

---

# **TextWrap.jl Documentation**

***Release 0.1***

**Carlo Baldassi**

June 17, 2013



# CONTENTS



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()`.



# PYTHON MODULE INDEX

t

TextWrap, ??





# PYTHON MODULE INDEX

t

TextWrap, ??