 'int'>)
```

A.1.9 stringtools.stp.stp: Test processing no clear_buffer after first data_sync

This test was passed with the state: Success.

Info	Processing data with an unsufficient start pattern.

```
STP:        data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP:        no clear buffer/ data_sync (31) received => changing state STP_STATE_ESCAPE_1 ->
↪  STP_STATE_IDLE
```

Success	Return value list if processing incorrect start of frame is correct (Content [None, None] and Type is <type 'list'>).

```
Result: [ None, None ] (<type 'list'>)
Expectation: result = [ None, None ] (<type 'list'>)
```

Success	State after processing incorrect start of frame is correct (Content 0 and Type is <type 'int'>).

```
Result: 0 (<type 'int'>)
Expectation: result = 0 (<type 'int'>)
```

Info Processing data with an unsufficient start pattern (two times sync).

```
STP:        data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP:        2nd data sync (3a) received => keep state
```

Success Return value list if processing data_sync twice is correct (Content [None, None] and Type is <type 'list'>).

```
Result: [ None, None ] (<type 'list'>)
Expectation: result = [ None, None ] (<type 'list'>)
```

Success State after processing data_sync twice is correct (Content 1 and Type is <type 'int'>).

```
Result: 1 (<type 'int'>)
Expectation: result = 1 (<type 'int'>)
```

A.1.10 stringtools.stp.stp: Test processing incorrect end frame pattern

This test was passed with the state: Success.

Info Processing data with an unsufficient end pattern.

```
STP:        data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP:        clear buffer (3c) received => changing state STP_STATE_ESCAPE_1 ->
↪  STP_STATE_STORE_DATA
STP:        data sync (3a) received => changing state STP_STATE_STORE_DATA ->
↪  STP_STATE_ESCAPE_2
STP:        data (64) received => changing state STP_STATE_ESCAPE_2 -> STP_STATE_IDLE
```

Success Return value list if processing data_sync and data again after start of frame is correct (Content [None, None, None, None, None, None] and Type is <type 'list'>).

```
Result: [ None, None, None, None, None, None ] (<type 'list'>)
Expectation: result = [ None, None, None, None, None, None ] (<type 'list'>)
```

Success State after processing data_sync and data again after start of frame is correct (Content 0 and Type is <type 'int'>).

```
Result: 0 (<type 'int'>)
Expectation: result = 0 (<type 'int'>)
```

Info Processing data with an unsufficient end pattern (start pattern instead of end pattern).

```
STP:       data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP:       clear buffer (3c) received => changing state STP_STATE_ESCAPE_1 ->
↪  STP_STATE_STORE_DATA
STP:       data sync (3a) received => changing state STP_STATE_STORE_DATA ->
↪  STP_STATE_ESCAPE_2
STP:       clear buffer (3c) received => chunking "(2): 74 65" and changing state
↪  STP_STATE_ESCAPE_2 -> STP_STATE_STORE_DATA
```

Success Return value list if processing 2nd start of frame is correct (Content [None, None, None, None, None, None] and Type is <type 'list'>).

```
Result: [ None, None, None, None, None, None ] (<type 'list'>)
Expectation: result = [ None, None, None, None, None, None ] (<type 'list'>)
```

Success State after processing 2nd start of frame is correct (Content 3 and Type is <type 'int'>).

```
Result: 3 (<type 'int'>)
Expectation: result = 3 (<type 'int'>)
```

Success Buffer size after processing 2nd start of frame is correct (Content 0 and Type is <type 'int'>).

```
Result: 0 (<type 'int'>)
Expectation: result = 0 (<type 'int'>)
```

Info Processing data with an unsufficient end pattern (two times sync instead of end pattern).

```
STP:       data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP:       clear buffer (3c) received => changing state STP_STATE_ESCAPE_1 ->
↪  STP_STATE_STORE_DATA
STP:       data sync (3a) received => changing state STP_STATE_STORE_DATA ->
↪  STP_STATE_ESCAPE_2
STP:       data sync (3a) received => changing state STP_STATE_ESCAPE_2 -> STP_STATE_ESCAPE_1
```

Success Return value list if processing data_sync twice after start of frame is correct (Content [None, None, None, None, None, None] and Type is <type 'list'>).

```
Result: [ None, None, None, None, None, None ] (<type 'list'>)
Expectation: result = [ None, None, None, None, None, None ] (<type 'list'>)
```

Success State after processing data_sync twice after start of frame is correct (Content 1 and Type is <type 'int'>).

```
Result: 1 (<type 'int'>)
Expectation: result = 1 (<type 'int'>)
```

A.1.11 stringtools.stp.stp: Test processing data after state corruption

This test was passed with the state: Success.

Info	Corrupting stp state and processing data.

```
STP:       unknown state (255) => changing state -> STP_STATE_IDLE and executing process with
↪  (3a)
STP:       data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP:       clear buffer (3c) received => changing state STP_STATE_ESCAPE_1 ->
↪  STP_STATE_STORE_DATA
```

Success	Return value list if processing start of a frame after state had been corrupted is correct (Content [None, None, None, None] and Type is <type 'list'>).

```
Result: [ None, None, None, None ] (<type 'list'>)
Expectation: result = [ None, None, None, None ] (<type 'list'>)
```

Success	State after processing start of a frame after state had been corrupted is correct (Content 3 and Type is <type 'int'>).

```
Result: 3 (<type 'int'>)
Expectation: result = 3 (<type 'int'>)
```

A.1.12 stringtools.csp.build_frame: Test CSP Frame creation

This test was passed with the state: Success.

Info	Creating testframe for ":testframe: for csp"

Success	STP-Frame is correct (Content ':testframe: for csp/n' and Type is <type 'str'>).

```
Result: ':testframe: for csp\n' (<type 'str'>)
Expectation: result = ':testframe: for csp\n' (<type 'str'>)
```

A.1.13 stringtools.csp.csp: Test CSP Frame processing

This test was passed with the state: Success.

Info	Processing testframe: ":testframe: for csp/n"

```
CSP:        Adding ':' to buffer
CSP:        Adding 't' to buffer
CSP:        Adding 'e' to buffer
CSP:        Adding 's' to buffer
CSP:        Adding 't' to buffer
CSP:        Adding 'f' to buffer
CSP:        Adding 'r' to buffer
CSP:        Adding 'a' to buffer
CSP:        Adding 'm' to buffer
CSP:        Adding 'e' to buffer
CSP:        Adding ':' to buffer
CSP:        Adding ' ' to buffer
CSP:        Adding 'f' to buffer
CSP:        Adding 'o' to buffer
CSP:        Adding 'r' to buffer
CSP:        Adding ' ' to buffer
CSP:        Adding 'c' to buffer
CSP:        Adding 's' to buffer
CSP:        Adding 'p' to buffer
CSP:        Message identified by seperator 0a
```

Success Processed STP-Frame is correct (Content ':testframe: for csp' and Type is <type 'str'>).

```
Result: ':testframe: for csp' (<type 'str'>)
Expectation: result = ':testframe: for csp' (<type 'str'>)
```

A.1.14 stringtools.csp.csp: Test processing wrong data type and wrong length

This test was passed with the state: Success.

Info Processing wrong data

```
CSP:        got wrong data [1, 2, 3]
```

Success Return value if processing wrong data is correct (Content None and Type is <type 'NoneType'>).

```
Result: None (<type 'NoneType'>)
Expectation: result = None (<type 'NoneType'>)
```

Success Buffer length after processing wrong data is correct (Content 0 and Type is <type 'int'>).

```
Result: 0 (<type 'int'>)
Expectation: result = 0 (<type 'int'>)
```

B Trace for testrun with python 3.6.7 (final)

B.1 Tests with status Success (14)

B.1.1 stringtools.hexlify: Test length information of the output of hexlify method

This test was passed with the state: Success.

Info	Checking test pattern with length 11.

Success	Length pattern of hexlify method is correct (Content '(11):' and Type is <class 'str'>).

```
Result: '(11):' (<class 'str'>)
Expectation: result = '(11):' (<class 'str'>)
```

B.1.2 stringtools.hexlify: Test data information of the output of hexlify method

This test was passed with the state: Success.

Info	Checking test pattern with length 11.

Success	Data pattern of hexlify method is correct (Content '11 27 10 74 65 73 74 64 61 74 61' and Type is <class 'str'>).

```
Result: '11 27 10 74 65 73 74 64 61 74 61' (<class 'str'>)
Expectation: result = '11 27 10 74 65 73 74 64 61 74 61' (<class 'str'>)
```

B.1.3 stringtools.linefeed_filter: Test output of filter

This test was passed with the state: Success.

Info	Checking test pattern with length 11.

Success	Returnvalue of linefeed_filter is correct (Content b'test//r//n123//r//n' and Type is <class 'bytes'>).

```
Result: b'test\\r\\n123\\r\\n' (<class 'bytes'>)
Expectation: result = b'test\\r\\n123\\r\\n' (<class 'bytes'>)
```

B.1.4 stringtools.gzip*: Test compress and extract method

This test was passed with the state: Success.

Info	Compressing Testdata

```
Finished to compress a string (compression_rate=0.277, consumed_time=0.0s).
```

Success	Zipped data with and without logger as argument is correct (Content (28): 1f 8b 08 00 8c b6 d4 5c 02 ff 63 60 20 15 08 fc 27 19 00 00 4e ba ce c0 65 00 00 00 and Type is <class 'bytes'>).

```
Result: (28): 1f 8b 08 00 8c b6 d4 5c 02 ff 63 60 20 15 08 fc 27 19 00 00 4e ba ce c0 65 00
↪  00 00 (<class 'bytes'>)
```

```
Expectation: result = (28): 1f 8b 08 00 8c b6 d4 5c 02 ff 63 60 20 15 08 fc 27 19 00 00 4e ba
↪  ce c0 65 00 00 00 (<class 'bytes'>)
```

Info	Extracting Testdata

```
Finished to extract a string (compression_rate=0.277, consumed_time=0.0s).
```

Success	Unzipped data with and without logger as argument is correct (Content (101): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff and Type is <class 'bytes'>).

```
Result: (101): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
↪  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff (<class 'bytes'>)
```

```
Expectation: result = (101): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
↪  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10
↪  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff (<class 'bytes'>)
```

Success	Result after compressing and extracting Testdata is correct (Content (101): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff and Type is <class 'bytes'>).

```
Result: (101): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
↪  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff (<class 'bytes'>)
```

```
Expectation: result = (101): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
↪  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10
↪  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff (<class 'bytes'>)
```

B.1.5 stringtools.stp.build_frame: Test STP Frame creation

This test was passed with the state: Success.

Info	Creating testframe for 'b'testframe: for stp"

Success	STP-Frame is correct (Content b':<testframe:= for stp:>' and Type is <class 'bytes'>).

```
Result: b':<testframe:= for stp:>' (<class 'bytes'>)
Expectation: result = b':<testframe:= for stp:>' (<class 'bytes'>)
```

B.1.6 stringtools.stp.stp: Test STP Frame processing

This test was passed with the state: Success.

Info	Processing testframe: 'b':<testframe:= for stp:>"

```
STP:        data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP:        clear buffer (3c) received => changing state STP_STATE_ESCAPE_1 ->
↪  STP_STATE_STORE_DATA
STP:        data sync (3a) received => changing state STP_STATE_STORE_DATA ->
↪  STP_STATE_ESCAPE_2
STP:        store sync value (3d) received => changing state STP_STATE_ESCAPE_2 ->
↪  STP_STATE_STORE_DATA
STP:        data sync (3a) received => changing state STP_STATE_STORE_DATA ->
↪  STP_STATE_ESCAPE_2
STP:        message completed (3e) => changing state STP_STATE_ESCAPE_2 -> STP_STATE_IDLE
```

Success	Processed STP-Frame is correct (Content b'testframe: for stp' and Type is <class 'bytes'>).

```
Result: b'testframe: for stp' (<class 'bytes'>)
Expectation: result = b'testframe: for stp' (<class 'bytes'>)
```

B.1.7 stringtools.stp.stp: Test processing wrong data type and wrong length

This test was passed with the state: Success.

Info	Processing wrong data

```
STP:        got wrong data [1, 2, 3]. Expecting a string or unicode with length 1.
```

Success	Return value if processing wrong data is correct (Content None and Type is <class 'NoneType'>).

```
Result: None (<class 'NoneType'>)
Expectation: result = None (<class 'NoneType'>)
```

Success	State after processing wrong data is correct (Content 0 and Type is <class 'int'>).

```
Result: 0 (<class 'int'>)
Expectation: result = 0 (<class 'int'>)
```

B.1.8 stringtools.stp.stp: Test processing data before start of frame

This test was passed with the state: Success.

Info	Processing data which is not an stp frame.

```
STP:        no data sync (31) received => ignoring byte
STP:        no data sync (32) received => ignoring byte
STP:        no data sync (33) received => ignoring byte
```

Success	Return value list if processing 3 bytes of data before start of frame is correct (Content [None, None, None] and Type is <class 'list'>).

```
Result: [ None, None, None ] (<class 'list'>)
Expectation: result = [ None, None, None ] (<class 'list'>)
```

Success	State after processing data before start of frame is correct (Content 0 and Type is <class 'int'>).

```
Result: 0 (<class 'int'>)
Expectation: result = 0 (<class 'int'>)
```

B.1.9 stringtools.stp.stp: Test processing no clear_buffer after first data_sync

This test was passed with the state: Success.

Info	Processing data with an unsufficient start pattern.

```
STP:        data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP:        no clear buffer/ data_sync (31) received => changing state STP_STATE_ESCAPE_1 ->
↪  STP_STATE_IDLE
```

Success	Return value list if processing incorrect start of frame is correct (Content [None, None] and Type is <class 'list'>).

```
Result: [ None, None ] (<class 'list'>)
Expectation: result = [ None, None ] (<class 'list'>)
```

Success	State after processing incorrect start of frame is correct (Content 0 and Type is <class 'int'>).

```
Result: 0 (<class 'int'>)
Expectation: result = 0 (<class 'int'>)
```

Info	Processing data with an unsufficient start pattern (two times sync).

```
STP:        data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP:        2nd data sync (3a) received => keep state
```

Success	Return value list if processing data_sync twice is correct (Content [None, None] and Type is <class 'list'>).

```
Result: [ None, None ] (<class 'list'>)
Expectation: result = [ None, None ] (<class 'list'>)
```

Success	State after processing data_sync twice is correct (Content 1 and Type is <class 'int'>).

```
Result: 1 (<class 'int'>)
Expectation: result = 1 (<class 'int'>)
```

B.1.10 stringtools.stp.stp: Test processing incorrect end frame pattern

This test was passed with the state: Success.

Info	Processing data with an unsufficient end pattern.

```
STP:        data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP:        clear buffer (3c) received => changing state STP_STATE_ESCAPE_1 ->
↪  STP_STATE_STORE_DATA
STP:        data sync (3a) received => changing state STP_STATE_STORE_DATA ->
↪  STP_STATE_ESCAPE_2
STP:        data (64) received => changing state STP_STATE_ESCAPE_2 -> STP_STATE_IDLE
```

Success	Return value list if processing data_sync and data again after start of frame is correct (Content [None, None, None, None, None, None] and Type is <class 'list'>).

```
Result: [ None, None, None, None, None, None ] (<class 'list'>)
Expectation: result = [ None, None, None, None, None, None ] (<class 'list'>)
```

Success	State after processing data_sync and data again after start of frame is correct (Content 0 and Type is <class 'int'>).

```
Result: 0 (<class 'int'>)
Expectation: result = 0 (<class 'int'>)
```

Info	Processing data with an unsufficient end pattern (start pattern instead of end pattern).

```
STP:       data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP:       clear buffer (3c) received => changing state STP_STATE_ESCAPE_1 ->
↪  STP_STATE_STORE_DATA
STP:       data sync (3a) received => changing state STP_STATE_STORE_DATA ->
↪  STP_STATE_ESCAPE_2
STP:       clear buffer (3c) received => chunking "(2): 74 65" and changing state
↪  STP_STATE_ESCAPE_2 -> STP_STATE_STORE_DATA
```

Success Return value list if processing 2nd start of frame is correct (Content [None, None, None, None, None, None] and Type is <class 'list'>).

```
Result: [ None, None, None, None, None, None ] (<class 'list'>)
Expectation: result = [ None, None, None, None, None, None ] (<class 'list'>)
```

Success State after processing 2nd start of frame is correct (Content 3 and Type is <class 'int'>).

```
Result: 3 (<class 'int'>)
Expectation: result = 3 (<class 'int'>)
```

Success Buffer size after processing 2nd start of frame is correct (Content 0 and Type is <class 'int'>).

```
Result: 0 (<class 'int'>)
Expectation: result = 0 (<class 'int'>)
```

Info Processing data with an unsufficient end pattern (two times sync instead of end pattern).

```
STP:       data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP:       clear buffer (3c) received => changing state STP_STATE_ESCAPE_1 ->
↪  STP_STATE_STORE_DATA
STP:       data sync (3a) received => changing state STP_STATE_STORE_DATA ->
↪  STP_STATE_ESCAPE_2
STP:       data sync (3a) received => changing state STP_STATE_ESCAPE_2 -> STP_STATE_ESCAPE_1
```

Success Return value list if processing data_sync twice after start of frame is correct (Content [None, None, None, None, None, None] and Type is <class 'list'>).

```
Result: [ None, None, None, None, None, None ] (<class 'list'>)
Expectation: result = [ None, None, None, None, None, None ] (<class 'list'>)
```

Success State after processing data_sync twice after start of frame is correct (Content 1 and Type is <class 'int'>).

```
Result: 1 (<class 'int'>)
Expectation: result = 1 (<class 'int'>)
```

B.1.11 stringtools.stp.stp: Test processing data after state corruption

This test was passed with the state: Success.

Info	Corrupting stp state and processing data.

```
STP:       unknown state (255) => changing state -> STP_STATE_IDLE and executing process with
↪  (3a)
STP:       data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP:       clear buffer (3c) received => changing state STP_STATE_ESCAPE_1 ->
↪  STP_STATE_STORE_DATA
```

Success	Return value list if processing start of a frame after state had been corrupted is correct (Content [None, None, None, None] and Type is <class 'list'>).

```
Result: [ None, None, None, None ] (<class 'list'>)
Expectation: result = [ None, None, None, None ] (<class 'list'>)
```

Success	State after processing start of a frame after state had been corrupted is correct (Content 3 and Type is <class 'int'>).

```
Result: 3 (<class 'int'>)
Expectation: result = 3 (<class 'int'>)
```

B.1.12 stringtools.csp.build_frame: Test CSP Frame creation

This test was passed with the state: Success.

Info	Creating testframe for 'b':testframe: for csp"

Success	STP-Frame is correct (Content b':testframe: for csp/n' and Type is <class 'bytes'>).

```
Result: b':testframe: for csp\n' (<class 'bytes'>)
Expectation: result = b':testframe: for csp\n' (<class 'bytes'>)
```

B.1.13 stringtools.csp.csp: Test CSP Frame processing

This test was passed with the state: Success.

Info	Processing testframe: 'b':testframe: for csp/n"

```
CSP:        Adding b':' to buffer
CSP:        Adding b't' to buffer
CSP:        Adding b'e' to buffer
CSP:        Adding b's' to buffer
CSP:        Adding b't' to buffer
CSP:        Adding b'f' to buffer
CSP:        Adding b'r' to buffer
CSP:        Adding b'a' to buffer
CSP:        Adding b'm' to buffer
CSP:        Adding b'e' to buffer
CSP:        Adding b':' to buffer
CSP:        Adding b' ' to buffer
CSP:        Adding b'f' to buffer
CSP:        Adding b'o' to buffer
CSP:        Adding b'r' to buffer
CSP:        Adding b' ' to buffer
CSP:        Adding b'c' to buffer
CSP:        Adding b's' to buffer
CSP:        Adding b'p' to buffer
CSP:        Message identified by seperator 0a
```

Success Processed STP-Frame is correct (Content b':testframe: for csp' and Type is <class 'bytes'>).

```
Result: b':testframe: for csp' (<class 'bytes'>)
Expectation: result = b':testframe: for csp' (<class 'bytes'>)
```

B.1.14 stringtools.csp.csp: Test processing wrong data type and wrong length

This test was passed with the state: Success.

Info Processing wrong data

```
CSP:        got wrong data [1, 2, 3]
```

Success Return value if processing wrong data is correct (Content None and Type is <class 'NoneType'>).

```
Result: None (<class 'NoneType'>)
Expectation: result = None (<class 'NoneType'>)
```

Success Buffer length after processing wrong data is correct (Content 0 and Type is <class 'int'>).

```
Result: 0 (<class 'int'>)
Expectation: result = 0 (<class 'int'>)
```

C Test-Coverage

C.1 stringtools

The line coverage for stringtools was 100.0%
The branch coverage for stringtools was 100.0%

C.1.1 stringtools.__init__.py

The line coverage for stringtools.__init__.py was 100.0%
The branch coverage for stringtools.__init__.py was 100.0%

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
"""
stringtools (Stringtools)
=========================

**Author:**

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

**Description:**

    This Module supports functionality around string operations.

**Submodules:**

* :mod:`stringtools.csp`
* :mod:`stringtools.stp`
* :func:`gzip_compress`
* :func:`gzip_extract`
* :func:`hexlify`

**Unittest:**

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

from stringtools import stp
from stringtools import csp
__DEPENDENCIES__ = ['report', 'stringtools']

import sys
if sys.version_info < (3,0):
    from cStringIO import StringIO
import gzip
import logging
import time


__DESCRIPTION__ = """The Module {\\tt %s} is designed to support functionality for strings (e.g.
    transfer strings via a bytestream, compressing, extracting, ...).
For more Information read the sphinx documentation.""" % __name__.replace('_', '\_')
"""The Module Description"""
__INTERPRETER__ = (2, 3)
```

```
"""The Tested Interpreter-Versions"""

__all__ = ['gzip_compress',
           'gzip_extract',
           'hexlify',
           'csp',
           'stp']


def gzip_compress(s, compresslevel=9, logger=None):
    """
    Method to compress a stream of bytes.

    :param str s: The bytestream (string) to be compressed
    :param int compresslevel: An optional compressionn level (default is 9)
    :param logger: An optinal `Logger` instance, if logging should be used
    :return: The compressed bytestream
    :rtype: str
    """
    rv = None
    t = time.time()
    if sys.version_info >= (3, 0):
        rv = gzip.compress(s, compresslevel)
    else:
        buf = StringIO()
        f = gzip.GzipFile(mode='wb', compresslevel=compresslevel, fileobj=buf)
        try:
            f.write(s)
        finally:
            f.close()
            rv = buf.getvalue()
            buf.close()
    if logger is not None and rv is not None:
        logger.debug('Finished to compress a string (compression_rate=%.3f, consumed_time=%.1fs).
    ', len(rv) / float(len(s)), time.time() - t)
    return rv


def gzip_extract(s, logger=None):
    """
    Method to extract data from a compress stream of bytes.

    :param str s: The compressed bytestream (string) to be extracted
    :param logger: An optinal `Logger` instance, if logging should be used
    :return: The extracted data
    :rtype: str
    """
    t = time.time()
    rv = None
    if sys.version_info >= (3,0):
        rv = gzip.decompress(s)
    else:
        inbuffer = StringIO(s)
        f = gzip.GzipFile(mode='rb', fileobj=inbuffer)
        try:
            rv = f.read()
        finally:
            f.close()
            inbuffer.close()
    if logger is not None and rv is not None:
        logger.debug('Finished to extract a string (compression_rate=%.3f, consumed_time=%.1fs).'
    , len(s) / float(len(rv)), time.time() - t)
    return rv
```

```
106
107
108 def hexlify(s):
109     """Method to hexlify a string.
110
111     :param str s: A string including the bytes to be hexlified.
112     :returns: The hexlified string
113     :rtype: str
114
115     **Example:**
116
117     .. literalinclude:: ../../stringtools/_examples_/hexlify.py
118
119     Will result to the following output:
120
121     .. literalinclude:: ../../stringtools/_examples_/hexlify.log
122     """
123     rv = '(%d):' % len(s)
124     for byte in s:
125         if sys.version_info >= (3, 0):
126             rv += ' %02x' % byte
127         else:
128             rv += ' %02x' % ord(byte)
129     return rv
130
131
132 def linefeed_filter(s):
133     """Method to change linefeed and carriage return to '\\\\n' and '\\\\r'
134
135     :param str s: A string including carriage return and/ or linefeed.
136     :returns: A string with converted carriage return and/ or linefeed.
137     :rtype: str
138     """
139     if sys.version_info >= (3, 0):
140         return s.replace(b'\r', b'\\r').replace(b'\n', b'\\n')
141     else:
142         return s.replace('\r', '\\r').replace('\n', '\\n')
```

C.1.2 stringtools.csp.py

The line coverage for stringtools.csp.py was 100.0%
The branch coverage for stringtools.csp.py was 100.0%

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 #
4 """
5 csp (Character seperation protocol)
6 ==================================
7
8 **Author:**
9
10 * Dirk Alders <sudo-dirk@mount-mockery.de>
11
12 **Description:**
13
14     This module is a submodule of :mod:`stringtools` and creates an frame to transmit and receive
       messages via an serial interface.
15
16 **Submodules:**
```

```
17
18 * :class:`stringtools.csp.csp`
19 * :func:`stringtools.csp.build_frame`
20
21 **Unittest:**
22
23         See also the :download:`unittest <../../stringtools/_testresults_/unittest.pdf>`
       documentation.
24 """
25
26 import report
27
28 import sys
29
30 DATA_SEPERATOR = b"\n"
31
32
33 class csp(report.logit):
34     """This class extracts messages from an "csp-stream".
35
36     **Example:**
37
38     .. literalinclude:: ../../stringtools/_examples_/csp.csp.py
39
40     Will result to the following output:
41
42     .. literalinclude:: ../../stringtools/_examples_/csp.csp.log
43     """
44     LOG_PREFIX = 'CSP:'
45
46     def __init__(self, seperator=DATA_SEPERATOR):
47         self.__rx_buffer__ = b""
48         self.__seperator__ = seperator
49
50     def process(self, b, logger=None):
51         """
52         This processes a byte out of a "stp-stream".
53
54         :param str b: A single byte
55         :param logger: An optinal `Logger` instance, if a logging should be used
56         :returns: The extracted message or None, if no message is identified yet
57         :rtype: str
58         """
59         if sys.version_info >= (3, 0):
60             if type(b) is not int:
61                 self.logit_warning(logger, 'got wrong data %s', repr(b))
62                 return None
63             b = bytes([b])
64         else:
65             if type(b) not in [str, unicode] or len(b) != 1:
66                 self.logit_warning(logger, 'got wrong data %s', repr(b))
67                 return None
68         #
69         if b == self.__seperator__:
70             self.logit_debug(logger, 'Message identified by seperator %02x', ord(self.
       __seperator__))
71             rv = self.__rx_buffer__
72             self.__rx_buffer__ = b""
73             return rv
74         else:
75             self.logit_debug(logger, 'Adding %s to buffer', repr(b))
76             self.__rx_buffer__ += b
77             return None
```

```
78
79
80 def build_frame(msg, seperator=DATA_SEPERATOR):
81     """This Method builds an "csp-frame" to be transfered via a stream.
82
83     :param str data: A String (Bytes) to be framed
84     :returns: The "csp-framed" message to be sent
85     :rtype: str
86
87     **Example:**
88
89     .. literalinclude:: ../../stringtools/_examples_/csp.build_frame.py
90
91     Will result to the following output:
92
93     .. literalinclude:: ../../stringtools/_examples_/csp.build_frame.log
94     """
95     return msg + seperator
```

C.1.3 stringtools.stp.py

The line coverage for stringtools.stp.py was 100.0%
The branch coverage for stringtools.stp.py was 100.0%

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 #
4 """
5 stp (Serial transfer protocol)
6 ==============================
7
8 **Author:**
9
10 * Dirk Alders <sudo-dirk@mount-mockery.de>
11
12 **Description:**
13
14     This module is a submodule of :mod:`stringtools` and creates an serial frame to transmit and
      receive messages via an serial interface.
15
16 **Submodules:**
17
18 * :class:`stringtools.stp.stp`
19 * :func:`stringtools.stp.build_frame`
20
21 **Unittest:**
22
23         See also the :download:`unittest <../../stringtools/_testresults_/unittest.pdf>`
      documentation.
24 """
25
26 import report
27 import stringtools
28
29 import sys
30
31 DATA_SYNC = b'\x3a'
32 """The data sync byte"""
33 DATA_CLEAR_BUFFER = b'\x3c'
34 """The clear buffer byte ('\\\\x3a\\\\x3c' -> start of message)"""
```

```
35 DATA_VALID_MSG = b'\x3e'
36 """The valid message byte ('\\\\x3a\\\\x3e' -> end of message)"""
37 DATA_STORE_SYNC_VALUE = b'\x3d'
38 """The store sync value byte ('\\\\x3a\\\\x3d' -> '\\\\x3a' inside a message)"""
39
40 STP_STATE_IDLE = 0x00
41 """Idle state definition (default)"""
42 STP_STATE_ESCAPE_1 = 0x01
43 """Escape 1 state definition ('\\\\x3a\\\\x3c' found)"""
44 STP_STATE_ESCAPE_2 = 0x02
45 """Escape 2 state definition ('\\\\x3a' found inside a message)"""
46 STP_STATE_STORE_DATA = 0x03
47 """Store data state definition (start of message found; data will be stored)"""
48
49
50 class stp(report.logit):
51     """This class extracts messages from an "stp-stream".
52
53     **Example:**
54
55     .. literalinclude:: ../../stringtools/_examples_/stp.stp.py
56
57     Will result to the following output:
58
59     .. literalinclude:: ../../stringtools/_examples_/stp.stp.log
60     """
61     LOG_PREFIX = 'STP:'
62
63     def __init__(self):
64         self.state = STP_STATE_IDLE
65         self.buffer = ""
66
67     def process(self, b, logger=None):
68         """
69         This processes a byte out of a "stp-stream".
70
71         :param str b: A single byte
72         :param logger: An optinal `Logger` instance, if a logging should be used
73         :returns: The extracted message or None, if no message is identified yet
74         :rtype: str
75         """
76         if sys.version_info >= (3, 0):
77             if type(b) is not int:
78                 self.logit_warning(logger, 'got wrong data %s. Expecting a string or unicode with
     length 1.', repr(b))
79                 return None
80             b = bytes([b])
81         else:
82             if type(b) not in [str, unicode] or len(b) != 1:
83                 self.logit_warning(logger, 'got wrong data %s. Expecting a string or unicode with
     length 1.', repr(b))
84                 return None
85
86         if self.state == STP_STATE_IDLE:
87             if b == DATA_SYNC:
88                 self.state = STP_STATE_ESCAPE_1
89                 self.logit(logger, report.logging.DEBUG, 'data sync (%02x) received => changing
    state STP_STATE_IDLE -> STP_STATE_ESCAPE_1', ord(b))
90             else:
```

```
91                  self.logit_warning(logger, 'no data sync (%02x) received => ignoring byte', ord(b
    ))
92              return None
93          if self.state == STP_STATE_ESCAPE_1:
94              if b == DATA_CLEAR_BUFFER:
95                  self.buffer = b""
96                  self.state = STP_STATE_STORE_DATA
97                  self.logit(logger, report.logging.DEBUG, 'clear buffer (%02x) received =>
    changing state STP_STATE_ESCAPE_1 -> STP_STATE_STORE_DATA', ord(b))
98              elif b != DATA_SYNC:
99                  self.state = STP_STATE_IDLE
100                 self.logit_warning(logger, 'no clear buffer/ data_sync (%02x) received =>
    changing state STP_STATE_ESCAPE_1 -> STP_STATE_IDLE', ord(b))
101             else:
102                 self.logit_warning(logger, '2nd data sync (%02x) received => keep state', ord(b))
103             return None
104         if self.state == STP_STATE_STORE_DATA:
105             if b == DATA_SYNC:
106                 self.state = STP_STATE_ESCAPE_2
107                 self.logit(logger, report.logging.DEBUG, 'data sync (%02x) received => changing
    state STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2', ord(b))
108             else:
109                 self.buffer += b
110             return None
111         if self.state == STP_STATE_ESCAPE_2:
112             if b == DATA_CLEAR_BUFFER:
113                 self.logit_warning(logger, 'clear buffer (%02x) received => chunking "%s" and
    changing state STP_STATE_ESCAPE_2 -> STP_STATE_STORE_DATA', ord(b), stringtools.hexlify(self.
    buffer))
114                 self.buffer = b""
115                 self.state = STP_STATE_STORE_DATA
116             elif b == DATA_VALID_MSG:
117                 self.state = STP_STATE_IDLE
118                 self.logit(logger, report.logging.DEBUG, 'message completed (%02x) => changing
    state STP_STATE_ESCAPE_2 -> STP_STATE_IDLE', ord(b))
119                 return self.buffer
120             elif b == DATA_STORE_SYNC_VALUE:
121                 self.state = STP_STATE_STORE_DATA
122                 self.logit(logger, report.logging.DEBUG, 'store sync value (%02x) received =>
    changing state STP_STATE_ESCAPE_2 -> STP_STATE_STORE_DATA', ord(b))
123                 self.buffer += DATA_SYNC
124             elif b == DATA_SYNC:
125                 self.state = STP_STATE_ESCAPE_1
126                 self.logit_warning(logger, 'data sync (%02x) received => changing state
    STP_STATE_ESCAPE_2 -> STP_STATE_ESCAPE_1', ord(b))
127             else:
128                 self.state = STP_STATE_IDLE
129                 self.logit_warning(logger, 'data (%02x) received => changing state
    STP_STATE_ESCAPE_2 -> STP_STATE_IDLE', ord(b))
130             return None
131
132         self.logit(logger, report.logging.CRITICAL, 'unknown state (%s) => changing state ->
    STP_STATE_IDLE and executing process with (%02x)', self.state, ord(b))
133         self.state = STP_STATE_IDLE
134         self.process(b[0], logger)
135         return None
136
137
138 def build_frame(data):
```

```
139     """This Method builds an "stp-frame" to be transfered via a stream.
140
141     :param str data: A String (Bytes) to be framed
142     :returns: The "stp-framed" message to be sent
143     :rtype: str
144
145     **Example:**
146
147     .. literalinclude:: ../../stringtools/_examples_/stp.build_frame.py
148
149     Will result to the following output:
150
151     .. literalinclude:: ../../stringtools/_examples_/stp.build_frame.log
152     """
153     rv = DATA_SYNC + DATA_CLEAR_BUFFER
154
155     for byte in data:
156         if sys.version_info >= (3, 0):
157             byte = bytes([byte])
158         if byte == DATA_SYNC:
159             rv += DATA_SYNC + DATA_STORE_SYNC_VALUE
160         else:
161             rv += byte
162
163     rv += DATA_SYNC + DATA_VALID_MSG
164     return rv
```