
TextWrap.jl Documentation

Release 0.1

Carlo Baldassi

December 18, 2016

Python Module Index	3
Python Module Index	5

This module provides the function `wrap` which parses an input text and reorganizes its white space so that it can be printed with a fixed screen width, optionally indenting it. It also provides the two convenience functions `print_wrapped` and `println_wrapped`.

Here is a quick example:

```
julia> using TextWrap

julia> text = "This text is going to be wrapped around in lines no longer than 20 characters.";

julia> println_wrapped(text, width=20)
This text is going
to be wrapped around
in lines no longer
than 20 characters.
```

It's very similar to Python's `textwrap` module, but the interface is slightly different.

wrap(*string*, *width::Integer* = 70, *initial_indent::Union{String, Integer}* = "", *subsequent_indent::Union{String, Integer}* = "", *break_on_hyphens::Bool* = true, *break_long_words::Bool* = true, *replace_whitespace::Bool* = true, *expand_tabs::Bool* = true, *fix_sentence_endings::Bool* = false)

Parses *string* and returns a new string in which newlines are inserted as appropriate in order for each line to fit within a specified width.

The behaviour can be controlled via optional keyword arguments, whose meaning is:

- **width**: the maximum width of the wrapped text, including indentation.
- **initial_indent**: indentation of the first line. This can be any string (shorter than *width*), or it can be an integer number (lower than *width*).
- **subsequent_indent**: indentation of all lines except the first. Works the same as *initial_indent*.
- **break_on_hyphens**: this flag determines whether words can be broken on hyphens, e.g. whether “high-precision” can be split into “high-” and “precision”.
- **break_long_words**: this flag determines what to do when a word is too long to fit in any line. If true, the word will be broken, otherwise it will go beyond the desired text width.
- **replace_whitespace**: if this flag is true, all whitespace characters in the original text (including newlines) will be replaced by spaces.
- **expand_tabs**: if this flag is true, tabs will be expanded in-place into spaces. The expansion happens before whitespace replacement.
- **fix_sentence_endings**: if this flag is true, the wrapper will try to recognize sentence endings in the middle of a paragraph and put two spaces before the next sentence in case only one is present.

print_wrapped(*text...*; *options...*)

print_wrapped(*io*, *text...*; *options...*)

println_wrapped(*text...*; *options...*)

println_wrapped(*io*, *text...*; [*options*])

These are just like the standard `Base.print()` and `Base.println()` functions (they print multiple arguments and accept an optional `IO` first argument), except that they wrap the result, and accept keyword arguments with the options to pass to `wrap()`.

t

TextWrap, ??

t

TextWrap, ??

P

`print_wrapped()` (in module `TextWrap`), [1](#)
`println_wrapped()` (in module `TextWrap`), [1](#)

T

`TextWrap` (module), [1](#)

W

`wrap()` (in module `TextWrap`), [1](#)