LaTeX for Undergraduates[*]

Patrick Bartol, Will Irvin, Andrew Lounsbury,
Josie Millican, Madison Pearson, John Puckett,
Adriana Rosalia, Alizza Schremp, Ryland Smith

Math Club
Tennessee Technological University
Cookeville, TN, 38505

Fall 2020 – Spring 2021

To put a title on a simple homework assignment, you only need lines `33-41` and `46-47` in the code (`LaTeX_for_Undergraduates.tex`) instead of an entire `titlepage`, but you don't necessarily have to use this method. You can put the title, your name, and the date on the paper however you'd like.

As you read this, use the keyboard shortcut Ctrl + F to search for specific things in the code so that you can compare the code precisely to what's being printed.

---

[*]This paper was jointly written by the TTU Math Club; however, any singular first-person pronouns refer to Andrew.

Throughout this paper, there are some commands from the accompanying `mymacros` package, but we forgo further reference to `mymacros.sty`. If you use a command from this tutorial in another document and it says the command is undefined, you probably need to install `mymacros` and include it in the preamble.

Contents

1 Setting Up

1.1 MiKTeX

Click here to download MiKTeX. Install it to the default directory. When it asks if MiKTeX should install packages on the fly, say "**yes**."

1.2 LaTeX Editors: Visual Studio Code

We recommend Visual Studio Code for Windows users and (maybe) Mac users.[1] To download the User Installer for VS Code, click here. When it gives the option to add the "Open with Code" action to file and directory context menus, **check both** boxes.

Open `LaTeX_for_Undergraduates.tex` in VS Code. With the button on the left sidebar, install the following:

LaTeX Workshop

Code Spell Checker

Notepad++ keymap

Initially, there is a wide scrollbar that shows a zoomed-out image of your code. You can toggle this under `View -> Show Minimap`.

Click the gear in the bottom left, select `Settings`, and change the following:[2]

| Editor: Word Wrap	**on**
Auto Find in Selection	**multiline**
LaTeX Workshop > Message > Error: Show	**uncheck** the box
Workbench: Startup Editor	**none**

Try building the file with the green button in the top-right.

Windows users may receive an error in the bottom left regarding a file `perl.exe` from a programming language called **Perl**. Click here to download Perl. There is a choice between *Strawberry Perl* and *ActiveState Perl*. A Stack Exchange page said that ActiveState Perl is better for beginners, so it's probably fine to download that one.

Now, you may need to add the path to your Perl installation to the Path variable. If so, search your computer for the option to edit your *Environment Variables* in your *System Properties*. Select `Environment Variables...` Under *System variables*, select **Path** and click `Edit...`.

---

[1]If you can't or don't want to use VS Code, see here and here.
[2]There is a button at the top right that opens a `json` file with all overridden settings.
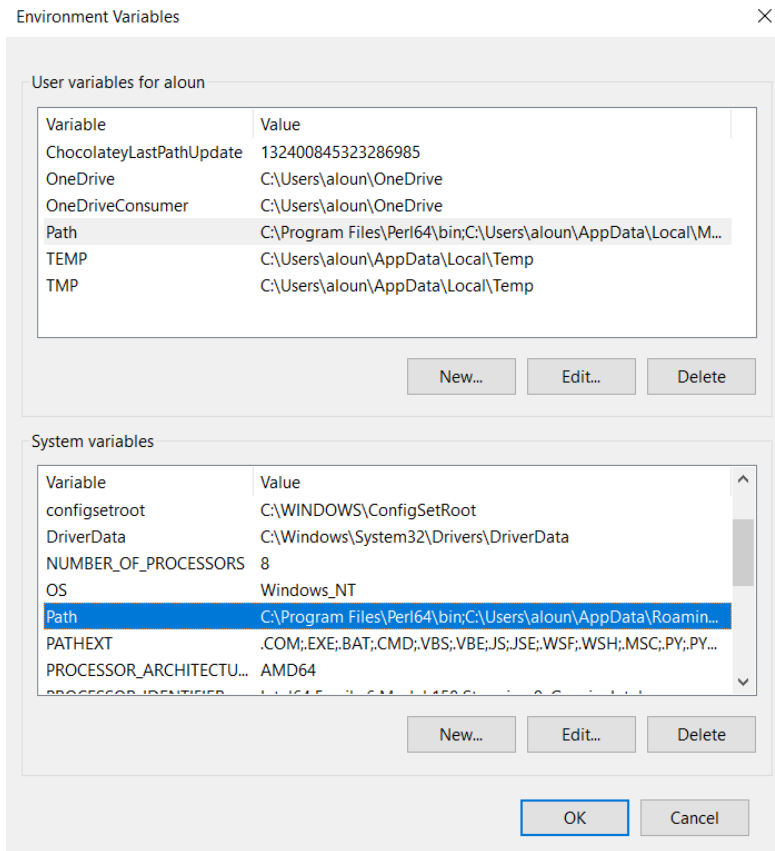
Add the path to your Perl installation with the `New` button on the right:
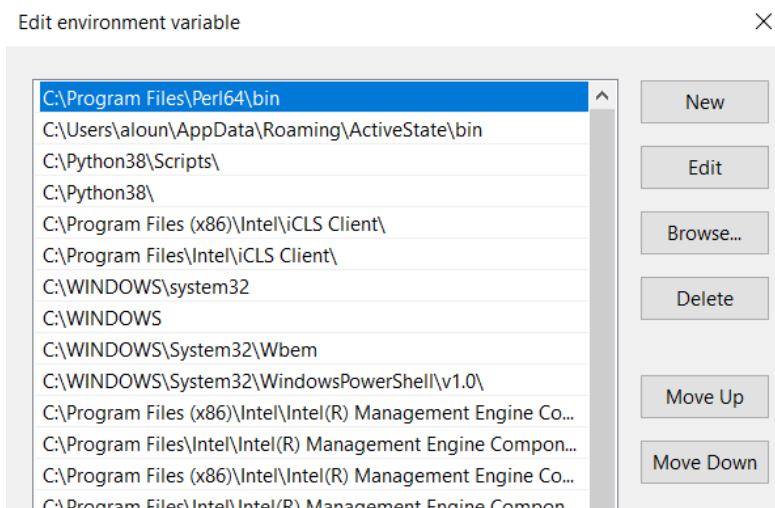
Try building the file again to ensure everything works properly. It will likely take longer than usual because it has to install the necessary packages, because it has to generate the auxiliary files for the first time, and because this is a nineteen-page document. Be sure to press the button in the top right with the magnifying glass to open the `pdf` and see the result of the compilation.
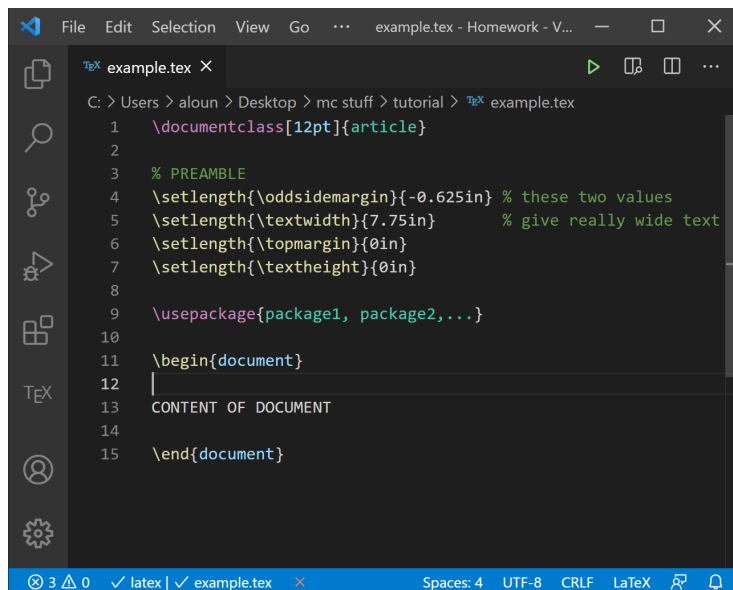
1.3   Creating a Document in VS Code

Create new files and open files under `File`. Regarding the three buttons ▷ ▯ ▯ in the top-right corner, the play button on the left compiles the file, the middle button with the magnifying glass displays the `pdf`, and the button on the right splits the editor so that you can open multiple files or open a duplicate view of the same file.[3] Note, however, that VS Code compiles every time you save.

View error messages by clicking the icons in the bottom-left corner.

The font size `[12pt]` is optional and may be set to sizes other than `12`. The `article` class is for regular documents.[4] The preamble is everything between `\documentclass{article}` and `\begin{document}`, in other words, your margins and text dimensions, which are optional, and your packages.

Regarding the content of the document, writing in LaTeX can generally be summarized as (1) typing regular text, (2) typing math with `$...$`, (3) displaying math with `\[...\]`, and (4) learning commands and environments as you need them. Any text contained in `\begin{⟨env-name⟩}...\end{⟨env-name⟩}` is in the ⟨env-name⟩ environment.

---

[3]I've noticed that when I close my laptop with a document and its `pdf` open, that `pdf` will no longer update when I recompile. Closing the `pdf` in the viewer and opening it again seems to consistently fix this.

[4]The `beamer` class allows us to make presentation slides and posters in LaTeX.

1.4 Keyboard Shortcuts in VS Code

Remember to make ample use of the following shortcuts.

Source[5]	Shortcut	Action
VSC	Ctrl + A	highlight all
N++	Ctrl + D	duplicate
LW	Ctrl + E	select current environment
VSC	Ctrl + F & Ctrl + H	search & replace
LW	Ctrl + K	kill compiler process
N++	Ctrl + Q	toggle comments
LW	Ctrl + T	toggle between $\backslash[\ldots\backslash]$ and the `equation*` environment
LW	Ctrl + W	wrap in environment
VSC	Ctrl + Z & Ctrl + Y	undo & redo (even autocompletion)
VSC	Double Click	highlight string
VSC	Highlight + Delimiter	surround with $\{\ldots\}$, $(\ldots)$, $[\ldots]$, $\$\ldots\$$
VSC	Shift + Click	highlight everything between the text cursor and your mouse cursor
VSC	Tab & Shift + Tab	tab/untab multiple lines; next/previous argument in a macro
VSC	Tab or Enter	autocomplete suggested text
VSC	Shift + Enter	prevent an autocompletion

These save lots of time. If your editor doesn't have them, you may want to look through your settings and add these shortcuts or change them if they're different from what you want them to be. In VS Code, you can view your keyboard shortcuts in the settings at the bottom-left.[6]

1.4.1 Autocompletion

VS Code allows us to autocomplete any string of regular text provided the string has already been typed into the document once. It doesn't give the option to autocomplete "Nullstellensatz" in this document because we haven't typed it yet, but if we type "Nullstellensatz" again, it gives us the option.

For commands, autocompletion doesn't trigger in new, unsaved files nor for commands that haven't already been typed *and saved* into the document once. Upon typing `\msum` for the first time, we don't have the option to autocomplete it and successively enter its arguments, but upon adding the three $\{\ldots\}$ to the first instance, saving the document, and typing `\msum` a second time, we can press Tab and VS Code automatically types `\msum{|}{}{}` with the braces ready and the text cursor inside the first set of braces.

---

[5]VSC = VS Code; LW = LaTeX Workshop; N++ = Notepad++ keymap

NOTE : These LaTeX Workshop shortcuts must be added in the shortcut menu.

[6]There is a button at the top right in the shortcuts menu that opens the `json` file with all overridden shortcuts.

2   Text

Regular text is said to be in paragraph mode.

2.1   Formatting Text

Comment things out with `%`.[7]
Create a new line without indentation using `\\`.
Create a new paragraph with indentation either with `\par` or by leaving an empty line in the code.
An empty line with a comment symbol will not create a new line. See code. We can stop the indentation on an empty line like this.[8]

1. `\par` doesn't indent

   when placed in certain environments such as `enumerate`.

   Nor does an empty line.

It's good to use these in such a way that makes your code look similar to what's being printed. Using `\\` along with an empty line will create an empty line in both your code and your document.

This will produce *italicized* text.
This will produce **boldface** text.
This will produce SMALL CAPS text.
This will produce `typewriter (teletype)` font.

Changing the ⟨*font-size*⟩ option in `\documentclass[`⟨*font-size*⟩`]{article}` will affect the way the following sizes work. For instance, `[12pt]` makes `\huge` and `\Huge` appear the same.

Whereof one cannot speak, thereof one must be silent.
Whereof one cannot speak, thereof one must be silent.
Whereof one cannot speak, thereof one must be silent.
Whereof one cannot speak, thereof one must be silent.
Whereof one cannot speak, thereof one must be silent.
Whereof one cannot speak, thereof one must be silent.
Whereof one cannot speak, thereof one must be silent.
Whereof one cannot speak, thereof one must be silent.

---

[7]There is a `comment` environment in the `verbatim` package for creating block comments. The benefit of this is that we can minimize environments in VS Code so that we don't have to constantly scroll through code we're not reading. We could otherwise highlight the multiple lines we want to comment out and press Ctrl + Q (or an equivalent shortcut).

[8]We wouldn't use `\par` unless we specifically wanted the indentation.

This is centered.

This is left-justified.

This is right-justified.

You can insert horizontal space   like this, or insert vertical space

like this.

2.2   Lists

1. This is `enumerate`.[9]

label #2 Though it numbers items automatically, it's good to label the items so that you can tell the difference between all the `\item`'s in your code. Note that you can change the labels with brackets.

3. (a) Once-nested `enumerate` environment
   (b)   i. Twice-nested
        ii. A. Thrice-nested
            B. This is the maximum number of times you can nest them.

- This is `itemize`.

- Good labeling for `itemize` might be "dot", "dash", "star", "point," but the utility of this depends on whether you put `\item` on its own line or the same line as your text.

- – Once-nested `itemize` environment
    – ∗ Twice-nested
      ∗ · Thrice-nested
         · This is the maximum number of times you can nest them.

There is also	the `tabular`	environment.
It's like an	array without	math mode.
Each column alignment	must be	indicated.
`l` = left-justified		
	`c` = centered	
		`r` = right-justified

---

[9]The `enumitem` package provides additional functionality for `enumerate`.

3 Math

Mathematical text is said to be in math mode.

3.1 Formatting Math

$Mathmodeisitalicizedbydefault.$
This will produce regular text with whitespace.
This will produce **bold text with whitespace.**
This will produce *italicized text with whitespace.*
This will produce SMALL CAPS TEXT WITH WHITESPACE.
This will produce **boldfacemathwithoutwhitespace.**[10]
This will produce roman(nonitalicized serif font)textwithoutwhitespace.

Theorem (Commands & Trouble-Shooting)**.** *Regarding LaTeX commands, there is plenty of information at your disposal on the internet. You will likely be able to solve most of the problems you encounter and find commands and packages you need by searching for them on the internet.*

Proof. Search for a list of commands such as this or this. Search for a package that suites your needs. Search for the error your compiler is giving. Stack Exchange, Overleaf, and Wikibooks are fantastic resources.

Under the LaTeX menu on the left sidebar in VS Code, there are two Snippet panels with a myriad of symbols to browse through. Relying on these menus is ill-advised, but being aware of them may still be worthwhile. Here are just a few commands worth noting:

1. $\sin\theta_1 = \cos\theta_2 = \max\{0,1\} = \min\{1,2\} = 3 \bmod 2 = \sqrt[4]{1} = \pm 1$

2. $\exists \ell \in \cap_{i=1}^n A_i$ such that $f \circ g(\ell) = \gcd(7,\ell) = \operatorname{lcm}(7,\ell) = 10 = \underbrace{1 + \cdots + 1}_{10 \text{ times}}$

3. (a) $\dim V = \operatorname{rank}(T) + \dim \operatorname{Null}(T)$, where $T \in \mathcal{L}(V,W)$
   (b) $|\mathcal{P}(S)| = 2^{|S|}$

4. $\mathscr{L}\{f(t-a)\mathscr{U}(t-a)\} = e^{-as}F(s)$

5. $\{x_n\}_{n=1}^\infty \subset \mathbb{R}$ is such that $\inf x_n \leq \sup x_n$ and $\liminf x_n \leq \limsup x_n$. ∎

Notation 1. $\{\ldots\}$ are required around subscripts and superscripts with more than one character (except for commands such as ∞, greek letters, etc.)

Claim. *The statement $x_m^{n+1} + x_{m+1}^n \approx 2^\pi \nless 8 \leq 2^\infty$ is*

1. *in inline math mode and*

2. *has none of the extra white-space from the code.*

---

[10]See Notation 7 for a possible alternative.

Proof of (1). Since blah blah blah, we have blah blee blah, which gives hm hm hm, blah blah. Hence, bloo blah blee bloo. Therefore, the statement is in inline math mode.

Proof of (2). Note that we temporarily changed the \qedsymbol in the previous proof environment so that it wouldn't show up twice. Also, note that we can change the title of any environment with brackets. ∎

Proposition (Display-Math Mode)**.** *The summation*

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \in \Theta(n^3)$$

is in display-math mode. Note that the first letters of commands for capital Greek letters are capitalized and that there are some lowercase Greek letters with variations whose commands begin with var*, such as φ and ε.*

Proof. Omitted ∎

Corollary (equation & multline)**.** *The equations*

$$\mathbf{I}(\mathbf{V}(I)) = \sqrt{I} \tag{1}$$

and

$$\begin{aligned} f_x = 3x^2y^4 + 6xy^5 + 3y^6 - 8x^3y^2 - 18x^2y^3 - 12xy^4 - 2y^5 \\ + 5x^4 + 12x^3y + 9x^2y^2 + 2xy^3 \end{aligned} \tag{2}$$

are respectively in the equation *and* multline *environments, which number single lines of display-math. The* multline *environment is useful for splitting single lines.*

Proof. Omitted ∎

Notation 2 (Extensible Block Delimiters)**.** The commands \left and \right resize the succeeding character to fit whatever's between them. The null delimiters \left. and \right. respectively leave the left and right character blank, but there must be both a \left⟨*something*⟩ and a \right⟨*something*⟩. Note that we can create extensible angle brackets with \left< and \right>:

$$\left\langle \frac{x}{2}, \frac{y}{2} \right\rangle = \left\{ h_1\left(\frac{x}{2}\right) + h_2\left(\frac{y}{2}\right) \,\Big|\, h_1, h_2 \in \mathbf{F}[x_1, \ldots, x_n] \right\}.$$

Notation 3 (Punctuation)**.** When display-math mode ends a phrase, the punctuation *must* be inside the display. For instance, it is true that

$$(0,1) = \bigcup_{n=1}^{\infty} \left(0 + \frac{1}{n}, 1 - \frac{1}{n} \right) \neq \varnothing,$$

but it may be true that $\gamma \in \mathbb{Q}$ or that $\gamma \notin \mathbb{Q}$. Hence, we may write $\gamma \stackrel{?}{\in} \mathbb{Q}$.

Notation 4 (Readability)**.** Separate long strings of equalities over several lines of code. It's better to write

```
15
= 10 + 5
= 3 + 7 + 5
= 1 + 2 + 3 + 4 + 5
```

rather than

```
15 = 10 + 5 = 3 + 7 + 5 = 1 + 2 + 3 + 4 + 5
```

Notation 5 (Such That)**.** There are two ways to say "such that" in a set definition. If $I, J \subseteq \mathbf{F}[x_1, \ldots, x_n]$ are ideals, where $\mathbf{F}$ is a field, then

$$\sqrt{I} = \{f \in \mathbf{F}[x_1, \ldots, x_n] : f^m \in I \text{ for some } m \in \mathbb{N}\},$$

and

$$I : J = \{f \in \mathbf{F}[x_1, \ldots, x_n] \mid fg \in I \; \forall g \in J\}.^{11}$$

Notation 6 (Forcing Display-Math)**.** Various notations get scrunched up in inline math: $\sum_{i=1}^{n}, \prod_{i=1}^{n}, \bigcap_{i=1}^{n}, \bigcup_{i=1}^{n}, \lim_{n\to\infty}, \sup_n(b_n), \int_a^b, \max_n\{\delta_n\}$, etc.

We can often override this by putting \limits after each command. So, we can write $\sum\limits_{i=1}^{n}, \prod\limits_{i=1}^{n}, \bigcap\limits_{i=1}^{n}, \bigcup\limits_{i=1}^{n}, \lim\limits_{n\to\infty}, \sup\limits_{n}(b_n), \max\limits_{n}\{\delta_n\}$.

Integrals can be put in \displaystyle: $\displaystyle\int_a^b$, but \limits does provide an alternative way of writing them: $\int\limits_a^b$.

This is not necessary when these are being displayed, except for \int\limits.

$$\sum_{i=1}^{n}, \quad \prod_{i=1}^{n}, \quad \bigcap_{i=1}^{n}, \quad \bigcup_{i=1}^{n}, \quad \lim_{n\to\infty}, \quad \sup_{n}(b_n), \quad \max_{n}\{\delta_n\}, \quad \int_a^b, \text{ and } \int\limits_a^b$$

Notation 7 (Bold Math Symbols)**.** In math mode, mathematical symbols will not be affected by \mathbf{}, as in $\sin \pi \notin \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$. To remedy this, use \boldsymbol{} as in $\boldsymbol{\sin \pi \notin \{1, 2, 3\}}$.

When nesting block delimiters inside of each other, it is often beneficial to use an abbreviation of \boldsymbol to bold the outer delimiters. For instance, $g^{-1}\boldsymbol{(}(c, d)\boldsymbol{)}$ is a little better than $g^{-1}((c, d))$.

3.2 Displaying Multiple Lines of Math

There are several ways to display multiple lines of math. Some environments have predefined numbering. Add a * when you `begin` and `end` the environment to remove numbering. Note that \\ creates a new row.

---

[11] This is not the | character on your keyboard. It's \mid. Also, note the use of \set and the extensible \Mid in Notation 2 above.

3.2.1 `gather`

The `gather` environment is like a multiline `center` environment that is in math mode by default. Let $V_1, \ldots, V_n \subseteq \mathbf{F}^n$ be varieties, where $\mathbf{F}$ is a field. Then,

$$\varnothing \subseteq V_1 \subseteq \cdots \subseteq V_n \tag{3}$$
$$\Rightarrow \mathbf{I}(\varnothing) \supseteq \mathbf{I}(V_1) \supseteq \cdots \supseteq \mathbf{I}(V_n) \tag{4}$$
$$\text{(toggle individual line numbers with } \backslash\texttt{notag)}$$
$$\Rightarrow \mathbf{V}(\mathbf{I}(\varnothing)) \subseteq \mathbf{V}(\mathbf{I}(V_1)) \subseteq \cdots \subseteq \mathbf{V}(\mathbf{I}(V_n)) \tag{5}$$
$$\Rightarrow \varnothing = \overline{\varnothing} \subseteq \overline{V_1} \subseteq \cdots \subseteq \overline{V_n}. \tag{6}$$

Notation 8 (Dots)**.** The command $\backslash$`ldots` produces ellipses at the bottom of the line, and $\backslash$`cdots` produces ellipses in the middle of the line.

The command $\backslash$`dots` is supposed to be able to tell the difference between cases that require lower dots and cases that require center dots. There are a few instances where it doesn't get it right, but I tend to use both $\backslash$`cdots` and $\backslash$`dots`. See §3.2.4 for an example of when $\backslash$`dots` doesn't work properly.

Both $\backslash$`ldots` and $\backslash$`dots` work in paragraph and math mode, but $\backslash$`cdots` only works in math mode.

3.2.2 `align` & `alignat`

The `align` and `alignat` environments are in math mode by default. In the `align` environment, `&` indicates an alignment.

$$f'(x) = \int f''(x)\,dx = \iint f'''(x)\,dxdx = \iiint f^{(4)}(x)\,dxdxdx = \frac{d}{dx}f(x)$$
$$= \frac{d^2}{dx^2}\int f(x)\,dx$$
$$= \frac{\partial}{\partial x}f(x)\,dx$$

$$A = \{x \in S \mid n_a(x) \geq 1\} \qquad \overline{A} = \{x \in S \mid n_a(x) = 0\}$$
$$B = \{x \in S \mid n_b(x) \geq 2\} \qquad \overline{B} = \{x \in S \mid n_b(x) < 2\}$$

In the `alignat` environment, the number of alignments must be indicated, and every alignment past the first requires two `&`'s.

$$|\overline{A_1} \cap \overline{A_2}| = |S| - |A_1| - |A_2|$$
$$+ |A_1 \cap A_2|$$
$$= 10 - 1 - 2$$
$$+ 3$$
$$= 10$$

3.2.3 `array`

Math mode must be specified for arrays.
`l` = left-justified

$$(\forall r > 0)(\exists q \neq p)[q \in B(p, r) \cap E]$$
$$\Rightarrow p \in E'$$

`c` = centered

$$(\forall r > 0)(\exists q \neq p)[q \in B(p, r) \cap E]$$
$$\Rightarrow p \in E'$$

`r` = right-justified

$$(\forall r > 0)(\exists q \neq p)[q \in B(p, r) \cap E]$$
$$\Rightarrow p \in E'$$

Arrays may also be used to display charts in math mode, and can be nested inside other arrays. For example, the Invisible Hand game G_1 and the Prisoner's Dilemma G_2 are two arrays inside of another array.

G_1	A	B
A	$\mathbf{8,8}$	$\mathbf{6},6$
B	$6,\mathbf{6}$	$2,2$

G_2	A	B
A	$3,3$	$1,\mathbf{5}$
B	$\mathbf{5},1$	$\mathbf{2,2}$

3.2.4 Matrices

Math mode must be specified for matrices.

$$\begin{bmatrix} 2 & 2 \\ 0 & 1 \end{bmatrix} \xrightarrow{r_1 \mapsto \frac{1}{2}r_1} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \xrightarrow{r_1 \mapsto -r_2 + r_1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{pmatrix} 2 & 2 \\ 0 & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

Use the `array` environment to separate rows and columns in a matrix.

$$[A \mid I_2] = \left[\begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ 3 & 4 & 0 & 1 \end{array} \right] \qquad \left[\begin{array}{c|c} a & b \\ \hline c & d \end{array} \right]$$

Use $\cdots$, $\vdots$, and $\ddots$ to apostrophize matrix entries. Note that I made a command that rotates $\ddots$ to my liking and that `\dots` doesn't know that it should be displaying `\cdots`.

$$\begin{bmatrix} \sigma_1 & 0 & \cdots & & \cdots & 0 & \cdots & 0 \\ 0 & \ddots & & & & \vdots & & \vdots \\ \vdots & & \sigma_r & & & & \ddots & \\ 0 & & & 0 & & \vdots & & \vdots \\ 0 & 0 & \cdots & & \ddots & 0 & \cdots & 0 \end{bmatrix}$$

The `kbordermatrix` package displays matrices with row and column labels. Set the delimiters `\kbldelim` and `\kbrdelim` in the preamble. See line `12`.

$$\begin{array}{c} \begin{array}{cc} a & b \end{array} \\ \begin{array}{c} a \\ b \end{array} \left(\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right) \end{array}$$

3.2.5 `cases`

Definition. Let X be an infinite set. Then, $d : X \times X \to \{0, 1\}$ defined by

$$d(p, q) = \begin{cases} 1, & p \neq q \\ 0, & p = q \end{cases}$$

is formatted with the `cases` environment, which requires math mode. Either $(p, q) \mapsto 0$ or $(p, q) \mapsto 1 \ \forall (p, q) \in X \times X$, and $d(p, q) = 0 \iff p = q$.

3.3 Escape Sequences and White-space in Math Mode

Use $\backslash$ to insert characters that would otherwise be used in LaTeX syntax. Some examples are $\_$, %, #, $, {, }, and similarly, `\lbrack` and `\rbrack`.

Additionally, there are various ways to insert a space in math mode (but some of these may also be used in paragraph mode.)

`\!` (negative thin)	$\phi(x)\forall x$	$\rightarrow\!\!\!\!\leftarrow$
(normal)	$\phi(x)\forall x$	$\rightarrow\leftarrow$
`\,`	$\phi(x)\,\forall x$	$\rightarrow\leftarrow$
`\:`	$\phi(x)\:\forall x$	$\rightarrow\leftarrow$
`\;`	$\phi(x)\;\forall x$	$\rightarrow\leftarrow$
`\ ` (slash space)	$\phi(x)\ \forall x$	$\rightarrow\leftarrow$
$\sim$ (tilde)	$\phi(x)\ \forall x$	$\rightarrow\ \leftarrow$
`\quad`	$\phi(x)\quad\forall x$	$\rightarrow\quad\leftarrow$
`\qquad`	$\phi(x)\qquad\forall x$	$\rightarrow\qquad\leftarrow$

Tildes are an example of unbreakable space, meaning the adjacent strings won't be separated over line breaks.[12] E.g.,

- -
string1 string2

Rather than printing   string1
string2,
it keeps the two strings on the same line. If we remove the space between the last dash and string1, both strings will move to the previous line. This could be useful for keeping inline mathematical statements together; it may sometimes be difficult to read a formula broken up over two lines.

Tildes are unbreakable in math mode and in paragraph mode. Some other options here are only unbreakable in paragraph mode; e.g., $(\backslash,)$, $(\backslash:)$, and $(\backslash;)$.

---

[12] Note that, for unknown reasons, the two tildes above didn't produce the same space, so tildes are perhaps unreliable.

4 Macros

The syntax `\newcommand{\macroname}{`⟨*definition*⟩`}` creates a macro instruction invoked with `\macroname` that inserts the ⟨*definition*⟩[13] wherever `\macroname` appears. The syntax `\newcommand*{\macroname}[n]{`⟨*definition*⟩`}`[14] is invoked with `n` arguments ⟨a_1⟩, . . . , ⟨a_n⟩ as `\macroname{`⟨a_1⟩`}`· · ·`{`⟨a_n⟩`}`. In the ⟨*definition*⟩, `#i` indicates where the `i`-th argument ⟨a_i⟩ will be placed, and `\hfill` inserts white-space until a space—the space in a matrix entry, for instance—is filled.

$$
\begin{bmatrix} 1 \\ 2 \\ 333 \\ 44500 \\ 5 \end{bmatrix} \quad \begin{bmatrix} 9 \\ 77 \\ 3008 \end{bmatrix} \quad \begin{bmatrix} a & b & c & d \end{bmatrix} \quad (1, 2, 3, 4, 5) \quad (a_1, \ldots, a_n)
$$

$$
\begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad \begin{bmatrix} u_1 & | & u_2 & | & u_3 \end{bmatrix} \quad \begin{bmatrix} v_1^T \\ \hline v_2^T \\ \hline v_3^T \\ \hline v_4^T \end{bmatrix}
$$

In general, it's good to reduce the number of keys you have to press as much as possible; however, VS Code's autocompletion already allows us to avoid typing many things. Thus, the benefit of shortening some commands with macros may merely be that it makes our code more concise, but this depends on personal preference. For instance, you may not care whether your code is populated with the few extra letters in `\mathbf` as opposed to those in `\mbf`.

4.1 Troubleshooting with `\end{document}`

One way to locate elusive errors is to put an `\end{document}` right in the middle of the document. If the error goes away, it was in the portion of the document you cut off. Move the command up or down and repeat until you've found the error. In such a scenario it helps to have a command `\ed` to do this testing.

---

[13]Macros can either be written on one line for potentially faster compile times or left in a readable format, but compile times won't be much faster in the former case since LaTeX Workshop keeps auxiliary files. The first compilation is long, but subsequent compilations are quick because most of the compilation is already done. Hence, saving memory by reducing the number of tokens isn't very important since most of those tokens aren't recompiled. In the latter case, it's good practice to put `%`'s at the end of most lines to prevent whitespace created by `EOL` (end of line) characters. FYI: 2 `EOL` = empty line in code = `\par`.

[14]`*` creates a "short" command that invokes error messages if the ⟨*definition*⟩ is broken up over paragraphs (`\par` or an empty line). The utility here is that this will also make your compiler tell you the line number if you leave an opening brace unclosed.
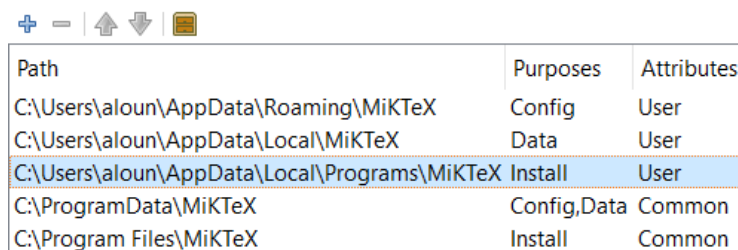
5   Packages

Packages provide commands for doing various typesetting tasks. For example, the `units` package allows us to write a/b instead of $\frac{a}{b}$ and the `xcolor` package allows us to <mark>highlight</mark> things. MiKTeX can install packages so that we don't have to deal with `sty` (style) files. It should do this automatically.[15]

5.1   Creating and Manually Installing Packages

Macros should be written in a separate file and added to the beginning of your document. The most prudent way to do this is to create a package by giving it the `.sty` file extension and putting these two lines at the beginning:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{⟨package-name⟩}
```

Create a folder named ⟨*package-name*⟩ containing the file. In MiKTeX Console, go to `Settings -> Directories`. Select the **Install** directory and open it with the square button.[16]

Path	Purposes	Attributes
C:\Users\aloun\AppData\Roaming\MiKTeX	Config	User
C:\Users\aloun\AppData\Local\MiKTeX	Data	User
C:\Users\aloun\AppData\Local\Programs\MiKTeX Install		User
C:\ProgramData\MiKTeX	Config,Data	Common
C:\Program Files\MiKTeX	Install	Common

On macOS, this directory will be

`/Users/<user_name>/Library/Application Support/MiKTeX/texmfs/install`

Then, place the ⟨*package-name*⟩ folder under `tex -> latex`, and under `Tasks` on the menubar, select **Refresh file name database**. You should now be able to use commands from ⟨*package-name*⟩ in any `tex` file without having to put the `sty` file in the same location.

It would be wise to pin the file or folder containing your commands to VS Code's icon and the File Explorer icon on your taskbar for easy access.

---

[15]In other words, it should do this "on the fly."

If not, once you've downloaded a package from CTAN, it should contain some `sty` files, in which case you should follow the steps laid out it §5.1.

If instead of `sty` files there are `ins` (installation) files, you must run those `ins` files to produce the required `sty` files. The simplest way to do this is to open the `ins` file in your editor and compile it as you would a `tex` file.

When all else fails, the makeshift way to use a package is to copy its `sty` files directly to the folder containing your `tex` file.

[16]An ignominious design flaw of MiKTeX is that it creates a file `miktex-console.lock` under `C:\Users\<user_name>` that prevents MiKTeX Console from opening initially. Simply delete this file and this problem shouldn't persist.

5.2   Images: `graphicx`

A picture of black writing on a whiteboard with the **shadows**, **contrast**, and **brightness** maximized, the **saturation** minimized, and possibly some other settings adjusted as needed—such as increasing the **brilliance** or slightly increasing the **exposure**—often gives an image whose white background nearly matches the white page and whose black writing nearly matches the black print.

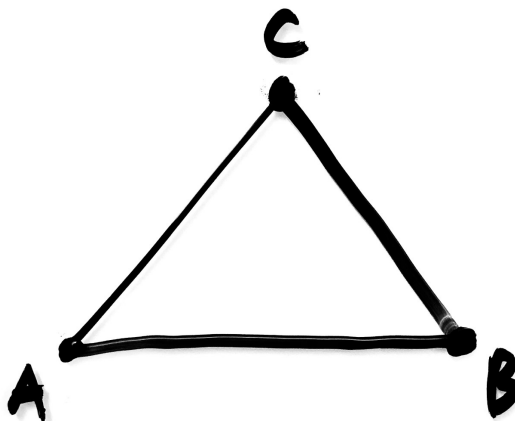Use the `center` environment or `\centering` if it's in a `figure` environment.[17,18]

Figure 1: $\triangle ABC$

Set the path to the folder with your images with `\graphicspath{{⟨path⟩}}`.
The file extensions of the images aren't required.
The `graphicx` package allows us to easily size images with `scale`, `width`, or `height`. Compare the options of the two images above.

---

[17]For subfigures, consider using the `subfigure` environment from the `subcaption` package.

[18]The option `[h]` stands for "here" and indicates that we want the figure to be placed exactly where it is in the code. Sometimes this is not enough, and stronger options are required. Ordered by strength, these are `[h]` < `[!h]` < `[H]`, where H comes from the `float` package.

5.3 Diagrams: PGF/Ti*k*Z

The `pgf` and `tikz` packages provide a very complex set of commands to draw shapes, graphs, and intricate diagrams straight into the document. You will probably never need to use them, but as before, your best resource for learning PGF/Ti*k*Z is the internet, so look for a some tutorials that might be helpful such as this.

Many packages build off of Ti*k*Z so that we can use it for specific tasks without doing all the work. For examples, there are the `tikz-feynman` and `tikzducks` packages.

Note that in the LaTeX menu on the left sidebar in VS Code, under the same Snippet panels mentioned in §3.1, there are Ti*k*Z panels that offer the opportunity to play around with various features from PGF/Ti*k*Z.

5.4 Poetry in LaTeX

The `verse` package allows us to typeset poetry in LaTeX. See `other sample files -> verse`.