Small and Easy, but Beautiful Design For You

KiTTy

Beautiful Design for you …

Contents

List of Figures

List of Tables

Chapter 1

Introduction

First, this section introduces a KiTTy itself and shows a comparison to alternative softwares, and introduces a value of KiTTy and use case of KiTTy. And also, it shows supported features.

1.1 What is KiTTy

KiTTy means **Ki**nx **T**iny **T**ypesetting, which is a simple typesetting system implemented by Kinx. It also provides a translator from Markdown, then you can typeset a Markdown document and can get a beautiful document. This document itself is also the example typeset by this system.

The objective is similar to LaTeX, it is an objective to typeset beautifully for a document managed as a text file. To be concretely, it is never going to be alternative, but the objective is being more useful in the use case like your personal situation by followings.

- Keeping a small system.
- Pretty beautiful output.
- Directly output as PDF file.

KiTTy is small but it has a simple feature to typeset it beautifully, and it is a typesetting system to output PDF directly from Markdown document.

1.1.1 Versus LaTeX

LaTeX is a huge system. It is extendable, it is supported by many people, and it is a typesetting system which can make a beautiful document. KiTTy is also a typesetting system which has a same objective as LaTeX, but it is provided as a small system with limited features.

Being huge of LaTeX system causes a complexibility of an installation. Besides, there are multiple distributions to provide many functionality of TeX and LaTeX. That is the reason why people have to think first what distribution should be used. Everybody can use KiTTy soon because KiTTy is included in Kind as a standard library,

Only one cons point is that limited features[1] are provided. And also the compilation speed is *slow*. It takes about 4 minutes to compile this document. The performance is one of problems, but this system is focusing what is used as a small and a personal use.

---

[1] About "limited features", see Chapter 3 Features Overview.

1.1.2 Versus Word

The Word is a famous WYSIWYG[2] Word Processor, but it is based on a different concept. The word processor can edit a document as it shows, but it usually saves a document as a binary style. Therefore, in general, you need an own software, which is "Word" in this case, to show its content. With KiTTy, same as LaTeX, you can see its content directly because it is a text style, and can edit it with your familier editor.

Saving as a text, it can be processed easily by another software, and also showing diff in a version control like Git is very easy. This is quite important and a mandatory requirement, especially for the management of source code. Besides, about the structure of a document, it is difficult from its looking to distinguish if it is structured or not. For example, you can not find its chapter number is correctly assigned, and you can not find if it would corrected when you change a layout or a location of sentences. In particular, there is a case sometimes you have met unlucky cases as someone has not set it up correctly.

The structure is clearly written in text style in KiTTy, it can correctly realize to make a cross-reference, numbering a chapter number, a figure number, a table number, and a page number. Instead, it is a cons point not to show the document as it shows, like WYSIWYG. You have to compile first to show the actual document to be typeset.

1.1.3 Versus Vivliostyle

CSS Typesetting is based on the Web Technology, and CSS Typesetting now has a most great future. Therefore CSS Typesetting is recommended if you want to do typeset in earnest. Especially Vivliostyle[3] is a very hopeful project.

As for KiTTy, it is important to be a small system and easy to use. Although time will solve, CSS Typesetting is still making it standard in progress. Therefore some of the specification has not been clarified yet, and it is possible to assume it will be changed in future. As this is a big project and making it standardized, it will be a system to use in a real commercial use, but the system itself will be getting huge.

Said again, a main point of view in KiTTy is a small and easy to use.

1.1.4 Conclusion

If you want to use a typesetting system right now in a real earnest, you should use LaTeX. If you want to study a typesetting system in the future and use it, you had better use a CSS Typesetting like Vivliostyle. Moreover, in sumerize, KiTTy will useful in following situations.

- Wants to use a small system, instead of the huge system like TeX.
- Wants to manage a diff by a version control system like Git.
- Wants to control the structure of a document and make a cross-reference correctly.

---

[2] This means **W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et.
[3] https://vivliostyle.org/

This system is provided to the person[4] who think the TEX system is very huge as a system for creating a small document, and who wants to manage the document as a text file. In particular, if you want to manage a document and a diff by the system like Git, it is quite difficult to manage it by Word which is realized by the style of WYSIWYG. It is a main target of use cases not to manage by a word processor and not to use a huge system.

1.2 Supported Features

1.2.1 Typesetting Features

KiTTy supports following features. See "Chapter 3 Features Overview" for a concrete content of features. By the way, kerning is not supported so far.

- Hyphenation, justification, and line-breaking algorithm
- Widows and Orphans
- Multiple columns
- Itemization
- Math formula and equations
- Image
- Chart
- Table
- Font
- Color
- Ligature
- Programming Code Block
- Title, Cover Page, and Contents
- Chapter and Section
- Cross-Reference
- Quotation
- Footnote

1.2.2 Typesetting Features for Japanese

KiTTy also supports following feautres for the own requirement in Japanese. If you want to add any other languages, I can not lead you because I am sorry I do not have any knowkedge for other languages. As it has been difficult to care about the extension point for other languages, it might need to add a lot of features or many points of fixes. But it will be possible to extend features because this is an open source.

- Japanese hyphenation
- Japanese Ruby like Difficult Kanji (how to read)

---

[4] This means the person like me. In fact, the first objective was to create a user manual of my own project.

1.2.3 PDF Features

Although this is not available when it is printed on a paper, the following features are supported.

- External Link by URL
- Link by Cross-Reference
- Bookmark

Chapter 2

Getting Started

This chapter introduces about preparing to start using KiTTy in practice and how to use through simple examples. By the way, there are differences about how to prepare between Windows and Linux, but you can get a same result as long as you use prepared fonts in KiTTy. Therefore you can use both without problems.

2.1 Installation

There are 2 steps to install it.

1. Install Kinx
2. Install KiTTy package

2.1.1 Linux

Download the following modules first on Linux. The version `v0.15.2` is the version which includes KiTTy library officially. When the latest version is different, rewrite the version number and do it. Make the work directory and change the directory.

```
$ mkdir temp
$ cd temp
```

Download Kinx module and extract it first, and then download KiTTy package and extract it.

```
$ curl -L \
    https://github.com/Kray-G/kinx/releases/download/v0.15.2/package_linux-amd64.tar.gz \
    --output package_linux-amd64.tar.gz
$ tar -xvf package_linux-amd64.tar.gz
$ curl -L \
    https://github.com/Kray-G/kinx/releases/download/v0.15.2/package_kitty.zip \
    --output package_kitty.zip
$ unzip package_kitty.zip
```

You will find the folder of the name of the version number, and move into it and execute a `install.sh` command.

```
$ cd v0.15.2
$ sudo ./install.sh
```

That is all for installation. Let's confirm the location to be installed.

```
$ which kinx
/usr/bin/kinx
```

2.1.2 Windows

Download 2 latest modules from Release Page when it is Windows.

- package_win64.zip
- package_kitty.zip

Extract each, and then copy files in `package_kitty.zip` to `lib` folder[1].

> The `fonts` and `phantomjs` folder should be right under `lib` folder. Please note that the files in an archive file might not be that structure.

2.2 Build

You do not have to build it usually, you can use a prebuilt modules following the procedure of "2.1 Installation". If you dare to build it, you can do it by the procedure as follows.

2.2.1 Linux

Clone it from Github, and do `make`.

```
$ git clone https://github.com/Kray-G/kinx.git
$ cd kinx
$ make
```

Install it.

```
$ git clone https://github.com/Kray-G/kinx.git
$ cd kinx
$ sudo make install
$ sudo make kitty-install
```

2.2.2 Windows

Clone it from Github, and do `make`.

```
$ git clone https://github.com/Kray-G/kinx.git
$ cd kinx
$ make.cmd
```

Now no installer has been provided, but you can use this system in the build environment. Use the system in the build environment.

---

[1] The installer for Windows is now not provided yet, but it will be prepared in the future.

2.3 hello, world

Create a following document and save it as the name of `helloworld.md`.

```
% Hello Kinx Tiny Typesetting
% Your name
% October 7, 2020

<param style="ArticleA4"/>

# Greeting
hello, world
```

Execute `kxkitty` command as below, and `helloworld.pdf` will be generated.

```
$ kxkitty helloworld.md
```

By the way, now you need write the document as above, but I will plan that it can be a little more simple document.

Chapter 3

Features Overview

This chapter will explain the feature's overview. As this document itself has been generated by this system, you can realize all what this document realize. Let's try various things first.

3.1 Typesetting Features

3.1.1 Hyphenation, Justification, and Line-Breaking Algorithm

This system supports a Hyphenation based on Franklin M. Liang algorithm. And also this supports a justification with that hyphenation algorithm.

Line-breaking is based on Knuth-Plass Line Breaking algorithm. This algorithm is what is controlled by Box, Glue, and Penalty,

This Kinx TT has supported some kind of TEX algorithms, so the final output would be very beautiful. You can check it on your eyes yourself as this document was generated by this system. On the other hand, there are some bad points below as a trade off.

Fig 3.1   Hyphenation and Justification

and it is same as the algorithm implemented in TEX. These hyphenation and line-breaking argorithm are the well-known as the best practice in the typesetting so far. Note that it is not all same as the output by TEX, because this is implemented by Kinx.

3.1.2 Widows and Orphans

In some case, this system provides a part of a penalty control for Widows and Orphans. Note that this is not perfect for all cases. It is available for following cases.

- Deterrent to put a section name to the bottom of page.
    - In this case, a section name is moved to a next page.
- Deterrent to put only a top line in a paragraph to the bottom of page.
    - In this case, all lines in a paragraph is moved to a next page.
    - When only a section name is put to the bottom of page as a result, a section name is also moved to the next page.
- Deterrent to put only a last line in a paragraph to the top of a next page.
    - In this case, last 2 lines are moved to a next page.

Those above mechanisms are automatically done. You do not have to do anything in your document. But those are not perfect for all cases, therefore use `<pagebreak />` command when you feel it is not good layout, and you can insert a page break there.

3.1.3 Multiple columns

You can use multiple columns. Write `<set-column value="N"/>`, and the document is set to N columns. When you get back to only 1 column, use `<set-column value="1"/>`. The line will be back to the top of a next column when the text is reached at the bottom of page.

You can set the column height with `height` parameter. `height` parameter can be the value with unit like `10em`. Writing `<set-column value="2" height="12em" />` will set 2 columns with the height of 12em.

About footnotes, it is not set each column. footnotes are always set against total width of a page. The following example is a part of *Alice' s Adventures in Wonderland*[1] with 2 columns.

CHAPTER I. Down the Rabbit-Hole

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice "without pictures or conversations?"

So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

There was nothing so VERY remark-able in that; nor did Alice think it so VERY much out of the way to hear the Rabbit say to itself, "Oh dear! Oh dear! I shall be late!" (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-POCKET, and looked at it, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.

The height is set to 30em here. As the system does not automatically set the height, you have to set it yourself one by one if you want to set the height. The line is reached at the bottom of page, the text will be back to the top of a next column automatically. When you do not set the height, the text will get back to the top of a next column automatically when it reached at the bottom of a page.

---

[1] Alice' s Adventures in Wonderland - Lewis Carroll

3.1.4 Itemization

There are 2 types of itemization with a symbol and a number. The following is an example of itemization with a symbol.

```
1  *   Level 1
2      *   Level 2
3          *   Level 3
4              *   Level 4
```

This is formatted as follows.

- Level 1
  - Level 2
    - Level 3
      * Level 4

The next example is a numbered itemization. The number will be adjusted automatically.

```
1  1.  Level 1
2      1.  Level 2
3          1.  Level 3
4              1.  Level 4
5              1.  Level 4
```

This is formatted as follows.

1. Level 1
   (a) Level 2
       i. Level 3
           A. Level 4
           B. Level 4

It is possible to use both and mix it in the same list. The next example is a mixed one.

```
1  *   Level 1
2      1.  Level 2
3          *   Level 3
4              1.  Level 4
```

This is formatted as follows.

- Level 1
  (a) Level 2
      - Level 3
          A. Level 4

3.1.5 Math formula and equations

KiTTy includes LaTeX, and output Math formula and equations. There are 2 styles of standalone style and inline style as an output style.

3.1.5.1 Standalone Style

When it is a standalone style, it is written as a Code Block with `Math` language. This is shown in an independent line.

```math:label=Math1
\begin{aligned}
    \int_{-\infty}^{\infty} f(x) dx &= \sqrt{\pi}
\end{aligned}
```

For above, the output will be below. It is not necessary to set the option of `label`, but you can use 1 for the reference to Math equations if you set the `label`.

$$\int_{-\infty}^{\infty} f(x)dx = \sqrt{\pi} \tag{1}$$

Note that `label` is supported not by KaTeX but by KiTTy. Therefore you have to control `label` when there are 2 or more Math equations.

```math:label=Math2(0.2)/Math3(0.6)
\begin{aligned}
    E &= mc^2 \\
    m &= \frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}} \\
\end{aligned}
```

For above, the label of Math2 and Math3 will be put at 20% and 60% from the top of the location where the Math equations is.

$$E = mc^2 \tag{2}$$

$$m = \frac{m_0}{\sqrt{1 - \frac{v^2}{c^2}}} \tag{3}$$

By this, writing `\ref{Math2}` creates the reference to the Math 2, and writing `\ref{Math3}` creates the refenrence to the Math 3.

3.1.5.2 Inline Style

When using inline Math, use $ around the Math equations. For example, writing `$E = mc^2$` shows $E = mc^2$. And writing a symbol with height like integral, for example, writing `$\\int\_{-\\infty}^{\\infty} f(x) dx = \\sqrt{\\pi}$` as the same equation as Math 1 will show $\int_{-\infty}^{\infty} f(x)dx = \sqrt{\pi}$. By the way, note that `\` and `_` have to be escaped because it is the same symbol of Markdown.

You can use `\\displaystyle` inside `$` if you want to use a large type of fomula. For example, writing `$\\displaystyle\\int\_{-\\infty}^{\\infty} f(x) dx = \\sqrt{\\pi}$` shows $\displaystyle\int_{-\infty}^{\infty} f(x)dx = \sqrt{\pi}$. But anyway it is not recommended because the height is different and not beautiful.

3.1.6 Image

For Image, you can use a Markdown syntax for Image as is, but the option will be written in the position of `alt`, and then the syntax will be the style of `![options](path)`. It is possible to insert an image as a standalone style, an inline style, and a floating style.

3.1.6.1 Standalone Image

To show a standalone image, write it as an independent paragraph between blank lines.

```
![scale=0.6](kinxlogo.png)
```

For above, the image will be inserted as below. The image width is adjusted to 60% for the page width by the option of `scale=0.6`. The ratio of width and height is stayed.

3.1.6.2 Inline Image

Here is an example of inline style. When inserting an image as inline style, you write it directly inside text like the following.

```
The file icon is changed to ![scale=0.08,offsetY=-5.0](zip256.png).
```

In this case, "The file icon is changed to ." will appear. You have to adjust `scale` and `offsetY` according to the original size of image.

3.1.6.3 Floating Image

To float an image, use `float=left` or `float=right` to the option. The image of "Figure 3.1 Hyphenation and Justification" at the page of 9 is one of example of this.

Fig 3.2   Damselfly

The picture on the left is provided as a Public Domain[2]. The image is located as a floating image like this. The option of `scale` can be used with a floating image, but its size is limited within 70%[3] of the area wthout margins.

And also, you can write multiple paragraphs around a floating image. When a text of a paragraph is finally reached at the bottom of a floating image, the text width of a line will be automatically back to the width of the page, and the text is located around a image naturally.

In this case, the right picture is an example of a floating image which is located to the right of the text. The image is also Public Domain as same as the above picture.

There is a note that the top of an image have to be set to the top of a paragraph. You can not put an image inside a paragraph. If

Fig 3.3   F14 Tomcats

there is an image you want to float when some paragraph starts, the image will be located to the left or right which you specified when paragraph starts.

3.1.6.4 Image Options

You can use options as follows.

Table 3.1 Image Options

Option	Value	Meaning
float	left, right	A location of a floating image.
scale	0.0∼ 1.0	A scale against the area between margins.
caption	Text	A caption of an image.
box	BOX_NORMAL	Output a box with a normal line.
	BOX_THIN	Output a box with a thin line.
	BOX_THICK	Output a box with a thick line.
padding	Real number	The margin between an image and a text.

---

[2] https://free-images.com/
[3] The margin between an image and a text is included in this 70%.

3.1.7 Chart

Chart is also supported. It supports both a stanalone style and a floating style.

3.1.7.1 Standalone Chart

For a standalone style of chart, use a Code Block with the language of `chart`. For example, it is written in JSON structure as below. In addition to chart information like `width` and `height`, Chart.js[4] data itself have to be put in the field of `options`.

```chart
{
    width: 800,
    height: 400,
    fontSize: 16,
    scale: 1.0,
    caption: "Radar Chart Example",
    options: {
        type: "radar",
        data: {
            labels: [["Eating", "Dinner"], ["Drinking", "Water"],
                "Sleeping", ["Designing", "Graphics"],
                "Coding", "Cycling", "Running"],
            datasets: [{
                label: "My First dataset",
                backgroundColor: "rgba(255, 0, 0, 0.2)",
                borderColor: "red",
                pointBackgroundColor: "red",
                data: [ 10.1, 80.0, 72.2, 73.3, 55.0, 68.5, 92.0 ]
            }, {
                label: "My Second dataset",
                backgroundColor: "rgba(0, 0, 255, 0.2)",
                borderColor: "blue",
                pointBackgroundColor: "blue",
                data: [ 30.9, 77.1, 49.9, 50.0, 67.8, 71.0, 22.8 ]
            }]
        },
        options: {
            legend: {
                position: "top",
            },
            scale: {
                ticks: {
                    beginAtZero: true
                }
            }
        }
    }
}
```

This will be shown as Figure 3.4 Radar Chart Example.
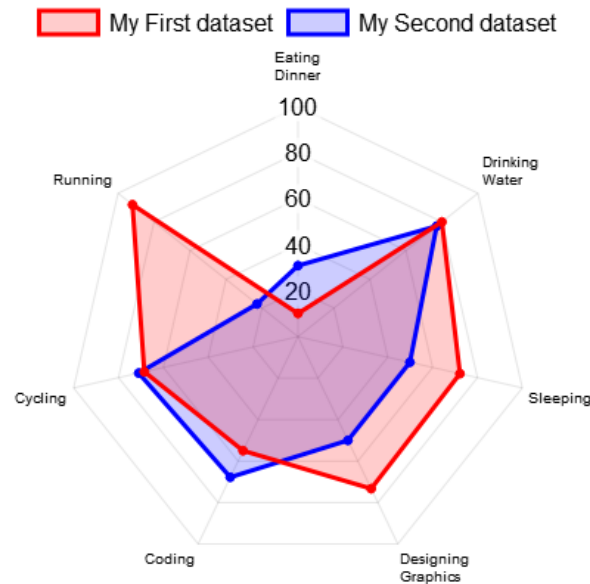
---

[4] https://www.chartjs.org/

Fig 3.4 Radar Chart Example

3.1.7.2 Floating Chart

A floating chart is also supported as same as an Image.

Like above, you can set `float: { right: true }` to the option, the chart will be floated and the next paragraph will be located around that chart.

In this case, it is a line chart. `scale` means a width against the area without margins as same as an image. You can put the caption for both a standalone style and a floating style, and a chart will inserted to the

Fig 3.5   Line Chart Example

document as an image. This caption will be shown in the List Of Figures when setting to output Table Of Contents. And also it is same as an floating image, a paragraph is naturally located under the chart.
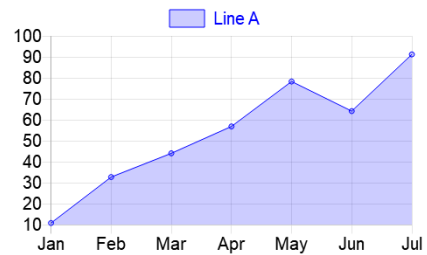
The following is an example of the code. By the way, in this time `options` is omitted due to space limitations, but the `options` means the option of Chart.js.

```chart
{
    float: { right: true },
    width: 480, height: 300, scale: 0.4, caption: "Line Chart Example",
    options: {
        type: "line",
        ...(omitted)
    }
}
```

3.1.8 Table

3.1.8.1 Markdown Table

Table is also supported. The table can be written as a normal Markdown table, and the table is automatically generated.

For example, "Table 3.1 Image Options" is written as below. Use a `<context />` tag for the option which is not directly specified as a Markdown syntax.

```
1  <context label="Table:ImageOptions"/>
2  <context caption="Image Options"/>
3  | Option  |     Value      |                 Meaning                 |
4  | ------- | -------------- | --------------------------------------- |
5  | `float` | `left`, `right`| A location of a floating image.         |
6  | `scale` | 0.0 ~ 1.0      | A scale against the area between margins.|
```

Using a normal Markdown syntax, you can make it centering, or aligning to the left or right. Besides, you can use Math formula inside your table. If the cell content is becoming too long, you can use `<context cell-i-j="..." />` to write a long cell content. In this case, (i, j) means the location without a header. For example, $(0, 0)$ means `A1` cell in the following table. You see i means a row number and j means a column number.

```
1   <context label="Table:TableExample"/>
2   <context caption="Example of Table"/>
3   <context vline-left="single"/>
4   <context vline-right="single"/>
5   <context vline-inside="single"/>
6   <context hline-header="double"/>
7   <context hline-inside="single"/>
8   <context cell-2-1="$\displaystyle\int_{-\infty}^{\infty} f(x) dx = \sqrt{\pi}$"/>
9   | Left |        Center        |        Right         |
10  | :--- | :------------------: | -------------------: |
11  | A1   | Aligned to the center.| Aligned to the right.|
12  | A2   |     Cell $(1,1)$     |     Cell $(1,2)$     |
13  | A3   |          -           |     Cell $(2,2)$     |
```

Here is the output result below.

Table 3.2 Example of Table

Left	Center	Right
A1	Aligned to the center.	Aligned to the right.
A2	Cell $(1, 2)$	Cell $(1, 2)$
A3	$\displaystyle\int_{-\infty}^{\infty} f(x)dx = \sqrt{\pi}$	Cell $(2, 2)$

3.1.8.2 Table Options

`<context />` tag can specify the parameters which can not be specified in a normal Markdown syntax. "Table 3.3 Table Options" shows the list of the parameters which can be specified for the table. Those parameters are used as a temporary and they will be removed right after it being used. Therefore the parameters specified once can not used for multiple tables. You have to set it again if you want to use the same parameters for a next table.

But you can change a default value for the parameters except `label` and `caption`. For that, change the value for the name with `-default` at the end of the parameter name. For example, you can change the default value of `vline-left` by changing the value for `vline-left-default`.

Table 3.3 Table Options

Name	Value	Meaning
`label`	Text	The label for Cross-Reference.
`caption`	Text	The caption of a table.
`vline-left`	`single`, `double`, `false`	Vertical line type of the left on a table (`false` by default).
`vline-right`	Same as above	Vertical line type of the right on a table (`false` by default).
`vline-inside`	Same as above	Vertical line type inside a table (`false` by default).
`hline-top`	Same as above	Horizontal line type of the top on a table (`single` by default).
`hline-bottom`	Same as above	Horizontal line type of the bottom on a table (`single` by default).
`hline-header`	Same as above	Horizontal line type under a header (`single` by default).
`hline-inside`	Same as above	Horizontal line type inside a table (`false` by default).
`cell-i-j`	Text	The text in the cell of (i, j). i is a row number and j is a column number, and both are 0 origin.
`limit-column`	Integer number	The column number for the minimum width to determine a cell width limitation.
`limit-width`	Real number	The minimum width of a cell width.

3.1.9 Font

3.1.9.1 Bold, Italic, BoldItalic

Bold、 *Italic*、 ***BoldItalic*** is shown by a normal Markdown syntax. Here is examples.

Table 3.4 How to write Bold, Italic, and BoldItalic

Markdown	Output
`**Bold**`	**Bold**
`*Italic*`	*Italic*
`***BoldItalic***`	***BoldItalic***

3.1.9.2 Use Fonts

If you want to use a font which is not loaded by default, load it by yourself, and you can use it. The following shows how to load a font file. It is necessary to specify 4 parameters separated by comma.

```
1   <font-load info="Name,type,shape,FileName.ttf" />
```

You can freely use the font after loading once. The parameter meaning is as follows.

- `Name` is the own name of the font you can specify.
- `type` is a font type. It should be `serif`, `sans`, `monotype`.
- `shape` is a font shape. It should be `regular`、 `bold`、 `italic`、 `bolditalic`.
    - `regular` means a normal shape used in the normal sentence.
    - `bold`, `italic`, and `bolditalic` is for the shape meant by that name.

Use `\font` command to change the font as a following example. By the way, you have to escape \ of a command name, [, and] like \\, \[, and \] because it is a Markdown special character. The following example shows how to use `Parisienne-Regular.ttf` font[5].

```
1   <font-load info="Parisienne,serif,regular,Parisienne-Regular.ttf" />
2   - Changing the font is available only with a scope like
3   ''\\font\[name=Parisienne\]{This is a pen.},''
4   and the font will be restored here.
```

This shows as below. This is available inside a scope of `\font` command as you see in the following example.

- Changing the font is available only with a scope like "*This is a pen.*," and the font will be restored here.

---

[5] This TrueType font file is already included in the KiTTy package, but it is not loaded by default. Load it by yourself to use it.

3.1.9.3 Font Size (Direct)

Font size is specified by `size` parameter of `\font` command.

```
1  ''After this, the font size will be \\font\[size=7pt\]{shrink to 7pt}.
2  And after this, the font size will be \\font\[size=15pt\]{extend to 15pt}.''
3  will be shown.
4  And also, for example, `\font[size=1.2em]{1.2 times larger size}` shows
5  \\font\[size=1.2em\]{1.2 times larger size} will be shown.
```

"After this, the font size will be shrink to 7pt. And after this, the font size will be extend to 15pt." will be shown. And also, for example, `\font[size=1.2em]{1.2 times larger size}` shows 1.2 times larger size will be shown.

You can use the unit of size as you see in the above example. The unit you can specified is as follows.

Table 3.5 Unit for Font Size

Unit	Meaning
em	Relative size based on the current font size.
ex	Relative size based on the x height at the current font size.
px	By Pixel.
pt	By Point.
pc	By Pica, which means 1/6 inch.
mm	By Millimeter.
cm	By Centimeter.
in	By Inch.

3.1.9.4 Font Size (Relative)

Use `\bigger`、`\smaller` to change the size with a relative size. The size will be changed by +1 point for `\bigger` or -1 point for `\smaller` inside its scope.

```
1  This is an example of `\bigger`.
2  \\bigger{This is, \\bigger{this is, \\bigger{this is} a sentence,} a sentence}.
3  For `\smaller`,
4  \\smaller{This is, \\smaller{this is, \\smaller{this is} a sentence,} a sentence}.
```

This is an example of `\bigger`. This is, this is, this is a sentence, a sentence. For `\smaller`, This is, this is, this is a sentence, a sentence.

3.1.10 Color

3.1.10.1 Text Color

Use \color to change the text color. Here are examples below.

```
 1  *  \\color\[red\]{\\bold{Red}.
 2     This line should be colored by the name of `red`.}
 3  *  \\color\[green\]{\\bold{Green}.
 4     This line should be colored by the name of `green`.}
 5  *  \\color\[blue\]{\\bold{Blue}.
 6     This line should be colored by the name of `blue`.}
 7  *  \\color\[cyan1\]{\\bold{Cyan}.
 8     This line should be colored by the name of `cyan1`.}
 9  *  \\color\[magenta1\]{\\bold{Magenta}.
10     This line should be colored by the name of `magenta1`.}
11  *  \\color\[yellow\]{\\bold{Yellow}.
12     This line should be colored by the name of `yellow`.}
13  *  \\color\[R=0,G=64,B=255\]{\\bold{RGB}.
14     This line should be colored by RGB value of `RGB=0,64,255`.}
15  *  \\color\[C=0.5,M=0.8,Y=0.2,K=0.0\]{\\bold{CMYK}.
16     This line should be colored by CMYK value of `CMYK=0.5,0.8,0.2,0.0`.}
```

- **Red**. This line should be colored by the name of red.
- **Green**. This line should be colored by the name of green.
- **Blue**. This line should be colored by the name of blue.
- **Cyan**. This line should be colored by the name of cyan1.
- **Magenta**. This line should be colored by the name of magenta1.
- **Yellow**. This line should be colored by the name of yellow.
- **RGB**. This line should be colored by RGB value of RGB=0,64,255.
- **CMYK**. This line should be colored by CMYK value of CMYK=0.5,0.8,0.2,0.0.

See ?? about supported color names.

3.1.11 Ligature and Special Character

3.1.11.1 Ligature

Now the following 5 ligatures is only supported.

- fi ... fi
- fl ... fl
- ff ... ff
- ffi ... ffi
- ffl ... ffl

3.1.11.2 " and "

both " and " are automatically replaced by writing 2 single quotes.

```
1  This is an ''example'' of double-quote.
```

This is an "example" of double-quote, will be shown.

3.1.11.3 Backquote

You can use a special command because backquote is same as a special character in Markdown. For this purpose, the special tag of `<backqN />` where `N` is a number of backquote can be used. Write `<backq3 />` to show ` ``` `, and write `<backq2 />` to show ` `` `. If you want to use `Mototype`, you can use `<monotype-backqN />`.

3.1.12 Programming Code Block

3.1.12.1 Code Block

Programming Code is written as a Code Block. It shows with a line number and a shadow box by default.

```
1  ```
2  class Test {
3      public test() {
4          # Test Method.
5      }
6  }
7  ```
```

Above example will show the following code[6]。

```
1  class Test {
2      public test() {
3          # Test Method.
4      }
5  }
```

Write the following to erase a line number and to be without a shadow.

```
1  ```:lineNumber=false,box=BOX_NORMAL
2  class Test {
3      public test() {
4          # Test Method.
5      }
6  }
7  ```
```

---

[6] It is hard to distinguish those because the same style, but the code with ` ``` ` is the example to write.

Here is the result.

```
class Test {
   public test() {
       # Test Method.
   }
}
```

3.1.12.2   Code Block Option

Specify `true` or `false` to the `lineNumber` option. The available values for the `box` option are as follows.

Table 3.6 The available values for box

値	意味
BOX_NORMAL	Show a box with a normal line width.
BOX_THIN	Show a box with a thin line width.
BOX_THICK	Show a box with a thick line width.
BOX_SHADOW	Show a box with a shadow. (by default)

3.1.13   Title, Cover Page, and Contents

3.1.13.1   Specify Title Content

KiTTy will automatically generate a title page, a cover page, and a table of contents. The necessary parameters have to be set before the first chapter title. The following example is the example of parameters used by this document. A title, author, and date is set by `%` at the head of a line. You should write only `%` for no information because its order can not be changed. The paragraph right after the setting by `%` will be a sub title.

```
1  % KiTTy
2  % Kray-G, Mr.Diamond Global Blue Publisher
3  % September 18, 2020
4
5  Small and Easy, but Beautiful Design For You
```

3.1.13.2   Other Parameters for Title

About the other parameters for a title, specify those by `<param />` tag before the first chapter title. The following is the example in this document.

```
1  <param style="JBookA4"/>
2  <param titleSize="78.8"/>
3  <param subtitleSize="14.4"/>
4  <param backgroundImage="back.jpg"/>
```

The parameter which can be set depends on the style. This example shows parameters which can be set in the `BookA4` style. The meaning of each parameter is as follows.

- This document will use the `BookA4` style.
- The size of a title text is 78.8pt.
- The size of a sub title text is 14.4pt.
- The cover page will use the `back.jpg` for the background image.

3.1.13.3 Parameters for Table Of Contents

To set the following command, and the document will show a table of contents. By `with` parameter, you can choose if the document will show a list of figures and a list of table[7].

```
1  <toc with="lof,lot"/>
```

3.1.13.4 Customize Style

There are default values of parameters for each style. Use `<style-info />` tag to change those default vlues. For example, write the tag like a following to change the chapter design. See ?? for the chaptr design.

```
1  <style-info name="chapter.style" value="BigChapter3" />
```

See ?? for details of style parameters which you can change.

3.1.14 Chapter and Section

3.1.15 Cross-Reference

3.1.16 Quotation

3.1.17 Footnote

---

[7] Set `lof` for a list of figures, and `lot` for a list of tables. If you want to show both, set it separated by comma like `lof,lot`.