Unittest for `stringtools`

December 25, 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	77981dbdc5e8fd54960c4f914c083602
Dependencies	

1.2 Unittest Information

Unittest Information	
Version	d720b191532eb97d77fbb9aafb67181a
Testruns with	python 2.7.17 (final), python 3.6.9 (final)

1.3 Test System Information

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

2 Statistic

2.1 Test-Statistic for testrun with python 2.7.17 (final)

Number of tests	**18**
Number of successfull tests	**18**
Number of possibly failed tests	**0**
Number of failed tests	**0**
Executionlevel	Full Test (all defined tests)
Time consumption	0.021s

2.2 Test-Statistic for testrun with python 3.6.9 (final)

Number of tests	**18**
Number of successfull tests	**18**
Number of possibly failed tests	**0**
Number of failed tests	**0**
Executionlevel	Full Test (all defined tests)
Time consumption	0.020s

2.3 Coverage Statistic

Module- or Filename	Line-Coverage	Branch-Coverage
stringtools	100.0%	97.1%
stringtools.__init__.py	100.0%	
stringtools.csp.py	100.0%	
stringtools.stp.py	100.0%	

3 Tested Requirements

3.1 Stream Definition

A Stream is from class bytes for python3 and from type str for python2.

3.2 Stream to Human readable String

3.2.1 Hexadecimal Values

Description
A Stream shall be converted to a human readable String containing all bytes as hexadecimal values seperated by a Space.

Reason for the implementation
Make non printable characters printable.

Fitcriterion
A stream shall be converted at least once and the hex values shall exist in the returnvalue in the correct order.

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

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (23)
Start-Time:	2019-12-25 16:38:16,495
Finished-Time:	2019-12-25 16:38:16,496
Time-Consumption	0.001s
Testsummary:	
Info	Checking test pattern de ad be ef (<type 'str'>).
Success	Pattern included all relevant information in the correct order.

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (23)
Start-Time:	2019-12-25 16:38:16,855
Finished-Time:	2019-12-25 16:38:16,856
Time-Consumption	0.001s
Testsummary:	
Info	Checking test pattern de ad be ef (<class 'bytes'>).
Success	Pattern included all relevant information in the correct order.

3.2.2 Number of Bytes

Description
The Length of a Stream surrounded by brakets shall be included in the human readable string.

Reason for the implementation
Show the length of a Stream without counting the seperated values.

Fitcriterion
The described pattern including the decimal number of bytes is included in the string for at least one Stream.

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

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (24)
Start-Time:	2019-12-25 16:38:16,496
Finished-Time:	2019-12-25 16:38:16,496
Time-Consumption	0.000s
Testsummary:	
Info	Checking test pattern with length 4.
Success	'(4)' is in '(4): de ad be ef' at position 0

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (24)
Start-Time:	2019-12-25 16:38:16,856
Finished-Time:	2019-12-25 16:38:16,857
Time-Consumption	0.000s
Testsummary:	
Info	Checking test pattern with length 4.
Success	'(4)' is in '(4): de ad be ef' at position 0

3.2.3 CRLF-Filter

Description
The module stringtools shall have a method to replace carriage return and line feed to their escaped representation.

Reason for the implementation
Replace these characters to make output printable (e.g. for logging a string based protocol).

Fitcriterion
Filter at least one string and check at least one CR and one LF representation.

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

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (25)
Start-Time:	2019-12-25 16:38:16,496
Finished-Time:	2019-12-25 16:38:16,497
Time-Consumption	0.000s

Testsummary:

Info	Checking test pattern with length 4.
Success	Returnvalue of linefeed_filter is correct (Content 'test//r//n123//r//n' and Type is <type 'str'>).

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (25)
Start-Time:	2019-12-25 16:38:16,857
Finished-Time:	2019-12-25 16:38:16,857
Time-Consumption	0.000s

Testsummary:

Info	Checking test pattern with length 4.
Success	Returnvalue of linefeed_filter is correct (Content b'test//r//n123//r//n' and Type is <class 'bytes'>).

3.3 Stream Compression

3.3.1 Compress

Description
The module stringtools shall have a method compressing a Stream with gzip.

Reason for the implementation
Speed up transfer with low transfer rate.

Fitcriterion
Compressed Stream is extractable and results in the original data.

Testresult

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

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (28)
Start-Time:	2019-12-25 16:38:16,497
Finished-Time:	2019-12-25 16:38:16,498
Time-Consumption	0.001s

Testsummary:

Info	Compressing Streams result in differnt streams with the same input stream. Therefore the test will compare the decompressed data.
Info	Compressing stream: (30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff
Info	Extracting stream: (26): 1f 8b 08 00 68 82 03 5e 02 ff 63 60 40 01 ff 51 01 00 2d 8a 7d de 1e 00 00 00
Success	Extracted data is correct (Content (30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff and Type is <type 'str'>).

Testresult

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (28)
Start-Time:	2019-12-25 16:38:16,857
Finished-Time:	2019-12-25 16:38:16,858
Time-Consumption	0.001s

Testsummary:

Info	Compressing Streams result in differnt streams with the same input stream. Therefore the test will compare the decompressed data.
Info	Compressing stream: (30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff
Info	Extracting stream: (26): 1f 8b 08 00 68 82 03 5e 02 ff 63 60 40 01 ff 51 01 00 2d 8a 7d de 1e 00 00 00
Success	Extracted data is correct (Content (30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff and Type is <class 'bytes'>).

3.3.2 Extract

Description

The module stringtools shall have a method extracting a Stream with gzip.

Reason for the implementation

Speed up transfer with low transfer rate.

Fitcriterion

Extracted Stream is equal to the original compressed data.

Testresult

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

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (29)
Start-Time:	2019-12-25 16:38:16,498
Finished-Time:	2019-12-25 16:38:16,498
Time-Consumption	0.001s

Testsummary:

Info	Extracting stream: (26): 1f 8b 08 00 34 e0 04 5d 02 ff 63 60 40 01 ff 51 01 00 2d 8a 7d de 1e 00 00 00
Success	Extracted data is correct (Content '(30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff' and Type is <type 'str'>).

Testresult

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (29)
Start-Time:	2019-12-25 16:38:16,858
Finished-Time:	2019-12-25 16:38:16,859
Time-Consumption	0.000s

Testsummary:

Info	Extracting stream: (26): 1f 8b 08 00 34 e0 04 5d 02 ff 63 60 40 01 ff 51 01 00 2d 8a 7d de 1e 00 00 00
Success	Extracted data is correct (Content '(30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff' and Type is <class 'str'>).

3.4 Carriagereturn Seperation Protocol (CSP)

The Carriagereturn Seperation Protocol shall use carriage return as the end pattern for message seperation.

3.4.1 Frame creation

Description

The CSP module shall support a method to create a Frame from a stream.

Reason for the implementation

Simple message or frame generation for streams (e.g. Keyboard (user input), RFID-Reader, . . .).

Fitcriterion

Creation of a testframe and checking the result.

Testresult

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

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (33)
Start-Time:	2019-12-25 16:38:16,499
Finished-Time:	2019-12-25 16:38:16,499
Time-Consumption	0.000s

Testsummary:

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

Testresult

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (33)
Start-Time:	2019-12-25 16:38:16,859
Finished-Time:	2019-12-25 16:38:16,859
Time-Consumption	0.000s

Testsummary:

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

3.4.2 Frame creation error

Description

The Frame creation Method shall raise ValueError, if a frame separation character is in the Source-String.

Reason for the implementation

String including separation charcter will be splitted in pieces while processing after transport.

Fitcriterion

ValueErroro is raised for at least one String including the separation character.

Testresult

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

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (34)
Start-Time:	2019-12-25 16:38:16,499
Finished-Time:	2019-12-25 16:38:16,499
Time-Consumption	0.000s

Testsummary:

Info	Creating testframe for ":testframe: for csp"
Success	CSP-Frame is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).

Testresult

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (34)
Start-Time:	2019-12-25 16:38:16,859
Finished-Time:	2019-12-25 16:38:16,859
Time-Consumption	0.000s

Testsummary:

Info	Creating testframe for 'b':testframe: for csp"
Success	CSP-Frame is correct (Content <class 'ValueError'> and Type is <class 'type'>).

3.4.3 Frame processing

Description

The CSP Module shall support a class including a method to process stream snipets of variable length. This Method shall return an empty list, if no frame has been detected, otherwise it shall return a list including detected frame(s).

Reason for the implementation

Support message analysis of a stream with every size.

Fitcriterion

At least one frame given in at least two snippets is identified correctly.

Testresult

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

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (35)
Start-Time:	2019-12-25 16:38:16,499
Finished-Time:	2019-12-25 16:38:16,500
Time-Consumption	0.001s

Testsummary:

Info	Processing testframe: ":testframe: for csp/n" in two snippets
Success	First processed CSP-Snippet is correct (Content [] and Type is <type 'list'>).
Success	Final processed CSP-Frame is correct (Content [':testframe: for csp'] and Type is <type 'list'>).

Testresult

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (35)
Start-Time:	2019-12-25 16:38:16,859
Finished-Time:	2019-12-25 16:38:16,860

Time-Consumption	0.001s

Testsummary:

Info	Processing testframe: 'b':testframe: for csp/n" in two snippets
Success	First processed CSP-Snippet is correct (Content [] and Type is <class 'list'>).
Success	Final processed CSP-Frame is correct (Content [b':testframe: for csp'] and Type is <class 'list'>).

3.4.4 Frame processing - Input data type error

Description

If the input data is not bytes for python3 or str for python 2, the process method shall raise TypeError.

Reason for the implementation

Type restriction.

Fitcriterion

At least the following types return TypeError (list, int, str for python3, unicode for python 2).

Testresult

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

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (36)
Start-Time:	2019-12-25 16:38:16,500
Finished-Time:	2019-12-25 16:38:16,502
Time-Consumption	0.002s

Testsummary:

Info	Processing wrong data (list)
Success	Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).
Success	Buffer still empty is correct (Content '' and Type is <type 'str'>).
Info	Processing wrong data (int)
Success	Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).
Success	Buffer still empty is correct (Content '' and Type is <type 'str'>).
Info	Processing wrong data (unicode)
Success	Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).
Success	Buffer still empty is correct (Content '' and Type is <type 'str'>).

Testresult

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

Testrun:	python 3.6.9 (final)

Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (36)
Start-Time:	2019-12-25 16:38:16,860
Finished-Time:	2019-12-25 16:38:16,861
Time-Consumption	0.001s

Testsummary:

Info	Processing wrong data (list)
Success	Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).
Success	Buffer still empty is correct (Content b'' and Type is <class 'bytes'>).
Info	Processing wrong data (int)
Success	Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).
Success	Buffer still empty is correct (Content b'' and Type is <class 'bytes'>).
Info	Processing wrong data (str)
Success	Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).
Success	Buffer still empty is correct (Content b'' and Type is <class 'bytes'>).

3.5 Serial Transfer Protocol (STP)

The Serial Transfer Protocol shall use a start pattern and an end pattern to identify a message in a stream. Both patterns shall be a two byte values starting with the same (sync-)byte.

3.5.1 Frame creation

Description
A frame creation method shall create a frame out of given input data.

Reason for the implementation
Message or Frame generation for streams (e.g. data transfer via bluetooth, ethernet, ...).

Fitcriterion
Creation of a testframe and checking the result.

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

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (40)
Start-Time:	2019-12-25 16:38:16,502
Finished-Time:	2019-12-25 16:38:16,502
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'>).

Testresult

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (40)
Start-Time:	2019-12-25 16:38:16,861
Finished-Time:	2019-12-25 16:38:16,862
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'>).

3.5.2 Frame creation - Start pattern and end pattern inside a message

Description

The frame creation method shall support existance of the start or end pattern in the data to be framed.

Reason for the implementation

Possibility to send any kind of data (including the patterns).

Fitcriterion

Creation of a testframe out of data including at least one start pattern and one end pattern and checking the result.

Testresult

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

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (41)
Start-Time:	2019-12-25 16:38:16,502
Finished-Time:	2019-12-25 16:38:16,503
Time-Consumption	0.000s

Testsummary:

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

Testresult

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (41)
Start-Time:	2019-12-25 16:38:16,862
Finished-Time:	2019-12-25 16:38:16,862
Time-Consumption	0.000s

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

3.5.3 Frame processing

Description

The STP Module shall support a class including a method to process stream snipets of variable length. This Method shall return an empty list, if no frame has been detected, otherwise it shall return a list including detected frame(s).

Reason for the implementation

Support message analysis of a stream with every size.

Fitcriterion

At least one frame given in at least two snippets is identified correctly.

Testresult

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

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (42)
Start-Time:	2019-12-25 16:38:16,503
Finished-Time:	2019-12-25 16:38:16,504
Time-Consumption	0.001s

Testsummary:	
Info	Processing testframe: ":<testframe for stp:>"
Success	First processed STP snippet is correct (Content [] and Type is <type 'list'>).
Success	Final processed STP snippet is correct (Content ['testframe for stp'] and Type is <type 'list'>).

Testresult

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (42)
Start-Time:	2019-12-25 16:38:16,862
Finished-Time:	2019-12-25 16:38:16,863
Time-Consumption	0.001s

Testsummary:	
Info	Processing testframe: 'b':<testframe for stp:>"
Success	First processed STP snippet is correct (Content [] and Type is <class 'list'>).
Success	Final processed STP snippet is correct (Content [b'testframe for stp'] and Type is <class 'list'>).

3.5.4 Frame processing - Input data type error

Description

If the input data is not bytes for python3 or str for python 2, the process method shall raise TypeError.

Reason for the implementation

Type restriction.

Fitcriterion

At least the following types return TypeError (list, int, str for python3, unicode for python 2).

Testresult

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

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (43)
Start-Time:	2019-12-25 16:38:16,504
Finished-Time:	2019-12-25 16:38:16,505
Time-Consumption	0.001s

Testsummary:

Info	Processing wrong data (list)
Success	Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).
Success	Buffer still empty is correct (Content '' and Type is <type 'str'>).
Info	Processing wrong data (int)
Success	Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).
Success	Buffer still empty is correct (Content '' and Type is <type 'str'>).
Info	Processing wrong data (unicode)
Success	Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).
Success	Buffer still empty is correct (Content '' and Type is <type 'str'>).

Testresult

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (43)
Start-Time:	2019-12-25 16:38:16,863
Finished-Time:	2019-12-25 16:38:16,864
Time-Consumption	0.002s

Testsummary:

Info	Processing wrong data (list)
Success	Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).
Success	Buffer still empty is correct (Content b'' and Type is <class 'bytes'>).

Info	Processing wrong data (int)
Success	Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).
Success	Buffer still empty is correct (Content b'' and Type is <class 'bytes'>).
Info	Processing wrong data (str)
Success	Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).
Success	Buffer still empty is correct (Content b'' and Type is <class 'bytes'>).

3.5.5 Frame processing - Start pattern and end pattern inside a message

Reason for the implementation

Possibility to send any kind of data (including the patterns).

Testresult

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

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (44)
Start-Time:	2019-12-25 16:38:16,505
Finished-Time:	2019-12-25 16:38:16,506
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 'list'>).

Testresult

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

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (44)
Start-Time:	2019-12-25 16:38:16,865
Finished-Time:	2019-12-25 16:38:16,866
Time-Consumption	0.002s
Testsummary:	
Info	Processing testframe: 'b':<testframe for :=<stp:⇒:>''
Success	Processed STP-Frame is correct (Content [b'testframe for :<stp:>'] and Type is <class 'list'>).

3.5.6 Frame processing - Data before the start pattern

Description

Data before the start pattern shall be ignored. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.15!

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (45)
Start-Time:	2019-12-25 16:38:16,506
Finished-Time:	2019-12-25 16:38:16,507
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 'list'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.15!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (45)
Start-Time:	2019-12-25 16:38:16,867
Finished-Time:	2019-12-25 16:38:16,869
Time-Consumption	0.002s

Testsummary:

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

3.5.7 Frame processing - Incorrect start patterns

Description

On receiving an incorrect start pattern, STP shall stay in ESCAPE_1, in case of data sync was received twice or back to state IDLE in all other faulty start patterns starting with data sync. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.16!

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (46)
Start-Time:	2019-12-25 16:38:16,507
Finished-Time:	2019-12-25 16:38:16,511
Time-Consumption	0.004s

Testsummary:

Info	Processing data with an insufficient start pattern.

Success	Return value list if processing incorrect start of frame is correct (Content [[]] 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 insufficient start pattern (two times sync).
Success	Return value list if processing data_sync twice is correct (Content [[]] and Type is <type 'list'>).
Success	State after processing data_sync twice is correct (Content 1 and Type is <type 'int'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.16!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (46)
Start-Time:	2019-12-25 16:38:16,869
Finished-Time:	2019-12-25 16:38:16,871
Time-Consumption	0.002s

Testsummary:

Info	Processing data with an insufficient start pattern.
Success	Return value list if processing incorrect start of frame is correct (Content [[]] 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 insufficient start pattern (two times sync).
Success	Return value list if processing data_sync twice is correct (Content [[]] and Type is <class 'list'>).
Success	State after processing data_sync twice is correct (Content 1 and Type is <class 'int'>).

3.5.8 Frame processing - Incorrect end pattern

Description

On receiving an incorrect end pattern, STP shall change to state STORE_DATA, in case of a start pattern, to ESCAPE_1, in case of data sync was received twice or back to state IDLE in all other faulty end patterns starting with data sync. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.17!

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (47)
Start-Time:	2019-12-25 16:38:16,511
Finished-Time:	2019-12-25 16:38:16,516
Time-Consumption	0.005s

Testsummary:

Info	Processing data with an insufficient end pattern.

Success	Return value list if processing data_sync and data again after start of frame is correct (Content [[]] 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'>).
Success	Buffer size after processing data with insufficient end pattern is correct (Content 0 and Type is <type 'int'>).
Info	Processing data with an insufficient end pattern (start pattern instead of end pattern).
Success	Return value list if processing 2nd start of frame is correct (Content [[]] 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 insufficient 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 [[]] 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'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.17!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (47)
Start-Time:	2019-12-25 16:38:16,871
Finished-Time:	2019-12-25 16:38:16,876
Time-Consumption	0.005s

Testsummary:

Info	Processing data with an insufficient end pattern.
Success	Return value list if processing data_sync and data again after start of frame is correct (Content [[]] 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'>).
Success	Buffer size after processing data with insufficient end pattern is correct (Content 0 and Type is <class 'int'>).
Info	Processing data with an insufficient end pattern (start pattern instead of end pattern).
Success	Return value list if processing 2nd start of frame is correct (Content [[]] 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 insufficient 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 [[]] 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'>).

3.5.9 Frame processing - After state corruption

Description

The state of STP shall be set to IDLE, after an unknown state was recognised. The currently processed data shall be

processed again. An error shall be given to the logger.

Reason for the implementation
Robustness against wrong or corrupted data.

Testresult
This test was passed with the state: **Success**. See also full trace in section A.1.18!

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (48)
Start-Time:	2019-12-25 16:38:16,516
Finished-Time:	2019-12-25 16:38:16,519
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 [[]] 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'>).
Success	Buffer size after corrupting stp state is correct (Content 2 and Type is <type 'int'>).

Testresult
This test was passed with the state: **Success**. See also full trace in section B.1.18!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (48)
Start-Time:	2019-12-25 16:38:16,877
Finished-Time:	2019-12-25 16:38:16,878
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 [[]] 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'>).
Success	Buffer size after corrupting stp state is correct (Content 2 and Type is <class 'int'>).

A Trace for testrun with python 2.7.17 (final)

A.1 Tests with status Info (18)

A.1.1 Hexadecimal Values

Description
A Stream shall be converted to a human readable String containing all bytes as hexadecimal values seperated by a Space.

Reason for the implementation
Make non printable characters printable.

Fitcriterion
A stream shall be converted at least once and the hex values shall exist in the returnvalue in the correct order.

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

Info	Checking test pattern de ad be ef (<type 'str'>).

Success	Pattern included all relevant information in the correct order.

```
Return value of hexlify is (4): de ad be ef
Using upper string for comparison: (4): DE AD BE EF
"DE" found in "(4): DE AD BE EF"... Reducing pattern
"AD" found in "AD BE EF"... Reducing pattern
"BE" found in "BE EF"... Reducing pattern
"EF" found in "EF"... Reducing pattern
```

A.1.2 Number of Bytes

Description
The Length of a Stream surrounded by brakets shall be included in the human readable string.

Reason for the implementation
Show the length of a Stream without counting the seperated values.

Fitcriterion
The described pattern including the decimal number of bytes is included in the string for at least one Stream.

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

Info	Checking test pattern with length 4.

Success	'(4)' is in '(4): de ad be ef' at position 0

A.1.3 CRLF-Filter

Description
The module stringtools shall have a method to replace carriage return and line feed to their escaped representation.

Reason for the implementation
Replace these characters to make output printable (e.g. for logging a string based protocol).

Fitcriterion
Filter at least one string and check at least one CR and one LF representation.

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

Info	Checking test pattern with length 4.

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 Compress

Description
The module stringtools shall have a method compressing a Stream with gzip.

Reason for the implementation
Speed up transfer with low transfer rate.

Fitcriterion
Compressed Stream is extractable and results in the original data.

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

Info	Compressing Streams result in differnt streams with the same input stream. Therefore the test will compare the decompressed data.

Info	Compressing stream: (30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

```
GZIP: Finished to compress a string (compression_rate=0.867, consumed_time=0.0s).
```

Info	Extracting stream: (26): 1f 8b 08 00 68 82 03 5e 02 ff 63 60 40 01 ff 51 01 00 2d 8a 7d de 1e 00 00 00

```
GZIP: Finished to extract a string (compression_rate=0.867, consumed_time=0.0s).
```

Success	Extracted data is correct (Content (30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff and Type is <type 'str'>).

```
Result: (30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff
↪  ff ff ff ff (<type 'str'>)
Expectation: result = (30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff (<type 'str'>)
```

A.1.5 Extract

Description
The module stringtools shall have a method extracting a Stream with gzip.

Reason for the implementation
Speed up transfer with low transfer rate.

Fitcriterion
Extracted Stream is equal to the original compressed data.

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

Info	Extracting stream: (26): 1f 8b 08 00 34 e0 04 5d 02 ff 63 60 40 01 ff 51 01 00 2d 8a 7d de 1e 00 00 00

```
GZIP: Finished to extract a string (compression_rate=0.867, consumed_time=0.0s).
```

Success	Extracted data is correct (Content '(30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff' and Type is <type 'str'>).

```
Result: '(30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff
↪  ff ff ff ff' (<type 'str'>)
Expectation: result = '(30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff' (<type 'str'>)
```

A.1.6 Frame creation

Description
The CSP module shall support a method to create a Frame from a stream.

Reason for the implementation
Simple message or frame generation for streams (e.g. Keyboard (user input), RFID-Reader, . . .).

Fitcriterion
Creation of a testframe and checking the result.

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

Info	Creating testframe for ":testframe: for csp"

Success	CSP-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.1.7 Frame creation error

Description
The Frame creation Method shall raise ValueError, if a frame separation character is in the Source-String.

Reason for the implementation
String including separation charcter will be splitted in pieces while processing after transport.

Fitcriterion
ValueErroro is raised for at least one String including the separation character.

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

Info	Creating testframe for ":testframe: for csp"

Success	CSP-Frame is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).

```
Result: <type 'exceptions.ValueError'> (<type 'type'>)
Expectation: result = <type 'exceptions.ValueError'> (<type 'type'>)
```

A.1.8 Frame processing

Description
The CSP Module shall support a class including a method to process stream snipets of variable length. This Method shall return an empty list, if no frame has been detected, otherwise it shall return a list including detected frame(s).

Reason for the implementation
Support message analysis of a stream with every size.

Fitcriterion
At least one frame given in at least two snippets is identified correctly.

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

Info	Processing testframe: ":testframe: for csp/n" in two snippets

```
CSP: Leaving data in buffer (to be processed next time): (10): 3a 74 65 73 74 66 72 61 6d 65
CSP: 1 messages identified
```

Success	First processed CSP-Snippet is correct (Content [] and Type is <type 'list'>).

```
Result: [  ] (<type 'list'>)
Expectation: result = [  ] (<type 'list'>)
```

Success	Final processed CSP-Frame is correct (Content [':testframe: for csp'] and Type is <type 'list'>).

```
Result: [ ':testframe: for csp' ] (<type 'list'>)
Expectation: result = [ ':testframe: for csp' ] (<type 'list'>)
```

A.1.9 Frame processing - Input data type error

Description
If the input data is not bytes for python3 or str for python 2, the process method shall raise TypeError.

Reason for the implementation
Type restriction.

Fitcriterion
At least the following types return TypeError (list, int, str for python3, unicode for python 2).

Testresult

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

Info	Processing wrong data (list)

Success	Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).

```
Result: <type 'exceptions.ValueError'> (<type 'type'>)
Expectation: result = <type 'exceptions.ValueError'> (<type 'type'>)
```

Success	Buffer still empty is correct (Content '' and Type is <type 'str'>).

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

Info	Processing wrong data (int)

Success	Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).

```
Result: <type 'exceptions.ValueError'> (<type 'type'>)
Expectation: result = <type 'exceptions.ValueError'> (<type 'type'>)
```

Success	Buffer still empty is correct (Content '' and Type is <type 'str'>).

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

Info	Processing wrong data (unicode)

Success	Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).

```
Result: <type 'exceptions.ValueError'> (<type 'type'>)
Expectation: result = <type 'exceptions.ValueError'> (<type 'type'>)
```

Success	Buffer still empty is correct (Content '' and Type is <type 'str'>).

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

A.1.10 Frame creation

Description

A frame creation method shall create a frame out of given input data.

Reason for the implementation
Message or Frame generation for streams (e.g. data transfer via bluetooth, ethernet, ...).

Fitcriterion
Creation of a testframe and checking the result.

Testresult
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.1.11 Frame creation - Start pattern and end pattern inside a message

Description
The frame creation method shall support existance of the start or end pattern in the data to be framed.

Reason for the implementation
Possibility to send any kind of data (including the patterns).

Fitcriterion
Creation of a testframe out of data including at least one start pattern and one end pattern and checking the result.

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

Info	Creating testframe including start and end pattern 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.1.12 Frame processing

Description
The STP Module shall support a class including a method to process stream snipets of variable length. This Method shall return an empty list, if no frame has been detected, otherwise it shall return a list including detected frame(s).

Reason for the implementation

Support message analysis of a stream with every size.

Fitcriterion

At least one frame given in at least two snippets is identified correctly.

Testresult

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: start pattern (3a 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: end pattern (3a 3e) received => storing message and changing state STP_STATE_ESCAPE_2 ->
↪  STP_STATE_IDLE
```

Success	First processed STP snippet is correct (Content [] and Type is <type 'list'>).

```
Result: [  ] (<type 'list'>)
Expectation: result = [  ] (<type 'list'>)
```

Success	Final processed STP snippet is correct (Content ['testframe for stp'] and Type is <type 'list'>).

```
Result: [ 'testframe for stp' ] (<type 'list'>)
Expectation: result = [ 'testframe for stp' ] (<type 'list'>)
```

A.1.13 Frame processing - Input data type error

Description

If the input data is not bytes for python3 or str for python 2, the process method shall raise TypeError.

Reason for the implementation

Type restriction.

Fitcriterion

At least the following types return TypeError (list, int, str for python3, unicode for python 2).

Testresult

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

Info	Processing wrong data (list)

Success	Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).

```
Result: <type 'exceptions.ValueError'> (<type 'type'>)
Expectation: result = <type 'exceptions.ValueError'> (<type 'type'>)
```

Success	Buffer still empty is correct (Content '' and Type is <type 'str'>).

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

Info	Processing wrong data (int)

Success	Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).

```
Result: <type 'exceptions.ValueError'> (<type 'type'>)
Expectation: result = <type 'exceptions.ValueError'> (<type 'type'>)
```

Success	Buffer still empty is correct (Content '' and Type is <type 'str'>).

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

Info	Processing wrong data (unicode)

Success	Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).

```
Result: <type 'exceptions.ValueError'> (<type 'type'>)
Expectation: result = <type 'exceptions.ValueError'> (<type 'type'>)
```

Success	Buffer still empty is correct (Content '' and Type is <type 'str'>).

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

A.1.14 Frame processing - Start pattern and end pattern inside a message

Reason for the implementation

Possibility to send any kind of data (including the patterns).

Testresult

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: start pattern (3a 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 pattern (3a 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: store sync pattern (3a 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: end pattern (3a 3e) received => storing message and changing state STP_STATE_ESCAPE_2 ->
↪  STP_STATE_IDLE
```

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

```
Result: [ 'testframe for :<stp:>' ] (<type 'list'>)
Expectation: result = [ 'testframe for :<stp:>' ] (<type 'list'>)
```

A.1.15 Frame processing - Data before the start pattern

Description

Data before the start pattern shall be ignored. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

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

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

```
STP: no data sync (5f) received => ignoring byte
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 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: end pattern (3a 3e) received => storing message and changing state STP_STATE_ESCAPE_2 ->
↪  STP_STATE_IDLE
```

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

```
Result: [ 'testframe for stp' ] (<type 'list'>)
Expectation: result = [ 'testframe for stp' ] (<type 'list'>)
```

A.1.16 Frame processing - Incorrect start patterns

Description

On receiving an incorrect start pattern, STP shall stay in ESCAPE_1, in case of data sync was received twice or back to state IDLE in all other faulty start patterns starting with data sync. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

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

Info	Processing data with an insufficient start pattern.

```
Sending ':1' to stp.
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: no start pattern (3a 31) received => changing state STP_STATE_ESCAPE_1 -> STP_STATE_IDLE
```

Success	Return value list if processing incorrect start of frame is correct (Content [[]] and Type is <type 'list'>).

```
Result: [ [  ] ] (<type 'list'>)
Expectation: result = [ [  ] ] (<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 insufficient start pattern (two times sync).

```
Sending '::' to stp.
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 [[]] and Type is <type 'list'>).

```
Result: [ [  ] ] (<type 'list'>)
Expectation: result = [ [  ] ] (<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.1.17 Frame processing - Incorrect end pattern

Description

On receiving an incorrect end pattern, STP shall change to state STORE_DATA, in case of a start pattern, to ESCAPE_1, in case of data sync was received twice or back to state IDLE in all other faulty end patterns starting with data sync. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

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

Info	Processing data with an insufficient end pattern.

```
Sending ':<te:d' to stp.
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 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
STP: Chunking "(2): 74 65" from buffer
```

Success	Return value list if processing data_sync and data again after start of frame is correct (Content [[]] and Type is <type 'list'>).

```
Result: [ [  ] ] (<type 'list'>)
Expectation: result = [ [  ] ] (<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'>)
```

Success	Buffer size after processing data with insufficient end pattern is correct (Content 0 and Type is <type 'int'>).

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

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

```
Sending ':<te:<' to stp.
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 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: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_2 ->
↪  STP_STATE_STORE_DATA
STP: Chunking "(2): 74 65" from buffer
```

Success Return value list if processing 2nd start of frame is correct (Content [[]] and Type is <type 'list'>).

```
Result: [ [  ] ] (<type 'list'>)
Expectation: result = [ [  ] ] (<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 insufficient end pattern (two times sync instead of end pattern).

```
Sending ':<te::' to stp.
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 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: second data sync (3a) received => changing state STP_STATE_ESCAPE_2 -> STP_STATE_ESCAPE_1
STP: Chunking "(2): 74 65" from buffer
```

Success Return value list if processing data_sync twice after start of frame is correct (Content [[]] and Type is <type 'list'>).

```
Result: [ [  ] ] (<type 'list'>)
Expectation: result = [ [  ] ] (<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.1.18 Frame processing - After state corruption

Description

The state of STP shall be set to IDLE, after an unknown state was recognised. The currently processed data shall be processed again. An error shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

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

Info	Corrupting stp state and processing data.

```
Sending ':<te' to stp.
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->
↪  STP_STATE_STORE_DATA
Setting state of stp to 255.
Sending ':<te' to stp.
STP: unknown state (255) => adding value (3a) back to data again and changing state ->
↪  STP_STATE_IDLE
STP: Chunking "(2): 74 65" from buffer
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 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 [[]] and Type is <type 'list'>).

```
Result: [ [  ] ] (<type 'list'>)
Expectation: result = [ [  ] ] (<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'>)
```

Success	Buffer size after corrupting stp state is correct (Content 2 and Type is <type 'int'>).

```
Result: 2 (<type 'int'>)
Expectation: result = 2 (<type 'int'>)
```

B Trace for testrun with python 3.6.9 (final)

B.1 Tests with status Info (18)

B.1.1 Hexadecimal Values

Description
A Stream shall be converted to a human readable String containing all bytes as hexadecimal values seperated by a Space.

Reason for the implementation
Make non printable characters printable.

Fitcriterion
A stream shall be converted at least once and the hex values shall exist in the returnvalue in the correct order.

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

Info	Checking test pattern de ad be ef (<class 'bytes'>).

Success	Pattern included all relevant information in the correct order.

```
Return value of hexlify is (4): de ad be ef
Using upper string for comparison: (4): DE AD BE EF
"DE" found in "(4): DE AD BE EF"... Reducing pattern
"AD" found in "AD BE EF"... Reducing pattern
"BE" found in "BE EF"... Reducing pattern
"EF" found in "EF"... Reducing pattern
```

B.1.2 Number of Bytes

Description
The Length of a Stream surrounded by brakets shall be included in the human readable string.

Reason for the implementation
Show the length of a Stream without counting the seperated values.

Fitcriterion
The described pattern including the decimal number of bytes is included in the string for at least one Stream.

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

Info	Checking test pattern with length 4.

Success	'(4)' is in '(4): de ad be ef' at position 0

B.1.3 CRLF-Filter

Description
The module stringtools shall have a method to replace carriage return and line feed to their escaped representation.

Reason for the implementation
Replace these characters to make output printable (e.g. for logging a string based protocol).

Fitcriterion
Filter at least one string and check at least one CR and one LF representation.

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

Info	Checking test pattern with length 4.

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 Compress

Description
The module stringtools shall have a method compressing a Stream with gzip.

Reason for the implementation
Speed up transfer with low transfer rate.

Fitcriterion
Compressed Stream is extractable and results in the original data.

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

Info	Compressing Streams result in differnt streams with the same input stream. Therefore the test will compare the decompressed data.

Info	Compressing stream: (30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

```
GZIP: Finished to compress a string (compression_rate=0.867, consumed_time=0.0s).
```

Info	Extracting stream: (26): 1f 8b 08 00 68 82 03 5e 02 ff 63 60 40 01 ff 51 01 00 2d 8a 7d de 1e 00 00 00

```
GZIP: Finished to extract a string (compression_rate=0.867, consumed_time=0.0s).
```

Success	Extracted data is correct (Content (30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff and Type is <class 'bytes'>).

```
Result: (30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff
↪  ff ff ff ff (<class 'bytes'>)
Expectation: result = (30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff (<class 'bytes'>)
```

B.1.5 Extract

Description
The module stringtools shall have a method extracting a Stream with gzip.

Reason for the implementation
Speed up transfer with low transfer rate.

Fitcriterion
Extracted Stream is equal to the original compressed data.

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

Info	Extracting stream: (26): 1f 8b 08 00 34 e0 04 5d 02 ff 63 60 40 01 ff 51 01 00 2d 8a 7d de 1e 00 00 00

```
GZIP: Finished to extract a string (compression_rate=0.867, consumed_time=0.0s).
```

Success	Extracted data is correct (Content '(30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff' and Type is <class 'str'>).

```
Result: '(30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff
↪  ff ff ff ff' (<class 'str'>)
Expectation: result = '(30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff ff' (<class 'str'>)
```

B.1.6 Frame creation

Description
The CSP module shall support a method to create a Frame from a stream.

Reason for the implementation
Simple message or frame generation for streams (e.g. Keyboard (user input), RFID-Reader, . . .).

Fitcriterion
Creation of a testframe and checking the result.

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

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

Success	CSP-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.1.7 Frame creation error

Description
The Frame creation Method shall raise ValueError, if a frame separation character is in the Source-String.

Reason for the implementation
String including separation charcter will be splitted in pieces while processing after transport.

Fitcriterion
ValueErroro is raised for at least one String including the separation character.

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

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

Success	CSP-Frame is correct (Content <class 'ValueError'> and Type is <class 'type'>).

```
Result: <class 'ValueError'> (<class 'type'>)
Expectation: result = <class 'ValueError'> (<class 'type'>)
```

B.1.8 Frame processing

Description
The CSP Module shall support a class including a method to process stream snipets of variable length. This Method shall return an empty list, if no frame has been detected, otherwise it shall return a list including detected frame(s).

Reason for the implementation
Support message analysis of a stream with every size.

Fitcriterion
At least one frame given in at least two snippets is identified correctly.

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

Info	Processing testframe: 'b':testframe: for csp/n" in two snippets

```
CSP: Leaving data in buffer (to be processed next time): (10): 3a 74 65 73 74 66 72 61 6d 65
CSP: 1 messages identified
```

Success	First processed CSP-Snippet is correct (Content [] and Type is <class 'list'>).

```
Result: [  ] (<class 'list'>)
Expectation: result = [  ] (<class 'list'>)
```

Success	Final processed CSP-Frame is correct (Content [b':testframe: for csp'] and Type is <class 'list'>).

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

B.1.9 Frame processing - Input data type error

Description
If the input data is not bytes for python3 or str for python 2, the process method shall raise TypeError.

Reason for the implementation
Type restriction.

Fitcriterion
At least the following types return TypeError (list, int, str for python3, unicode for python 2).

Testresult

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

Info	Processing wrong data (list)

Success	Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).

```
Result: <class 'ValueError'> (<class 'type'>)
Expectation: result = <class 'ValueError'> (<class 'type'>)
```

Success	Buffer still empty is correct (Content b'' and Type is <class 'bytes'>).

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

Info	Processing wrong data (int)

Success	Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).

```
Result: <class 'ValueError'> (<class 'type'>)
Expectation: result = <class 'ValueError'> (<class 'type'>)
```

Success	Buffer still empty is correct (Content b'' and Type is <class 'bytes'>).

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

Info	Processing wrong data (str)

Success	Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).

```
Result: <class 'ValueError'> (<class 'type'>)
Expectation: result = <class 'ValueError'> (<class 'type'>)
```

Success	Buffer still empty is correct (Content b'' and Type is <class 'bytes'>).

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

B.1.10 Frame creation

Description

A frame creation method shall create a frame out of given input data.

Reason for the implementation
Message or Frame generation for streams (e.g. data transfer via bluetooth, ethernet, ...).

Fitcriterion
Creation of a testframe and checking the result.

Testresult
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.1.11 Frame creation - Start pattern and end pattern inside a message

Description
The frame creation method shall support existance of the start or end pattern in the data to be framed.

Reason for the implementation
Possibility to send any kind of data (including the patterns).

Fitcriterion
Creation of a testframe out of data including at least one start pattern and one end pattern and checking the result.

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

Info	Creating testframe including start and end pattern 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.1.12 Frame processing

Description
The STP Module shall support a class including a method to process stream snipets of variable length. This Method shall return an empty list, if no frame has been detected, otherwise it shall return a list including detected frame(s).

Reason for the implementation

Support message analysis of a stream with every size.

Fitcriterion

At least one frame given in at least two snippets is identified correctly.

Testresult

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: start pattern (3a 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: end pattern (3a 3e) received => storing message and changing state STP_STATE_ESCAPE_2 ->
↪  STP_STATE_IDLE
```

Success	First processed STP snippet is correct (Content [] and Type is <class 'list'>).

```
Result: [  ] (<class 'list'>)
Expectation: result = [  ] (<class 'list'>)
```

Success	Final processed STP snippet is correct (Content [b'testframe for stp'] and Type is <class 'list'>).

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

B.1.13 Frame processing - Input data type error

Description

If the input data is not bytes for python3 or str for python 2, the process method shall raise TypeError.

Reason for the implementation

Type restriction.

Fitcriterion

At least the following types return TypeError (list, int, str for python3, unicode for python 2).

Testresult

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

Info	Processing wrong data (list)

Success	Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).

```
Result: <class 'ValueError'> (<class 'type'>)
Expectation: result = <class 'ValueError'> (<class 'type'>)
```

Success	Buffer still empty is correct (Content b'' and Type is <class 'bytes'>).

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

Info	Processing wrong data (int)

Success	Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).

```
Result: <class 'ValueError'> (<class 'type'>)
Expectation: result = <class 'ValueError'> (<class 'type'>)
```

Success	Buffer still empty is correct (Content b'' and Type is <class 'bytes'>).

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

Info	Processing wrong data (str)

Success	Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).

```
Result: <class 'ValueError'> (<class 'type'>)
Expectation: result = <class 'ValueError'> (<class 'type'>)
```

Success	Buffer still empty is correct (Content b'' and Type is <class 'bytes'>).

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

B.1.14 Frame processing - Start pattern and end pattern inside a message

Reason for the implementation

Possibility to send any kind of data (including the patterns).

Testresult

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: start pattern (3a 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 pattern (3a 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: store sync pattern (3a 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: end pattern (3a 3e) received => storing message and changing state STP_STATE_ESCAPE_2 ->
↪  STP_STATE_IDLE
```

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

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

B.1.15 Frame processing - Data before the start pattern

Description

Data before the start pattern shall be ignored. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

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

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

```
STP: no data sync (5f) received => ignoring byte
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 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: end pattern (3a 3e) received => storing message and changing state STP_STATE_ESCAPE_2 ->
↪  STP_STATE_IDLE
```

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

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

B.1.16 Frame processing - Incorrect start patterns

Description

On receiving an incorrect start pattern, STP shall stay in ESCAPE_1, in case of data sync was received twice or back to state IDLE in all other faulty start patterns starting with data sync. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

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

Info	Processing data with an insufficient start pattern.

```
Sending b':1' to stp.
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: no start pattern (3a 31) received => changing state STP_STATE_ESCAPE_1 -> STP_STATE_IDLE
```

Success	Return value list if processing incorrect start of frame is correct (Content [[]] and Type is <class 'list'>).

```
Result: [ [  ] ] (<class 'list'>)
Expectation: result = [ [  ] ] (<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 insufficient start pattern (two times sync).

```
Sending b'::' to stp.
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 [[]] and Type is <class 'list'>).

```
Result: [ [  ] ] (<class 'list'>)
Expectation: result = [ [  ] ] (<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.1.17 Frame processing - Incorrect end pattern

Description

On receiving an incorrect end pattern, STP shall change to state STORE_DATA, in case of a start pattern, to ESCAPE_1, in case of data sync was received twice or back to state IDLE in all other faulty end patterns starting with data sync. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

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

Info	Processing data with an insufficient end pattern.

```
Sending b':<te:d' to stp.
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 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
STP: Chunking "(2): 74 65" from buffer
```

Success	Return value list if processing data_sync and data again after start of frame is correct (Content [[]] and Type is <class 'list'>).

```
Result: [ [  ] ] (<class 'list'>)
Expectation: result = [ [  ] ] (<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'>)
```

Success	Buffer size after processing data with insufficient end pattern is correct (Content 0 and Type is <class 'int'>).

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

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

```
Sending b':<te:<' to stp.
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 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: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_2 ->
↪  STP_STATE_STORE_DATA
STP: Chunking "(2): 74 65" from buffer
```

Success Return value list if processing 2nd start of frame is correct (Content [[]] and Type is <class 'list'>).

```
Result: [ [  ] ] (<class 'list'>)
Expectation: result = [ [  ] ] (<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 insufficient end pattern (two times sync instead of end pattern).

```
Sending b':<te::' to stp.
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 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: second data sync (3a) received => changing state STP_STATE_ESCAPE_2 -> STP_STATE_ESCAPE_1
STP: Chunking "(2): 74 65" from buffer
```

Success Return value list if processing data_sync twice after start of frame is correct (Content [[]] and Type is <class 'list'>).

```
Result: [ [  ] ] (<class 'list'>)
Expectation: result = [ [  ] ] (<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.1.18 Frame processing - After state corruption

Description

The state of STP shall be set to IDLE, after an unknown state was recognised. The currently processed data shall be processed again. An error shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

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

Info	Corrupting stp state and processing data.

```
Sending b':<te' to stp.
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->
↪  STP_STATE_STORE_DATA
Setting state of stp to 255.
Sending b':<te' to stp.
STP: unknown state (255) => adding value (3a) back to data again and changing state ->
↪  STP_STATE_IDLE
STP: Chunking "(2): 74 65" from buffer
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 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 [[]] and Type is <class 'list'>).

```
Result: [ [  ] ] (<class 'list'>)
Expectation: result = [ [  ] ] (<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'>)
```

Success	Buffer size after corrupting stp state is correct (Content 2 and Type is <class 'int'>).

```
Result: 2 (<class 'int'>)
Expectation: result = 2 (<class 'int'>)
```

C Test-Coverage

C.1 stringtools

The line coverage for stringtools was 100.0%
The branch coverage for stringtools was 97.1%

C.1.1 stringtools.__init__.py

The line coverage for stringtools.__init__.py was 100.0%
The branch coverage for stringtools.__init__.py was 97.1%

```
#!/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.

**Module Documentation:**

"""

from stringtools import stp
from stringtools import csp
__DEPENDENCIES__ = []

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

logger_name = 'STRINGTOOLS'
logger = logging.getLogger(logger_name)

```

```
__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):
    """
    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)
    :return: The compressed bytestream
    :rtype: str

    **Example:**

    .. literalinclude:: ../examples/gzip_compress.py

    Will result to the following output:

    .. literalinclude:: ../examples/gzip_compress.log
    """
    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 rv is not None:
        logger.debug('GZIP: 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):
    """
    Method to extract data from a compress stream of bytes.

    :param str s: The compressed bytestream (string) to be extracted
    :return: The extracted data
    :rtype: str

    **Example:**

    .. literalinclude:: ../examples/gzip_extract.py

    Will result to the following output:

    .. literalinclude:: ../examples/gzip_extract.log
    """
```

```
110     t = time.time()
111     rv = None
112     if sys.version_info >= (3, 0):
113         rv = gzip.decompress(s)
114     else:
115         inbuffer = StringIO(s)
116         f = gzip.GzipFile(mode='rb', fileobj=inbuffer)
117         try:
118             rv = f.read()
119         finally:
120             f.close()
121             inbuffer.close()
122     if rv is not None:
123         logger.debug('GZIP: Finished to extract a string (compression_rate=%.3f, consumed_time
    =%.1fs).', len(s) / float(len(rv)), time.time() - t)
124     return rv
125
126
127 def hexlify(s):
128     """Method to hexlify a string.
129
130     :param str s: A string including the bytes to be hexlified.
131     :returns: The hexlified string
132     :rtype: str
133
134     **Example:**
135
136     .. literalinclude:: ../examples/hexlify.py
137
138     Will result to the following output:
139
140     .. literalinclude:: ../examples/hexlify.log
141     """
142     rv = '(%d):' % len(s)
143     for byte in s:
144         if sys.version_info >= (3, 0):
145             rv += ' %02x' % byte
146         else:
147             rv += ' %02x' % ord(byte)
148     return rv
149
150
151 def linefeed_filter(s):
152     """Method to change linefeed and carriage return to '\\\\n' and '\\\\r'
153
154     :param str s: A string including carriage return and/ or linefeed.
155     :returns: A string with converted carriage return and/ or linefeed.
156     :rtype: str
157     """
158     if sys.version_info >= (3, 0):
159         return s.replace(b'\r', b'\\r').replace(b'\n', b'\\n')
160     else:
161         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 97.1%

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
"""
csp (Carriage-Return seperation protocol)
=========================================

**Author:**

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

**Description:**

    This module is a submodule of :mod:`stringtools` and creates an frame to transmit and receive messages via an serial interface.

**Submodules:**

* :class:`stringtools.csp.csp`
* :func:`stringtools.csp.build_frame`
"""

import stringtools

import logging
import sys

logger_name = 'STRINGTOOLS'
logger = logging.getLogger(logger_name)

DATA_SEPERATOR = b'\n'


class csp(object):
    """This class extracts messages from an "csp-stream".

    **Example:**

    .. literalinclude:: ../examples/csp.csp.py

    Will result to the following output:

    .. literalinclude:: ../examples/csp.csp.log
    """
    LOG_PREFIX = 'CSP:'

    def __init__(self, seperator=DATA_SEPERATOR):
        self.__buffer__ = b''
        self.__seperator__ = seperator

    def process(self, data):
        """
        This processes a byte out of a "stp-stream".

        :param bytes data: A byte stream
        :returns: A list of the extracted message(s)
        :rtype: list
        """
```

```
58          if sys.version_info <= (3, 0):
59              if type(data) is unicode:
60                  raise TypeError
61          #
62          rv = (self.__buffer__ + data).split(self.__seperator__)
63          self.__buffer__ = rv.pop()
64          if len(self.__buffer__) != 0:
65              logger.debug('%s Leaving data in buffer (to be processed next time): %s', self.
    LOG_PREFIX, stringtools.hexlify(self.__buffer__))
66          if len(rv) != 0:
67              logger.debug('%s %d messages identified', self.LOG_PREFIX, len(rv))
68          return rv
69
70
71  def build_frame(msg, seperator=DATA_SEPERATOR):
72      """This Method builds an "csp-frame" to be transfered via a stream.
73
74      :param str data: A String (Bytes) to be framed
75      :returns: The "csp-framed" message to be sent
76      :rtype: str
77
78      **Example:**
79
80      .. literalinclude:: ../examples/csp.build_frame.py
81
82      Will result to the following output:
83
84      .. literalinclude:: ../examples/csp.build_frame.log
85      """
86      if seperator in msg:
87          raise ValueError
88      else:
89          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 97.1%

```
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
```

```
22 import stringtools
23
24 import logging
25 import sys
26
27 logger_name = 'STRINGTOOLS'
28 logger = logging.getLogger(logger_name)
29
30 DATA_SYNC = b'\x3a'
31 """The data sync byte"""
32 DATA_CLEAR_BUFFER = b'\x3c'
33 """The clear buffer byte ('\\\\x3a\\\\x3c' -> start of message)"""
34 DATA_VALID_MSG = b'\x3e'
35 """The valid message byte ('\\\\x3a\\\\x3e' -> end of message)"""
36 DATA_STORE_SYNC_VALUE = b'\x3d'
37 """The store sync value byte ('\\\\x3a\\\\x3d' -> '\\\\x3a' inside a message)"""
38
39 STP_STATE_IDLE = 0x00
40 """Idle state definition (default)"""
41 STP_STATE_ESCAPE_1 = 0x01
42 """Escape 1 state definition ('\\\\x3a\\\\x3c' found)"""
43 STP_STATE_ESCAPE_2 = 0x02
44 """Escape 2 state definition ('\\\\x3a' found inside a message)"""
45 STP_STATE_STORE_DATA = 0x03
46 """Store data state definition (start of message found; data will be stored)"""
47
48
49 class stp(object):
50     """This class extracts messages from an "stp-stream".
51
52     **Example:**
53
54     .. literalinclude:: ../examples/stp.stp.py
55
56     Will result to the following output:
57
58     .. literalinclude:: ../examples/stp.stp.log
59     """
60     LOG_PREFIX = 'STP:'
61
62     def __init__(self):
63         self.state = STP_STATE_IDLE
64         self.__buffer__ = b''
65         self.__clear_buffer__()
66
67     def __clear_buffer__(self):
68         if len(self.__buffer__) > 0:
69             logger.warning('%s Chunking "%s" from buffer', self.LOG_PREFIX, stringtools.hexlify(
    self.__buffer__))
70         self.__buffer__ = b''
71
72     def process(self, data):
73         """
74         This processes a byte out of a "stp-stream".
75
76         :param bytes data: A byte stream
77         :returns: The extracted message or None, if no message is identified yet
78         :rtype: str
79         """
80         if type(data) is list:
81             raise TypeError
82         if sys.version_info <= (3, 0):
83             if type(data) is unicode:
84                 raise TypeError
```

```
85          #
86          rv = []
87          #
88          while len(data) > 0:
89              if sys.version_info >= (3, 0):
90                  b = bytes([data[0]])
91              else:
92                  b = data[0]
93              data = data[1:]
94              #
95              if self.state == STP_STATE_IDLE:
96                  if b == DATA_SYNC:
97                      self.state = STP_STATE_ESCAPE_1
98                      logger.debug('%s data sync (%02x) received => changing state STP_STATE_IDLE
    -> STP_STATE_ESCAPE_1', self.LOG_PREFIX, ord(b))
99                  else:
100                     logger.warning('%s no data sync (%02x) received => ignoring byte', self.
    LOG_PREFIX, ord(b))
101             elif self.state == STP_STATE_ESCAPE_1:
102                 if b == DATA_CLEAR_BUFFER:
103                     logger.debug('%s start pattern (%02x %02x) received => changing state
    STP_STATE_ESCAPE_1 -> STP_STATE_STORE_DATA', self.LOG_PREFIX, ord(DATA_SYNC), ord(b))
104                     self.state = STP_STATE_STORE_DATA
105                     self.__clear_buffer__()
106                 elif b != DATA_SYNC:
107                     self.state = STP_STATE_IDLE
108                     logger.warning('%s no start pattern (%02x %02x) received => changing state
    STP_STATE_ESCAPE_1 -> STP_STATE_IDLE', self.LOG_PREFIX, ord(DATA_SYNC), ord(b))
109                 else:
110                     logger.warning('%s 2nd data sync (%02x) received => keep state', self.
    LOG_PREFIX, ord(b))
111             elif self.state == STP_STATE_STORE_DATA:
112                 if b == DATA_SYNC:
113                     self.state = STP_STATE_ESCAPE_2
114                     logger.debug('%s data sync (%02x) received => changing state
    STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2', self.LOG_PREFIX, ord(b))
115                 else:
116                     self.__buffer__ += b
117             elif self.state == STP_STATE_ESCAPE_2:
118                 if b == DATA_CLEAR_BUFFER:
119                     logger.warning('%s start pattern (%02x %02x) received => changing state
    STP_STATE_ESCAPE_2 -> STP_STATE_STORE_DATA', self.LOG_PREFIX, ord(DATA_SYNC), ord(b))
120                     self.state = STP_STATE_STORE_DATA
121                     self.__clear_buffer__()
122                 elif b == DATA_VALID_MSG:
123                     self.state = STP_STATE_IDLE
124                     logger.debug('%s end pattern (%02x %02x) received => storing message and
    changing state STP_STATE_ESCAPE_2 -> STP_STATE_IDLE', self.LOG_PREFIX, ord(DATA_SYNC), ord(b)
    )
125                     rv.append(self.__buffer__)
126                     self.__buffer__ = b''
127                 elif b == DATA_STORE_SYNC_VALUE:
128                     self.state = STP_STATE_STORE_DATA
129                     logger.debug('%s store sync pattern (%02x %02x) received => changing state
    STP_STATE_ESCAPE_2 -> STP_STATE_STORE_DATA', self.LOG_PREFIX, ord(DATA_SYNC), ord(b))
130                     self.__buffer__ += DATA_SYNC
131                 elif b == DATA_SYNC:
132                     self.state = STP_STATE_ESCAPE_1
133                     logger.warning('%s second data sync (%02x) received => changing state
    STP_STATE_ESCAPE_2 -> STP_STATE_ESCAPE_1', self.LOG_PREFIX, ord(b))
134                     self.__clear_buffer__()
135                 else:
```

```
136                     self.state = STP_STATE_IDLE
137                     logger.warning('%s data (%02x) received => changing state STP_STATE_ESCAPE_2
    -> STP_STATE_IDLE', self.LOG_PREFIX, ord(b))
138                     self.__clear_buffer__()
139             else:
140                 logger.error('%s unknown state (%s) => adding value (%02x) back to data again and
     changing state -> STP_STATE_IDLE', self.LOG_PREFIX, repr(self.state), ord(b))
141                 self.state = STP_STATE_IDLE
142                 self.__clear_buffer__()
143                 data = b + data
144         return rv
145
146
147 def build_frame(data):
148     """This Method builds an "stp-frame" to be transfered via a stream.
149
150     :param str data: A String (Bytes) to be framed
151     :returns: The "stp-framed" message to be sent
152     :rtype: str
153
154     **Example:**
155
156     .. literalinclude:: ../examples/stp.build_frame.py
157
158     Will result to the following output:
159
160     .. literalinclude:: ../examples/stp.build_frame.log
161     """
162     rv = DATA_SYNC + DATA_CLEAR_BUFFER
163
164     for byte in data:
165         if sys.version_info >= (3, 0):
166             byte = bytes([byte])
167         if byte == DATA_SYNC:
168             rv += DATA_SYNC + DATA_STORE_SYNC_VALUE
169         else:
170             rv += byte
171
172     rv += DATA_SYNC + DATA_VALID_MSG
173     return rv
```