

# 日本語 の俺的最強チートシート

Bony\_Chops

2020 年 12 月 9 日

## 概 要

ここを書くには

---

```
1 \begin{abstract}
2   ここを書くには...
3 \end{abstract}
```

---

## 0 はじめに

Hello hackers! これから Word から日本語 を使い始めるキミや、ある程度日本語 には慣れたけどコマンドを忘れがちなキミに向けたチートシートです。

### 0.1 付録について

この PDF や  $\text{T}_\text{E}\text{X}$  ソース、付録に掲載しているソースは全て GitHub 上で公開しています。実践する際にぜひお役立てください。

🐼BonyChops / latex-cheatsheet (<https://github.com/BonyChops/latex-cheatsheet>)

### 0.2 環境

本チートシートは以下の環境を使っていることを想定して書いています。特殊な仕様意外は基本的に同じだと思います。

表 1: 筆者の環境

OS	Ubuntu 20.04
エディタ	Visual Studio Code
環境	TeX Live 2017
コンパイルスクリプト	ptex2pdf
documentclass	jarticle

また、本ガイドと同じ環境で行う場合、以下の sty ファイルが必要です。インターネットから探してきて、tex ファイルと同じディレクトリに置くか、付録 B を参考にして、sty ファイルをインストールしてください。

- jlisting.sty

# 1 コマンド集

いっぱいあります。載せる順番は適当です。

## 1.1 章・節・項を構成する … \section · \subsection · \subsubsection

章・節・項を構成する

### 1.1.1 レポート風

#### コマンド

```
1 \section{目的}
2 今回はりんごの剥き方を理解することを目的とする。
3
4 \section{理論}
5 本章では本実験に必要な理論をまとめる。
6
7 \subsection{包丁について}
8 本節では包丁についてをまとめる。
9
10 \subsubsection{さばき方}
11 本項では $\cdots$
```

#### 実行結果

## 1 目的

今回はりんごの剥き方を理解することを目的とする。

## 2 理論

本章では本実験に必要な理論をまとめる。

### 2.1 包丁について

本節では包丁についてをまとめる。

#### 2.1.1 さばき方

本項では …

### 1.1.2 番号をなくす

コマンドの最後に\*をつけよう

#### コマンド

```
1 \section{番号ありの章}
2 \subsection{番号ありの節}
3 \subsubsection{番号ありの項}
4
5 \section*{番号なしの章}
6 \subsection*{番号なしの節}
7 \subsubsection*{番号なしの項}
```

#### 実行結果

## 1 番号ありの章

### 1.1 番号ありの節

#### 1.1.1 番号ありの項

## 番号なしの章

### 番号なしの節

#### 番号なしの項

## 1.2 箇条書き … itemize · enumerate · description

### 1.2.1 記号付き

#### コマンド

```
1 カレーの材料
2 \begin{itemize}
3   \item ジャがいも
4   \item にんじん
5   \item 玉ねぎ
6   \item 牛肉
7   \item ルー
8 \end{itemize}
```

#### 実行結果

カレーの材料

- ジャがいも
- にんじん
- 玉ねぎ
- 牛肉
- ルー

### 1.2.2 数字付き

#### コマンド

```
1 カレーのつくりかた
2 \begin{enumerate}
3   \item 材料を切る
4   \item 炒める
5   \item 水に材料を入れる
6   \item ルーを入れる
7   \item 完成
8 \end{enumerate}
```

#### 実行結果

カレーのつくりかた

1. 材料を切る
2. 炒める
3. 水に材料を入れる
4. ルーを入れる
5. 完成

### 1.2.3 見出しつき

見出しの後に改行するには `\mbox{}\` で。

#### コマンド

```
1 カレーづくりで大切にすること
2 \begin{description}
3   \item[味見] 美味しいカレーを作ら
4     しょう。
5   \item[見た目]\mbox{}\ 見た目が良
6     いカレーを作らしょう。
7   \item[愛情]\mbox{}\ 愛を込めたカ
8     レーを作らしょう。
9 \end{description}
```

#### 実行結果

カレーづくりで大切にすること

味見 美味しいカレーを作らしょう。

見た目  
見た目が良いカレーを作らしょう。

愛情  
愛を込めたカレーを作らしょう。

### 1.3 画像 … includegraphics

#### コマンド

```
1 \begin{figure}[H]
2   \center
3   \includegraphics[width=3cm]{./image/writing_man2_angry.png}
4   \caption{再提出にキレる学生 \copyright いらすとや}
5 \end{figure}
```

#### 実行結果



図 1: 再提出にキレる学生 © いらすとや

### 1.4 表 … table · tabular

#### コマンド

```
1 \begin{table}[H]
2   \center
3   \caption{当たりくじの本数と賞金額}
4   \begin{tabular}{|c|c|}
5     \hline
6     賞金 & 本数 \\ \hline
7     10000 & 5 \\ \hline
8     1000 & 20 \\ \hline
9     200 & 75 \\ \hline
10    0 & 900 \\ \hline
11   \end{tabular}
12 \end{table}
```

#### 実行結果

表 2: 当たりくじの本数と賞金額

賞金	本数
10000	5
1000	20
200	75
0	900

レポートではこちらを使っても良いかもしれません。

## コマンド

```
1 \begin{table}[H]
2   \center
3   \caption{当たりくじの本数と賞金額}
4   \begin{tabular}{c|c}
5     賞金 & 本数 \\ \hline
6     10000 & 5 \\ \hline
7     1000 & 20 \\ \hline
8     200 & 75 \\ \hline
9     0 & 900 \\ \hline
10    \end{tabular}
11 \end{table}
```

## 実行結果

表 3: 当たりくじの本数と賞金額

賞金	本数
10000	5
1000	20
200	75
0	900

## 1.5 改行 … \\\

\\で改行した後は\\quad で 1 文字分開けましょう。

## コマンド

```
1 \begin{center}
2   \section*{走れメロス}
3 \end{center}
4
5 メロスは激怒した。必ず、かの \ruby{邪智暴虐}{じゃちぼうぎやく} の王を除かなければならぬと決意した。メロスには政治がわからぬ。メロスは、村の牧人である。笛を吹き、羊と遊んで暮して来た。けれども邪悪に対しては、人一倍に敏感であった。\\
6 \quad きょう未明メロスは村を出発し、野を越え山越え、十里はなれた此このシラクスの市にやって来た。メロスには父も、母も無い。女房も無い。十六の、内気な妹と二人暮しだ。
```

## 実行結果

### 走れメロス

メロスは激怒した。必ず、かの じゃちぼうぎやく 邪智暴虐 の王を除かなければならぬと決意した。メロスには政治がわからぬ。メロスは、村の牧人である。笛を吹き、羊と遊んで暮して来た。けれども邪悪に対しては、人一倍に敏感であった。

きょう未明メロスは村を出発し、野を越え山越え、十里はなれた此このシラクスの市にやって来た。メロスには父も、母も無い。女房も無い。十六の、内気な妹と二人暮しだ。

## 1.6 ふりがなを降る … ruby

むずか 難しい漢字に かんじ ruby ルビ を振ろう。

## 注意

この命令を使うには\begin{document}より前に\usepackage{okumacro}を宣言し、パッケージを読み込む必要があります。

## コマンド

```
1
2 \begin{itemize}
```

```
3 \item \ruby{邪}{じゃ}\ruby{智}{ち}\ruby{暴}{ぼう}\ruby{虐}{ぎやく}
```

```

4 \item \ruby{老}{ろう}\ruby{若}{に
  やく}\ruby{男}{なん}\ruby{女}{に
  よ}
5 \item \ruby{弱}{じゃく}\ruby{肉}{
  にく}\ruby{強}{きょう}\ruby{食}{し
  よく}
6 \item \ruby{焼}{やき}\ruby{肉}{に
  く}\ruby{定}{てい}\ruby{食}{しょ
  く}
7 \end{itemize}

```

実行結果

- じゃ ち ぼうぎやく 邪智暴 虐
- ろうにやくなんによ 老 若 男女
- じゃくにくきょうしよく 弱 肉 強 食
- やきにくていしよく 焼肉定 食

難しい漢字に対してだけでなく、特別な読み方をさせるときにも使えるでしょう。

コマンド

```

1 \begin{quote}
2   囚われた\ruby{屈辱}{くつじょく}
   は 反撃の\ruby{嚙矢}{こうし}だ
   城壁の\ruby{其}{その}\ruby{彼
   方}{かなた} 獲物を\ruby{屠}{ほ
   ぶ}る $\dagger$ \ruby{狩人}{イ
   エーガー} $\dagger$
3 \end{quote}
4 \begin{flushright}
5   \small{Linked Horizon \
   copyright 紅蓮の弓矢}
6 \end{flushright}

```

実行結果

囚くつじょくわれた屈辱は 反撃こうしの嚙矢だ 城  
壁その其かなたの彼方 獲物ほぶを屠イェーガーる † 狩人 †

紅蓮の弓矢 ©Linked Horizon

## 1.7 引用 … quote · quotation

文章を引用するときは、どこからどこまでが引用であるか、どこからの引用であるかを明確にする必要があります。引用する文章が長文であるときは quotation を用いると良いでしょう。

余談

引用と参考は別物であるため注意が必要です。

???「これひょうせつ剽窃だよねえ！」

コマンド

```

1 このとき、ある研究熱心で学生にとっても好かれ
  ている研究者が言いました。
2 \begin{quote}
3   私が学生時代の頃は $\cdots$
4 \end{quote}

```

実行結果

このとき、ある研究熱心で学生にとっても好かれて  
いる研究者が言いました。

私が学生時代の頃は …



## 1.8 文字寄せ … flushleft · center · flushright

コマンド

```
1 \begin{flushleft}
2   左寄せ
3 \end{flushleft}
4 \begin{center}
5   中央寄せ
6 \end{center}
7 \begin{flushright}
8   右寄せ
9 \end{flushright}
```

実行結果

左寄せ

中央寄せ

右寄せ

## 1.9 文字サイズ・書式

使いたい箇所に{\コマンド 文字列}というフォーマットで使います。

(例)

```
1 文字の大きさを{\Huge 変えます}
```

実行結果

文字の大きさを **変えます**

(例 2)

```
1 \textdagger {\it 純黒の\ruby{悪夢}{Nightmare}} \textdagger
```

実行結果

† 純黒の *Nightmare* 悪夢 †

## 文字サイズ [1]

コマンド	サイズ	見え方
\tiny	5pt	日本語
\scriptsize	7pt	日本語
\footnotesize	8pt	日本語
\small	9pt	日本語
\normalsize	10pt (標準)	日本語
\large	12pt	日本語
\Large	14.4pt	日本語
\LARGE	17.28pt	日本語
\huge	20.74pt	日本語
\Huge	24.88pt	日本語

## 書体 [2]

コマンド	名称	見え方
\rm	ローマン体 (標準)	スヌーピー Snoopy
\bf	ボールド体 (太字)	スヌーピー <b>Snoopy</b>
\it	イタリック体 (強調)	スヌーピー <i>Snoopy</i>
\sf	サンセリフ体	スヌーピー Snoopy
\sl	斜体	スヌーピー <i>Snoopy</i>
\sc	スモールキャップス (全部大文字)	スヌーピー SNOOPY
\tt	タイプライタ体	スヌーピー Snoopy
\gt	ゴシック体 (日本語)	スヌーピー Snoopy
\mc	明朝体 (日本語)	スヌーピー Snoopy

## 付録 A 環境

僕が毎回使っているプリセットです。🐼BonyChops / latex-cheatsheet に sample/getstarted.tex として配布しています。

---

```
1 \documentclass[a4j, titlepage]{jarticle}
2 \usepackage{float}
3 \usepackage[dvipdfmx]{graphicx}
4 \usepackage{multicol}
5 \usepackage{okumacro}
6 \usepackage{amssymb}
7 \usepackage{amsmath}
8 \usepackage{url}
9 \usepackage{ascmac}
10 \usepackage{fancyvrb}
11 \usepackage{otf}
12 \usepackage{here}
13 %\usepackage{mediabb}
14 \makeatletter
15 %https://qiita.com/ta_b0_/items/2619d5927492edbb5b03
16 \usepackage[listings,jlisting] %日本語のコメントアウトをする場合jlstlistingが必要
17 \usepackage[top=30truemm,bottom=30truemm,left=25truemm,right=25truemm]{geometry}
18 %ここからソースコードの表示に関する設定
19 \lstset{
20     basicstyle={\ttfamily},
21     identifierstyle={\small},
22     commentstyle={\small\itshape},
23     keywordstyle={\small\bfseries},
24     ndkeywordstyle={\small},
25     stringstyle={\small\ttfamily},
26     frame={tb},
27     breaklines=true,
28     columns=[1]{fullflexible},
29     numbers=left,
30     xrightmargin=0zw,
31     xleftmargin=3zw,
32     numberstyle={\scriptsize},
33     stepnumber=1,
34     numbersep=1zw,
35     lineskip=-0.5ex
36 }
37 %ここまでソースコードの表示に関する設定
38
39 \begin{document}
40
41 ここにテキストをテスト...
42
43 \end{document}
```

---

## 付録 B sty ファイルをインストールする

自分でパッケージを追加する場合、.sty ファイルを用意してあげる必要があります。その際、.tex がおいてあるディレクトリと一緒に置いてあげればいいのですが、jlisting のように、ほぼ毎回使うパッケージを毎回置くのは面倒ですね。そこでこの章では PC に sty をインストールする方法をまとめておきます。ここに書いてあるものは、以下の Qiita 記事とほぼ同じような内容になります (記事よりはちょっとだけ丁寧に書いたつもり)。違う点としては、Windows 環境を想定して書いているところですかね。

LaTeX で sty ファイルをインストールする - Qiita

(<https://qiita.com/BonyChops/items/bfc30e06ab1b86ff782c>)

### 1. sty ファイルが置かれているディレクトリを見つける

地味に大変でした。環境によっては違うので注意が必要です。

{TeX の西暦}には、T<sub>E</sub>X のバージョンの西暦 (2017 など) が入ります。

#### ソースコード 1: Windows の場合

```
1 c:/texlive/{TeX の西暦}/texmf-dist/tex/latex
```

#### ソースコード 2: Ubuntu の場合

```
1 /usr/share/texlive/texmf-dist/tex/latex
```

### 2. それっぽいディレクトリを見つける / なければ更にディレクトリを作成する

latex ディレクトリには更にディレクトリがいっぱいあり、その中に sty ファイルがあります。できれば適切なディレクトリに入れてあげたいところですが、正直どのディレクトリに入れても動くので、そこまで気にしなくても良いかもしれません。

また、探してみても、該当するものがない場合は自分でディレクトリを作る必要があります。

- (a) 該当するディレクトリがある場合例えば、jlisting の場合、listings というディレクトリがあるので、そこへコマンドプロンプト / ターミナルでアクセスします。

```
1 cd c:/texlive/{TeX の西暦}/texmf-dist/tex/latex
2 cd listings
```

- (b) 該当するディレクトリがない / あるかどうかわからない場合例えば、siunitx の場合、該当するものがない (と思う) ので、自分で作ってあげる必要があります。siunitx というディレクトリを作ってあげましょう。Linux の場合、管理者権限が必要です。

```
1 cd c:/texlive/{TeX の西暦}/texmf-dist/tex/latex
2 mkdir siunitx
3 cd siunitx
```

### 3. sty ファイルをダウンロードする sty を置くディレクトリにカレントディレクトリがセットされている状態で、せっくなのでそのままダウンロードまでやっちゃいましょう。基本的に何を使っても良いですが、Windows (10 限定?) / Linux どちらでも使える curl で落としてみましょう。

```
1 curl -L -O {url}
```

#### ソースコード 3: siunitx の例

```
1 curl -L -O https://www.sys.kth.se/docs/texlive/texmf-dist/tex/latex/siunitx/siunitx.sty
```

ただし、jlisting の場合、おそらく .sty は直接配布されていないので、圧縮ファイルを自分で展開してからディレクトリに入れてあげる必要があります。コピー時にも Linux の場合は管理者権限が必要です。

#### 4. 変更を適用する


変更を適用して完了です。Linux の場合は管理者権限が必要です。

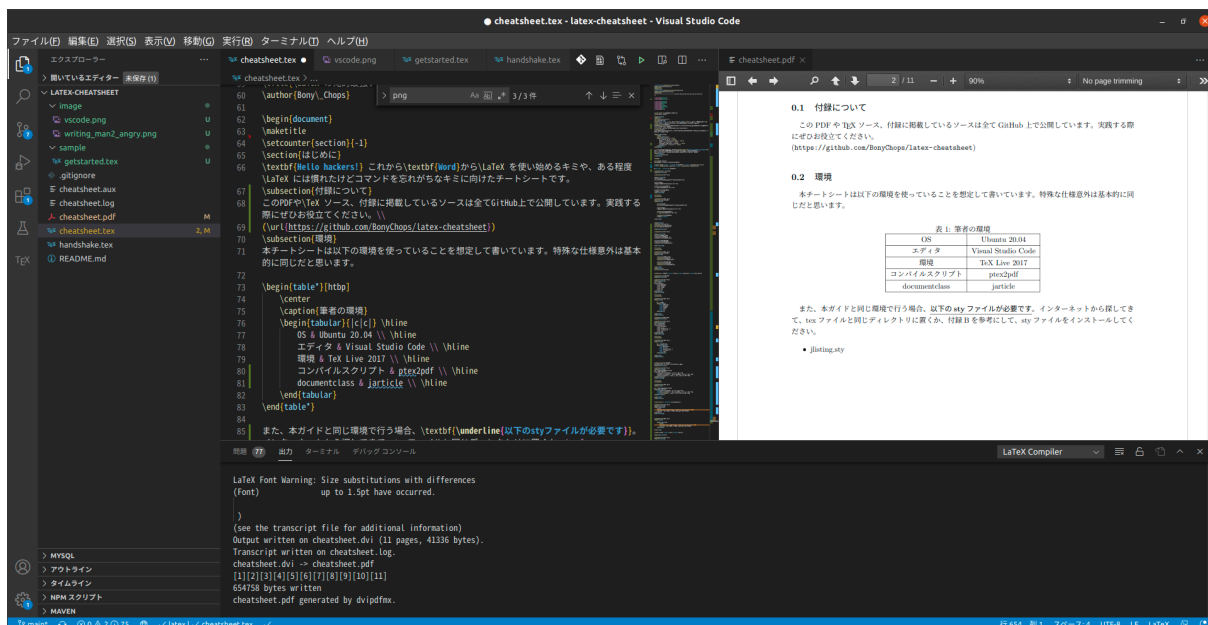
---

```
1 mktexlsr
```

---

## 付録 C VSCode で日本語 を扱う

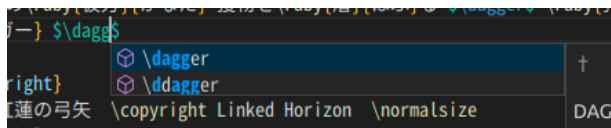
みなさん、エディタは何を使っていますか？ えっ、 $\text{\TeX}$  Works ですか…。 いえ、決して否定してはいませんが、もしテキストにインストールして勝手にはいってきたやつを使っているのであれば、もっと良いエディタもありますよ！ というご提案です。今回はその一つである  Visual Studio Code をご紹介します。



$\text{\TeX}$  Works と比較した時 …

### VSCode で書くメリット

- 複数のファイルを並べて置けるため、PDF と  $\text{\TeX}$  ファイルを見比べながらできる！
- コマンドを書く途中でサジェスト (候補) が出てくるためすぐ書ける！ 間違えない！



- サイドバーにエクスプローラがあるため複数ファイルがあっても把握しやすい！
- エラーが (比較的) わかりやすい！
- コンパイルしなくても、カーソルをのせると数式がプレビューされる！
- コマンドが色分けされて見やすい！

- Git が使えるから、レポートのバックアップも 5 秒以内！ (要ショートカット)
- Discord の Rich Presence で編集中のファイルを知ってもらえるから、レポートやってますよ自慢ができる！ (需要は俺)
- VSCode のショートカットが使える！

- 文字を選択した後、(, [, {, \$を入力すると、その記号 (とその対になる記号) で囲ってくれる！
- マルチカーソルで複数を一度に編集！
- Git 操作のショートカットを割り当てて、音速 Commit & Push！
- 自分でショートカットを細かく決められる！

- ダークテーマだから目に優しい！ (もちろんライトテーマもあるよ！好きなテーマを探そう)
- コンパイルが早くなるらしい！ (諸説あり)

## VSCode で書くデメリット

- $\text{\TeX}$  Works にある、「PDF のこの部分と  $\text{\TeX}$  ソースのここが対応してる」を表示してくれる機能がない
- 最初のセットアップが少々面倒かもしれない
- VSCode の PDF ビューワに、クリックしたところだけをズームする機能がない (探せばあるかも?)
- なんだかんだもう  $\text{\TeX}$  Works には戻れなくなるかもしれない

## セットアップ

1. まずは  $\text{\TeX}$  Live を通常通りインストールします ( $\text{\TeX}$  Works はあってもなくても OK)  
すでにインストールしてある場合はここを飛ばしてください。バージョンはおそらくいくつでも大丈夫だと思います (2016-2019 の動作確認済です)。インストール時に PATH (環境変数) を通すか聞かれた場合は、通すようお願いしておきましょう。聞かれなかったり、お願いし忘れた場合は自分でパスを通します。詳しくは次。
2.  $\text{\TeX}$  Live の PATH (環境変数) が通っているか確認する  
まず、コマンドプロンプト / ターミナルを開き `ptex2pdf` と入力し Enter。コマンドが見つからないと言われたら 2a を行う。
  - (a)  $\text{\TeX}$  Live の PATH (環境変数) を通す  
以下の PATH を通してください。通し方がわからない場合は、ここに書くと膨大になっちゃうので、こちらの記事をご覧ください。  
Windows 10 でのパスの通し方 - Qiita  
(<https://qiita.com/BonyChops/items/a56ea9d0194a9992fd16>)

### ソースコード 4: Windows の場合のパス

```
1 C:\texlive\{TeX の西暦}\bin\win32\
```

3. VSCode をインストールする  
Windows の場合、「VSCode で開く」を追加する等のオプションはできるだけつけておいたほうが良いでしょう。
4. VSCode に必要な拡張機能を入れておく  
VSCode を起動した後、左側のサイドバーにある拡張機能のアイコンをクリックすると、Marketplace にアクセスすることができます。ここで、VSCode スタータの皆様、俺的イチオシ拡張機能を列挙しておきます。正直何もわからない場合は以下を全部入れてもいいと思います ( は必須、それ以外は任意です)




LaTeX Workshop - サジェストやコンパイル、色々してくれる優れもの

Snippet Generator - この後作るユーザースニペットがめちゃくちゃ簡単に作れるようになります

- Japanese Language Pack for Visual Studio Code - VSCode が日本語になります
- Bracket Pair Colorizer 2 - 括弧の深度ごとに色分けされ、とにかくめちゃくちゃ見やすくなる
- Trailing Spaces - 行末に無駄なスペースが合った場合、そこが赤くなります
- zenkaku - 全角スペースを使っていた場合、そこが白くなります
- Discord Presence - Discord で自分の活動を自慢したい人向け
- Code Spell Checker - 英単語でスペルが間違っていた場合青色の波線で教えてくれる
- Settings Sync - Github のアカウントをリンクして、インストールした拡張機能やその設定をバックアップ。これで別環境への移行も簡単にできます
  - なんとそのうち VSCode が公式に設定のバックアップをできるようにするんだとか (しかも仕様はこの拡張機能とほとんど同じで、バックアップ先を Microsoft アカウントやGithub に指定可能) ! 興味がある人はこの拡張機能の代わりに公式のバックアップ機能を有効にしてもいいかもしれない (ただしまだプレビュー機能だそうです) .
- VS Code Counter - ワークスペース内のプロジェクトで、自分が何行書いたかを知れる

#### 5. 設定を構成する

Ctrl + Shift + P もしくは F1 でコマンドを入力します。"settings json" と入力し、基本設定: 設定 (JSON) を開くを選択し、bonychops/latex-cheetsheet の settings/setting.json の中身を貼り付けてください。

#### 6. ビルドしてみる

とりあえずこれで最低限の設定は完了したので、試しにビルドしてみましょう。tex のファイルを作成してみて、いろいろ書いて、Ctrl + Alt + B で正常にビルドできれば成功です。完成した PDF は Ctrl + Alt + V で確認できます。

## ユーザースニペットを使いこなす

早く書くために、スニペットを設定してみましょう。


(例)

\screen → Tab ↓

---

```
1 \begin{screen}
2
3 \end{screen}
```

---

1. 画面左下歯車をクリック
2. "ユーザー スニペット"
3. "新しい グローバル スニペット ファイル"
4. "latex"
5. bonychops/latex-cheetsheet の settings/snippet.json を貼り付ける



## ユーザースニペットを自作する

TeX は便利だけど、毎回プリセット (`\begin{document}` より上の部分) を書くのは大変 ... と思っている人必見！ さっきのスニペット機能を生かすと、毎回前回のレポートをコピーして ... という無駄な時間をなくし、あなたの大切な時間を守れます！

1. まず、設定したいプリセットを用意します。付録 A から丸パクリでも良いです
2. プリセットのソースファイルが選ばれている状態で `Ctrl + Shift + P` → "Generate snippet"
3. "latex"
4. name: 適当な名前
5. prefix: 呼び出す際のコマンドみたいなやつ
6. description: 特になし
7. 完成！
8. テストしてみる:  
    `Ctrl + A` で選んだ後全部消してみて、何も無い場所で prefix を打つと、登録したプリセットが現れます！

## 付録 D 日本語 トラブルシューティングガイド

### 参考文献

- [1] LaTeX コマンド集 - 文字サイズ <http://www.latex-cmd.com/style/size.html>
- [2] LaTeX コマンド集 - 書体 <http://www.latex-cmd.com/style/style.html>