textparser (version 1.4.0)

index
c:\dev\04_pydev\csutils3\csutils\textparser.py

```
################################################################################
# Textparser class eases basic operations like search or data extraction on textfiles.
#
# @package: csutils.textparser
# @author:  cwsoft
# @python:  3.6 or higher
################################################################################
```

Modules

os re

Classes

builtins.object
    Textparser

class **Textparser**(builtins.object)

```
Textparser(source)

Class to perform basic operations like search and data extraction on textfiles.
```

Methods defined here:

__init__(self, source)
```
Initalize Textparser object with data from textfile path or from input string.
```

__repr__(self)
```
Output string representation of the textparser object.
```

from_source(self, source)
```
Read all textlines from specified source into memory and store data in _lines.
Source can be a valid textfile path or an input string.
```

get_lines(self, rows=':', merge='\n', end='\n')
```
Return all textlines matching given row indices joined by merge char.
Row indices can be a slice or comma separated string, or a container with indices.
```

get_match(self, pattern, subpatterns=None, ignoreCase=True)
```
Return tuple with row index and textline of the first row, matching the given pattern.
To narrow possible matches, one can specify as many optional subpatterns as needed. Subpatterns are
evaluated relative to the line matching the main pattern using the specified rowOffset. Subpatterns
are defined as follows: subpatterns = [(rowOffset1, subPattern1),..., (rowOffsetN, subPatternN)].

Note: Patterns starting with 'rx:' will perform a regular expression search on the source lines.
Set ignoreCase=False to perform a case sensitive search on all specified search patterns.
Set findAll=False to return a tuple with row index and textline of the first matching result only.
```

get_matches(self, pattern, subpatterns=None, ignoreCase=True, findAll=True)
```
Return list of tuples with row index and textline for all rows, matching given main pattern.
To narrow possible matches, one can specify as many optional subpatterns as needed. Subpatterns are
evaluated relative to the line matching the main pattern using the specified rowOffset. Subpatterns
are defined as follows: subpatterns = [(rowOffset1, subPattern1),..., (rowOffsetN, subPatternN)].

Note: Patterns starting with 'rx:' will perform a regular expression search on the source lines.
Set ignoreCase=False to perform a case sensitive search on all specified search patterns.
Set findAll=False to return a tuple with row index and textline of the first matching result only.
```

get_numbered_source_lines(self, output=False, nbrFormat='5d')
```
Return source lines prepend by their corresponding row indices.
Set output=True to dump the result to the console stdout.
```

get_values(self, rows, cols=':', sep=None, merge=' ', end='\n')
```
Return all values matching the given row and column indices.
Row and col indices can be a slice or comma separated string, or a container with indices.
By default, column values are joined with merge char, rows are joined with end char.
Set merge="," and end=";" to join column values by comma and lines by semicolon.
```

Static methods defined here:

write(path, lines, append=True)
```
Write or append input lines to textfile defined by the path string.
```

Readonly properties defined here:

lines

```
Return number of textlines from input source.
```

source

```
Return source of the imported data as string.
```

Data descriptors defined here:

__dict__

```
dictionary for instance variables (if defined)
```

__weakref__

```
list of weak references to the object (if defined)
```