最強の LaTeX 環境構築

yuma　　リュカ

2024 年 4 月 5 日

1 序文

1.1 編集方針

曖昧さを排除したできるだけ再現性の高い LaTeX [*1] [*2] 環境構築に対する処方箋を書きたいという思いでこの資料を書いた。しかし環境構築後にも難関が待ち受けている。そのために環境構築後の設定の変更等にも配慮して資料を工夫した。

- LaTeX を使えるようになりたい。さらに高性能エディターで書きたい。
- 自分がものすごく苦労した環境構築を 1 日でかつ理解しながら終わらせたい。
- さらに、その先を自分で学習できるようにしたい。
- 苦労した環境構築の記憶をとどめておいてまた困った時に見返したい。
- 大学 1 年生にも分かりやすく伝えたい

そこで初心者に LaTeX を使えるように、さらには自分好みにカスタマイズできるようになってもらうための資料とすることを編集の基本方針としたい。

この文章は、LaTeX を Visual Stadio Code (以下 VS Code) での環境構築のために自分がしたことのすべてを書いている。技術的解説もできるだけするように努めた。

これから VS Code 設定と、LaTeX 設定の両面のアプローチから環境構築設定を解説したいと思う。可能な限り初心者に分かりやすくそして自分が忘れないように書いたつもりである。この資料が LaTeX に入門する一助となれば筆者としても幸甚の極みである。

1.2 この資料の特徴

この資料の目標を以下に挙げる。

1. LuaLaTeX を用いて、VSCode 上で編集そして pdf の生成ができるような環境を設定すること。
2. さらに VSCode で LaTeX を高速にかつ便利に書くための環境を作ること。
3. LaTeX に関わる外部ツールを用いてより強力に LaTeX を編集できること。

この資料は、LaTeX の解説書ではない。VS Code で LaTeX 文書を編集するために必要な IT 知識と設定の方法を述べる。そのあと VS Code の基本機能そして LaTeX を便利に書くためのツールをできるだけ紹介する。美文書書作成入門 [2] は、LaTeX の機能の解説については大変充実した良書ではあるものの、環境構築についてはあまり触れていない。またほかのサイトなどを探しても、IT 知識が少しでもないとまったく分からない。初心者だと取り返しのつかなくなることがあるのではないかと考えて行動できなくなる。この状況を回避するためにこの資料を作った。

しかし、この資料にも至らぬ点が山ほどあると思われる。自分もすべてを理解しているわけではない。この資料の記述は自分がおこなってできたことだけを記述している。それ以外のことを知りたい場合は、付録の参

[*1] 日本では「ラテフ」と読むことが多い。海外では「レイテック」「ラテック」派がおおい。LaTeX の読み方については、[1] を参照すること。

[*2] LaTeX は Leslie Lamport によって開発された組版システムのことである。TeX よりもより簡単に記述できるようになっている。

考文献[*3]や LaTeX workshop の readme[6] を読むとよい。

1.2.1 設定用コード等について

設定ファイルの内容等、必要なソースコードはこの資料中に記載するとともに、コピー&ペーストの利便性も考えて markdown ファイルでも提供する。

1.3 この資料の読み方

この資料は、細かな説明などは意図せずに省かれている可能性があるため、製作者のもとでこの資料を使う形式を想定している。したがってそれ以外の場合での利用は自分で参考文献等を読まれることを強くオススメする。資料制作者の立ち合いのもとでの教授のあと使用するという形式でも十分使える資料になっている。

LaTeX の導入の後はこの資料がリファレンスとしても外部ツールや設定の一覧としても使えるように構成をしている。

注意として LaTeX 構文についての記述は必要最小限度にとどめている。したがって、より詳しい LaTeX の構文やマクロの詳しい組み方より発展的な扱い方等は [2][7] 等を参照する必要がある。

1.4 免責事項

この資料を使ったことによる損害等は、著者は一切の責任を負いません。

1.5 質問対応等について

この資料の範囲の質問は学術サーバー内の LaTeX サークルで行っています。チャンネル LaTeX サークルで質問をする際はフォーラム形式にて行ってください。

またこの資料の最新版は学術サーバー内 LaTeX サークル内および Git Hub 上で随時更新予定です。

また LaTeX に関する有用な知識やテクニック、有用なプリアンブル、外部ツールなどは是非 LaTeX サークル内でのプリアンブル投稿フォーラムに寄せてください。皆さんの協力によって大学内に LaTeX をより簡単により使えるようにそして広く普及させましょう。

この資料よりも発展的・高度な質問に関しては、[8] などを参照する必要がある。

[*3] [3],[4],[5] 等を参考にするとよい。このサイトから多くのことを学んだ。しかし最低限すぎるし何より必要十分な解説はなされていない。

目次

第 I 部
導入

2 LaTeX, VS Code とは何か？

2.1 LaTeX とは何か？

LaTeX とは、文書執筆ツールとして使われる、組版処理系の一つである。LaTeX を使う利点をいくつか挙げる。

- 数式がきれいに書ける
  - Word で書こうとするととんでもない数のクリックと精密なエイム力を要求される[*4]
- 見た目と論理構造を分離できる
- 修正、再利用が容易
- git でバージョン管理できる
  - 共同編集が可能
- 数学系、物理系では論文執筆のデファクトスタンダード
- 貧弱なスペックのパソコンでも編集作業がやりやすい
- 無料

VS Code の機能を使ってテンプレートを作っておけば簡単に体裁の整った文書を作成でき、内容のみに集中して執筆することが可能になる。数式交じりのレポートを作成するときなど、大学生 1 年生でも恩恵を受ける場面は多いはずである。

LaTeX と関わりの深い言葉として TeX があるが、この 2 つはまったく違うものであることを注意しておこう。[*5]たまに口頭でテフということもあるが、[*6]ユーザーが見える範囲で TeX があらわれることはほとんどなく、大方 LaTeX のことを指していると考えてよい。このあたりの話は美文書書作成入門 [2] に詳しい。

この資料では、ローカル (手元の PC) に LaTeX をインストールし、VS Code で編集ができる環境を整える。インストール不要な環境を提供するサービスに LaTeX works editer や cloud LaTeX というものもあるが、両者を使って比較してみると、カスタマイズ性の高いローカル環境の威力がわかると思う。

2.2 VS Code とは何か？

VS Code とは、Visual Stadio Code の略でマイクロソフト社が提供する統合開発環境である。要は高性能なエディターであり、後述する拡張機能の存在によって LaTeX との親和性も高い。[*7]VS Code は、

- 無料で頒布されている

[*4] これは決して Word を貶しているわけではない。そのように Word を使うことが間違っているのである。Word も LaTeX も同じ文書執筆ツールではあるが、それぞれに長所短所があり使うべき場所というものがある。

[*5] TeX はプログラム言語であり、LaTeX は TeX を用いて記述されたソフトウェアの集合体である。LaTeX は数多くのソフトウェアが組み合わさって一つの文書を出力する。

[*6] TeX は日本では「テフ」が多いが海外では「テック」と呼ばれることが多い。

[*7] テキストエディタは Windows 標準搭載のメモ帳などが有名である。また TeX 専用のテキストエディタとしては TeX Live にある TeXworks 等が有名である。この資料では今後のトレンドを考えて VSCode への導入を推奨する。

- 動作が軽い
- クロスプラットフォームである（さまざまな OS に対応している）
- 最新トレンド全部入り
- 拡張がしやすく、さまざまな機能がある

という特徴がある。

3 cloud LaTeX の使い方

美文書作成入門 [2] や一週間で LaTeX の基礎が学べる本 [7] 等を用いて、LaTeX にふれることでこの後の流れが少し分かりやすくなるかもしれない。そこで、環境構築せずとも LaTeX が使える環境として cloud LaTeX について解説する。

cloud LaTeX は、オンライン上で LaTeX を扱うことができる。環境構築をすることなくすぐに LaTeX 文章を書くことができるという点は、LaTeX の記法に慣れるという面では、有用である。したがってこの章では、cloud LaTeX の使い方について簡単に説明して LaTeX に慣れてもらいたい。初めに `https://cloudlatex.io/ja`[9] のサイトに行きユーザー名やメールアドレスと任意のパスワードを設定する。すると設定したメールアドレスにメールが来るので、メールアドレスの受信確認をクリックする。アカウント登録が完了する。マイページから、新規プロジェクトの追加を選んで LaTeX 文書を書き始めることができる。cloud LaTeX は VS Code と同等の機能があるので最初に使うにはとてもいい教材である。データはすべて cloud LaTeX のサーバ上に保存されるのでデータが消える心配はほとんどない。また、共同作業も可能である。しかし、共同作業をより効率的に行いたい場合は、Overleaf のほうが優れている可能性もある。Overleaf は GitHub と連携することができるために論文執筆の際に有効となる。どちらがいいかは個人によるが、すぐに始めるならば cloud LaTeX をオススメする。

第 II 部
環境構築

4 ローカル環境構築の基本用語

4.1 LaTeX 環境設定用 IT 用語の理解

環境

LaTeX が機能するための設定やハードソフト等の存在や設定などを合わせて環境という。

ローカル

オンライン上ではなくパソコンのこと。ローカル環境とは自分のパソコンの環境のこと。

リモート

(ネットワークを介して接続できる) 離れたところにある

ユーザー

自分が設定した名前 自分のことユーザーの名前のフォルダーが最上位にあると考えておけばよい。[*8]

デフォルト

初期設定のこと。

拡張子

ファイルの種別を示す。

隠しファイル

エクスプローラー（よくファイルを使う場所）からは見えないファイル。たとえば、.latexrc 等の. の前に何もついていないファイルのこと。開き方 エクスプローラーを開いて上にある表示タブを開いて表示タブにする。表示タブから隠しファイルを開いて、チェックマークをつける。これにより、今まで見えなかったファイルが見えるようになる。

GUI

GUI とは “Graphical User Interface（グラフィカルユーザーインターフェース）” の略であり、ユーザーの使いやすさを重視し、アイコンやボタンなどを用いて直感的にわかりやすくコンピューターに指令を出せるようにしたユーザーインターフェース。ユーザーインターフェースとは、ユーザーとコンピューター間での情報のやり取りの方法、仲立ちするものである。GUI ではキーボードに加えてマウスなどのポインティングデバイスやタッチパネルによって視覚的に操作できる。Word や PowerPoint などがこれに当たる。

CUI

CUI とは “Character User Interface（キャラクターユーザーインターフェース）” の略である。あらかじめ決まっている文字や命令を コマンドといい、コマンドを入力してコンピュータに指示をだすユーザーインターフェース。CUI では主にキーボードによるコマンドの文字情報のみで操作する。Windows ではコマンドプロンプトがこれに当たる。

ディストリビューション

パッケージの仕方・配布形態の異なるソフトウェアパッケージのことであり、TeXlive は日本で最も使われている LaTeX ディストリビューションである。

ディレクトリ

ファイルが入っている階層を指し示す言葉。階層のことであり、日本語の意味では住所録の意味。ドライブ直下に置くとは、最上部のフォルダーの中に入れることを意味する。ファイルの現在位置を指し示す言葉で、ルートディレクトリとは、

ディレクトリとフォルダは、

- GUI で操作するときはフォルダーという。
- CUI で操作するときはディレクトリと呼ぶ

という違いがあるのみであり指している対象としては同一のものである。

ルートディレクトリ　フォルダーの階層の最上位のフォルダーを意味する。

ホームディレクトリ　ユーザーフォルダーのこと。

カレントディレクトリ　現在自分がいるフォルダーのこと。

ここで注意! LaTeX や、その他のプログラミング言語は日本語つまり全角のファイル名は、探すことが

[*8] Windows はユーザーアカウントを用いて、PC を使用するユーザー (利用者) を識別しています。また、ユーザーごとにホームディレクトリが割り当てられているため、ユーザーごとの設定に利用されることがあります。

できない。エラーを出す。このため、すべてのファイル名や、パソコンでの設定した名前は、かならず半角英数字にすることを覚えておく必要がある。全角文字だけでなく半角スペースもエラーの原因となりうることを注意しておく。代替案として英語表記する、スペースを表現する方法として、スペースはアンダースコア (_) あるいはハイフン (-) に置き換える、などの方法がある。

パス

ディレクトリでそのファイルの住所位置を指定するもの。

グローバル

すべてのユーザーやそのアプリケーション内ですべてに設定されるような設定のこと。

コマンドプロンプト

Windows の操作をコマンドでおこなうためのシステム。

開き方 1

コマンドプロンプトをスタートメニュー（Windows アイコンのタブ）の中の検索窓からコマンドプロンプトと打ち、マウスで出てきたコマンドプロンプトをクリックする。

開き方 2

Windows キー +R で「ファイル名を指定して実行」という窓が出てくるのでその指定する窓に Cmd と打ち Enter を打つことで、コマンドプロンプトを実行できる。

開き方 3

エクスプローラーのパスの入力欄に `cmd` と入力して Enter を押すことでもコマンドプロンプトを開くことができる。

リポジトリ

プロジェクトを構成するプログラムのソースコードやドキュメント、関連する各種のデータやファイルなどを一元的に管理する格納場所のこと

4.2 LaTeX 用語の理解

ビルド

LaTeX では、記述したソースコードに問題がないかどうかの解析を行い、問題がなければ実行可能なファイルに変換すること。

コンパイラ

コンパイラとは機械が読み取れる言語に移すことのできるソフトウェア等のこと。

コンパイル

ソースコードを機械が翻訳できる言語に移すことで、ビルドの中の一連の作業に入っている。

プリアンブル

`\documentclass` から`\begin{document}`までの中にある設定のこと。

5 エクスプローラーの設定

拡張子と隠しファイルを表示するように設定する。

1. エクスプローラーを起動する。

2. 拡張子を表示したいフォルダを開き、メニューの「表示」タブをクリックする。
3. さらに [表示] を選択し、「ファイル名拡張子」「隠しファイル」にチェックを入れる。

ファイル名の後ろに.txt 等のファイルの種類を示す文字列が表示されるようになり、今まで見えなかったファイルが見えるようになる。

6 TeX Live 導入

6.1 TeX Live インストール

TeX Live は、次のページからダウンロードすると良い。[*9]

`https://www.tug.org/texlive/acquire-netinstall.html`

をクリックして、ページ上のリンク install-tl-windows.exe をクリックする。この時、警告が出るが無視して大丈夫。そして Next を押し続けて install を押す。これで、LaTeXpackage のほとんどすべてをダウンロードすることが可能。フルパッケージをダウンロードすると 3 時間程度かかる。

6.2 インストールを最短で終わらせる方法（最小構成でのインストール）

環境構築を早く行いたい場合や、PC にメモリがない場合には環境構築の容量を大きく制限する必要がある。インターネット回線が非常に遅い場合でもインストールを最短で終わらせる方法について記述する。

6.2.1 TeX Live インストーラーのインストール

TeX Live インストーラーをインストールすることがまず始めにある。次に、TeX Live インストーラーで**高度な設定**をクリックする。

次に TeX Live インストーラーで**高度な設定**をクリックする。ここで最小のパッケージでインストールするために次のことに気を付ける。

- 「ディレクトリ」は無変更
- 「選択したもの」欄で変更２つ行う。
  - スキーム (インストールするパッケージの種類を大まかに決める) を変更する。basic スキームを選択。
  - 下のカスタマイズに入り、言語欄から日本語と英語を追加。右の他のコレクション欄から ‘LaTeX 推奨パッケージ’ を追加。

これが終われば右下のインストールをクリックする。

これをすれると一時間から一時間半程度でインストールすることができる。[*10]

[*9] LaTeX はたくさんのソフトウェアからなるシステムであるため、配布用に必要なソフトウェア群をパッケージしたディストリビューションの形でインストールする。今回インストールするのは TeX Live というディストリビューションである。

[*10] これらのインストール時間は目安であり通常は回線の速さによって変化する。

6.3 インストールが正常に行われたことを確認する

インストール完了後、コマンドプロンプトを起動して、`latex -v` と打ち込む。この時に、3.141592653 のような version 情報が出てこれば正常に完了している。[*11]

7 Visual Stadio Code の導入

7.1 Visual Stadio Code のインストール

Visual Studio Code は次のページからダウンロードするとよい。

```
https://code.visualstudio.com/download
```

をクリックしてそのあとの画面の中から今回の場合は、自分の環境に合わせて (Windows か Mac の) ファイルをダウンロードする。ファイルを開き、追加タスクの変更は触らないようにして Next を押し続ければインストールできる。

7.2 VS code の日本語化

VS Code をショートカットから開いて、積み木のようなアイコン Extentions（拡張機能）をクリック。検索窓に Japanese と打って、Japanese Language Pack for Visual Studio Code を選択してインストールする。これで日本語化が完了する。このように拡張機能を入れることで VS Code の機能をより使いやすいものにすることができる。

8 latexmk の導入

8.1 latexmk とは

LaTeX 文書を手動でコンパイルしようとすると複数種類のコマンドを適切な順番で複数回実行しなければならない。latexmk とは、各種 LaTeX のビルドコマンドを自動で一括で実行してくれるツールである。

8.2 latexmk のインストール

TeX の導入で最小構成を選択した人は最初から latexmk は入っていないので TeX Live Manager からダウンロードする必要がある。

TeX Live Manager をスタート画面から検索して開く。TeX Live Manager を起動して待機中になったら、検索欄から ‘latexmk’ と検索する。すると、しばし待機した後に二つの ‘latexmk’ が出てくる。

- latexmk
- latexmk.win32

の二つが出てくるので、これらをチェックボックスをクリックして選択項目をインストール を選択し latexmk

[*11] これは、TeX 製作者 Donald E.Knuth 教授の意向で version up ごとに円周率 π へ近づいて行く。

を選択してインストールする。これで latexmk のインストールが完了する。

今回の環境構築では LuaLaTeX を用いるので lualatex-math というパッケージをダウンロードする必要がある。フルパッケージで TeX Live をインストールした場合には入っている。また、TeX Live Manager ですでに入っていると表示される場合は問題ない。[10] 参照

8.3 latexmk の設定

ホームディレクトリ[*12] 直下に.latexmkrc というテキストファイルを作成し、テキストエディタで開く。ここでテキストエディタは VS Code を使うこと。VS Code で UTF-8 [*13]にしなければならない。markdown ファイルに記載された latexmk の設定をコピーして貼り付け、保存する。同じ内容をソースコード 1 に示す。

ソースコード 1 latexmk の設定

```
1  # 通常の LaTeX ドキュメントのビルドコマンド
2  $latex = 'uplatex %O -kanji=utf8 -no-guess-input-enc -synctex=1 -interaction=
       nonstopmode %S';
3  # pdfLaTeX のビルドコマンド
4  $pdflatex = 'pdflatex %O -synctex=1 -interaction=nonstopmode %S';
5  # LuaLaTeX のビルドコマンド
6  $lualatex = 'lualatex %O -synctex=1 -interaction=nonstopmode %S';
7  # XeLaTeX のビルドコマンド
8  $xelatex = 'xelatex %O -no-pdf -synctex=1 -shell-escape -interaction=
       nonstopmode %S';
9  # Biber, BibTeX のビルドコマンド
10 $biber = 'biber %O --bblencoding=utf8 -u -U --output_safechars %B';
11 $bibtex = 'pbibtex %O %B';
12 # makeindex のビルドコマンド
13 $makeindex = 'upmendex %O -o %D %S';
14 # dvipdf のビルドコマンド
15 $dvipdf = 'dvipdfmx %O -o %D %S';
16 # dvipd のビルドコマンド
17 $dvips = 'dvips %O -z -f %S | convbkmk -u > %D';
18 $ps2pdf = 'ps2pdf.exe %O %S %D';
19
20 # PDF の作成方法を指定するオプション
21 ## $pdf_mode = 0; PDF を作成しない。
22 ## $pdf_mode = 1; $pdflatex を利用して PDF を作成。
23 ## $pdf_mode = 2; $ps2pdf を利用して .ps ファイルから PDF を作成。
24 ## pdf_mode = 3; $dvipdf を利用して .dvi ファイルから PDF を作成。
25 ## $pdf_mode = 4; $lualatex を利用して .dvi ファイルから PDF を作成。
26 ## $pdf_mode = 5; xdvipdfmx を利用して .xdv ファイルから PDF を作成。
27 $pdf_mode = 4;
```

[*12] PC>ローカルディスク>ユーザー>ユーザー名、と辿れるディレクトリのこと

[*13] UTF-8 とは Unicode 用の符号化方式の一つである。UTF-16 もある。VS Code ではデフォルト設定として UTF-8 となっている。しかしほかのエディタ (例えばメモ帳等) ではデフォルトで UTF-8 になっていない可能性がある。他エディタで latexmk をつくり、VS Code で開いた際に UTF-8 以外のときにはステータスバーから UTF-8 に直す必要がある。[11]

```
28
29  # PDF viewer の設定
30  if ($^O eq 'MSWin32') {
31    $pdf_previewer = "start %S"; # "start %S": .pdf に関連付けられた既存のソフトウェアで表示する。
32  } else {
33    $pdf_previewer = "open %S";
34  }
35
36  ## Windows では SyncTeX(PDF をビューアーで開いたまま中身の更新が可能で更新がビューアーで反映される機能）が利用できる SumatraPDF 等が便利。
37  ## ぜひ SyncTeX 機能のあるビューアーをインストールしよう。
38  ## SumatraPDF: https://www.sumatrapdfreader.org/free-pdf-reader.html
39  ## $pdf_previewer = 'SumatraPDF -reuse-instance';
```

8.4 latexmk の設定の解説

9 LaTeX workshop の設定

VS Code の設定は、setting.json [*14]というファイルに記載されている。setting.json を開く方法はいくつかある。

1 つ目

VS Code 左下の設定マーク（歯車マーク）をクリックして、「設定」を選択する。右上端にあるファイルに矢印がついたアイコンをクリックする。

2 つ目

キーボードのショートカットキーを用いて、ctrl+shift+P と入力することで、コマンドパレットを出現させてそこに Preferences:Open User Setting と打ち込む。

3 つ目

Ctrl+ , で設定画面を開く。さらに右上端にあるファイルに矢印がついたアイコンをクリックする。

の 2 つの方法があり熟練したら 2 番目の方法の方が効率が良い。

setting.json を開いたら、{}の中に markdown ファイルに記載された LaTeX workshop の設定をコピーして書き加え、保存する。同じ内容をソースコード 2 に示す。もし開いた時点で{}以外に何か書き込まれていた場合、{}の最後の要素にコンマをつけて、設定をコピーして書き加え、保存する。

ソースコード 2 LaTeX workshop の設定

```
1  {
2  // LaTeXの設定
3  "editor.wordSeparators": "./\\()\"'-:,.;<>~!@#$%^&*|+=[]{}`~?　、。「」【】『』（）！？ てにをはがのともへでや",
4
```

[*14] .json はジェイソンと読む

```
5   // 設定: LaTeX Workshop
6
7   // LaTeX Workshop ではビルド設定を「Tool」と「Recipe」という2つで考える
8   // Tool: 実行される1つのコマンド。コマンド（command）と引数（args）で構成される
9   // Recipe: Tool の組み合わわせを定義する。Tool の組み合わせ（tools）で構成される。
10  // tools の中で利用される Tool は "latex-workshop.latex.tools" で定義されている必要
        がある。
11
12  // latex-workshop.latex.tools: Tool の定義
13  "latex-workshop.latex.tools": [
14    // latexmk を利用した lualatex によるビルドコマンド
15    {
16      "name": "Latexmk (LuaLaTeX)",
17      "command": "latexmk",
18      "args": [
19        "-f", "-gg", "-lualatex", "-synctex=1", "-interaction=nonstopmode", "-
              file-line-error", "%DOC%"
20      ]
21    },
22    // latexmk を利用した xelatex によるビルドコマンド
23    {
24      "name": "Latexmk (XeLaTeX)",
25      "command": "latexmk",
26      "args": [
27        "-f", "-gg", "-xelatex", "-synctex=1", "-interaction=nonstopmode", "-file
              -line-error", "%DOC%"
28      ]
29    },
30    // latexmk を利用した uplatex によるビルドコマンド
31    {
32      "name": "Latexmk (upLaTeX)",
33      "command": "latexmk",
34      "args": [
35        "-f", "-gg", "-synctex=1", "-interaction=nonstopmode", "-file-line-error
              ", "%DOC%"
36      ]
37    },
38    // latexmk を利用した platex によるビルドコマンド
39    // 古い LaTeX のテンプレートを使いまわしている（ドキュメントクラスが jreport や
          jsreport ）場合のため
40    {
41      "name": "Latexmk (pLaTeX)",
42      "command": "latexmk",
43      "args": [
44        "-f", "-gg", "-pv", "-latex='platex'", "-latexoption='-kanji=utf8 -no-
              guess-input-env'", "-synctex=1", "-interaction=nonstopmode", "-file-
```

```
            line-error", "%DOC%"
      ]
    },

    {
      "name": "pLaTeX",
      "command": "platex",
      "args": [
        "%DOC%","-file-line-error","-interaction=nonstopmode","-halt-on-error"
      ],
      "env": {}
    },
    {
      "name": "dvipdfmx",
      "command": "dvipdfmx",
      "args": [
        "-V 4",
        "%DOC%"
      ]
    },
    {
      "name": "Biber",
      "command": "biber",
      "args": [
        "%DOCFILE%"
      ]
    },
],

  // latex-workshop.latex.recipes: Recipe の定義
"latex-workshop.latex.recipes": [
      // LuaLaTeX で書かれた文書のビルドレシピ
      {
        "name": "LuaLaTeX",
        "tools": [
          "Latexmk (LuaLaTeX)"
        ]
      },
      // XeLaTeX で書かれた文書のビルドレシピ
      {

        "name": "XeLaTeX",
        "tools": [
          "Latexmk (XeLaTeX)"
        ]
      },
```

```
90      // LaTeX(upLaTeX) で書かれた文書のビルドレシピ
91      {
92        "name": "upLaTeX",
93        "tools": [
94          "Latexmk (upLaTeX)"
95        ]
96      },
97      // LaTeX(pLaTeX) で書かれた文書のビルドレシピ
98      {
99        "name": "pLaTeX + dvipdfmx",
100       "tools": [
101         "pLaTeX", // 相互参照のために2回コンパイルする
102         "pLaTeX",
103         "dvipdfmx"
104       ]
105     },
106     {
107       "name": "pLaTeX + Biber + dvipdfmx",
108       "tools": [
109         "pLaTeX",
110         "Biber",
111         "pLaTeX",
112         "pLaTeX",
113         "dvipdfmx"
114       ]
115     },
116 ],
117
118 // latex-workshop.latex.magic.args: マジックコメント付きの LaTeX ドキュメントをビルドする設定
119 // '%!TEX' で始まる行はマジックコメントと呼ばれ、LaTeX のビルド時にビルドプログラムに解釈され、
120 // プログラムの挙動を制御する事ができる。
121 // 参考リンク: https://blog.miz-ar.info/2016/11/magic-comments-in-tex/
122 "latex-workshop.latex.magic.args": [
123   "-f", "-gg", "-pv", "-synctex=1", "-interaction=nonstopmode", "-file-line-error", "%DOC%"
124 ],
125
126 // latex-workshop.latex.clean.fileTypes: クリーンアップ時に削除されるファイルの拡張子
127 // LaTeX 文書はビルド時に一時ファイルとしていくつかのファイルを生成するが、最終的に必要となるのは
128 // PDF ファイルのみである場合などが多い。また、LaTeX のビルド時に失敗した場合、失敗時に生成された
129 // 一時ファイルの影響で、修正後のビルドに失敗してしまう事がよくある。そのため、一時的な
```

```
        ファイルを
130     // 削除する機能（クリーンアップ）が LaTeX Workshop には備わっている。
131     "latex-workshop.latex.clean.fileTypes": [
132         "*.aux", "*.bbl", "*.blg", "*.idx", "*.ind", "*.lof", "*.lot", "*.out", "*.
                toc",
133          "*.acn", "*.acr", "*.alg", "*.glg", "*.glo", "*.gls", "*.ist", "*.fls",
                 "*.log",
134          "*.fdb_latexmk",
135         // for Beamer files
136         "_minted*", "*.nav", "*.snm", "*.vrb",
137     ],
138
139     // latex-workshop.latex.autoClean.run: ビルド失敗時に一時ファイルのクリーンアップを
            行うかどうか
140     // 上記説明にもあったように、ビルド失敗時に生成された一時ファイルが悪影響を及ぼす事がある
            ため、自動で
141     // クリーンアップがかかるようにしておく。
142
143     "latex-workshop.latex.autoClean.run": "onBuilt",
144
145     // latex-workshop.view.pdf.viewer: PDF ビューアの開き方
146     // VSCode 自体には PDF ファイルを閲覧する機能が備わっていないが、
147     // LaTeX Workshop にはその機能が備わっている。
148     // "tab" オプションを指定すると、今開いているエディタを左右に分割し、右側に生成された
            PDFを表示するようにしてくれる
149     // この PDF ビュアーは LaTeX のビルドによって更新されると同期して内容を更新してくれる。
150     "latex-workshop.view.pdf.viewer": "tab",
151
152     // latex-workshop.latex.autoBuild.run: .tex ファイルの保存時に自動的にビルドを行う
            かどうか
153     // LaTeX ファイルは .tex ファイルを変更後にビルドしないと、PDF ファイル上に変更結果が
            反映されないため、
154     // .tex ファイルの保存と同時に自動的にビルドを実行する設定があるが、文書が大きくなるに連
            れてビルドにも
155     // 時間がかかってしまい、ビルドプログラムの負荷がエディタに影響するため、無効化しておく。
156     "latex-workshop.latex.autoBuild.run": "never",
157
158     "[tex]": {
159         // スニペット補完中にも補完を使えるようにする
160         "editor.suggest.snippetsPreventQuickSuggestions": false,
161         // インデント幅を2にする
162         "editor.tabSize": 2
163     },
164
165     "[latex]": {
166         // スニペット補完中にも補完を使えるようにする
```

```
167       "editor.suggest.snippetsPreventQuickSuggestions": false,
168       // インデント幅を2にする
169       "editor.tabSize": 2
170
171   },
172
173
174   "[bibtex]": {
175       // インデント幅を2にする
176       "editor.tabSize": 2
177   },
178
179
180
181   // ---------- LaTeX Workshop ----------
182
183   // 使用パッケージのコマンドや環境の補完を有効にする
184   "latex-workshop.intellisense.package.enabled": true,
185   "window.zoomLevel": -1,
186   "catex.greek-completion": {
187
188     "name": "CaTeX Greek Completion",
189     "languages": [
190       "latex"
191     ],
192     "triggers": [
193       ":"
194     ],
195     "dictionary": "$GIM/defaults/greeks.json",
196     "renderMode": "latex"
197   },
198
199   // 、を,に変えることができる設定latexindent.yamlを用いて設定。
200
201 }
```

9.1 自動ビルド機能の設定

```
"files.autoSave": "afterDelay",
  "latex-workshop.latex.autoBuild.run": "never",
```

は、自動ビルド機能である。しかし、自動ビルドは、このファイルのように大きすぎると非常に重くなってしまう。LaTeX は、コンパイルと、書くことを分離することで、高度なレイアウトの最適化を可能にしている。これは never にしておくことをオススメする。

10　ローカル環境でビルドする

ここまでの内容が行えていれば、十分に高機能な環境構築ができているはずである。ローカル環境でビルドできることを確認してみよう。ソースコード 3 の内容の.tex ファイルを作成し、VScode 上でビルドする。

ソースコード 3　LuaLaTeX のソースコード

```
1  \documentclass{ltjsarticle}
2  % ltjsarticle: lualatex 用の 日本語 documentclass
3  % LuaLaTeX以外のタイプセットエンジンを使ってビルドする場合は、 \documentclass[dvipdfmx
       ]{jsarticle} などのように適当に書き換える。
4
5  \begin{document}
6
7  \title{はじめての\LaTeX }
8  \author{Meidai}
9  \maketitle
10 \section{はじめての\LaTeX Lua\LaTeX }
11 %\section{はじめての\LaTeX Lua\TeX}
12
13
14 \subsection{小見出し！ }
15 Hello world!
16 今日は\LaTeX を覚えていってください。
17 \LaTeX + VSCode は最強の組み合わせ。
18
19 \end{document}
```

第 III 部
LaTeX 解説

11　TeX live

11.1　TeX live とは？

TeX live とは、TeX の国際的なディストリビューションである。LaTeX や TeX のパッケージを入手することができる。

そして、LaTeX のパッケージを一括で管理することができる機能が TeX Live に標準で実装されている TeX Live Manager である。

11.2　TeX live Manager の使い方

TeX Live Manager はスタートメニューから TeX live Manager と検索すると開くことができる。

TeX live Manager を開くと現在 TeX live のパッケージのうちでローカルにインストール済みのものが表示される。状態を「未インストール」に変更すると、自分がローカルにインストールしていないパッケージが表示される。検索欄で必要なパッケージを検索して選択、「選択項目をインストール」と操作することでインストールができる。

LaTeX 導入の段階で「高度な設定」を用いてインストールした場合には必要最小限のパッケージしかインストールされていないのでここからインストール必要がある。

12 LaTeX の基本機能

LaTeX の特徴の一つに文書の論理的構造と視覚的なレイアウトを分けて考えることができることがある。例えば、「はじめに」という名前の節があれば文書に`\section{はじめに}`と書いておく。この文書をコンパイルするに際し、`\section{...}`という命令がどのような見た目になるかはスタイルファイルに記述されており、それを元に目的のファイルが生成される。LaTeX はこのような構成をしているため文章を書くときはデザインのことを考えずに済むし、デザインを変えたいときはスタイルファイルを変更するだけで済むのである。

我々が LaTeX で文書を作る時には、この利点を最大限に享受するため、直接見た目を指示するような命令はなるべく使わないようにするべきだろう。

12.1 コンパイル

- ctrl+Alt+B のショートカットキーでおこなう。
- VS Code 右上の緑色の三角マークをクリックして行う。
- 左のタブの Build LaTeX project の中の Recipe: からおこなう。

ただし、今回の設定では、上の 2 つの方法でコンパイルすると LuaLaTeX で出力される。このために、pLaTeX などを使いたい場合は、左のタブからおこなう必要がある。

13 エラーの解決方法

LaTeX で文書を執筆する上で、数々のエラーに遭遇することになるだろう。赤い文字に怒られたように感じるかもしれないが、それは勘違いである。エラーメッセージは、修正するヒントを与えてくれている心強い味方であることを忘れてはならない。

エラーが起きた時の典型的な解決手順を順番に解説する。

13.1 エラーメッセージの表示方法

VSCode でのビルド時にエラーが起きた場合、一度 VScode はビルドを再試行する。この際右下に Recipe terminated with error. Retry building the project. というポップアップが表示される。2 回目のビルドでもエラーが起きた場合、Recipe terminated with error. というポップアップが表示される。まずは Open compiler log ボタンをクリックしてエラーメッセージを表示しよう。

13.2　原因の切り分け

エラーメッセージが出ていることを確認したら、エラーが出ていそうな部分をコメントアウトしてからビルドするなどして、エラーが出ていない部分を確定させる。こうすることで、見当違いの場所に修正を施してしまうといった事態を避けることができる。

13.3　エラーメッセージの読み方

原因となる部分が分かったら、エラーメッセージを読む。エラーメッセージの中に

```
1 c:/<path>:xxx: Undefined control sequence.
2 l.xxx ...<エラー直前のソース>
3                                 <エラー位置のソース
```

というような内容が書かれた部分があるはずだ。そのうち一番最初に出てくるものの解決にとりかかろう。

まずはエラーの種類をよく読もう。英語だからと読み飛ばしてはいけない。エラーメッセージにはエラーが起こった個所と原因が書いてある。この場合は xxx 行目で Undefined control sequence, すなわち未定義のコマンドが使われたことが原因のようだ。原因となったコマンドをを適切なコマンドに置き換えることでエラーは解決するだろう。

原因からすぐに解決方法が分かる場合は、その方法を実行しよう。そうでない場合はエラーメッセージをそのままコピーして検索すると、大体は偉大な先人が残してくれた解決策が見つかる。

13.4　よくあるエラー

Undefined control sequence.

コントロールシークエンス、つまりコマンドが定義されていないというエラーである。コマンドのスペルミス、あるいはコマンドの後のスペースの入れ忘れを疑うべきである。

Paragraph ended before \align was complete.

align 環境は空行を入れるとエラーが起きる。

File ended while scanning use of \@xdblarg.

`\@xdblarg` の内容を読んでいる間にファイルが終わってしまったという趣旨のエラーである。解決方法の推測が難しいが、かっこの閉じ忘れでこのエラーが起きる。

13.5　それでも解決しなかったら

LaTeX の基本的な機能を使っているうちは大抵は上の方法で解決する。もしそれでも解決しない場合は、マニュアルを参照することを検討しよう。

特にパッケージの使い方に関してはマニュアルが一番信頼できる情報源である。

13.5.1 LaTeX のマニュアル

texdoc は TeXLive に含まれるドキュメントを検索できるコマンドである。例えば texdoc について調べたい場合は、ターミナルで texdoc texdoc と実行すると texdoc の使い方が書かれた pdf ファイルが開く。

14 新規 package の導入

新規パッケージつまり、インターネット上や、自分で作ったスタイルファイル等を使いたいときに、めんどくさいことになる。パッケージファイルの拡張子は ".sty" であり、このパッケージを適切な場所に配置しなければならない。しかし、".sty" が入っていない場合がある。これに対処するためには次セクション

14.1 ".sty" ファイルが入っていない場合

".ins" ファイルが重要になる。このファイル形式は、パッケージ本体と ".ins" ファイルが入っている.dtx を LaTeX で.dtx ファイルを実行する必要がある。
たとえば tools.ins というファイルがあった場合には、.dtx ファイルを同じディレクトリに保存する。次にコマンドプロンプトを起動して、以下のように、入力する。

```
latex tools.ins
```

すると、同じフォルダー内に.sty ファイルが生成される。この.sty ファイルの保存場所は、次セクション参照。

14.2 TeX Directory Structure とは何か？

LaTeX のパッケージなどは、TDS（TeX Directory Structure）にしたがって各ディレクトリに配置されている。今回は TeX Live なので、エクスプローラーから windows フォルダーの中の texlive フォルダーの中の 2022 年フォルダーの中の texmf-dist の中の tex の中の latex フォルダーの中に各パッケージのフォルダーがあり、その中に各スタイルファイルフォルダーが保存されている。といった構成になっている。これは非常に見にくいし、読みにくいので、ここからは、これを

```
C:\texlive\2022\texmf-dist\tex\latex
```

のように書くと簡便になる。そしてこれが Windows のディレクトリの記述の仕方である。この中の tex フォルダーにはたくさんのフォルダーがあるが、最初に覚えておくべきは 2 つで、

folder 名	意味
tex	TeX の操作に関するフォルダーで、latex フォルダーがある。
font	フォントにかかわるフォルダー。

であり、このことから、.sty ファイルの保存場所は、

```
C:\texlive\2022\texmf-dist\tex\latex
```

よりも下のファイルに加えれば良いことがわかった。フォントファイルについては、以下参照。https://texwiki.texjp.org/?TeX%20%E3%81%AE%E3%83%87%E3%82%A3%E3%83%AC%E3%82%AF%E3%83%88%E3%8

3%AA%E6%A7%8B%E6%88%90

14.3 パッケージを使うための一覧表の更新

TeX Live 環境では、上のように.sty ファイルを保存しても動かない。これは、ls-R というファイルが存在していてこれが、TeX が必要なパッケージを探すためのパスを与えている。そのために、この一覧表を更新しなければならない。このためには、コマンドプロンプトを開き、次のように命令するだけでいい。

```
mktexlsr
```

と入力することで一覧表を更新することができる。

14.4 使用するべき package

package は最小構成でおこなうべきである。なぜなら、LuaLaTeX はかなり最近のものであり、package が LuaLaTeX に対応していないということもかなりの可能性としてある。そのため、参考になるのは参考文献のサイト [12] になる。

第 IV 部
VS Code 解説

15 VS Code の基本機能

デフォルトで使える VS Code の基本機能を解説する。

15.1 検索/置換機能

VS Code の基本機能として、置換機能がついている。このためマクロを組まずとも、を, に変えることができる。コマンドは、Ctrl+H キーを押すと検索と置換が出てきて、Enter キーを押すとひとつづつ置換、Crl+Alt+Enter キーを押すとすべてが置換される。また置換を閉じるのは Esc キーである。

また ctrl+shift+F を押すことで検索サイドバーを開くことができる。

15.2 コメントアウト機能

LaTeX は % から行末までをコメントと認識する。しかしこれをいちいち書くのは面倒くさい。さらにプログラム言語によりコメント機能はまったく違う。この時に、Ctrl+/ を使うことで、その時に編集しているプログラム言語に応じて適切な構文でコメントアウトすることができる。

15.3 各種ショートカット

ショートカットキー	機能
Ctrl+Alt+B	ビルドを実行する
Ctrl+Alt+V	pdfviewer を起動する
Ctrl+click	SyncTeX の利用（pdf 側）
Ctrl+Alt+J	SyncTeX の利用（コード側）
Ctrl+H キー	置換パレットの表示
Crl+Alt+Enter	置換の全置換
Ctrl+/	行のコメントアウト機能（全言語共通）
Ctrl+@	VS Code のターミナルの起動
Shift+Alt+F	latexindent を実行する
Ctrl+Space	コード補完を再表示
Ctrl+Shift+ + or -	ズームの程度を調整する

ただし、ショートカットキーによってビルドした場合は、Build LaTeX project の最初に記述されているコマンドをビルドコマンドとして呼び出す。これらのショートカットキーはデフォルトで設定されているもので、好きなキーに変更することもできる。

15.4 pdf viewer について

VSCode には pdfvewer が入っている。この機能について解説する。LaTeX で生成した文章は外部の viewer（例えば edge や chrome など）でも見ることができる。しかし、VSCode では SynicTeX や検索機能等も使うことができる。

検索機能 文書内の文字列を検索することができる。Ctrl+F を押すことで文章内の検索をかけることができる。

PDF の目次を見る PDF の目次を見ることができる。この目次は bookmark パッケージを用いたときまたはその pdf がそもそもブックマークがついていた時に限るが目次から簡単に飛ぶことができる。これは、pdfviewer の上のタスクバーの一番左のアイコンをクリックすることで見ることができる。

16 VS Code の設定について

16.1 VS Code の設定について

VS Code の設定は、setting.json というすべてに適応される設定と、workspace 設定という、workspace と呼ばれるファイルを何個か入れた空間について設定ができる。さらにファイルごとにも設定ができる。この優先順位は、setting.json 、work space 設定、ファイル設定の順に高い。これの設定のおかげで、さまざまな設定がおこなえるようになる。ワークスペースが 1 つのアプリケーションのように使うことができる。

16.2 wordwrap

wordwrap とは、このエディターの中で入力している中で pdfview 等にしているときに文章が見えている間で改行を自動で行ってくれる設定である。この設定は最初、自動改行しない設定になっている。この時には、書いている行が右にシフトしてしまい、マウス又は、エンドキー等で次の行にしなければならない。また見るときも水平スクロールをしなければならない。しかしこの機能を有効にすることで、自動で改行し、文章の視認性が高くなる。

16.3 スニペット

スニペットと自動補完

VSCode には引数や入力を補完するシステムがある。これを自動補完という。[*15]これは、VSCode の基本的な機能の一つである。これは特に設定の必要はない。入力者はサジェスト[*16]に表示された提案の中から、Tab キーや space キーで補完することができる。

スニペットは非常に長いコードをも格納することができ非常に短いコードでそのコードを呼び出すことができる点で有用である。

スニペットとは、コードの中で何回も使うだろうコードを少しの記述でそれを呼び出す一連の動作のことをいう。スニペットの語源は、短い単語のことである。

16.3.1 ユーザースニペットの導入

まずスニペットを導入する。左下の歯車から「ユーザースニペットの構成」を選択する。検索窓が出てくるので、latex と入力して latex.json を開く。latex.json にも、{}があるので、その中に次のコードをコピーする。

```
{
  "report":{
     "prefix": "report",
     "body": [
        "\\documentclass[a4paper,11pt]{ltjsarticle}",
        "",
        "",
        "% 数式",
        "\\usepackage{amsmath,amsfonts}",
        "\\usepackage{bm}",
        "% 画像",
        "\\usepackage{graphics}",
        "\\usepackage{graphicx}",
        "\\usepackage{here} %画像の表示位置調整用",
        "\\usepackage{type1cm}",
```

[*15] VSCode 等の microsoft 製品ではこの自動補完のことを固有の機能名でインテリセンスと呼ぶ。Microsoft 製品の公式ドキュメント等はインテリセンスと読んでいる。この資料ではより一般的な自動補完ということにする。

[*16] サジェストは “suggest” に由来し、予測変換の意味がある。

```
16
17            "",
18            "%A4: 21.0 x 29.7cm",
19            "${4}",
20            "",
21            "\\begin{document}",
22            "",
23            "\\title{${5}}",
24            "\\author{${6}}",
25            "\\date{${7:\\today}}",
26            "\\maketitle",
27            "",
28            "",
29            "$0",
30            "",
31            "",
32            "\\end{document}"
33        ],
34        "description": "授業レポート用テンプレート"
35    }
36  }
```

と書く。たとえば、report と latex で打つと、report のひな形が出てくるようになる。このように、スニペットはプリアンブル部等を簡単に早く書くことができるようになる。

16.4　スニペットの書き方

ユーザースニペットのフォーマットは次のように書く。スニペットも setting.json と同じように、’,’ で各設定を区切る。

ソースコード 4　latex.json

```
1
2   {
3   "[ スニペットの名前 ]": {
4     "prefix": "[ 呼び出すときのショートカット]",
5     "body":[
6       "[ 出力されるコードの1行目]",
7       "[ 出力されるコードの2行目]",
8       "...",
9     ],
10    "description": "[ スニペットの説明文]"
11  }
12  }
```

のように書く。

入力値の補足

入力値の中のコマンドに$n というのがあるがこれは、スニペットを記述するときに入力するためのカーソルが次にどこにいけば良いかを入力するための引数である。たとえば、次の例では、プログラムを載せるための環境をスニペットにより定義している。

```
{
  "report":{
      "prefix": "report",
      "body": [
          "\\documentclass[a4paper,11pt]{ltjsarticle}",
          "",
          "",
          "% 数式",
          "\\usepackage{amsmath,amsfonts}",
          "\\usepackage{bm}",
          "% 画像",
          "\\usepackage{graphics}",
          "\\usepackage{graphicx}",
          "\\usepackage{here} %画像の表示位置調整用",
          "\\usepackage{type1cm}",

          "",
          "%A4: 21.0 x 29.7cm",
          "${4}",
          "",
          "\\begin{document}",
          "",
          "\\title{${5}}",
          "\\author{${6}}",
          "\\date{${7:\\today}}",
          "\\maketitle",
          "",
          "",
          "$0",
          "",
          "",
          "\\end{document}"
      ],
      "description": "授業レポート用テンプレート"
  }
}
```

このように書いたときに、カーソルが$1 を記述した後に Tab キーを押すと$2 にカーソルが移動するように書くことができる。また、さらに、$1 が 2 つあるが、これらは同じことを記述したい場合にはたとえば参照をタイトルと同じにしたい場合、両方一度に入力することができる。さらに、body の末尾に$0 を入れておけ

ば、入力終了後に次の行からすぐに書き始めることができる。

ユーザースニペットに関して必要な注意また他の言語への応用等は [13][14][15] 参照するとよい。

16.5 禅モード

禅モードは Ctrl+K Z を用いると使うことができるモードでこのモードでは下のステータスバー等がなくなりコーディング作業に集中できるモードになっている。この禅モードから抜け出すためには Esc キーを押すことで抜け出すことができる。ここでは、ショートカットキーを使うことでさまざまな操作をすることができる。そのためにコーディング作業をするときにこのモードを使うと集中することができる。

第 V 部
VS Code 外部ツール

17 LaTeX workshop

17.1 snippet View

LaTeX Workshop から提供されているスニペットには一覧パネルがある。この一覧パネルはよく使う数学記号についてのスニペットに対応している。アクティビティバーの TeX のパネルから使うことがあるのである。これは GUI で使うことができるので LaTeX で What you see is what you get が可能になる。このバーはドラッグすることでコントロールパネルや、Secondary Side Bar に移動することができる。

17.2 文字数カウント

文字数をカウントするにはいくつかの方法がある。

方法 1 Ctrl+A を用いてすべて選択して下のステータスバーを見ると 何個選択という表示が現れる。

利点

- 拡張機能無しで文字数を数えることができる
- どの拡張子でも使える。

欠点

- ステータスバーに常に表示されていないこと。
- 文字数とは関係のないところもカウントされること。例えばプリアンブルなど

方法 2 ltexworkshop の設定を用いる。コマンドパネルを開いて、“latex-workshop.texcount” のコマンドを用いる。

欠点

- ステータスバーに常に表示されていないこと。
- 毎回打つのは面倒くさい

方法 3 設定に書き込む方法 setting.json に次を{}のなかに書き込む

ソースコード 5 wordCount

```
1     // LaTeX-workshopの設定で文字数をカウントする。
2     "latex-workshop.texcount.autorun": "onSave",
3     // LaTeX-Workshopの設定で文字数をカウントのタイミングを調整する。
4     "latex-workshop.texcount.interval": 1000,
```

利点

- ステータスバーに表示され楽
- 毎回打つ手間がない

欠点

- tex 以外の拡張子では使えない。

WordCounter を用いる方法　利点

- どの拡張子でも使える。
- 読了時間などのカスタムができる。

欠点

- TeX のコマンドまで認識してしまう。
- ステータスバーが窮屈になる。

ことが挙げられる。

17.3 SyncTex を使う

SyncTex とは何か？

TeX のソースファイルと PDF でカーソルの位置を同期する機能のこと。これを使えば、pdf 上の表示がどのソースに関係しているかということがすぐにわかる。

SyncTeX の設定

SyncTeX は、ビルドするときに形成されるファイル*.synctex.gz を使って動作する。

コマンドラインで SyncTeX を使う場合にはコマンドで

ソースコード 6　synctex

```
1  -synctex
```

をつけることで.synctex のファイルを生成することができる。

また、

ソースコード 7　synctex

```
1  -synctex=1
```

のオプションをつけた場合には、.synctex.gz という形式の圧縮ファイルが生成される。

ソースコード 8　synctex

```
1  -synctex=-1
```

の場合には、.synctex という形式のファイルが生成される。

LaTeXworkshop でのコンパイルの際には、

ソースコード 9 SyncTeX

```
1  "args": ["-synctex=1"]
```

のように追記することで、SyncTeX ファイルを生成することができる。

また、このファイルを消さないためには、

`"latex-workshop.latex.clean.fileTypes":`

の記述の中から、

`"*.synctex.gz"`、

の記述を消去すればよい。最初から無ければ問題ない。これにより、VS Code の pdfvier において、SyncTeX が利用できるようになる。

SyncTeX の具体的な利用の仕方

pdfviewer 上で Ctrl キーを押しながらマウスでコードをみたい場所におき、左クリックすることでそのコードの位置に飛ぶことができる。逆にコードから pdf に飛びたいときは、選択範囲をマウスで示して、またはカーソルをおいて、Ctrl+Alt+J で飛ぶことができる。(コマンドパレットから SyncTeX と入れてコマンドを実行させてもよい。)

17.4 シンタックスハイライト

シンタックスハイライトとは、LaTeX の文章などを編集するためのコマンドに色をつけることでその対応やコードの可読性を高めることができる VS Code の機能である。しかしながら、LaTeX workshop はそれに対応しているのだが VS Code のカラーテーマを Visual Studio Dark または Visual Studio Light にしているとうまく機能してくれない。したがってダークテーマでシンタックスハイライトを使いたい場合は、デフォルト設定の Dark+ を使用するとよい。又はほかのカラーテーマならば良い。

17.5 cloud LaTeX との連携

cloud LaTeX を使うことでパソコン内部に LaTeX 環境を構築しなくとも LaTeX を使うことができる。cloud LaTeX は cloud LaTeX と同一のサーバー内とローカルのこの VS Code と連携することが可能であり、それにより、ローカルとサーバーで同期ができるために非常に便利なものとなっている。

利点

- データがサーバ上で保存するためにデータが消える心配はない。
- コンパイルがサーバ上でおこなわれるために自分のパソコンに負荷がかからないそのために自分のパソコンではできない autobuild が可能。
- cloud LaTeX のシステムやインストールされたパッケージを簡単に使うことができる。

欠点

- VS Code からサーバーを立ち上げることができないために、最初に cloud LaTeX の方に入る必要がある。
- 定期的な対応の更新をしなければならない。
- ローカル上ではコンパイルできないためにオフライン状態でコンパイルできない。

等が挙げられる。

cloud LaTeX との連携をするための設定

導入の設定は、md 形式のファイルに添付されているものを setting.json に入れればよい。cloud LaTeX 参照

17.6 LaTeXWorkshop の設定に困ったら

LaTeXworkshop の設定に困った場合には、[16] 又は、公式ドキュメントである [6] を読むとよい。しかしながらこれらは英語であるのでこれがいやな場合は、[17][18][19][20][21] が公式ドキュメントを日本語に訳したような書き方をしているためにこれを読むとよい。[22] 等は VS Code と latex について解説している数少ない wiki のページである。[23][24][25] は変わった方法で LaTeX を VSCode に導入している。また VSCode の解説については、[26][27][28] 等を参考にすると良い。

18 cloudlatex

18.1 cloudlatex とは何か？

cloudlatex とは、株式会社アカリクが運営している LaTeX のビルドをリモート環境で行うサービスのことである。この cloudlatex を用いるとローカル環境に LaTeX がインストールされていなくとも LaTeX 文章を書くことができる。

18.2 cloudlatex の VS Code での拡張機能

18.2.1 設定方法

LaTeX のコードを自前の環境で書き、コンパイルをリモート環境で行うことができる。まず VSCode 上の拡張機能から cloudlatex の拡張機能をインストールする。次に自分の cloudlatex のマイページから右上のユーザー名をクリックしてプルダウンを開き「プラグイン連携」をクリックする。プラグインとは、拡張機能のことである。次にそこで生成されるトークンを二つコピーする。「client」「access-token」の 2 つがわかっていれば OK である。次に VSCode 上で CL と書いたアイコンが生成されるためにそこをクリックして「Set account」をクリックする。「メールアドレス」「client」「access-token」を順番に入力する。「Your account has been validated!」という通知が出れば連携の設定は完了する。

18.2.2 使い方

使い方は Cloud Latex で作成したプロジェクトを Visual Studio Code で編集することができる。ここでまず初めに Project ID を取得する必要がある。URL を確認する

「https://cloudlatex.io/projects/XXXXXX/edit」となっている 6 桁の数字「XXXXXX」が Project ID である。次に自分のローカル環境にその cloudlatex 連携をするためのディレクトリを作る。

エクスプローラーからそのディレクトリを開き、次に VSCode 上で「Project Setting」をクリックすると「設定」のファイルが開く。ここでワークスペース設定から「Cloudlatex: Enabled」にチェックを入れ、「Cloudlatex: Project ID」に先ほど確認した数字を入力する。「Project files have been synchronized!」という通知が出てくれば成功である。これでローカル環境に構築することなくリモート環境で LaTeX のコンパイルができる。[*17]LaTeX のコンパイルは初期設定では、Ctrl+S を押して保存する。又は VSCode の autosave されたときにコンパイルされる設定になってる。またこの機能のみを使う latexworkshop 設定は [29] を参照するとよい。[*18]

この機能によって VSCode の強力な編集機能をフルに使うことができる。

19 Ultra Math Preview

19.1 Ultra Math Preview とは何か

Ultra Math Preview は、LaTeX Workshop よりも強力な数式プレビューができる拡張機能である。パッケージで定義されたコマンドもプレビューすることができる。さらにスタイルをカスタマイズすることができるため、LaTeX Workshop 標準のプレビューよりもより利便性が高いものになっている。このプレビューは Markdown でも用いることができる。

- 数式を打つとプレビューが即座に出てくる
- LaTeX と Markdown で同じマクロがプレビューに使える
- プレビューが透過できる
- プレビューの上からその下にある文字をクリックすることができる

19.2 Ultra Math Preview の設定

setting.json の{}の中に次を入れることで latexworkshop が対応していないパッケージの数式のレンダリングをすることができる。[30]

ソースコード 10 Ultra Math Preview

```
1  "umath.preview.renderer": "mathjax",
```

[*17] しかしここで多くの場合エラーが発生することが多い。アカリクの cloudlatex のサーバーの問題かわからないが容量の大きいファイル等になるとサーバーと接続できないことが多い。この場合は時間をおくか又は新しく Project ID を入れなおすとよい。

[*18] また同時に Cloud 上でも保存することができる。この場合だと保存したときに常にコンパイルするためにサーバーに負荷がかかりやすくよく接続が切れてしまう。接続が切れたときに LaTeX のコンパイルエラーが出てしまう。この場合 LaTeX のエエラーなのか接続の問題なのかという見分けがつきにくくなる。また latexworkshop 標準搭載のエラーメッセージを吐くことはないため慣れないうちはリモート環境での執筆をオススメする。

```
"umath.preview.macros": [
    "\\require{physics}",
    "\\require{HTML}",
    "\\require{mathtools}",
    "\\require{mhchem}",
    "\\require{empheq}",
    "\\scriptsize{}",
    "\\newcommand{\\bm}[1]{\\boldsymbol{#1}}"]
  "umath.preview.position": "top",
  "umath.preview.customCSS": [
    "background-color: rgba(0, 0, 0, 1);" ,
  ],
```

19.3 Ultra Math Preview の設定の解説

設定の意味について記述する。

umath.preview.toggleMathPreview　プレビューを出すか出さないかを制御することができる。

umath.preview.renderer　レンダリング（変換）に MathJax を使うか KaTeX を使うかを選択することができる。

umath.preview.macros　LaTeX のマクロを定義してレンダリングすることができる。

umath.preview.position　プレビューを出す位置を変えることができる。

umath.preview.customCSS　プレビューの見た目を変える CSS

umath.preview.closeAllPreview　表示されたプレビューを非表示にすることができる。デフォルトでは Escape キーを押すことで非表示にすることができる。

19.3.1 umath.preview.macros

LaTeXworkshop でも数式の表示は可能だが拡張性は低い。しかし、この拡張機能は、`\newcommand` で記述することができるマクロを表示することができる。

require+extension **名**　このコマンドは MathJax の固有のコマンドであり LaTeX のパッケージのコマンドをレンダリングすることができるものである。

scriptsize　このコマンドは、UltraMathPreview の表示を小さくすることができるコマンドである。これを使うことで表示サイズが pdf と同レベルになり全体が見やすくなる。

bm **の定義**　このコマンドは、bm パッケージの`\bm` をレンダリングするために必要なものである。このコマンドは`\bm` を`\newcommand` で定義している。

UltraMathPreview は定義をしてやれば Mathjax に入っていないパッケージでもよく使うのものであればレンダリングすることができる。マクロの制作方法については [2] に詳しい。

19.3.2 umath.preview.position

ホバーの出す位置を変更することができる。ホバーの出す位置は`"top"`または`"bottom"`である。

19.3.3 umath.preview.customCSS

この設定は、プレビューの背景の色を変更することができる。

ソースコード 11 color

```
"umath.preview.customCSS": [
  "background-color: rgba(0, 0, 0, 1);" ,
],
```

について、rgba(0, 0, 0, 1) の引数を変更することで色を変化させることができる。左から順に red,green,blue であり、最後の数が透過率である。この設定は黒色で透過なしになっている。ライトモードの人は rgba(255,255 ,255 , 1) 等にするとよい。

20 LaTeX 表制作

20.1 Table Generator

Table Generator は、LaTeX での表を書く際に非常に有用なサイトである。このサイトでは word とほとんど同等の環境の GUI を用いることで LaTeX 記法での表に変換してくれるサイトである。https://www.tablesgenerator.com/[31] で書くことができる。さらにこれは html 等にも対応しているなどさまざまなマークアップ言語に対応している。

また CSV ファイルの読み込みもすることができる。([32] 参照。)

20.2 csv2tabular

csv2tabular は excel 形式の表や、excel の表をコピペするだけで LaTeX 形式の表に変換してくれる外部ツールである。これは、レポートの制作の時などでも便利である。このサイトは、[33] https://rra.yahansugi.com/scriptapplet/csv2tabular/ から見ることができる。またこの表は、tabular 形式になるが環境名は表示されないので、次の環境を加える必要がある。

```
\begin{table}[h]
  \caption{キャプション名を書く}
  \label{ラベル名を書く}
  \centering
   \begin{tabular}{clll}% ここは中央ぞろえまたは左右ぞろえを決める。
    % ここに変換したコードを載せる。
  \end{tabular}
\end{table}
```

のように書く必要がある。([34] 参照) また、Table Generator のように、セルの結合等には対応していないため、形式的にまとまっているデータに対しては有効であるが、複雑な表は処理が難しい、複雑な表を書く場合は Table Generator を使う方がよい。

21　テキスト校正くん

テキスト校正くんは、tex ファイルや md ファイル等の日本語文章を校閲することができる。これはインストールするだけで利用できる。例えば「です・ます」調と「だ・である」調の混在等を教えてくれる。

22　LaTeX 数式コマンドを手書きで打つ

22.1　Myscript math とは何か

LaTeX コマンドを打つのが面倒だったり数式が非常に長く文字よりも手で書いた方が早く書けるときには `https://webdemo.myscript.com/views/math/index.html` のサイトを使うとよい。このサイトは Myscript([35] 参照) が運営しているサイトであり Apple Pencil や手書きで書いた文字を認識し、LaTeX 形式の数式コマンドに変換してくれるウエブ上のサービスである。無料で使うことができる。

22.2　Myscript Math の使い方

このサイトでは pple Pencil や手書きで書いた文字を認識し LaTeX コマンドを生成できる。また簡単な数式の場合はグラフ描画することができる。

23　キーボードショートカット

23.1　キーボードショートカットとは何か？

キーボードショートカットとは Windows やアプリケーションの機能を選択するような操作を、マウスやタッチパッドではなくキーボードの複数のキーを組み合わせて行う操作のことである。この操作を覚えることでさまざまな PC 上で行う操作をマウス無しで行うことができるようになり、より早くさまざまなタスクを遂行できるようになる。[*19]

ここでは VS Code のキーボードショートカットの設定の仕方について記述する。

23.2　キーボードショートカットの変更方法

23.2.1　キーボードショートカットの開き方

キーボードショートカットの変更方法には 3 通りの開きかたがある。

方法 1　左下の設定からキーボードショートカットを選択

方法 2　Ctrl+K Ctrl+S を押す

方法 3　コマンドパレット Ctrl+shift+P から preferences:Open keyboard Shortcuts または、preferences:Open keyboard Shortcuts(json) を選択する。

[*19] マウスでの操作が早い場合も多々ある。例えば powerpoint 等の GUI であるソフトである。このようなグラフィック中心のソフトウェアはマウスでの操作の方がよい。逆に VS Code はその気になればほとんどの操作をコマンドで遂行することができる。このようなマウスを使うべき操作と使う必要のないコマンド操作の区別をつけることは PC を扱ううえで早く作業できるかできないかの違いになる。

これでキーボードショートカットの設定画面に移る。

23.3　キーボードショートカットの設定方法 (規定の方法)

キーボードショートカットの設定画面 (json でないほう) を表示すると、コマンドが設定されている場所とされていない操作がある。

コマンドが定義されていないところを選択してエンターキー又はクリックをするとキーバインドの設定画面に移る。この際にショートカットにしたいキーを打つことでその操作のショートカットキーを定義することができる。[*20]

この設定画面からキーボードショートカットをすぐに確認できるためにすぐに公式ドキュメントを検索するまでもないキーボードショートカットの検索にも使うことができる。

23.4　キーボードショートカットの設定方法 (JSON)

キーボードショートカットの設定は JSON ファイルからも行うことができる。この場合はネットから必要なキーバインドの定義を持ってくるとき等に有用なやり方になる。キーボードショートカットの記述方法は次のようになる。より詳細なキーボードショートカットの方法は以下のように行う。

ソースコード 12　キーボードショートカットの変更方法 (json)

```
1   // 既定値を上書きするには、このファイル内にキー バインドを挿入しますauto[]
2 [
3   {
4     "key": "cmd+n",
5     "command": "explorer.newFile",
6     "when": "explorerViewletFocus"
7   },
8 ]
```

このように、"key"には、キーバインドを打ち込み、"command"には拡張機能またはこの VS Code のシステム自体に設定されている機能を記述する。

次に"when"はそのキーバインドがいつどのようなときにそのキーバインドを打ったときに其のコマンドを使用するのかということを決めることができる。

またこの"when"を設定する際には、VS Code の公式ドキュメントである https://code.visualstudio.com/docs/getstarted/keybindings#_when-clause-contexts で見つけるとよい。この中で https://code.visualstudio.com/api/references/when-clause-contexts#conditional-operators が自分の状況にあったものが見つかると思う。([36][37] 等、または公式ドキュメント [26] 参照)

またキーバインドを簡単に表示する方法として keybindings.json ファイルを開いているときに Ctrl+K Ctrl+K のキーを押すと次のコマンドが表示される。残りはユーザーが"command","when"を記述すればよい。

ソースコード 13　keybindings.json

```
1   {
2   "key": "入力したキー",
```

[*20] キーボードショートカットの設定はソースからデフォルトかユーザー定義かということがわかるようになっている。

```
3   "command": "commandId",
4   "when": "editorTextFocus"
5 }
```

24 Git Hub Copilot

24.1 Git Hub Copilot とは

Git Hub Copilot とは、Git Hub が提供する生成型 AI を VSCode に拡張した機能である。この Git Hub Copilot は、本来はプログラミングの補助として考えられたものであるが、LaTeX にも使うことができる。LaTeX に用いると、Copilot はユーザーの入力に対して次の文章の生成や、数式の計算等を行うことができる。

24.2 導入方法

まずは Git Hub に登録しよう。GitHub とは、文章管理ソフトのことである。このソフトについての詳細は [38] を参照するとよい。この Git Hub に登録することで Git Hub Copilot を使うことができる。Git Hub Copilot は、`https://github.com` から登録することができる。このサイトは学生は無料で Pro プランを使うことができる。是の登録には数日かかる。登録が完了したら、VSCode の拡張機能から Git Hub Copilot をインストールする。この時 GitHub へのログインを求められるので、先ほど登録したアカウントでログインする。これで Git Hub Copilot の導入は完了する。

24.3 使用方法

使用方法としては、Copilot はユーザーの入力に対して文章の生成ができる。例えば、「数学の定義を書く」と入力すると、次のような文章を生成する。

> 数学とは、数とその構造、変化、空間、量、不確実性などの概念を扱う学問である。数学は、抽象化された概念を形式的に定義し、それらの間の関係を厳密に推論することによって、数学的な対象についての知識を得る。数学は、抽象化、論理的推論、算術、幾何学、統計学、計算科学、数論などの様々なトピックを扱う。数学は、数とその構造、変化、空間、量の研究である。数学は、数学的オブジェクトの研究を含む。数学は、自然科学、工学、医学、経済学などの他の分野で使用される数学的なモデルを開発するのに役立つ。

のような文章を生成することができる。また計算などもすることができる。実際の使用例については、[39] を参照するとよい。

25 LaTeX で図を用いる

25.1 Mathcha

Mathcha とは、LaTeX での tikz を簡単に直観的に書くことができるものである。GUI で簡単に描画することができるのでお勧めである

25.1.1 Mathcha の使い方

Mathcha の使い方は、[40] を参照するとよい。

25.2 I Love TikZ

I Love TikZ は、LaTeX での tikz を簡単にかつペイントの要領で書いた図やイラストを Tikz 形式に変換することができるものである。東京工業大学のデジタル創作同好会が作成した。[41]GUI がとても使いやすくさらに無料である。このサイトは `https://hackathon-22spring-13.github.io/client/` から見ることができる。

25.3 I love TikZ の使い方

I love TikZ の使い方は url 先のページにもあるように絵やイラストをかいて変換ボタンを押すだけである。それで変換することができる。非常に簡単。

25.4 Python で描画したグラフを Tikz に変換する

tikzplotlib は Pyplot で書いたグラフを次のようにすることで Tikz に変換することができる。

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  import tikzplotlib
4
5  x = np.linspace(0, 2 * np.pi, 100)
6  y = np.sin(x)
7
8  plt.plot(x, y)
9  plt.xlabel("$x$")
10 plt.ylabel("$\sin(x)$")
11 plt.title("Plot of $\sin(x)$")
12
13 tikzplotlib.save("sinx_plot_tikz.tex")
```

このように書くと tex ファイルが出力されるために、そのファイルを LaTeX で読み込むことで図を表示することができる。この方法は、[42] を参照するとよい。このようにすることで TikZ での図を簡単に作成するかつ、編集することができる。

25.5 Draw.io

Draw.io は、図を描くためのツールである。Draw.io はフローチャート・レイアウト・組織図などの図を作ることができる。またアイコンなども簡単に作れる。

Draw.io はもともと `https://draw.io/` から無料で使うことができるツールである。さらに VSCode の拡張機能としても使うことができる。この拡張機能は [43] を参照するかまたは VSCode の拡張機能から Draw.io

を検索するとよい。

25.5.1　Draw.io 拡張機能の使い方

VSCode の拡張機能から Draw.io Integration を検索して、インストールする。VSCode から拡張機能を開くには新規ファイルの作成から拡張子を.drawio または.dio にすることで開くことができる。LaTeX にそのままインポートしたい場合はファイル名を.drawio.png や drawio.svg にすることでそのまま png 画像や svg 画像としてインポートすることができる。

ファイルの拡張子を.drawio や.dio にした場合は上のメニューバーから File を選択して Export as から画像形式を選択することで画像として保存することができる。

付録 A　Windows ショートカットキー

A.1　ショートカットキーについて

VS Code ではさまざまなショートカットが存在する。しかし、それをより活用し、ひいては PC 全体の効率的な活用を目指すことができる。例えば、マウスを使う場合と比べて時間効率では、年間 120 時間の時短が見込める等の報告がある。([44] などを参照。)

ここでは Windows の起動から VS Code, ブラウザでのショートカットについて記述する。

付録 B　ショートカットキー一覧

ショートカットキーは一つのキーのみではなく複数キーの組む合わせも存在する。

コマンド	機能
Alt+Z	wordwrap の変更
Ctrl+Alt+B	ビルドを実行する
Ctrl+Alt+V	pdfviewer を起動する
Ctrl+click	SyncTeX の利用（pdf 側）
Ctrl+Alt+J	SyncTeX の利用（コード側）
Ctrl+H キー	置換パレットの表示
Crl+Alt+Enter	置換の全置換
Ctrl+/	行のコメントアウト機能（全言語共通）
Ctrl+@	VS Code のターミナルの起動
Ctrl+Shift+X	拡張機能タブを開く
Shift+Alt+F	latexindent を実行する
Ctrl+Space	コード補完を再表示
Ctrl+Shift+M	LaTeX のエラーメッセージを表示する
Ctrl+shift+M	数式環境で数式のプレビューを表示する
Ctrl+B	サイドバーの表示の設定
Ctrl+Shift++ or −	ズームの程度を調整する
Ctrl+K F	ワークスペースを閉じる
Ctrl+shift+L	マルチカーソル (同じ文字)
Ctrl+K Z	禅モード (Esc で取り消し)
Ctrl+Space	suggest の表示

次の参考文献は、この資料を書き上げるために用いた資料一覧である。この資料を読み通すことができたならば、きっとこれらのサイトも読むことができるだろう。さらに便利な使い方や、自分なりの設定をすることもできるだろう。

良い LaTeX ライフ happy LaTeXing

索引

参考文献

[1] Leslie Lamport. **文書処理システム** *LaTeX*. アスキー, 1990.

[2] 奥村晴彦. *LaTeX* **美文書作成入門**. 技術評論社, 2020.

[3] @passive-radio. 【大学生向け】LaTeX 完全導入ガイド Windows 編（2022 年）. https://qiita.com/passive-radio/items/623c9a35e86b6666b89e#4-snippet-%E3%81%AE%E3%82%B9%E3%82%B9%E3%83%A1%E6%96%87%E7%AB%A0%E4%BD%9C%E6%88%90%E3%81%AE%E3%82%82%E3%81%A3%E3%81%A8%E5%8A%B9%E7%8E%87%E5%8C%96.

[4] @rainbartown. VS Code で最高の LaTeX 環境を作る（2020）. https://qiita.com/rainbartown/items/d7718f12d71e688f3573.

[5] 日記.【Windows】VS Code+pLaTeX（+LuaLaTeX）環境を構築した. https://everykalax.hateblo.jp/entry/2022/12/15/144238.

[6] James-Yu. Latex-workshop/readme.md. https://github.com/James-Yu/LaTeX-Workshop/blob/master/README.md.

[7] 明松真司. *1* **週間で** *LaTex* **の基礎が学べる本** . impress, 2022.

[8] Tex wiki. https://okumuralab.org/tex/.

[9] Cloudlatex. https://cloudlatex.io/ja.

[10] alpaca honke(あるかっぱ/アルパカ本家). [初心者]latex のインストールから使い方までこれ一本！2022 版. https://qiita.com/alpaca-honke/items/f30a2d04eedaa3c36a21#pdf%E3%83%93%E3%83%A5%E3%83%BC%E3%83%AF%E3%81%AE%E3%82%A4%E3%83%B3%E3%82%B9%E3%83%88%E3%83%BC%E3%83%AB%E8%A8%AD%E5%AE%9A.

[11] JavaDrive. https://www.javadrive.jp/vscode/setting/index4.html.

[12] TeX Wiki. TeX のディレクトリ構成（TDS）. https://texwiki.texjp.org/?TeX%20%E3%81%AE%E3%83%87%E3%82%A3%E3%83%AC%E3%82%AF%E3%83%88%E3%83%AA%E6%A7%8B%E6%88%90.

[13] Web 備忘録. Web 備忘録 VS Code でスニペットを自作する方法（2019）. https://webbibouroku.com/Blog/Article/VSCode-snippets.

[14] Web 備忘録. VS Code で自作のスニペットを登録する. https://qiita.com/michawo/items/051da6ce6d9daf9784fb.

[15] Web 業界で働く人を少しだけ手助けするメディア. Visual Studio Code ユーザースニペットの使い方まとめ. https://web-guided.com/620/#:~:text=%E3%82%B9%E3%83%8B%E3%83%9A%E3%83%83%E3%83%88%E3%82%92%E5%B1%95%E9%96%8B%E3%81%97%E3%81%9F%E6%99%82%E3%81%AB%E3%80%81%E3%82%AB%E3%83%BC%E3%82%BD%E3%83%AB%E3%82%92%E4%BB%BB%E6%84%8F%E3%81%AE%E4%BD%8D%E7%BD%AE%E3%81%AB%E8%A8%AD%E5%AE%9A%E3%81%99%E3%82%8B%E3%81%93%E3%81%A8%E3%81%8C%E5%87%BA%E6%9D%A5%E3%81%BE%E3%81%99%E3%80%82%20%241%20%E3%82%92%E8%A8%98%E8%BF%B0%E3%81%97%E3%81%9F%E7%AE%87%E6%89%80%E3%81%AB%E3%82%AB%E3%83%BC%E3%82%BD%E3%83%AB%E3%81%8C%E5%87%BA%E7%8F%BE%E3%81%97%E3%80%81%20%242%20%E3%80%81%20%243%20%E3%81%A8%E8%A8%98%E8%BF%B0%E3%82%92%E8%BF%BD%E5%8A%A0%E3%81%97%E3%81%9F%E5%A0%B4%E5%90%88%E3%81%AF%E3%80%81%20%EF%BC%BB、%EF%BC%BD%20%E3%82%AD%E3%83%BC%E3%82%92%E6%8A%BC%E3%81%99%E3%81%93%E3%81%A8%E3%81%A7%E3%80%81%E3%81%9D%E3%82%8C%E3%82%89%E3%81%AE%E4%BD%8D%E7

%BD%AE%E3%81%AB%E3%82%AB%E3%83%BC%E3%82%BD%E3%83%AB%E3%81%8C%E7%A7%BB%E5%8B%95%E3%81%97%E3%81%BE%E3%81%99%E3%80%82%20%22body%22%3A%20%5B%20%22console.log%20%28%27%241%27%29%3B%22%2C%20%22%242%22%20%5D.

[16] James-Yu. home. https://github.com/James-Yu/LaTeX-Workshop/wiki.

[17] Yarakashi_Kikohshi. LaTeX Workshop を使いこなす. https://qiita.com/Yarakashi_Kikohshi/items/a9357dd469320ffb65a0.

[18] Yarakashi_Kikohshi. LaTeX Workshop をもう少し使いこなす. https://qiita.com/Yarakashi_Kikohshi/items/1a275f2046b002e398b3.

[19] Yarakashi_Kikohshi. LaTeX Workshop をもっと使いこなす. https://qiita.com/Yarakashi_Kikohshi/items/1f2225c7e28aad498998.

[20] Yarakashi_Kikohshi. Zotero と LaTeX Workshop で bib ファイルを扱いこなす. https://qiita.com/Yarakashi_Kikohshi/items/8f720643543ba175f7cc.

[21] Yarakashi_Kikohshi. LaTeX Workshop の数式プレビューを使いこなす. https://qiita.com/Yarakashi_Kikohshi/items/4570bba51787e47a03c6.

[22] TeX Wiki. Visual Studio Code/LaTeX. https://texwiki.texjp.org/?Visual%20Studio%20Code%2FLaTeX#he95e080.

[23] hikozaru1202. エラーしながら学ぶ VS Code に Latex を導入 1. https://qiita.com/hikozaru1202/items/4189bfc52a99b7c32968.

[24] @hikozaru1202. エラーしながら学ぶ VS Code に Latex を導入 2. https://qiita.com/hikozaru1202/items/befa7ddb6ea1b8920c92.

[25] @hikozaru1202. エラーしながら学ぶ VS Code に Latex を導入 3. https://qiita.com/hikozaru1202/items/ce4c916f8d763b4006a9#latexmkrc%E3%81%AE%E4%BD%9C%E6%88%90.

[26] visual studio Code. https://code.visualstudio.com/docs/sourcecontrol/github#_cloning-a-repository.

[27] リブロワークス. *Visual Studio Code* **完全入門**. impress, 2022.

[28] TeX フォーラム. VS Code で SyncTeX が使えない. https://okumuralab.org/tex/mod/forum/discuss.php?d=3075&parent=18242.

[29] onigiri. CloudLaTeX と VScode 連携によって最強の Tex 環境を手に入れる. https://nkgtt.hatenablog.jp/entry/2020/12/24/000000.

[30] migawariw.【LaTeX】Ultra Math Preview で数式プレビューを超強化. https://qiita.com/migawariw/items/1d3ab752f1ef261d6bcb.

[31] Tables Generator. https://www.tablesgenerator.com/.

[32] tablesgenerator about. https://www.tablesgenerator.com/about.

[33] csv2tabular. https://rra.yahansugi.com/scriptapplet/csv2tabular/.

[34] Tex の表作成ジェネレーター（csv2tabular）を活用して表を作る方法. https://hibikanblog.net/blog-entry-774.html.

[35] Myscript. https://webdemo.myscript.com/.

[36] 今泉大樹.【小ネタ】極力キーボードから手を離したくないライト vimmer が vscode に設定しているショートカット（基本的なファイル操作編）. https://dev.classmethod.jp/articles/vscode_file_operation_shortcut_setting/.

[37] @x5dwimpejx. Vs code のキーボードショートカットの変更の設定. https://qiita.com/x5dwimpejx/items/5623ccfae992fcff5510.
[38] https://github.com/.
[39] https://www.sejuku.net/blog/25535.
[40] https://www.mathcha.io/.
[41] https://trap.jp/post/1623/.
[42] https://pypi.org/project/tikzplotlib/.
[43] https://marketplace.visualstudio.com/items?itemName=hediet.vscode-drawio.
[44]【最速仕事術】効率が 24 倍アップ！「脱マウス」の極意. https://www.nomura.co.jp/el_borde/article/0010/.
[45] san.com. CaTeX（軽鳥怪鳥）で快適 LaTeX ライフ in VS Code. https://konn-san.com/articles/2018-11-26-happy-latex-with-catex.html.
[46] @skikkh. VS Code で快適 LaTeX 環境を構築する方法. https://qiita.com/skikkh/items/707e8a5def368a69e9a6.
[47] **アカリク**. Cloud LaTeX Extension for Visual Studio Code. https://github.com/cloudlatex-team/cloudlatex-VSCode-extension/blob/main/docs/README_ja.md.
[48] Yarakashi-Kikohshi. VS Code で編集して Cloud LaTeX でタイプセットする. https://gist.github.com/Yarakashi-Kikohshi/e554045b77d35bd132eb976034625023.
[49] mtk_birdman's blog.【Windows】Visual Studio Code で Cloud LaTeX の実行環境を構築する（2022.02.16）. https://mtkbirdman.com/windows-visual-studio-code-cloud-latex-install.
[50] Cui と gui. https://www.pc-master.jp/words/cui-gui.html.
[51] 川崎庸市. *Visual Studio Code* **の教科書**. マイナビ, 2021.
[52] TsubasaTakeda. LaTeX を VS Code で書く（初心者のレポート向け）. https://qiita.com/TsubasaTakeda/items/bed2856c6bd427268144#%E6%8B%A1%E5%BC%B5%E6%A9%9F%E8%83%BD%E3%82%92%E8%BF%BD%E5%8A%A0.
[53] James-Yu. FAQ and common issues. https://github.com/James-Yu/LaTeX-Workshop/wiki/FAQ#syntax-highlighting-does-not-work-for-most-elements.
[54] lualatexlab.【基本】latex マニュアルを参照しよう. https://lualatexlab.blog.fc2.com/blog-entry-5.html.
[55] wikipedia. https://ja.wikipedia.org/wiki/LaTeX#cite_note-FOOTNOTELamport19905-1.
[56] LuaLaTeX Lab.【基本】LuaLaTeX の PDF を便利にしよう ～hyperref パッケージほか～. https://lualatexlab.blog.fc2.com/blog-entry-43.html.
[57] 日本 markdown ユーザー会. https://qiita.com/migawariw/items/1d3ab752f1ef261d6bcb.
[58] Yarakashi_Kikohshi. LaTeX でいろんなパッケージを usepackage する. https://qiita.com/Yarakashi_Kikohshi/items/97f9f920fb23974e0011.
[59] Stack overflow. https://stackoverflow.com/.
[60] TeX Wiki. SyncTeX. https://texwiki.texjp.org/?SyncTeX.
[61] @t_kemmochi. uplatex で SyncTeX するための最低限の VS Code 設定. https://qiita.com/t_kemmochi/items/dd38bbf2b823c770d1ec.
[62] 情報科学屋さんを目指す人のメモ. PDF をクリックして対応する LaTeX ソースにジャンプする方法

（TeXworks+SyncTeX）. https://did2memo.net/2015/03/05/latex-pdf-source-jump/.
[63] Tex wiki. https://texwiki.texjp.org/?TeX%20Live.
[64] 三秀舎ブックスラボ. http://www.kksanshusha.jp/booklab/archives/1184#footnote_0_1184.
[65] Latex で pdf 出力する際の tips. https://blog.miz-ar.info/2015/09/latex-hyperref-tips/.
[66] 12345. VS Code の便利なショートカットキー. https://qiita.com/12345/items/64f4372fbca041e949d0.
[67] @TomK. VS Code のマルチカーソル練習帳. https://qiita.com/TomK/items/3b1f5be07d708d7bd6c5.
[68] かわさきしんじ. VS Code でテキストの折り返しを設定するには. https://atmarkit.itmedia.co.jp/ait/articles/1807/27/news035.html.
[69] Microsoft サポート. Word で unicodemath および latex を使用して行形式の数式を入力する. https://support.microsoft.com/ja-jp/office/word-%E3%81%A7-unicodemath-%E3%81%8A%E3%82%88%E3%81%B3-latex-%E3%82%92%E4%BD%BF%E7%94%A8%E3%81%97%E3%81%A6%E8%A1%8C%E5%BD%A2%E5%BC%8F%E3%81%AE%E6%95%B0%E5%BC%8F%E3%82%92%E5%85%A5%E5%8A%9B%E3%81%99%E3%82%8B-2e00618d-b1fd-49d8-8cb4-8d17f25754f8.
[70] @Yarakashi_Kikohshi. エラーと警告を読むゾ. https://qiita.com/Yarakashi_Kikohshi/items/4ede21b06d094ad3b89e#-%E6%AD%A3%E3%81%97%E3%81%8F%E3%83%AD%E3%82%B0%E3%82%92-latex-workshop-%E3%81%AB%E5%BC%95%E3%81%8D%E6%B8%A1%E3%81%99.
[71] TeX Wiki. LaTeX のエラーメッセージ. https://texwiki.texjp.org/?LaTeX%20%E3%81%AE%E3%82%A8%E3%83%A9%E3%83%BC%E3%83%A1%E3%83%83%E3%82%BB%E3%83%BC%E3%82%B8.
[72] HP 社. Windows 10 のコマンドプロンプトとは？ その起動方法と使用例を紹介. https://jp.ext.hp.com/techdevice/windows10sc/27/.
[73] Django Girls のチュートリアル. コマンドライン（コマンドプロンプト）とその関連について学べるサイト. https://tutorial.djangogirls.org/ja/.
[74] Yarakashi-Kikohshi. Customize Syntax Highlight for LaTeX Workshop. https://gist.github.com/Yarakashi-Kikohshi/948720cb69d0baafe71e62ec6cb2cb54.
[75] @ta_b0_. LaTeX にソースコードを【美しく】貼る方法. https://qiita.com/ta_b0_/items/2619d5927492edbb5b03.
[76] Yarakashi-Kikohshi. なるべくデフォルトのまま使いたい人へ. https://qiita.com/Yarakashi_Kikohshi/items/e9270af54569640fe80f.
[77] IT 用語辞典. ビルド【build】. https://e-words.jp/w/%E3%83%93%E3%83%AB%E3%83%89.html.
[78] 大橋. ルートディレクトリって結局どこ？https://codor.co.jp/django/root-directry.
[79] KERI's Lab. VS Code で TeX を書こう. https://www.kerislab.jp/posts/2019-01-14-VSCode-latex/#:~:text=SyncTex%20%E3%81%A8%E3%81%AF%E3%80%81TeX%20%E3%81%AE%E3%82%BD%E3%83%BC%E3%82%B9%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB%E3%81%A8%20PDF%20%E3%81%A7%E3%82%AB%E3%83%BC%E3%82%BD%E3%83%AB%E3%81%AE%E4%BD%8D%E7%BD%AE%E3%82%92%E5%90%8C%E6%9C%9F%E3%81%99%E3%82%8B%E6%A9%9F%E8%83%BD%E3%81%A7%E3%81%99%E3%80%82%20VSCode%20%E3%81%AE、PDF%20%E3%83%93%E3%83%A5%E3%83%BC%E3%83%AF%E3%81%AF%20SyncTex%20%E3%81%AB%E5%AF%BE%E5%BF%9C%E3%81%97%E3%81%A6%E3%81%84%E3%82%8B%E3%81%AE%E3%81%A7%E3%80%81TeX%20%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB%E3%81%A8%20PDF%20%E3%83%95%E3%82%A1%E3%82%

A4%E3%83%AB%E3%81%AE%E8%A9%B2%E5%BD%93%E7%AE%87%E6%89%80%E3%82%92%E8%A1%8C%E3%81%8D%E6%9D%A5%E3%81%99%E3%82%8B%E3%81%93%E3%81%A8%E3%81%8C%E3%81%A7%E3%81%8D%E3%81%BE%E3%81%99%E3%80%82.

[80] TeX Wiki. LaTeX の警告メッセージ. https://texwiki.texjp.org/?LaTeX%20%E3%81%AE%E8%AD%A6%E5%91%8A%E3%83%A1%E3%83%83%E3%82%BB%E3%83%BC%E3%82%B8.

[81] @t_kemmochi. イマドキの LaTeX の書き方入門. https://qiita.com/t_kemmochi/items/78064daaa3903b7925ab#%E4%BB%98%E9%8C%B2-%E7%92%B0%E5%A2%83%E6%A7%8B%E7%AF%89.

[82] @yt1114. SyncTex で 2 時間消費…VS Code で SyncTex を適用するときの注意点. https://qiita.com/yt1114/items/a50c97dafb4d193c0198.

[83] https://hackathon-22spring-13.github.io/client/.

[84] https://qiita.com/Hosi121/items/a0f7e7eb1b40b06c8459.