Introduction to LaTeX

Rwitaban Goswami and Mihir Vahanwala

June 25, 2020

Contents

1 Welcome to LaTeX

1.1 LaTeX? What? Why? How?

I know what you are thinking. Why do we need these fancy LaTeX documents that need a whole lot of hassle and pain to setup? Couldn't we simply use our trusty old MS-Word, or better still, Notepad? What is LaTeX anyway? Is it a software, or is it a programming language?

Simply put, it's a document preparation system. Well, you may ask, so is MS-Word, what's the difference?

Well, MS-Word is a WYSIWYG processor (What You See Is What You Get). Why is LaTeX better? Because for many kind of things, what you want to see is very hard to get in MS-Word, but in LaTeX it is deadass easy, you write a single line of code, and you're done!

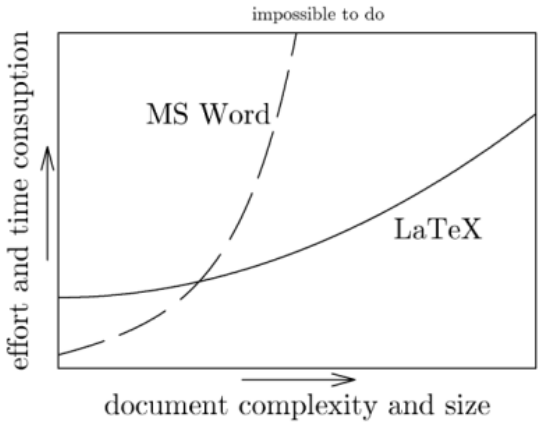
LaTeX will help you draw diagrams, create table of contents, and of course, write mathematical equations, which is what its most known for among non LaTeX people ;)

But the most important thing LaTeX does is it separates the content from the presentation. The content often goes between tags, and the presentation of that content is defined by the properties of the tags. Sounds familiar? Yes, Just like HTML, LaTeX is also a markup language, so it is terribly easy to understand.

1.2 Motivation

LaTeX (pronounced lay-tech or laa-tech, and definitely not lay-teks which will make others think you have some kind of a weird fetish) is a free, powerful, and absolutely indispensable markup tool to typeset elegant technical documents. LaTeX allows us to write complex mathematical equations without much fuss; its environments save us the hassle of organising large documents manually; with LaTeX we can showcase code and render almost any scientific illustration. LaTeX is paradise for anyone who works in STEM. Once you have experience, you can typeset assignments, papers, articles and theses with unprecedented ease, using LaTeX.

This popular graph from Marko Pinteric is indeed apt:



1.3 Deciding you LaTeX workflow

Now if you want to work with LaTeX, you have to decide your LaTeX workflow. What do we mean by that?

Well, by itself, LaTeX is just a markup language. It just defines what your text should look like in your final document. If you write a .tex file (yes, that is the extension of LaTeX files), it is not going to convert itself to the final document (typically a pdf)

So you need some software to do that conversion for you. And as all software go, you can have your software either in GUI (normal software with a graphical interface) or CLI (like a PowerShell or a Bash terminal. Don't worry if you don't recognize these terms, it is not necessary for LaTeX)

If you are looking to work in development, or think you'll be finding yourself writing any sort of code in the future, I highly recommend switching to a CLI workflow for **all** *your work* **right now.**

We have listed the possible workflows in order of the familiarity with code writing or LaTeX skillset required, so pick and choose as you see fit:

1. Overleaf
   - Difficulty level: 

- Flow: GUI
- OS: Any, use on web browser
- Setup time: None
- Description: If you don't want to install any software, and have no experience in writing code before, and don't plan on writing huge documents and/or environments of yourself, you can go for this
- How to Use: Just log in on Overleaf.com, type your LaTeX code on the right hand and click 'Recompile' or hit CTRL+S to see your pdf appear magically on the left side

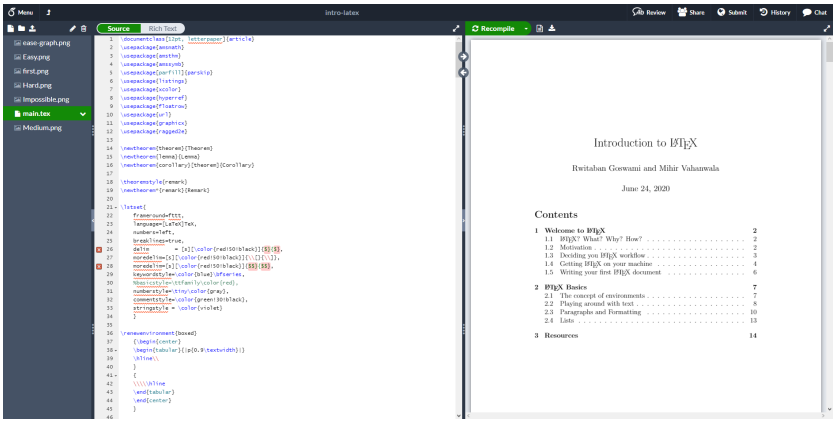


Figure 1: Overleaf

2. TeXStudio

- Difficulty level:
- Flow: GUI
- OS: Windows/Linux/OSX
- Setup time: As long as it takes to install

- Description: The wonder of this is that it provides an intuitive GUI for you to use, but the LaTeX distribution installation itself is separate from the GUI. So all you need to do is install this, and it takes care of the LaTeX distribution by itself. If you find the underlying distribution to be lacking in some packages, then it may be a bit of some pain to first find out which distribution TeXStudio is using behind the curtains, and then install packages for that. This will work nicely if you want to do the 4^{th} option but with a GUI instead of the text editor.
- How to Use: You have to install the setup according to your OS from `https://www.texstudio.org/#download`, and install the application. It provides a similar intuitive GUI to overleaf

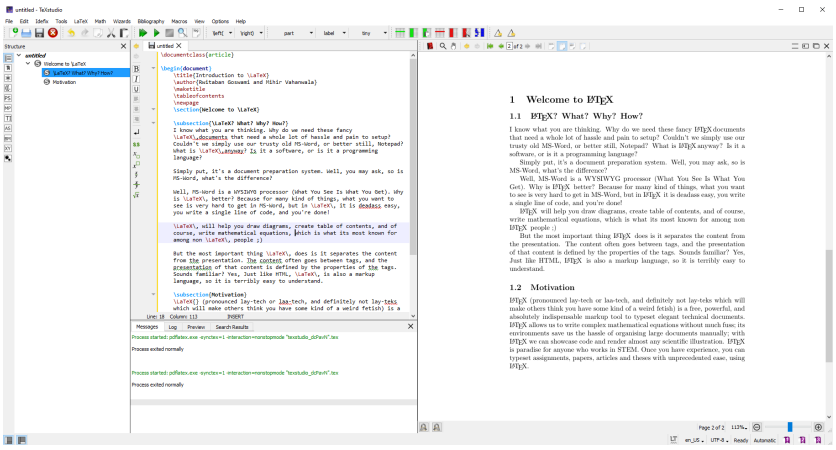


Figure 2: TexStudio

3. TeXWorks

- Difficulty level:
- Flow: GUI
- OS: Windows/Linux/OSX
- Setup time: As long as it takes to install

- Description: This works pretty similarly to 2. TeXStudio, in that it provides an intuitive GUI to use independent of the LaTeX distribution. It even lets you choose the LaTeX distribution you want, and you can setup packages for it accordingly.
- How to Use: You have to install the setup according to your OS from `http://www.tug.org/texworks/`, and install the application. It will provide a window for you to write your LaTeX code in, and when you compile it, it produces the pdf in a *separate* window

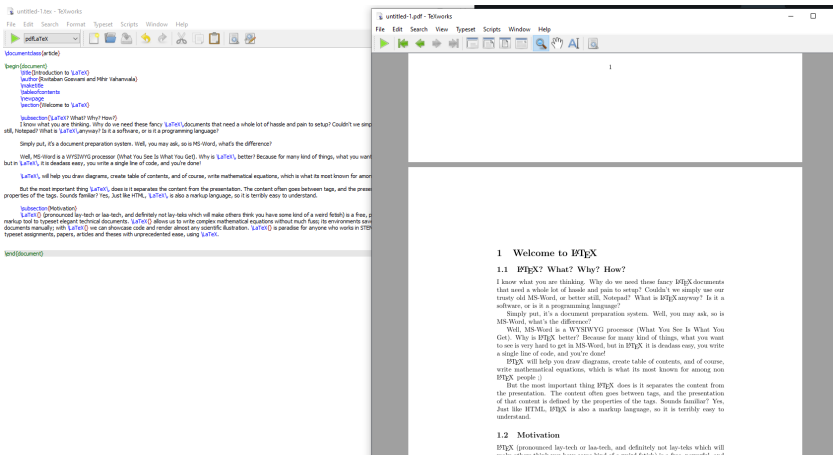


Figure 3: TexWorks

4. TeXLive + GUI Text Editor

- Difficulty level:
- Flow: CLI + Text Editor
- OS: Windows/Linux/OSX
- Setup time: As long as it takes to understand your OS's CLI, and a hell lot of time for all of TexLive packages to install
- Description: Technically, TexLive, is one of the LaTeX distributions that work under the hood for 2. TeXStudio or 3. TeXWorks.

Instead of using a GUI for utilizing the distribution, you are going to write your LaTeX code in a editor itself, and call the CLI command to compile your pdf file yourself.

- How to Use: Steps to get the TeXLive distribution depend on your OS

  (a) Linux: Open up your terminal and execute
  sudo apt−get install texlive #to get the most common packages (not including BiBLaTeX)
  or
  sudo apt−get install texlive−full #to get all of the packages , but the install time is high
  Now open up your favorite text editor (some non exhaustive options: Notepad++, Sublime Text, VSCode) and write your LaTeX code. Open up a terminal side by side. Also open the resulting pdf side by side. Whenever you want to see your pdf update, execute
  pdflatex /path/to/tex/file
  and your pdf file will be created in the same directory

  (b) Windows: Install TexLive from `https://www.tug.org/texlive/acquire-netinstall.html`. The rest of the instructions are the same as for Linux, except that you open a *cmd terminal* or a *powershell window* instead of the Linux terminal. Also, Windows may not automatically update your pdf once you execute
  pdflatex /path/to/tex/file
  so you may need to keep refreshing your pdf if your reader is a browser. Windows will also definitely now allow pdflatex to execute if your pdf is open in a reader like Acrobat

  (c) OSX: If you have homebrew, you can open your terminal and install the distribution via
  brew cask install mactex
  This will also install the TexWorks GUI along with it, so you could also use that. Otherwise it is pretty similar to working with Linux

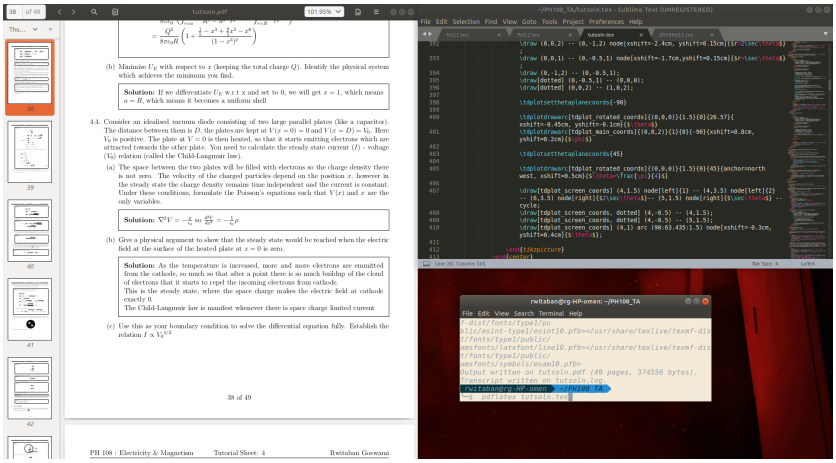


Figure 4: Sublime Text + TeXLive

5. TeXLive + VSCode + LaTeX workshop plugin

- Difficulty level:
- Flow: VSCode CLI + VSCode Editor + VSCode plugin (to view pdf)
- OS: Windows/Linux/OSX
- Setup time: Takes time and care to load plugin and all its dependancies
- Description: Does everything, from the CLI, to the text editor, to the pdf and its updation, from within VSCode itself
- How to Use: First install VSCode from `https://code.visualstudio.com/download`. Then install the LaTeX workshop plugin on VSCode by James Yu. Install TeXlive as from before. Install latexmk from `https://ctan.org/pkg/latexmk`. Once you finish setting up your VSCode settings, you are good to go. The plugin will automatically refresh the pdf for you as soon as the tex file is saved. It even does things like compile the table of contents,

the bibliography, and references in one go. It will even take your cursor from the point in the code to the point in the pdf! The only hassle is the installation

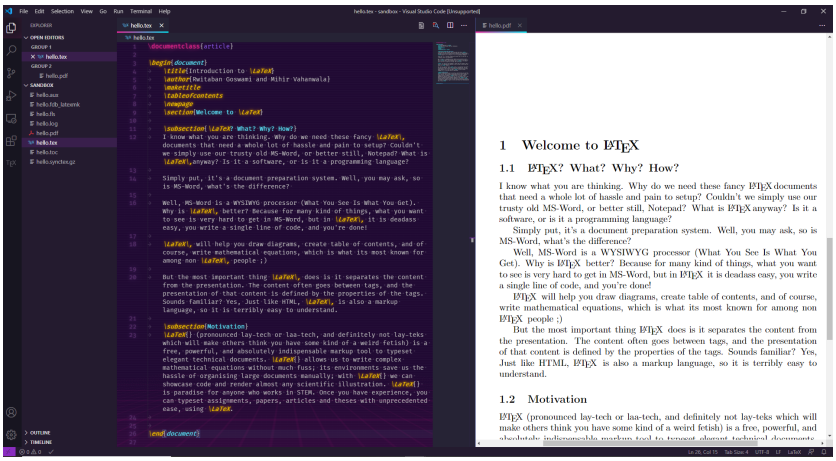


Figure 5: VSCode + TeXLive + Plugin

6. TeXLive + CLI Text Editor

- Difficulty level:
- Flow: CLI + CLI Editor
- OS: Linux/OSX (Windows but with WSL only)
- Setup time: Very steep learning curve to learn CLI text editors
- Description: Same as 4., but the text editor is now CLI. Do this if you are extremely familiar and regularly use a CLI text editor (like Vim, Nano, Emacs etc), or want to start learning it.
- How to Use: Setup of TeXLive is same as in 4. The CLI editor can also be installed through CLI. Usage is same as before, except that have only *two* windows open side by side, one with the pdf open and one with your text editor (which should have its own terminal inside it). You can also use this approach with Windows, but you can only get a CLI editor (like Vim), through WSL

(Windows Subsystem for Linux), which basically runs a command line version of Linux on your Windows (not like an emulator or a virtual machine)

Figure 6: Vim + TeXLive

1.4 Writing your first LaTeX document

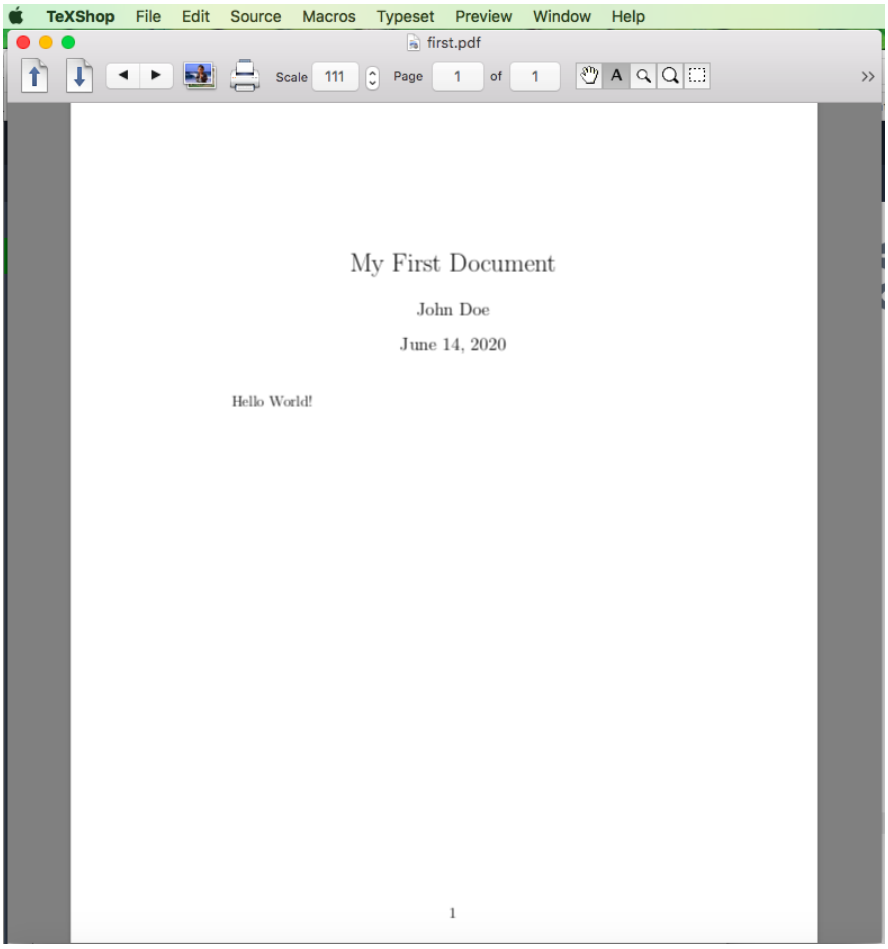
LaTeX is commonly used to typeset PDFs, and the typesetting code is written in a `.tex` file. Make a file, say, `first.tex`.

Type in the following code and then hit typeset (or use whatever flow you selected in the previous section):

```
1 \documentclass[12pt, letterpaper]{article}
2 %this tells LaTeX that the document is to be treated as
      an article, and activates features accordingly.
     Optional information about the font size and type of
      paper is also supplied.
3
4 %the percentage sign is used to make comments! This
     will not be read by the compiler!
5
6 %This part of the .tex file, before you declare \begin{
    document}, is called the PREAMBLE.
7
```

```
\begin{document}
\title{My First Document}
\author{John Doe}

%if you do not write the next line, the compile date
    will be typeset automatically
\date{June 14, 2020}

\maketitle

Hello World!
%This is a quiet comment.

\end{document}
```

Look, we did it ourselves, and here's what we got.



Try it!

2 LaTeX Basics

2.1 The concept of environments

Observe the source code of your first document. It has a preamble, and a body that is contained between `\begin{document}` and `\end{document}`. Every document has this structure.

The pair `\begin{document}` and `\end{document}` are said to define the *document environment.* All LaTeX files have the document environment, and the

`\documentclass{}` in their preamble, but that is not the only environment that is. Environments are used to format blocks of text in LaTeX documents. Environments are delimited by the opening tag `\begin` and the closing tag `\end` Everything inside those tags will be formatted specially, depending on the environment. Environments can be nested: of course, as a trivial example, all the environments we will use are contained in the document environment.

In this tutorial itself, we will see simple applications of environments to center-align text, and also to create lists. Later on in the course, the usage of environments will be much more involved.

2.2 Playing around with text

This is plain text. **This is bold text**. *This is italicised text. Another way.* THIS IS CAPITALISED TEXT. This is underlined text. ***This is bold text. Oh wait, it's italicised as well.*** This is blue text. I made a command to turn the default colour violet for the rest of this demo. Or do you prefer a touch of grey? I have fifty shades, actually. Darker. This colour is what light of wavelength 600 nm looks like. This turquoise is prepared by mixing blue, green and a hint of black. This colour is the complement of green. I wonder if it has a name more specific than pink. Back to violet, it's default. This is tiny text. This is small text. This is regular sized text. This is large text. This is huge text.

In order to get these shades, we used the `xcolor` package. To make the code below work, type `\usepackage{xcolor}` in the preamble, i.e. before the declaration `\begin{document}`

Here's how we did it.

```
This is plain text.
\textbf{This is bold text}.
\textit{This is italicised text.}
\emph{Another way.}
\textsc{This is capitalised text.}
\underline{This is underlined text.}
\textbf{\textit{This is bold text. Oh wait, it's
   italicised as well.}} \textcolor{blue}{This is blue
   text.}
\color{violet}
I made a command to turn the default colour violet for
   the rest of this demo.
\textcolor{black!70}{Or do you prefer a touch of grey?}
\textcolor{black!30}{I have fifty shades, actually.}
\textcolor{black}{Darker.}
\textcolor[wave]{600}{This colour is what light of
   wavelength 600 nm looks like.}
\textcolor{blue!40!green!55!black}{This turquoise is
   prepared by mixing blue, green and a hint of black.}
\textcolor{-green}{This colour is the complement of
   green. I wonder if it has a name more specific than
   pink.}
Back to violet, it's default.
\tiny{This is tiny text.}
\small {This is small text.}
This is regular sized text.
\large{This is large text.}
\huge{This is huge text.}
```

Play around with colours (or colors, if you prefer American), and show us a glimpse your favourite shade in the assignments!

2.3 Paragraphs and Formatting

We recommend using the `ragged2e` package to toggle between justifications. Type `\usepackage{ragged2e}` in the preamble.

The set of paragraphs in this box are Center Aligned using the center environment. To finish this paragraph and start a new one, you could use the enter key twice.

Like so. Another way to tell the 'compiler' that you'd like to typeset a new line, is to use double backslash.

See? It worked. There's yet another command to start a new paragraph, as you will see in the code.

There we go. Observe the code. We have a way of introducing vertical space between paragraphs.

Like this. We can also introduce horizontal space.

Here is the complete source code:

```
\begin{center}
    The set of paragraphs in this box are Center
       Aligned using the center environment. To finish
       this paragraph and start a new one, you could
       use the enter key twice.

    Like so. Another way to tell the 'compiler' that
       you'd like to typeset a new line, is to use
       double backslash. \\
    See? It worked. There's yet another command to
       start a new paragraph, as you will see in the
       code. \par
    There we go. Observe the code. We have a way of
       introducing vertical space between paragraphs.

    \vspace{1.5em}

```

```
10      Like this. \hspace{2.5em} We can also introduce
           horizontal space.
11  \end{center}
```

Before beginning this paragraph, we declared the command `\raggedright`, and now, as you can see, our text is left justified. We will now add a few inconsequential lines, just to make the paragraph bigger, and the effect more pronounced.

We will now add a few inconsequential lines, just to make the paragraph bigger, and the effect more pronounced. The left justification is here to stay, until we do something about it. Should we? Well, there's a lesson to teach, so...

Before beginning this paragraph, we declared the command `\raggedleft`, and now, as you can see, our text is right justified. We will now add a few inconsequential lines, just to make the paragraph bigger, and the effect more pronounced.

We will now add a few inconsequential lines, just to make the paragraph bigger, and the effect more pronounced. The right justification is here to stay, until we do something about it. Should we? Well, there's a lesson to teach, so...

Finally, we declared the command `\justifying` right before this paragraph so our text is justified as before, and all is good. We will now add a few inconsequential lines, just to make the paragraph bigger, and the effect more pronounced.

The beginning of this paragraph has a lot of indentation. This is because we issued the command `\setlength{\parindent}{7em}` right before it.

The super indentation is here to stay until we tinker with it again.

This is reminiscent of the time we'd use our fingers to decide indentation in our notebooks.

And this returns things to the way they were before.

All we did this time was declare `\setlength{\parindent}{0em}`. If you

want to switch off indentation for a particular paragraph, use `\noindent` right before it.

This document has `\usepackage[parfill]{parskip}` declared in its preamble. This is convenient to have vertical separation between paragraphs: a series of empty lines in the source code is typeset as an empty line in the PDF. And, for some reason, Mihir does not like the beginning of paragraphs to be indented. Make sure you import this *before* `ragged2e`.

We will show you the source code for the next few lines. Honestly, we have no idea why it sounds like 'clickbait'.

Rwitaban and Mihir wondered what's the best way to format paragraphs, till they came across this package.

There's enough space in the source to fit a truck, but they'll have no idea when we typeset it.

```
1 We will show you the source code for the next few lines
    . Honestly, we have no idea why it sounds like '
    clickbait'.
2
3 Rwitaban and Mihir wondered what's the best way to
    format paragraphs, till they came across this
    package.
4
5
6
7
8 There's enough space in the source to fit a truck, but
    they'll have no idea when we typeset it.
```

A question that must've popped in your mind: What exactly does `2.5em` mean in a command that deals with length? `em` is a unit of length that is roughly equal to the width of an uppercase 'M' in the current font. The math unit `mu` is related: `em` = 18 `mu`, where `em` is from the math symbols family.

Another unit defined this way is `ex`: roughly the height of a lowercase 'x' in the current font.

Then of course, there are more standard units like `pt`, `cm`, `mm` and `in` (which

stands for inch).

The most commonly used lengths include `\linewidth` (width of the line in the current environment), `\parskip` (vertical space between paragraphs), `\topmargin` (length of the top margin). The linked Overleaf tutorial is a helpful guide.

2.4 Lists

- This is a list.
- It is an unordered list.
- It is created using the 'itemize' environment.
  - Environments can be nested.
  - So can unordered lists.
- Look at the code below to see how it's done.

```
\begin{itemize}
    \item This is a list.
    \item It is an unordered list.
    \item It is created using the 'itemize' environment.
    \begin{itemize}
        \item Environments can be nested.
        \item So can unordered lists.
    \end{itemize}
    \item Look at the code below to see how it's done.
    %This is the code.
\end{itemize}
```

1. This is an ordered list.
2. The only difference in the code is, `itemize` is replaced by `enumerate`.
   (a) Nesting
      i. More nesting
         A. Man stop.
         B. This is the deepest nesting that is supported.

LaTeX automatically renders nested lists, both ordered and unordered, in distinct styles. Changing these styles is, in our opinion, a bit too finicky, and beyond the scope of this basic tutorial. You will learn more about it, once you gain mastery over environments and commands.

Here's a way to make descriptive lists:

MA 105 Calculus
MA 106 Linear Algebra
MA 108 Differential Equations

The code is simple enough:

```
\begin{description}
\item [\textbf{MA 105}] Calculus
\item [\textbf{MA 106}] Linear Algebra
\item [\textbf{MA 108}] Differential Equations
\end{description}
```

3 Resources

This, and subsequent tutorials are written with the intention to motivate and help you get started. We recommend you follow the tutorials and try out what we demonstrate yourself. The majority of learning will take place on

the job, that is, when you type in code yourself. Here are pointers to some helpful resources; they even helped us ~~blatantly copy~~ design this material.

- `https://www.overleaf.com/learn/latex/Tutorials`
- `https://www.latex-tutorial.com/tutorials/`

These sources are vast, and will prove useful whenever you sit down to write LaTeX. Also, if you run into a wall, a simple Google search is an underrated trick; you'll often find a helpful TeX StackExchange thread (or any similar forum) in the first few results. Moreover, we encourage you to follow the links in our tutorials, and check out the citations, if any.