
simpletex Documentation

Release v0.2.1

Samuel Li

Aug 06, 2018

Contents

1	Getting Started	3
2	API Documentation	11
	Python Module Index	17

simpletex is a Python library for automatically generating LaTeX documents. It is extremely easy to use:

```
from simpletex import write, save
from simpletex.document import Document, Section, Subsection

with Document(size='11pt'):
    with Section('Section Name'):
        with Subsection('Subsection Name'):
            write('Hello World!')

save('filename.tex')
```

For a quick introduction to using simpletex, please refer to the *Tutorial*. For an overview of simpletex's functionality, please see the *Examples*.

1.1 Installation Instructions

The easiest way to install `simpletex` is through the Python Package Index (PyPI). This can be done through the following command:

```
pip3 install simpletex
```

1.2 Tutorial

This tutorial covers the fundamentals, basic usage, advanced features, and third-party support of `simpletex`. For further usage examples, please see the *Examples*.

1.2.1 Fundamentals

The core of `simpletex` is the `Formatter` class; nearly all `simpletex` classes derive from this base class. Formatters apply formatting to (mostly) text. For example, the `Italics` formatter italicizes text. Formatters can be used by calling a `Formatter` instance.

```
>>> from simpletex.formatting.text import Italics
>>> print(Italics() ('Hello World!'))
\textit{Hello World!}
```

`Formatter` instances can also be used as context managers. Upon exit of the `with` block, all written text is passed to the formatter.

```
>>> from simpletex import write, dump
>>> with Italics():
    write('Hello')
    write('World')
```

(continues on next page)

(continued from previous page)

```
>>> dump()
'\\textit{Hello\\nWorld}'
```

1.2.2 Basic Usage

Creating Our First Document

```
from simpletex import write, save
from simpletex.document import Document

with Document():
    write('Hello World!')

save('hello_world.tex')
```

That wasn't so hard, was it?

Sections and Formatting

Let's create a document with two sections:

```
from simpletex import write, save
from simpletex.document import Document, Section

with Document():
    with Section('A'):
        write('First Section Text')
    with Section('B'):
        write('Second Section Text')

save('section_basics.tex')
```

Easy enough. Let's italicize all the section headings:

```
from simpletex import write, save
from simpletex.document import Document, Section
from simpletex.formatting.text import Italics

Section.heading.apply(Italics())

with Document():
    with Section('A'):
        write('First Section Text')
    with Section('B'):
        write('Second Section Text')

save('section_basics.tex')
```

Pretty good for two additional lines.

In the previous example, `Section.heading` is a `Style: a Formatter` which allows many other `Formatters` to be conveniently combined.

1.3 Examples

1.3.1 Basic Usage

Python Code

```
from simpletex import write, save
from simpletex.document import Document, Section, Subsection

with Document(size='11pt'):
    with Section('Section Name'):
        with Subsection('Subsection Name'):
            write('Hello World!')

save('filename.tex')
```

TeX Output

```
\documentclass[11pt]{article}

\usepackage[utf8]{inputenc}

\begin{document}
  \section{Section Name}
  \subsection{Subsection Name}
  Hello World!
\end{document}
```

PDF Output

1 Section Name

1.1 Subsection Name

Hello World!

1.3.2 Simple List and Text Formatting

Python Code

```

from simpletex import write, save
from simpletex.document import Document, Section, Subsection
from simpletex.formatting.text import (Bold, Italics, Underline,
                                       Emphasis, SmallCaps)
from simpletex.sequences import OrderedList, UnorderedList

UnorderedList.bullet = '>'

with Document(size='11pt'):
    with Section('Section Name'):
        with UnorderedList():
            write(Bold() ('Bold Text'))
            write(Italics() ('Italic Text'))
            write(Underline() ('Underlined Text'))
            with Underline():
                write('More Underlined Text')
        with Subsection('Subsection Name'):
            write('The quick brown fox jumps over the lazy dog.')
            with OrderedList():
                write(Emphasis() ('Emphasized Text'))
                write(SmallCaps() ('Small Caps'))

save('filename.tex')

```

TeX Output

```

\documentclass[11pt]{article}

\usepackage[utf8]{inputenc}

\begin{document}
  \section{Section Name}
  \begin{itemize}
    \item[>] \textbf{Bold Text}
    \item[>] \textit{Italic Text}
    \item[>] \underline{Underlined Text}
    \item[>] \underline{More Underlined Text}
  \end{itemize}
  \subsection{Subsection Name}
  The quick brown fox jumps over the lazy dog.
  \begin{enumerate}
    \item \emph{Emphasized Text}
    \item \textsc{Small Caps}
  \end{enumerate}
\end{document}

```

PDF Output

1 Section Name

- > **Bold Text**
- > *Italic Text*
- > Underlined Text
- > More Underlined Text

1.1 Subsection Name

The quick brown fox jumps over the lazy dog.

1. *Emphasized Text*
2. SMALL CAPS

1.3.3 Equations and Math

Python Code

```
from simpletex import write, save
from simpletex.document import Document, Section
from simpletex.math import (Equation,
                             Add, Subtract, Multiply, Divide)

with Document(size='11pt'):
    with Section('Inline Equations'):
        write('Example of the commutative property:')
        with Equation():
            with Multiply(symbol='x'):
                write(3)
                write(5)
```

(continues on next page)

(continued from previous page)

```

        write(Multiply(symbol='times')(5, 3))
        write(15)
    with Section('Display Equations'):
        write('If')
        with Equation():
            write('x')
            write(5)
        write('then:')
        with Equation(inline=False):
            with Divide():
                write(Add()('x', 1))
                write(3)
            write(Subtract()(7, 5))
save('filename.tex')

```

TeX Output

```

\documentclass[11pt]{article}

\usepackage[utf8]{inputenc}

\begin{document}
  \section{Inline Equations}
  Example of the commutative property:
   $3 \times 5 = 5 \times 3 = 15$ 
  \section{Display Equations}
  If
   $x = 5$ 
  then:
  
$$\frac{x+1}{3} = 7-5$$

\end{document}

```

PDF Output

1 Inline Equations

Example of the commutative property: $3 \times 5 = 5 \times 3 = 15$

2 Display Equations

If $x = 5$ then:

$$\frac{x+1}{3} = 7-5$$

1.3.4 XeTeX Support and Fonts

Python Code

```

from simpletex import write, write_break, save, usepackage
from simpletex.document import Document, Section, Subsection
from simpletex.formatting import Style
from simpletex.formatting.font import Font
from simpletex.formatting.text import Italics, SmallCaps
from simpletex.formatting.layout import Centering

title = Style()
title.apply(Font('Bebas Neue Bold', size=40))

subtitle = Style()
subtitle.apply(Font('Times New Roman', size=11))

Section.heading.apply(Font('Open Sans Semibold', size=16))

Subsection.heading.apply(Font('Open Sans Semibold', size=12))
Subsection.heading.apply(Italics())
Subsection.heading.apply(Centering())

usepackage('geometry', margin='0.5in')
with Document(size='11pt'):
    with Centering():
        write_break(title('Example Title Text'))
        with subtitle:
            write_break("Example Subtitle Text")
            write_break("More Subtitle Text")
    with Section('Section Name'):
        write('Example section text.')
        write(SmallCaps('Lorem ipsum dolor si amet.'))
        with Subsection('Subsection Name'):
            write('Hello World!')

save('filename.tex')

```

TeX Output

```

\documentclass[11pt]{article}

\usepackage[margin=0.5in]{geometry}
\usepackage[utf8]{inputenc}
\usepackage{fontspec}
\usepackage{xltextra}
\usepackage{anyfontsize}
\usepackage{titlesec}

\newfontfamily\BebasNeueBold[Mapping=tex-text]{Bebas Neue Bold}
\newfontfamily\TimesNewRoman[Mapping=tex-text]{Times New Roman}
\newfontfamily\OpenSansSemibold[Mapping=tex-text]{Open Sans Semibold}

\titleformat*{\subsection}{\centering\itshape\fontsize{12}{15}\OpenSansSemibold }
\titleformat*{\section}{\fontsize{16}{20}\OpenSansSemibold }

```

(continues on next page)

(continued from previous page)

```
\begin{document}
  \begin{center}
    {\fontsize{40}{52}\BebasNeueBold Example Title Text} \\
    {\fontsize{11}{14}\TimesNewRoman Example Subtitle Text} \\
    More Subtitle Text \\
  \end{center}
  \section{Section Name}
  Example section text.
  \textsc{Lorem ipsum dolor si amet.}
  \subsection{Subsection Name}
  Hello World!
\end{document}
```

PDF Output

EXAMPLE TITLE TEXT

Example Subtitle Text
More Subtitle Text

1 Section Name

Example section text. LOREM IPSUM DOLOR SI AMET.

1.1 *Subsection Name*

Hello World!

2.1 Core LaTeX

This section documents `simpletex`'s core LaTeX functionality.

2.1.1 Document Structure

This module provides utilities to define a document's basic structure.

```
class simpletex.document.Document (document_class: str = 'article', size: str = '12pt')
```

Bases: `simpletex.base.Environment`

A class which manages a LaTeX document.

Upon instantiation, sets the input encoding to UTF-8 and declares the document class. This class does NOT manage the preamble (imports, newcommand declarations, etc.). The preamble is managed by the `simpletex.Preamble` class.

Create an empty document using the given document class and font size.

document_class [str] The LaTeX document class name to use.

size [str] The default font size to use.

```
class simpletex.document.Section (name: str)
```

Bases: `simpletex.document.Title`

Represents a LaTeX section.

Create an empty section with the given name.

name [str] The section name.

```
class simpletex.document.Subsection (name: str)
```

Bases: `simpletex.document.Title`

Represents a LaTeX subsection.

Create an empty subsection with the given name.

name [str] The subsection name.

2.1.2 Text Formatting

This module provides formatters to perform basic text formatting.

class `simpletex.formatting.text.Bold` (*inline: bool = False*)
Bases: `simpletex.formatting.text.SimpleFormatter`

Applies bold formatting to text.

Create a new bold formatter.

inline [bool] If true, the inline (`\bfshape`) version of the formatter is used. Otherwise, use the default (`\textbf{}`) version.

class `simpletex.formatting.text.Italics` (*inline: bool = False*)
Bases: `simpletex.formatting.text.SimpleFormatter`

Applies italic formatting to text.

Create a new italics formatter.

inline [bool] If true, the inline (`\itshape`) version of the formatter is used. Otherwise, use the default (`\textit{}`) version.

class `simpletex.formatting.text.Underline`
Bases: `simpletex.formatting.text.SimpleFormatter`

Applies underline formatting to text.

Create a new underline formatter.

Note that inline formatting is not supported for underlines.

class `simpletex.formatting.text.SmallCaps` (*inline: bool = False*)
Bases: `simpletex.formatting.text.SimpleFormatter`

Applies small capital formatting to text.

Create a new small capital formatter.

inline [bool] If true, the inline (`\scshape`) version of the formatter is used. Otherwise, use the default (`\textsc{}`) version.

class `simpletex.formatting.text.Emphasis` (*inline: bool = False*)
Bases: `simpletex.formatting.text.SimpleFormatter`

Applies emphasis formatting to text.

Create a new emphasis formatter.

inline [bool] If true, the inline (`\em`) version of the formatter is used. Otherwise, use the default (`\emph{}`) version.

2.1.3 Lists and Sequences

This module provides formatters to create sequences and lists.

class `simpletex.sequences.OrderedList`
 Bases: `simpletex.base.Environment`
 Formats a given list of text as a numbered LaTeX list.
 Equivalent to the LaTeX `enumerate` environment.
 Create an empty numbered list.

class `simpletex.sequences.UnorderedList`
 Bases: `simpletex.base.Environment`
 Formats a given list of text as a bulleted LaTeX list.
 Equivalent to the LaTeX `itemize` environment.
 Create an empty bulleted list.

class `simpletex.sequences.Description`
 Bases: `simpletex.base.Environment`
 Formats a given list of key-value pairs as a LaTeX description list.
 Equivalent to the LaTeX `description` environment.
 Create an empty description list.

2.1.4 Document Layout

This module provides formatters to control document and text layout.

class `simpletex.formatting.layout.Centering` (*inline: bool = False*)
 Bases: `simpletex.base.Environment`
 Centers all contents.
 Equivalent to the LaTeX `center` environment.
 Create a new centering formatter.

inline [bool] If true, the `inline` (`\centering`) version of the formatter is used. Otherwise, use the default (`\begin{center}`) version.

class `simpletex.formatting.layout.Columns` (*number: int = 2*)
 Bases: `simpletex.base.Environment`
 Formats contents into columns.
 Equivalent to the LaTeX `multicols` environment.
 Create a new `multicols` environment.
 Automatically imports the required `multicol` package.
number [int] The number of columns to use.

2.1.5 Equations and Mathematics

This module provides formatters to create and display LaTeX equations.

class `simpletex.math.Equation` (*inline: bool = True*)
 Bases: `simpletex.core.Formatter`
 Formats text as a LaTeX equation.

Initialize an empty equation.

inline [bool] Chooses the equation format to use. If `True`, use the inline ($\$$) syntax. If `False`, use the display-style ($\$\$$) syntax.

class `simpletex.math.Add`

Bases: `simpletex.math.Operator`

Adds arguments together in symbolic form.

Initialize an empty addition operator.

class `simpletex.math.Subtract` (*inline: bool = False*)

Bases: `simpletex.math.Operator`

Subtracts two given arguments in symbolic form.

Initialize an empty subtraction operator.

class `simpletex.math.Multiply` (*symbol='dot'*)

Bases: `simpletex.math.Operator`

Multiplies arguments together in symbolic form.

Initialize an empty multiplication operator.

symbol [str or None] Chooses the multiplication symbol to use. If `None`, no explicit symbol is used. If `'.'` or `'dot'`, a dot-style (`\cdot`) operator is used. If `'x'`, `'cross'`, or `'times'`, a cross-style (`\times`) operator is used. If `'*'` or `'star'`, a star-style (`*`) operator is used.

class `simpletex.math.Divide` (*inline: bool = False*)

Bases: `simpletex.math.Operator`

Divides two given arguments in symbolic form.

Initialize an empty division operator.

inline [bool] Chooses division format to use. If `False`, use the display-style (`\frac{\{ \} \{ \}`) syntax. If `True`, use the inline (`/`) syntax.

class `simpletex.math.Matrix` (*brackets: str = '['*)

Bases: `simpletex.base.Environment`

Represents a matrix.

Can be constructed either with nested lists or a 2D numpy array. Numpy is not required.

Create a new empty matrix.

The type of brackets to use can be specified. Automatically imports the required package `amsmath`.

brackets [str] Type of brackets to use. Supported options are `''` (no brackets), `'('`, `'['`, `'{'`, `'|'`, and `'||'`.

2.2 XeTeX Font Support

This section documents simpletex's support for XeTeX's font customization features.

2.2.1 Font Formatting

This module provides formatters to change the font and size of text.

class `simpletex.formatting.font.Font` (*name: str, size: int = None, skip: int = None, inline: bool = False*)

Bases: `simpletex.core.Formatter`

Changes the font face (and optionally the size) of text.

Imports the required packages `fontspec` and `xltxtra` upon instantiation. If used, the document must be processed with XeTeX or another font-aware TeX processor.

Create a new font formatter using the given font name and size.

name [str] The name of the font to use.

size [int] The font size to use, in pt. If size is None, the font size will not be changed.

skip [int] The line skip to use in pt. If skip is not provided, but size is specified, automatically calculate the skip according to `skip = int(size*1.3)`.

inline [bool] If True, the inline (TeX directive) version of the formatter is used. Otherwise, use the command form.

class `simpletex.formatting.font.SizeSelector` (*size: int, skip: int = None*)

Bases: `simpletex.core.Formatter`

Changes the font size of text.

Allows specifying an optional skip parameter. Imports the required package `anyfontsize` on instantiation.

Create a new font size formatter with the given size and skip.

Automatically imports the required package `anyfontsize`.

size [int] The font size to use, in pt. If size is None, the formatter will do nothing.

skip [int] The line skip to use in pt. If skip is not provided, but size is specified, automatically calculate the skip according to `skip = int(size*1.3)`.

S

`simpletex.document`, [11](#)
`simpletex.formatting.font`, [14](#)
`simpletex.formatting.layout`, [13](#)
`simpletex.formatting.text`, [12](#)
`simpletex.math`, [13](#)
`simpletex.sequences`, [12](#)

A

Add (class in `simpletex.math`), 14

B

Bold (class in `simpletex.formatting.text`), 12

C

Centering (class in `simpletex.formatting.layout`), 13

Columns (class in `simpletex.formatting.layout`), 13

D

Description (class in `simpletex.sequences`), 13

Divide (class in `simpletex.math`), 14

Document (class in `simpletex.document`), 11

E

Emphasis (class in `simpletex.formatting.text`), 12

Equation (class in `simpletex.math`), 13

F

Font (class in `simpletex.formatting.font`), 14

I

Italics (class in `simpletex.formatting.text`), 12

M

Matrix (class in `simpletex.math`), 14

Multiply (class in `simpletex.math`), 14

O

OrderedList (class in `simpletex.sequences`), 12

S

Section (class in `simpletex.document`), 11

`simpletex.document` (module), 11

`simpletex.formatting.font` (module), 14

`simpletex.formatting.layout` (module), 13

`simpletex.formatting.text` (module), 12

`simpletex.math` (module), 13

`simpletex.sequences` (module), 12

SizeSelector (class in `simpletex.formatting.font`), 15

SmallCaps (class in `simpletex.formatting.text`), 12

Subsection (class in `simpletex.document`), 11

Subtract (class in `simpletex.math`), 14

U

Underline (class in `simpletex.formatting.text`), 12

UnorderedList (class in `simpletex.sequences`), 13