textwrap — Formatting Text Paragraphs

Purpose: Formatting text by adjusting where line breaks occur in a paragraph.

The `textwrap` module can be used to format text for output in situations where pretty-printing is desired. It offers programmatic functionality similar to the paragraph wrapping or filling features found in many text editors and word processors.

Example Data

The examples in this section use the module `textwrap_example.py`, which contains a string `sample_text`.

```
# textwrap_example.py

sample_text = '''
    The textwrap module can be used to format text for output in
    situations where pretty-printing is desired.  It offers
    programmatic functionality similar to the paragraph wrapping
    or filling features found in many text editors.
    '''
```

Filling Paragraphs

The `fill()` function takes text as input and produces formatted text as output.

```
# textwrap_fill.py

import textwrap
from textwrap_example import sample_text

print(textwrap.fill(sample_text, width=50))
```

The results are something less than desirable. The text is now left justified, but the first line retains its indent and the spaces from the front of each subsequent line are embedded in the paragraph.

```
$ python3 textwrap_fill.py

    The textwrap module can be used to format
text for output in     situations where pretty-
printing is desired.  It offers     programmatic
functionality similar to the paragraph wrapping
or filling features found in many text editors.
```

Removing Existing Indentation

The previous example has embedded tabs and extra spaces mixed into the middle of the output, so it is not formatted very cleanly. Removing the common whitespace prefix from all of the lines in the sample text with `dedent()` produces better results and allows the use of docstrings or embedded multiline strings straight from Python code while removing the formatting of the code itself. The sample string has an artificial indent level introduced for illustrating this feature.

```
# textwrap_dedent.py

import textwrap
from textwrap_example import sample_text

dedented_text = textwrap.dedent(sample_text)
print('Dedented:')
print(dedented_text)
```

The results are starting to look better.

```
$ python3 textwrap_dedent.py

Dedented:

The textwrap module can be used to format text for output in
situations where pretty-printing is desired.  It offers
programmatic functionality similar to the paragraph wrapping
or filling features found in many text editors.
```

Since “dedent” is the opposite of “indent,” the result is a block of text with the common initial whitespace from each line removed. If one line is already indented more than another, some of the whitespace will not be removed.

Input like

```
␣Line one.
␣␣␣␣Line two.
␣Line three.
```

becomes

```
Line one.
␣␣␣␣Line two.
Line three.
```

Combining Dedent and Fill

Next, the dedented text can be passed through `fill()` with a few different `width` values.

```
# textwrap_fill_width.py

import textwrap
from textwrap_example import sample_text

dedented_text = textwrap.dedent(sample_text).strip()
for width in [45, 60]:
    print('{} Columns:\n'.format(width))
    print(textwrap.fill(dedented_text, width=width))
    print()
```

This produces outputs in the specified widths.

```
$ python3 textwrap_fill_width.py

45 Columns:

The textwrap module can be used to format
text for output in situations where pretty-
printing is desired.  It offers programmatic
functionality similar to the paragraph
wrapping or filling features found in many
text editors.

60 Columns:

The textwrap module can be used to format text for output in
situations where pretty-printing is desired.  It offers
programmatic functionality similar to the paragraph wrapping
or filling features found in many text editors.
```

Indenting Blocks

Use the `indent()` function to add consistent prefix text to all of the lines in a string. This example formats the same example text as though it was part of an email message being quoted in the reply, using > as the prefix for each line.

```
# textwrap_indent.py
```

```
import textwrap
from textwrap_example import sample_text

dedented_text = textwrap.dedent(sample_text)
wrapped = textwrap.fill(dedented_text, width=50)
wrapped += '\n\nSecond paragraph after a blank line.'
final = textwrap.indent(wrapped, '> ')

print('Quoted block:\n')
print(final)
```

The block of text is split on newlines, the prefix is added to each line that contains text, and then the lines are combined back into a new string and returned.

```
$ python3 textwrap_indent.py

Quoted block:

>  The textwrap module can be used to format text
> for output in situations where pretty-printing is
> desired.  It offers programmatic functionality
> similar to the paragraph wrapping or filling
> features found in many text editors.

> Second paragraph after a blank line.
```

To control which lines receive the new prefix, pass a callable as the `predicate` argument to `indent()`. The callable will be invoked for each line of text in turn and the prefix will be added for lines where the return value is true.

```
# textwrap_indent_predicate.py

import textwrap
from textwrap_example import sample_text


def should_indent(line):
    print('Indent {!r}?'.format(line))
    return len(line.strip()) % 2 == 0


dedented_text = textwrap.dedent(sample_text)
wrapped = textwrap.fill(dedented_text, width=50)
final = textwrap.indent(wrapped, 'EVEN ',
                        predicate=should_indent)

print('\nQuoted block:\n')
print(final)
```

This example adds the prefix `EVEN` to lines that contain an even number of characters.

```
$ python3 textwrap_indent_predicate.py

Indent ' The textwrap module can be used to format text\n'?
Indent 'for output in situations where pretty-printing is\n'?
Indent 'desired.  It offers programmatic functionality\n'?
Indent 'similar to the paragraph wrapping or filling\n'?
Indent 'features found in many text editors.'?

Quoted block:

EVEN  The textwrap module can be used to format text
for output in situations where pretty-printing is
EVEN desired.  It offers programmatic functionality
EVEN similar to the paragraph wrapping or filling
EVEN features found in many text editors.
```

Hanging Indents

In the same way that it is possible to set the width of the output, the indent of the first line can be controlled independently of

In the same way that it is possible to set the width of the output, the indent of the first line can be controlled independently of subsequent lines.

```
# textwrap_hanging_indent.py

import textwrap
from textwrap_example import sample_text

dedented_text = textwrap.dedent(sample_text).strip()
print(textwrap.fill(dedented_text,
                    initial_indent='',
                    subsequent_indent=' ' * 4,
                    width=50,
                    ))
```

This makes it possible to produce a hanging indent, where the first line is indented less than the other lines.

```
$ python3 textwrap_hanging_indent.py

The textwrap module can be used to format text for
    output in situations where pretty-printing is
    desired.  It offers programmatic functionality
    similar to the paragraph wrapping or filling
    features found in many text editors.
```

The indent values can include nonwhitespace characters, too. The hanging indent can be prefixed with * to produce bullet points, for example.

Truncating Long Text

To truncate text to create a summary or preview, use `shorten()`. All existing whitespace, such as tabs, newlines, and series of multiple spaces, will be standardized to a single space. Then the text will be truncated to a length less than or equal to what is requested, between word boundaries so that no partial words are included.

```
# textwrap_shorten.py

import textwrap
from textwrap_example import sample_text

dedented_text = textwrap.dedent(sample_text)
original = textwrap.fill(dedented_text, width=50)

print('Original:\n')
print(original)

shortened = textwrap.shorten(original, 100)
shortened_wrapped = textwrap.fill(shortened, width=50)

print('\nShortened:\n')
print(shortened_wrapped)
```

If non-whitespace text is removed from the original text as part of the truncation, it is replaced with a placeholder value. The default value `[...]` can be replaced by providing a `placeholder` argument to `shorten()`.

```
$ python3 textwrap_shorten.py

Original:

 The textwrap module can be used to format text
for output in situations where pretty-printing is
desired.  It offers programmatic functionality
similar to the paragraph wrapping or filling
features found in many text editors.

Shortened:

The textwrap module can be used to format text for
output in situations where pretty-printing [...]
```

See also

- Standard library documentation for textwrap

string — Text Constants and Templates

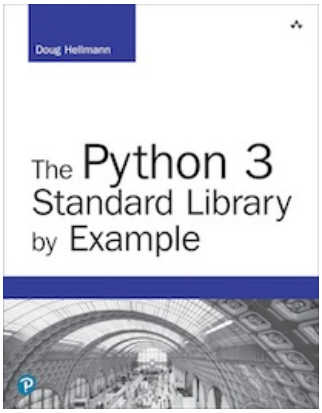
re — Regular Expressions

Quick Links

This page was last updated 2017-01-28.

Navigation

string — Text Constants and Templates
re — Regular Expressions



Get the book

The output from all the example programs from PyMOTW-3 has been generated with Python 3.7.1, unless otherwise noted. Some of the features described here may not be available in earlier versions of Python.

Looking for examples for Python 2?

© Copyright 2019, Doug Hellmann