Traversing LaTeX

The Harried Undergraduate's Guide

Hera Brown

The Oxford University Computer Science and Technology Society

February 22, 2025

What is LaTeX?

- LaTeX is a tool used to typeset technical documents.

What is LaTeX?

- LaTeX is a tool used to typeset technical documents.
- It's different from other document preparation systems you've probably used before (e.g. Word); instead of the output looking just like the input, in LaTeX you type both the text, and macros (like commands or functions) that control how the text looks.

What is LaTeX?

- LaTeX is a tool used to typeset technical documents.
- It's different from other document preparation systems you've probably used before (e.g. Word); instead of the output looking just like the input, in LaTeX you type both the text, and macros (like commands or functions) that control how the text looks.
- LaTeX documents look quite different before and after they've been processed!

What is LaTeX?

- LaTeX is a tool used to typeset technical documents.
- It's different from other document preparation systems you've probably used before (e.g. Word); instead of the output looking just like the input, in LaTeX you type both the text, and macros (like commands or functions) that control how the text looks.
- LaTeX documents look quite different before and after they've been processed!

```
\section{What is \LaTeX?}

\begin{frame}
\frametitle{What is \LaTeX?}
\begin{columns}
\begin{column}{0.5\textwidth}
  \begin{itemize}[]
    \item \LaTeX{} is a tool
      used to typeset technical
      documents.

      \item It's different from
      ...
```

Fig. 1. A sample of the code used to typeset this slide

What is LaTeX?

- LaTeX is a tool used to typeset technical documents.
- It's different from other document preparation systems you've probably used before (e.g. Word); instead of the output looking just like the input, in LaTeX you type both the text, and macros (like commands or functions) that control how the text looks.
- LaTeX documents look quite different before and after they've been processed!
- It might look a little scary at first, but once you get over the learning curve LaTeX becomes a very powerful tool.

```
\section{What is \LaTeX?}

\begin{frame}
\frametitle{What is \LaTeX?}
\begin{columns}
\begin{column}{0.5\textwidth}
  \begin{itemize}[]
    \item \LaTeX{} is a tool
      used to typeset technical
      documents.

      \item It's different from
      ...
```

Fig. 1. A sample of the code used to typeset this slide

A (very brief) History of LaTeX[1]

- LaTeX's genealogy can be traced back to TeX, a typesetting tool created by Donald Knuth in 1978. TeX is a more basic set of commands and macros than LaTeX, and still forms the basis of LaTeX — a lot of what we'll learn today is technically TeX, not LaTeX!

[1] Thanks, Wikipedia, for all this information!

A (very brief) History of LaTeX[1]

- LaTeX's genealogy can be traced back to TeX, a typesetting tool created by Donald Knuth in 1978. TeX is a more basic set of commands and macros than LaTeX, and still forms the basis of LaTeX — a lot of what we'll learn today is technically TeX, not LaTeX!
  To give you an idea of how stable TeX is as a tool, it's been stable since 1990. Only minor updates have been released since then (indeed, its version number is asymptotically approaching π).

[1] Thanks, Wikipedia, for all this information!

A (very brief) History of LaTeX[1]

- LaTeX's genealogy can be traced back to TeX, a typesetting tool created by Donald Knuth in 1978. TeX is a more basic set of commands and macros than LaTeX, and still forms the basis of LaTeX — a lot of what we'll learn today is technically TeX, not LaTeX!
  To give you an idea of how stable TeX is as a tool, it's been stable since 1990. Only minor updates have been released since then (indeed, its version number is asymptotically approaching π).
- LaTeX was developed by Leslie Lamport in the early 1980s, as a package of macros on top of TeX. The current standard, LaTeX 2_ε, was released in 1994.

[1] Thanks, Wikipedia, for all this information!

A (very brief) History of LaTeX[1]

- LaTeX's genealogy can be traced back to TeX, a typesetting tool created by Donald Knuth in 1978. TeX is a more basic set of commands and macros than LaTeX, and still forms the basis of LaTeX — a lot of what we'll learn today is technically TeX, not LaTeX!
  To give you an idea of how stable TeX is as a tool, it's been stable since 1990. Only minor updates have been released since then (indeed, its version number is asymptotically approaching π).
- LaTeX was developed by Leslie Lamport in the early 1980s, as a package of macros on top of TeX. The current standard, LaTeX 2_ε, was released in 1994.
  LaTeX has been designed with the intention of separating presentation from content; the intention is that you just need to type up your document semantically, and LaTeX will handle the difficult formatting bits. We'll see a lot of ways in which LaTeX makes things easier for us in this way in the talk.

[1] Thanks, Wikipedia, for all this information!

Why Should I Care?

What's in it for you? Well:

Why Should I Care?

What's in it for you? Well:

- It's the way to write papers and texts to the academic standard.

Why Should I Care?

What's in it for you? Well:

- It's the way to write papers and texts to the academic standard.
- Sometimes you'll get tutors who refuse to mark handwritten notes. Impress them with your LaTeX skills.

Why Should I Care?

What's in it for you? Well:

- It's the way to write papers and texts to the academic standard.
- Sometimes you'll get tutors who refuse to mark handwritten notes. Impress them with your LaTeX skills.
- If you're doing a third- or fourth-year project, you'll need to write a report. That report will probably need to be written in LaTeX.

Why Should I Care?

What's in it for you? Well:

- It's the way to write papers and texts to the academic standard.
- Sometimes you'll get tutors who refuse to mark handwritten notes. Impress them with your LaTeX skills.
- If you're doing a third- or fourth-year project, you'll need to write a report. That report will probably need to be written in LaTeX.
- It's just fun to use LaTeX! It's very rewarding to make beautiful documents.

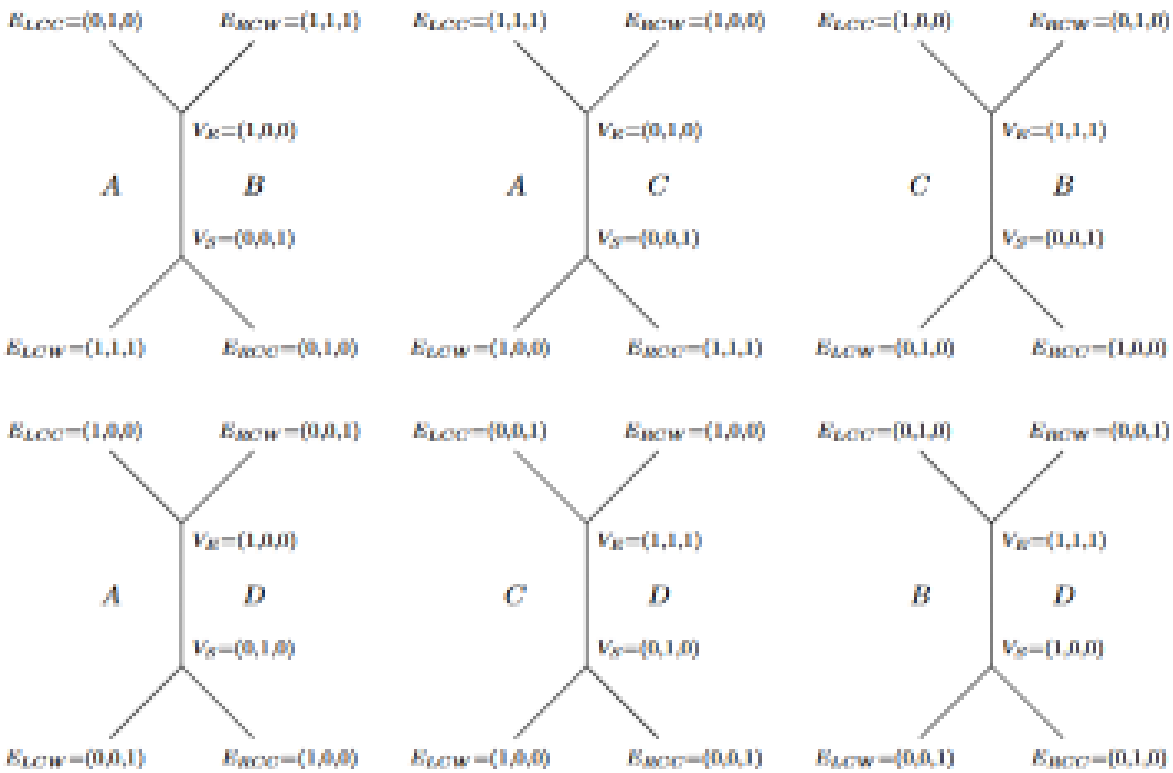
Some examples of what I've done

Want some examples? Well, here's what I've made so far this year:

Some examples of what I've done

Want some examples? Well, here's what I've made so far this year:

If using B-rep, then, the winged-edge structure starts off looking like:

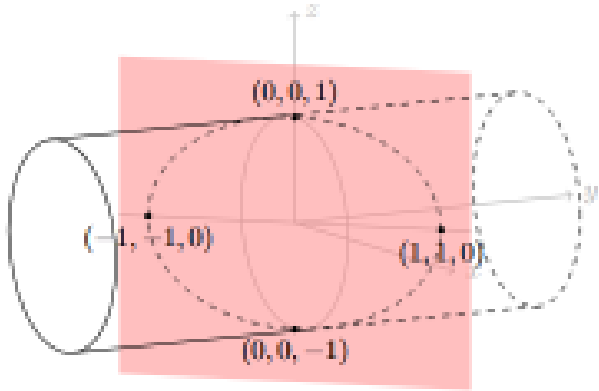


The final convex hull has an identical structure, but with every instance of $(1, 1, 1)$ replaced with $(3, 3, 3)$, with new corresponding faces. Said replacements are the intermediate edges added during the algorithm's running.

Some examples of what I've done

Want some examples? Well, here's what I've made so far this year:

Q5. (a) We get a cylinder lying on its side, cut diagonally, as below::



Note that the ends of the cylinders are only included above to make the diagram look like a cylinder; the cylinder really should be infinitely long.

Some examples of what I've done

Want some examples? Well, here's what I've made so far this year:

1. We may encode the transition relation on the above LTS as such:

Transition	x_1	x_1'	x_2	x_2'
$(0,0)$	0	0	0	0
$(0,1)$	0	0	0	1
$(0,2)$	0	1	0	0
$(0,3)$	0	1	0	1
$(1,0)$	0	0	1	0
$(1,1)$	0	0	1	1
$(1,2)$	0	1	1	0
$(1,3)$	0	1	1	1
$(2,2)$	1	1	0	0
$(2,3)$	1	1	0	1
$(3,2)$	1	1	1	0
$(3,3)$	1	1	1	1

From this and by inspection, we can write out an equivalent Boolean formula which captures the relation as:

$$\neg x_1 \vee x_1'$$

Which seems quite simplified.

Some examples of what I've done

Want some examples? Well, here's what I've made so far this year:

(e) Again, we'll follow Sider in LfP 9.7:

(1) $\forall\alpha\Box\phi \rightarrow \Box\phi$
(2) $\Box(\forall\alpha\Box\phi \rightarrow \Box\phi)$
(3) $\Diamond\forall\alpha\Box\phi \rightarrow \Diamond\Box\phi$
(4) $\Diamond\Box\phi \rightarrow \phi$
(5) $\Diamond\forall\alpha\Box\phi \rightarrow \phi$
(6) $\forall\alpha(\Diamond\forall\alpha\Box\phi \rightarrow \phi)$
(7) $\Diamond\forall\alpha\Box\phi \rightarrow \forall\alpha\phi$
(8) $\Box(\Diamond\forall\alpha\Box\phi \rightarrow \forall\alpha\phi)$
(9) $\Box\Diamond\forall\alpha\Box\phi \rightarrow \Box\forall\alpha\phi$
(10) $\forall\alpha\Box\phi \rightarrow \Box\Diamond\forall\alpha\Box\phi$
(11) $\forall\alpha\Box\phi \rightarrow \Box\forall\alpha\phi$

and so it follows that $\vdash_{\mathrm{SQML}} \Box\forall\alpha\phi \rightarrow \forall\alpha\Box\phi$.

Some examples of what I've done

Want some examples? Well, here's what I've made so far this year:

All these were put together in a week or so, for problem sheets. Once you get the hang of it, it's pretty easy to make pretty documents pretty fast.

What we'll do today

- Today we'll go through all the material you need to know to start typesetting your problem sheets in LaTeX.

What we'll do today

- Today we'll go through all the material you need to know to start typesetting your problem sheets in LaTeX.
- We'll be making an answer sheet to the MT22 Discrete Maths sheet 1. Hopefully you should know all the symbols we're using!

What we'll do today

- Today we'll go through all the material you need to know to start typesetting your problem sheets in LaTeX.
- We'll be making an answer sheet to the MT22 Discrete Maths sheet 1. Hopefully you should know all the symbols we're using!
- If at any point you have any questions, do stop me and ask them!

Getting Set Up

- To get started with LaTeX, you'll need something to type in, and something to turn your LaTeX code into a document (usually a pdf). There are a ton of ways to do this — it can be a little confusing trying to find the right tools for you!

Getting Set Up

- To get started with LaTeX, you'll need something to type in, and something to turn your LaTeX code into a document (usually a pdf). There are a ton of ways to do this — it can be a little confusing trying to find the right tools for you!
- For a proper editor that's free and open source, I'd recommend getting TeXworks (which is what I'm using to edit these slides in!). It's quick and easy to set up, and comes with a bunch of useful features (compiles many distributions of LaTeX, manages references, lets you split up your project into many files) and is easily extensible with your own features.

Getting Set Up

- To get started with LaTeX, you'll need something to type in, and something to turn your LaTeX code into a document (usually a pdf). There are a ton of ways to do this — it can be a little confusing trying to find the right tools for you!
- For a proper editor that's free and open source, I'd recommend getting TeXworks (which is what I'm using to edit these slides in!). It's quick and easy to set up, and comes with a bunch of useful features (compiles many distributions of LaTeX, manages references, lets you split up your project into many files) and is easily extensible with your own features.
- If you just want to play around and follow these slides for a bit, though, there are online editors as well — I've heard that Overleaf is good for this.

The Document Class

- Now you've got an editor open.
  How do you get started?

The Document Class

- Now you've got an editor open. How do you get started?
- You start by telling LaTeX what it's looking at! All documents in LaTeX start with the `\documentclass{...}` macro. You can put `book` in the middle if you're typesetting a book, `report` if you're typesetting a report, `article` if you're typesetting an article, and so on.

The Document Class

- Now you've got an editor open. How do you get started?
- You start by telling LaTeX what it's looking at! All documents in LaTeX start with the `\documentclass{...}` macro. You can put `book` in the middle if you're typesetting a book, `report` if you're typesetting a report, `article` if you're typesetting an article, and so on.
- Here let's write `\documentclass{article}` on our first line; I find that these look the best for problem sheets.

The Document Class

- Now you've got an editor open. How do you get started?
- You start by telling LaTeX what it's looking at! All documents in LaTeX start with the `\documentclass{...}` macro. You can put `book` in the middle if you're typesetting a book, `report` if you're typesetting a report, `article` if you're typesetting an article, and so on.
- Here let's write `\documentclass{article}` on our first line; I find that these look the best for problem sheets.

```
\documentclass{article}
```

Packages

- As in other programming languages, LaTeX has a set of packages which let you import a whole host of functionalities. The macro for importing a package is just `\usepackage{...}`.

```
\documentclass{article}
```

Packages

- As in other programming languages, LaTeX has a set of packages which let you import a whole host of functionalities. The macro for importing a package is just `\usepackage{...}`.
- Packages can let you typeset new symbols, organise documents in different ways, typeset other character sets, or even completely change how LaTeX works.

```
\documentclass{article}
```

Packages

- As in other programming languages, LaTeX has a set of packages which let you import a whole host of functionalities. The macro for importing a package is just `\usepackage{...}`.
- Packages can let you typeset new symbols, organise documents in different ways, typeset other character sets, or even completely change how LaTeX works.
- There's even a package that lets you typeset a little snowman.

```
\documentclass{article}
```

Packages

- As in other programming languages, LaTeX has a set of packages which let you import a whole host of functionalities. The macro for importing a package is just `\usepackage{...}`.
- Packages can let you typeset new symbols, organise documents in different ways, typeset other character sets, or even completely change how LaTeX works.
- There's even a package that lets you typeset a little snowman.
 .

```
\documentclass{article}
```

Packages

- As in other programming languages, LaTeX has a set of packages which let you import a whole host of functionalities. The macro for importing a package is just `\usepackage{...}`.
- Packages can let you typeset new symbols, organise documents in different ways, typeset other character sets, or even completely change how LaTeX works.
- There's even a package that lets you typeset a little snowman. ☃.
- Today, we'll be using the
  - `amsmath`,
  - `amssymb`, and
  - `enumitem`.

  packages.

```
\documentclass{article}
```

Packages

- As in other programming languages, LaTeX has a set of packages which let you import a whole host of functionalities. The macro for importing a package is just `\usepackage{...}`.
- Packages can let you typeset new symbols, organise documents in different ways, typeset other character sets, or even completely change how LaTeX works.
- There's even a package that lets you typeset a little snowman. ☃.
- Today, we'll be using the
  - `amsmath`,
  - `amssymb`, and
  - `enumitem`.

  packages.

```
\documentclass{article}
\usepackage{amsmath}
\usepackage{amssymb}
\usepackage{enumitem}
```

Your First Text

- We're just about ready to start typing. All that's left to do is tell LaTeX when the content of the document begins.

```
\documentclass{article}
\usepackage{amsmath}
\usepackage{amssymb}
\usepackage{enumitem}
```

Your First Text

- We're just about ready to start typing. All that's left to do is tell LaTeX when the content of the document begins.
- To do this, we just add `\begin{document}` and `\end{document}`, and type the content of the document in-between.

```
\documentclass{article}
\usepackage{amsmath}
\usepackage{amssymb}
\usepackage{enumitem}
```

Your First Text

- We're just about ready to start typing. All that's left to do is tell LaTeX when the content of the document begins.
- To do this, we just add `\begin{document}` and `\end{document}`, and type the content of the document in-between.

```
\documentclass{article}
\usepackage{amsmath}
\usepackage{amssymb}
\usepackage{enumitem}

\begin{document}
  Hello, world!
\end{document}
```

Your First Text

- We're just about ready to start typing. All that's left to do is tell LaTeX when the content of the document begins.
- To do this, we just add `\begin{document}` and `\end{document}`, and type the content of the document in-between.
- And you should now have your first LaTeX document!

```
\documentclass{article}
\usepackage{amsmath}
\usepackage{amssymb}
\usepackage{enumitem}

\begin{document}
  Hello, world!
\end{document}
```

Your First Text

- We're just about ready to start typing. All that's left to do is tell LaTeX when the content of the document begins.
- To do this, we just add `\begin{document}` and `\end{document}`, and type the content of the document in-between.
- And you should now have your first LaTeX document! There's not much to it at the moment. Let's add a title:

```
\documentclass{article}
\usepackage{amsmath}
\usepackage{amssymb}
\usepackage{enumitem}

\begin{document}
  Hello, world!
\end{document}
```

Your First Text

- We're just about ready to start typing. All that's left to do is tell LaTeX when the content of the document begins.
- To do this, we just add `\begin{document}` and `\end{document}`, and type the content of the document in-between.
- And you should now have your first LaTeX document! There's not much to it at the moment. Let's add a title:

```
\documentclass{article}
\usepackage{amsmath}
\usepackage{amssymb}
\usepackage{enumitem}

\title{Discrete Maths 1}
\author{Hera Brown}

\begin{document}
  \maketitle
  Hello, world!
\end{document}
```

Your First Text

- We're just about ready to start typing. All that's left to do is tell LaTeX when the content of the document begins.
- To do this, we just add `\begin{document}` and `\end{document}`, and type the content of the document in-between.
- And you should now have your first LaTeX document! There's not much to it at the moment. Let's add a title:
- And now you have a proper name and title.

```
\documentclass{article}
\usepackage{amsmath}
\usepackage{amssymb}
\usepackage{enumitem}

\title{Discrete Maths 1}
\author{Hera Brown}

\begin{document}
  \maketitle
  Hello, world!
\end{document}
```

Your First Text

- We're just about ready to start typing. All that's left to do is tell LaTeX when the content of the document begins.
- To do this, we just add `\begin{document}` and `\end{document}`, and type the content of the document in-between.
- And you should now have your first LaTeX document! There's not much to it at the moment. Let's add a title:
- And now you have a proper name and title.

Discrete Maths 1

Hera Brown
February 21, 2025

Hello world!

Organising Your Document

- Let's start typing answers to the problem sheet in, typing as we might into a basic text editor.

Organising Your Document

- Let's start typing answers to the problem sheet in, typing as we might into a basic text editor.

```
\begin{document}
  \maketitle

  Question 1 (i).  Here there
  are no elements of the set
  {n | n in Z and n2 < 0}, and
  so the following set would
  be 0.
  (ii).  Here we want the
  solutions of the equation
  n4 - 3n2 + 2n, and the set
  formed by these would be
  {-2, 0, 1}.

  ...
```

Organising Your Document

- Let's start typing answers to the problem sheet in, typing as we might into a basic text editor.
- This typesets to...

```
\begin{document}
  \maketitle

  Question 1 (i).  Here there
  are no elements of the set
  {n | n in Z and n2 < 0}, and
  so the following set would
  be 0.
  (ii).  Here we want the
  solutions of the equation
  n4 - 3n2 + 2n, and the set
  formed by these would be
  {-2, 0, 1}.

  ...
```

Organising Your Document

- Let's start typing answers to the problem sheet in, typing as we might into a basic text editor.
- This typesets to...

Discrete Maths 1

Hera Brown
February 21, 2025

Question 1 (i). Here there are no elements of the set n — n in Z and n2 $<$ 0, and so the following set would be 0. (ii). Here we want the solutions of the equation n4 - 3n2 + 2n, and the set formed by these would be -2,0,1. (iii) Here we want all the numbers of the form n2 - n where n $<$ 5. So the set would be 0, 2, 6, 12. (iv) Here it turns out that the set contains exactly the prime numbers. So the set would be 2,3,5,7,.... Question 2. (i) This set has at least min(—A—,—B—) elements, and at most —A—+—B— elements. (ii) This set has at least 0 elements, and at most min(—A—,—B—) elements.

Organising Your Document

- Let's start typing answers to the problem sheet in, typing as we might into a basic text editor.
- This typesets to...
- That doesn't look quite right. Some of the symbols have gone missing, or look different to how they should do. The text also hasn't been formatted very well; it's all one big block.

Discrete Maths 1

Hera Brown
February 21, 2025

Question 1 (i). Here there are no elements of the set n — n in Z and n2 < 0, and so the following set would be 0. (ii). Here we want the solutions of the equation n4 - 3n2 + 2n, and the set formed by these would be -2,0,1. (iii) Here we want all the numbers of the form n2 - n where n < 5. So the set would be 0, 2, 6, 12. (iv) Here it turns out that the set contains exactly the prime numbers. So the set would be 2,3,5,7,.... Question 2. (i) This set has at least min(—A—,—B—) elements, and at most —A—+—B— elements. (ii) This set has at least 0 elements, and at most min(—A—,—B—) elements.

Organising Your Document

- Let's start typing answers to the problem sheet in, typing as we might into a basic text editor.
- This typesets to...
- That doesn't look quite right. Some of the symbols have gone missing, or look different to how they should do. The text also hasn't been formatted very well; it's all one big block.
- Let's fix the latter issue first. How do we lay things out more nicely?

Discrete Maths 1

Hera Brown
February 21, 2025

Question 1 (i). Here there are no elements of the set n — n in Z and n2 < 0, and so the following set would be 0. (ii). Here we want the solutions of the equation n4 - 3n2 + 2n, and the set formed by these would be -2,0,1. (iii) Here we want all the numbers of the form n2 - n where n < 5. So the set would be 0, 2, 6, 12. (iv) Here it turns out that the set contains exactly the prime numbers. So the set would be 2,3,5,7,.... Question 2. (i) This set has at least min(—A—,—B—) elements, and at most —A—+—B— elements. (ii) This set has at least 0 elements, and at most min(—A—,—B—) elements.

Sections and Subsections

- Let's start by splitting things up into sections.

```
\begin{document}
  \maketitle

  Question 1 (i).  Here there
  are no elements of the set
  {n | n in Z and n2 < 0}, and
  so the following set would
  be 0.
  (ii).  Here we want the
  solutions of the equation
  n4 - 3n2 + 2n, and the set
  formed by these would be
  {-2, 0, 1}.

  ...
```

Sections and Subsections

- Let's start by splitting things up into sections.
- LaTeX has various semantic levels that you can fit things into (much like how HTML has <h1> through to <h6> tags).

```
\begin{document}
  \maketitle

  Question 1 (i).  Here there
  are no elements of the set
  {n | n in Z and n2 < 0}, and
  so the following set would
  be 0.
  (ii).  Here we want the
  solutions of the equation
  n4 - 3n2 + 2n, and the set
  formed by these would be
  {-2, 0, 1}.

  ...
```

Sections and Subsections

- Let's start by splitting things up into sections.
- LaTeX has various semantic levels that you can fit things into (much like how HTML has <h1> through to <h6> tags). These levels are:

```
\begin{document}
  \maketitle

  Question 1 (i).  Here there
  are no elements of the set
  {n | n in Z and n2 < 0}, and
  so the following set would
  be 0.
  (ii).  Here we want the
  solutions of the equation
  n4 - 3n2 + 2n, and the set
  formed by these would be
  {-2, 0, 1}.

  ...
```

Sections and Subsections

- Let's start by splitting things up into sections.
- LaTeX has various semantic levels that you can fit things into (much like how HTML has <h1> through to <h6> tags). These levels are:
  - \part

```
\begin{document}
  \maketitle

  Question 1 (i).  Here there
  are no elements of the set
  {n | n in Z and n2 < 0}, and
  so the following set would
  be 0.
  (ii).  Here we want the
  solutions of the equation
  n4 - 3n2 + 2n, and the set
  formed by these would be
  {-2, 0, 1}.

  ...
```

Sections and Subsections

- Let's start by splitting things up into sections.
- LaTeX has various semantic levels that you can fit things into (much like how HTML has <h1> through to <h6> tags). These levels are:
  - \part
  - \chapter

```
\begin{document}
  \maketitle

  Question 1 (i).  Here there
  are no elements of the set
  {n | n in Z and n2 < 0}, and
  so the following set would
  be 0.
  (ii).  Here we want the
  solutions of the equation
  n4 - 3n2 + 2n, and the set
  formed by these would be
  {-2, 0, 1}.

  ...
```

Sections and Subsections

- Let's start by splitting things up into sections.
- LaTeX has various semantic levels that you can fit things into (much like how HTML has <h1> through to <h6> tags). These levels are:
  - \part
  - \chapter
  - \section

```
\begin{document}
  \maketitle

  Question 1 (i).  Here there
  are no elements of the set
  {n | n in Z and n2 < 0}, and
  so the following set would
  be 0.
  (ii).  Here we want the
  solutions of the equation
  n4 - 3n2 + 2n, and the set
  formed by these would be
  {-2, 0, 1}.

  ...
```

Sections and Subsections

- Let's start by splitting things up into sections.
- LaTeX has various semantic levels that you can fit things into (much like how HTML has <h1> through to <h6> tags). These levels are:
  - \part
  - \chapter
  - \section
  - \subsection

```
\begin{document}
  \maketitle

  Question 1 (i).  Here there
  are no elements of the set
  {n | n in Z and n2 < 0}, and
  so the following set would
  be 0.
  (ii).  Here we want the
  solutions of the equation
  n4 - 3n2 + 2n, and the set
  formed by these would be
  {-2, 0, 1}.

  ...
```

Sections and Subsections

- Let's start by splitting things up into sections.
- LaTeX has various semantic levels that you can fit things into (much like how HTML has <h1> through to <h6> tags). These levels are:
  - \part
  - \chapter
  - \section
  - \subsection
  - \subsubsection

```
\begin{document}
  \maketitle

  Question 1 (i).  Here there
  are no elements of the set
  {n | n in Z and n2 < 0}, and
  so the following set would
  be 0.
  (ii).  Here we want the
  solutions of the equation
  n4 - 3n2 + 2n, and the set
  formed by these would be
  {-2, 0, 1}.

  ...
```

Sections and Subsections

- Let's start by splitting things up into sections.
- LaTeX has various semantic levels that you can fit things into (much like how HTML has <h1> through to <h6> tags). These levels are:
  - \part
  - \chapter
  - \section
  - \subsection
  - \subsubsection
  - \paragraph

```
\begin{document}
  \maketitle

  Question 1 (i).  Here there
  are no elements of the set
  {n | n in Z and n2 < 0}, and
  so the following set would
  be 0.
  (ii).  Here we want the
  solutions of the equation
  n4 - 3n2 + 2n, and the set
  formed by these would be
  {-2, 0, 1}.

  ...
```

Sections and Subsections

- Let's start by splitting things up into sections.
- LaTeX has various semantic levels that you can fit things into (much like how HTML has <h1> through to <h6> tags). These levels are:
  - \part
  - \chapter
  - \section
  - \subsection
  - \subsubsection
  - \paragraph
  - \subparagraph

```
\begin{document}
  \maketitle

  Question 1 (i).  Here there
  are no elements of the set
  {n | n in Z and n2 < 0}, and
  so the following set would
  be 0.
  (ii).  Here we want the
  solutions of the equation
  n4 - 3n2 + 2n, and the set
  formed by these would be
  {-2, 0, 1}.

  ...
```

Sections and Subsections

- Let's start by splitting things up into sections.
- LaTeX has various semantic levels that you can fit things into (much like how HTML has <h1> through to <h6> tags). These levels are:
  - \part
  - \chapter
  - \section
  - \subsection
  - \subsubsection
  - \paragraph
  - \subparagraph
- Let's add some sections to our code.

```
\begin{document}
  \maketitle

  Question 1 (i).  Here there
  are no elements of the set
  {n | n in Z and n2 < 0}, and
  so the following set would
  be 0.
  (ii).  Here we want the
  solutions of the equation
  n4 - 3n2 + 2n, and the set
  formed by these would be
  {-2, 0, 1}.

  ...
```

Sections and Subsections

- Let's start by splitting things up into sections.
- LaTeX has various semantic levels that you can fit things into (much like how HTML has <h1> through to <h6> tags). These levels are:
  - \part
  - \chapter
  - \section
  - \subsection
  - \subsubsection
  - \paragraph
  - \subparagraph
- Let's add some sections to our code.

```
\begin{document}
  \maketitle

  \section{Question 1}
  (i).  Here there are no
  elements of the set
  {n | n in Z and n2 < 0}, and
  so the following set would
  be 0.
  (ii).  Here we want the
  solutions of the equation
  n4 - 3n2 + 2n, and the set
  formed by these would be
  {-2, 0, 1}.
  ...

\section{Question 2}
  ...
```

Sections and Subsections

- Let's start by splitting things up into sections.
- LaTeX has various semantic levels that you can fit things into (much like how HTML has <h1> through to <h6> tags). These levels are:
  - \part
  - \chapter
  - \section
  - \subsection
  - \subsubsection
  - \paragraph
  - \subparagraph
- Let's add some sections to our code.
- Typesetting this gets us:

```
\begin{document}
  \maketitle

  \section{Question 1}
  (i).  Here there are no
  elements of the set
  {n | n in Z and n2 < 0}, and
  so the following set would
  be 0.
  (ii).  Here we want the
  solutions of the equation
  n4 - 3n2 + 2n, and the set
  formed by these would be
  {-2, 0, 1}.
  ...

\section{Question 2}
  ...
```

Sections and Subsections

- Let's start by splitting things up into sections.
- LaTeX has various semantic levels that you can fit things into (much like how `HTML` has `<h1>` through to `<h6>` tags). These levels are:
  - `\part`
  - `\chapter`
  - `\section`
  - `\subsection`
  - `\subsubsection`
  - `\paragraph`
  - `\subparagraph`
- Let's add some `sections` to our code.
- Typesetting this gets us:

Discrete Maths 1

Hera Brown
February 21, 2025

1 Question 1

(i). Here there are no elements of the set n — n in Z and n2 < 0, and so the following set would be 0. (ii). Here we want the solutions of the equation n4 - 3n2 + 2n, and the set formed by these would be -2,0,1. (iii) Here we want all the numbers of the form n2 - n where n < 5. So the set would be 0, 2, 6, 12. (iv) Here it turns out that the set contains exactly the prime numbers. So the set would be 2,3,5,7,....

2 Question 2

(i) This set has at least min(—A—,—B—) elements, and at most —A—+—B— elements. (ii) This set has at least 0 elements, and at most min(—A—,—B—) elements.

Sections and Subsections

- Let's start by splitting things up into sections.
- LaTeX has various semantic levels that you can fit things into (much like how `HTML` has `<h1>` through to `<h6>` tags). These levels are:
  - `\part`
  - `\chapter`
  - `\section`
  - `\subsection`
  - `\subsubsection`
  - `\paragraph`
  - `\subparagraph`
- Let's add some `sections` to our code.
- Typesetting this gets us:
- If we want to get rid of the number before the section, just replace `\section{...}` with `\section*{...}`.

Discrete Maths 1

Hera Brown
February 21, 2025

1 Question 1

(i). Here there are no elements of the set n — n in Z and n2 < 0, and so the following set would be 0. (ii). Here we want the solutions of the equation n4 - 3n2 + 2n, and the set formed by these would be -2,0,1. (iii) Here we want all the numbers of the form n2 - n where n < 5. So the set would be 0, 2, 6, 12. (iv) Here it turns out that the set contains exactly the prime numbers. So the set would be 2,3,5,7,....

2 Question 2

(i) This set has at least min(—A—,—B—) elements, and at most —A—+—B— elements. (ii) This set has at least 0 elements, and at most min(—A—,—B—) elements.

Sections and Subsections

- Let's start by splitting things up into sections.
- LaTeX has various semantic levels that you can fit things into (much like how `HTML` has `<h1>` through to `<h6>` tags). These levels are:
  - `\part`
  - `\chapter`
  - `\section`
  - `\subsection`
  - `\subsubsection`
  - `\paragraph`
  - `\subparagraph`
- Let's add some `sections` to our code.
- Typesetting this gets us:
- If we want to get rid of the number before the section, just replace `\section{...}` with `\section*{...}`.

Discrete Maths 1

Hera Brown
February 21, 2025

Question 1

(i). Here there are no elements of the set n — n in Z and n2 < 0, and so the following set would be 0. (ii). Here we want the solutions of the equation n4 - 3n2 + 2n, and the set formed by these would be -2,0,1. (iii) Here we want all the numbers of the form n2 - n where n < 5. So the set would be 0, 2, 6, 12. (iv) Here it turns out that the set contains exactly the prime numbers. So the set would be 2,3,5,7,....

Question 2

(i) This set has at least min(—A—,—B—) elements, and at most —A—+—B— elements. (ii) This set has at least 0 elements, and at most min(—A—,—B—) elements.

Lists and More Lists

- We'd also the subparts of the document to be spaced out nicely. For this, we'll use the `\begin{enumerate}` and `\end{enumerate}` macros.

```
\begin{document}
  \maketitle

  \section{Question 1}
  (i).  Here there are no
  elements of the set
  {n | n in Z and n2 < 0}, and
  so the following set would
  be 0.
  (ii).  Here we want the
  solutions of the equation
  n4 - 3n2 + 2n, and the set
  formed by these would be
  {-2, 0, 1}.
  ...

\section{Question 2}
  ...
```

Lists and More Lists

- We'd also the subparts of the document to be spaced out nicely. For this, we'll use the `\begin{enumerate}` and `\end{enumerate}` macros.
- Between these, we add items to the list by using the `\item` macro, as such:

```
\begin{document}
  \maketitle

  \section{Question 1}
  (i).  Here there are no
  elements of the set
  {n | n in Z and n2 < 0}, and
  so the following set would
  be 0.
  (ii).  Here we want the
  solutions of the equation
  n4 - 3n2 + 2n, and the set
  formed by these would be
  {-2, 0, 1}.
  ...

\section{Question 2}
  ...
```

Lists and More Lists

- We'd also the subparts of the document to be spaced out nicely. For this, we'll use the `\begin{enumerate}` and `\end{enumerate}` macros.
- Between these, we add items to the list by using the `\item` macro, as such:

```
\begin{document}
  \maketitle

  \section{Question 1}
  \begin{enumerate}
    \item Here there are no
    elements of the set
    {n | n in Z and n2 < 0},
    and so the following set
    would be 0.
    \item Here we want the
    solutions of the equation
    n4 - 3n2 + 2n, and the set
    formed by these would be
    {-2, 0, 1}.
    ...
  \end{enumerate}

\section{Question 2}
  ...
```

Lists and More Lists

- We'd also the subparts of the document to be spaced out nicely. For this, we'll use the `\begin{enumerate}` and `\end{enumerate}` macros.
- Between these, we add items to the list by using the `\item` macro, as such:
- And this looks like:

```
\begin{document}
  \maketitle

  \section{Question 1}
  \begin{enumerate}
    \item Here there are no
    elements of the set
    {n | n in Z and n2 < 0},
    and so the following set
    would be 0.
    \item Here we want the
    solutions of the equation
    n4 - 3n2 + 2n, and the set
    formed by these would be
    {-2, 0, 1}.
    ...
  \end{enumerate}

\section{Question 2}
  ...
```

Lists and More Lists

- We'd also the subparts of the document to be spaced out nicely. For this, we'll use the `\begin{enumerate}` and `\end{enumerate}` macros.
- Between these, we add items to the list by using the `\item` macro, as such:
- And this looks like:

Discrete Maths 1

Hera Brown
February 21, 2025

Question 1

1. Here there are no elements of the set n — n in Z and n2 < 0, and so the following set would be 0.
2. Here we want the solutions of the equation n4 - 3n2 + 2n, and the set formed by these would be -2,0,1.
3. Here we want all the numbers of the form n2 - n where n < 5. So the set would be 0, 2, 6, 12.
4. Here it turns out that the set contains exactly the prime numbers. So the set would be 2,3,5,7,....

Question 2

1. This set has at least min(—A—,—B—) elements, and at most —A—+—B— elements.
2. This set has at least 0 elements, and at most min(—A—,—B—) elements.

Lists and More Lists

- We'd also the subparts of the document to be spaced out nicely. For this, we'll use the `\begin{enumerate}` and `\end{enumerate}` macros.
- Between these, we add items to the list by using the `\item` macro, as such:
- And this looks like:
- If we want to change the labels from numbers to what we had before, we need to pass in an option to `\begin{enumerate}`. We do this by enclosing the option in square brackets. In this case, we'd write `\begin{enumerate}[label=(\roman*)]`.

Discrete Maths 1

Hera Brown
February 21, 2025

Question 1

1. Here there are no elements of the set n — n in Z and n2 $<$ 0, and so the following set would be 0.
2. Here we want the solutions of the equation n4 - 3n2 + 2n, and the set formed by these would be -2,0,1.
3. Here we want all the numbers of the form n2 - n where n $<$ 5. So the set would be 0, 2, 6, 12.
4. Here it turns out that the set contains exactly the prime numbers. So the set would be 2,3,5,7,....

Question 2

1. This set has at least min(—A—,—B—) elements, and at most —A—+—B— elements.
2. This set has at least 0 elements, and at most min(—A—,—B—) elements.

Lists and More Lists

- We'd also the subparts of the document to be spaced out nicely. For this, we'll use the `\begin{enumerate}` and `\end{enumerate}` macros.
- Between these, we add items to the list by using the `\item` macro, as such:
- And this looks like:
- If we want to change the labels from numbers to what we had before, we need to pass in an option to `\begin{enumerate}`. We do this by enclosing the option in square brackets. In this case, we'd write `\begin{enumerate}[label=(\roman*)]`.
- Updating the code, we get:

Discrete Maths 1

Hera Brown
February 21, 2025

Question 1

1. Here there are no elements of the set n — n in Z and n2 < 0, and so the following set would be 0.
2. Here we want the solutions of the equation n4 - 3n2 + 2n, and the set formed by these would be -2,0,1.
3. Here we want all the numbers of the form n2 - n where n < 5. So the set would be 0, 2, 6, 12.
4. Here it turns out that the set contains exactly the prime numbers. So the set would be 2,3,5,7,....

Question 2

1. This set has at least min(—A—,—B—) elements, and at most —A—+—B— elements.
2. This set has at least 0 elements, and at most min(—A—,—B—) elements.

Lists and More Lists

- We'd also the subparts of the document to be spaced out nicely. For this, we'll use the `\begin{enumerate}` and `\end{enumerate}` macros.
- Between these, we add items to the list by using the `\item` macro, as such:
- And this looks like:
- If we want to change the labels from numbers to what we had before, we need to pass in an option to `\begin{enumerate}`. We do this by enclosing the option in square brackets. In this case, we'd write `\begin{enumerate}[label=(\roman*)]`.
- Updating the code, we get:

Discrete Maths 1

Hera Brown
February 21, 2025

Question 1

(i) Here there are no elements of the set n — n in Z and n2 < 0, and so the following set would be 0.

(ii) Here we want the solutions of the equation n4 - 3n2 + 2n, and the set formed by these would be -2,0,1.

(iii) Here we want all the numbers of the form n2 - n where n < 5. So the set would be 0, 2, 6, 12.

(iv) Here it turns out that the set contains exactly the prime numbers. So the set would be 2,3,5,7,....

Question 2

(i) This set has at least min(—A—,—B—) elements, and at most —A—+—B— elements.

(ii) This set has at least 0 elements, and at most min(—A—,—B—) elements.

Typesetting Maths[2]

- We've now got the high-level structure of the document down. But each individual paragraph doesn't look so good. The notation's off entirely! What do we do about it?

Question 1

(i) Here there are no elements of the set n — n in Z and n2 $<$ 0, and so the following set would be 0.

...

[2] "Math" if you're American.

Typesetting Maths[2]

- We've now got the high-level structure of the document down. But each individual paragraph doesn't look so good. The notation's off entirely! What do we do about it?
- The answer lies in maths mode. Maths mode is where the hard word of typesetting mathematical symbols happens in LaTeX.

Question 1

(i) Here there are no elements of the set n — n in Z and n2 $<$ 0, and so the following set would be 0.

...

[2] "Math" if you're American.

Typesetting Maths[2]

- We've now got the high-level structure of the document down. But each individual paragraph doesn't look so good. The notation's off entirely! What do we do about it?
- The answer lies in maths mode. Maths mode is where the hard word of typesetting mathematical symbols happens in LaTeX.
- Typically, maths mode is entered with a $ symbol, and escaped with another $ symbol.

Question 1

(i) Here there are no elements of the set n — n in Z and n2 $<$ 0, and so the following set would be 0.

...

[2] "Math" if you're American.

Typesetting Maths[2]

- We've now got the high-level structure of the document down. But each individual paragraph doesn't look so good. The notation's off entirely! What do we do about it?
- The answer lies in maths mode. Maths mode is where the hard word of typesetting mathematical symbols happens in LaTeX.
- Typically, maths mode is entered with a `$` symbol, and escaped with another `$` symbol.
  So, for instance, `$\pi + \sqrt{e^3}$` would be typeset as $\pi + \sqrt{e^3}$.

Question 1

(i) Here there are no elements of the set n — n in Z and n2 $<$ 0, and so the following set would be 0.

...

[2] "Math" if you're American.

The Maths Environment

- With this in mind, let's try and typeset our document a bit more nicely.

```
\section{Question 1}
\begin{enumerate}
  \item Here there are no
  elements of the set
  {n | n in Z and n2 < 0},
  and so the following set
  would be 0.
...
```

The Maths Environment

- With this in mind, let's try and typeset our document a bit more nicely.
- First, let's add the dollar signs:

```
\section{Question 1}
\begin{enumerate}
  \item Here there are no
  elements of the set
  {n | n in Z and n2 < 0},
  and so the following set
  would be 0.
...
```

The Maths Environment

- With this in mind, let's try and typeset our document a bit more nicely.
- First, let's add the dollar signs:

```
\section{Question 1}
\begin{enumerate}
  \item Here there are no
  elements of the set
  ${n | n in Z and n2 < 0}$,
  and so the following set
  would be $O$.
...
```

The Maths Environment

- With this in mind, let's try and typeset our document a bit more nicely.
- First, let's add the dollar signs:
- Then macros to type the mathematical symbols:

```
\section{Question 1}
\begin{enumerate}
  \item Here there are no
  elements of the set
  ${n | n in Z and n2 < 0}$,
  and so the following set
  would be $O$.
...
```

The Maths Environment

- With this in mind, let's try and typeset our document a bit more nicely.
- First, let's add the dollar signs:
- Then macros to type the mathematical symbols:

```
\section{Question 1}
\begin{enumerate}
  \item Here there are no
  elements of the set
  ${n | n \in \mathbb{Z}
  \wedge n^2 < 0}$,
  and so the following set
  would be $\emptyset$.
...
```

The Maths Environment

- With this in mind, let's try and typeset our document a bit more nicely.
- First, let's add the dollar signs:
- Then macros to type the mathematical symbols:
- And finally, we need to make the curly brackets into macros, so that LaTeX can read them properly:

```
\section{Question 1}
\begin{enumerate}
  \item Here there are no
  elements of the set
  ${n | n \in \mathbb{Z}
  \wedge n^2 < 0}$,
  and so the following set
  would be $\emptyset$.
...
```

The Maths Environment

- With this in mind, let's try and typeset our document a bit more nicely.
- First, let's add the dollar signs:
- Then macros to type the mathematical symbols:
- And finally, we need to make the curly brackets into macros, so that LaTeX can read them properly:

```
\section{Question 1}
\begin{enumerate}
  \item Here there are no
  elements of the set
  $\{n | n \in \mathbb{Z}
  \wedge n^2 < 0\}$,
  and so the following set
  would be $\emptyset$.
...
```

The Maths Environment

- With this in mind, let's try and typeset our document a bit more nicely.
- First, let's add the dollar signs:
- Then macros to type the mathematical symbols:
- And finally, we need to make the curly brackets into macros, so that LaTeX can read them properly:
- Let's see what that looks like...

```
\section{Question 1}
\begin{enumerate}
  \item Here there are no
  elements of the set
  $\{n | n \in \mathbb{Z}
  \wedge n^2 < 0\}$,
  and so the following set
  would be $\emptyset$.
...
```

The Maths Environment

- With this in mind, let's try and typeset our document a bit more nicely.
- First, let's add the dollar signs:
- Then macros to type the mathematical symbols:
- And finally, we need to make the curly brackets into macros, so that LaTeX can read them properly:
- Let's see what that looks like...

Question 1

1. Here there are no elements of the set $\{n | n \in \mathbb{Z} \wedge n^2 < 0\}$, and so the following set would be $\emptyset$.

...

The Maths Environment

- With this in mind, let's try and typeset our document a bit more nicely.
- First, let's add the dollar signs:
- Then macros to type the mathematical symbols:
- And finally, we need to make the curly brackets into macros, so that LaTeX can read them properly:
- Let's see what that looks like...
- Brilliant! Let's get the rest of the document done...

Question 1

1. Here there are no elements of the set $\{n | n \in \mathbb{Z} \wedge n^2 < 0\}$, and so the following set would be $\emptyset$.

...

The Maths Environment

- With this in mind, let's try and typeset our document a bit more nicely.
- First, let's add the dollar signs:
- Then macros to type the mathematical symbols:
- And finally, we need to make the curly brackets into macros, so that LaTeX can read them properly:
- Let's see what that looks like...
- Brilliant! Let's get the rest of the document done...

Discrete Maths 1

Hera Brown
February 21, 2025

Question 1

(i) Here there are no elements of the set $\{n | n \in \mathbb{Z} \wedge n^2 < 0\}$, and so the following set would be $\emptyset$.

(ii) Here we want the solutions of the equation $n^4 - 3n^2 + 2n$, and the set formed by these would be $\{-2, 0, 1\}$.

(iii) Here we want all the numbers of the form $n^2 - n$ where $n < 5$. So the set would be $\{0, 2, 6, 12\}$.

(iv) Here it turns out that the set contains exactly the prime numbers. So the set would be $\{2, 3, 5, 7, \dots\}$.

Question 2

(i) This set has at least $\min(|A|, |B|)$ elements, and at most $|A| + |B|$ elements.

(ii) This set has at least 0 elements, and at most $\min(|A|, |B|)$ elements.

Greek Letters

- Most of the LaTeX macros used to typeset mathematics are pretty obvious. Probably the most obvious are the Greek letters, where you just type a backslash followed by the name of a letter.

Greek Letters

- Most of the LaTeX macros used to typeset mathematics are pretty obvious. Probably the most obvious are the Greek letters, where you just type a backslash followed by the name of a letter.
  If you want the lowercase letter, write the name in lowercase.
  Otherwise, make the first letter of the macro a capital.

Greek Letters

- Most of the LaTeX macros used to typeset mathematics are pretty obvious. Probably the most obvious are the Greek letters, where you just type a backslash followed by the name of a letter.
  If you want the lowercase letter, write the name in lowercase.
  Otherwise, make the first letter of the macro a capital.
- For instance, `$\alpha$` typesets to α, `$\beta$` typesets to β, and `$\gamma$` typesets to γ.

Greek Letters

- Most of the LaTeX macros used to typeset mathematics are pretty obvious. Probably the most obvious are the Greek letters, where you just type a backslash followed by the name of a letter.
  If you want the lowercase letter, write the name in lowercase.
  Otherwise, make the first letter of the macro a capital.
- For instance, `$\alpha$` typesets to α, `$\beta$` typesets to β, and `$\gamma$` typesets to γ.
  Similarly, `$\Gamma$` typesets to Γ, `$\Pi$` typesets to Π, and `$\Sigma$` typesets to Σ.

All the Symbols You'll Ever Need

- There are, of course, far more symbols than just the Greek letters. Some of the most obvious ones are typeset as follows:

All the Symbols You'll Ever Need

- There are, of course, far more symbols than just the Greek letters. Some of the most obvious ones are typeset as follows:
  - The logical symbols $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\forall$ and $\exists$ are typeset by `\neg`, `\wedge`, `\vee`, `\rightarrow`, `\forall` and `\exists` respectively.

All the Symbols You'll Ever Need

- There are, of course, far more symbols than just the Greek letters. Some of the most obvious ones are typeset as follows:
  - The logical symbols $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\forall$ and $\exists$ are typeset by `\neg`, `\wedge`, `\vee`, `\rightarrow`, `\forall` and `\exists` respectively.
  - The set-theoretic symbols $\{$, $\}$, $\cup$, $\cap$, $\subseteq$, $\in$, $\mathcal{P}$, $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{Q}$ can be typeset by `\{`, `\}`, `\cup`, `\cap`, `\subseteq`, `\in`, `\mathcal{P}`, `\mathbb{N}`, `\mathbb{Z}` and `\mathbb{Q}` respectively.

All the Symbols You'll Ever Need

- There are, of course, far more symbols than just the Greek letters. Some of the most obvious ones are typeset as follows:
  - The logical symbols $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\forall$ and $\exists$ are typeset by `\neg`, `\wedge`, `\vee`, `\rightarrow`, `\forall` and `\exists` respectively.
  - The set-theoretic symbols $\{$, $\}$, $\cup$, $\cap$, $\subseteq$, $\in$, $\mathcal{P}$, $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{Q}$ can be typeset by `\{`, `\}`, `\cup`, `\cap`, `\subseteq`, `\in`, `\mathcal{P}`, `\mathbb{N}`, `\mathbb{Z}` and `\mathbb{Q}` respectively.
  - Superscripts such as x^2 and e^{π}, and subscripts such as p_0 and τ_i, can be typeset using `^` and `_` respectively: so we could write `x^2`, `e^\pi`, `p_0` and `\tau_i`.

All the Symbols You'll Ever Need

- There are, of course, far more symbols than just the Greek letters. Some of the most obvious ones are typeset as follows:
  - The logical symbols $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\forall$ and $\exists$ are typeset by `\neg`, `\wedge`, `\vee`, `\rightarrow`, `\forall` and `\exists` respectively.
  - The set-theoretic symbols $\{$, $\}$, $\cup$, $\cap$, $\subseteq$, $\in$, $\mathcal{P}$, $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{Q}$ can be typeset by `\{`, `\}`, `\cup`, `\cap`, `\subseteq`, `\in`, `\mathcal{P}`, `\mathbb{N}`, `\mathbb{Z}` and `\mathbb{Q}` respectively.
  - Superscripts such as x^2 and e^π, and subscripts such as p_0 and τ_i, can be typeset using `^` and `_` respectively: so we could write `x^2`, `e^\pi`, `p_0` and `\tau_i`.
  - Equalities and inequalities such as $\leq$, $\geq$, $\neq$, $\sim$, $\approx$, $\prec$ and $\preceq$ are typeset by `\leq`, `\geq`, `\neq`, `\sim`, `\approx`, `\prec` and `\preceq` respectively.

All the Symbols You'll Ever Need

- There are, of course, far more symbols than just the Greek letters. Some of the most obvious ones are typeset as follows:
  - The logical symbols $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\forall$ and $\exists$ are typeset by `\neg`, `\wedge`, `\vee`, `\rightarrow`, `\forall` and `\exists` respectively.
  - The set-theoretic symbols $\{$, $\}$, $\cup$, $\cap$, $\subseteq$, $\in$, $\mathcal{P}$, $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{Q}$ can be typeset by `\{`, `\}`, `\cup`, `\cap`, `\subseteq`, `\in`, `\mathcal{P}`, `\mathbb{N}`, `\mathbb{Z}` and `\mathbb{Q}` respectively.
  - Superscripts such as x^2 and e^π, and subscripts such as p_0 and τ_i, can be typeset using `^` and `_` respectively: so we could write `x^2`, `e^\pi`, `p_0` and `\tau_i`.
  - Equalities and inequalities such as $\leq$, $\geq$, $\neq$, $\sim$, $\approx$, $\prec$ and $\preceq$ are typeset by `\leq`, `\geq`, `\neq`, `\sim`, `\approx`, `\prec` and `\preceq` respectively.
- If you ever see a symbol out in the wild, and can't recognise it, do visit https://detexify.kirelabs.org. It lets you draw in the symbol by hand, and gives you a list of what macros might typeset that symbol — it's very comprehensive!

Defining Your Own Macros

- All these macros are great. But what if we need more? What if we want to define our own?

Defining Your Own Macros

- All these macros are great. But what if we need more? What if we want to define our own?
- It turns out you can! Let's suppose that you want to typeset $[g]^{\alpha}_{f^u_{\beta}}$ over and over in a document. To typeset that, every time, you'd need to type `$[g]^\alpha_{f^u_\beta}$`.

Defining Your Own Macros

- All these macros are great. But what if we need more? What if we want to define our own?
- It turns out you can! Let's suppose that you want to typeset $[g]^{\alpha}_{f^u_{\beta}}$ over and over in a document. To typeset that, every time, you'd need to type `$[g]^\alpha_{f^u_\beta}$`. That's an awful lot of effort.
- But LaTeX comes to the rescue. You can define a new macro yourself by typing `\newcommand{\name}{...}`.

Defining Your Own Macros

- All these macros are great. But what if we need more? What if we want to define our own?
- It turns out you can! Let's suppose that you want to typeset $[g]^{\alpha}_{f^u_{\beta}}$ over and over in a document. To typeset that, every time, you'd need to type `$[g]^\alpha_{f^u_\beta}$`. That's an awful lot of effort.
- But LaTeX comes to the rescue. You can define a new macro yourself by typing `\newcommand{\name}{...}`.
  So, in this case, we'd write:

```
\newcommand{\g}{[g]^\alpha_{f^u_\beta}}
```

Defining Your Own Macros

- All these macros are great. But what if we need more? What if we want to define our own?
- It turns out you can! Let's suppose that you want to typeset $[g]^{\alpha}_{f^u_{\beta}}$ over and over in a document. To typeset that, every time, you'd need to type `$[g]^\alpha_{f^u_\beta}$`. That's an awful lot of effort.
- But LaTeX comes to the rescue. You can define a new macro yourself by typing `\newcommand{\name}{...}`.
  So, in this case, we'd write:

```
\newcommand{\g}{[g]^\alpha_{f^u_\beta}}
```

and then wherever we wanted to typeset $[g]^{\alpha}_{f^u_{\beta}}$, we'd just type `$\g$` instead. Neat!

Parameters

- You can even parametrise your macros. Suppose that you've constructed some complex formula that looks like $[A| \longleftrightarrow |\emptyset]$. The code to typeset that is `$[A|\longleftrightarrow|\emptyset]$`. It's a bit of a pain to typeset.

Parameters

- You can even parametrise your macros. Suppose that you've constructed some complex formula that looks like $[A| \longleftrightarrow |\emptyset]$. The code to typeset that is `$[A|\longleftrightarrow|\emptyset]$`. It's a bit of a pain to typeset. But you don't want to just have A in there; you want to extend the formula to talk about other variables.

Parameters

- You can even parametrise your macros. Suppose that you've constructed some complex formula that looks like $[A| \longleftrightarrow |\emptyset]$. The code to typeset that is `$[A|\longleftrightarrow|\emptyset]$`. It's a bit of a pain to typeset. But you don't want to just have A in there; you want to extend the formula to talk about other variables.
- Instead of defining a family of macros, each with a different variable in place of A, you can parametrise your macro as follows:

```
\newcommand{\exeq}[1]{[#1|\longleftrightarrow|\emptyset]}
```

  and then all you need to do is type, say, `$\exeq{\Sigma}$`, which then typesets as $[\Sigma| \longleftrightarrow |\emptyset]$.

Parameters

- You can even parametrise your macros. Suppose that you've constructed some complex formula that looks like $[A| \longleftrightarrow |\emptyset]$. The code to typeset that is `$[A|\longleftrightarrow|\emptyset]$`. It's a bit of a pain to typeset. But you don't want to just have A in there; you want to extend the formula to talk about other variables.
- Instead of defining a family of macros, each with a different variable in place of A, you can parametrise your macro as follows:

```
\newcommand{\exeq}[1]{[#1|\longleftrightarrow|\emptyset]}
```

  and then all you need to do is type, say, `$\exeq{\Sigma}$`, which then typesets as $[\Sigma| \longleftrightarrow |\emptyset]$.
- This generalises; if you replace the "1" in the above with "2" or "3", for instance, you can then define a macro with two or three parameters!

More Maths!

- There's even more you can do with the maths mode.

More Maths!

- There's even more you can do with the maths mode.
- Sometimes you have some big long formula that's too long to write inline, with the rest of the text. Say that you want to talk about a Taylor series:

More Maths!

- There's even more you can do with the maths mode.
- Sometimes you have some big long formula that's too long to write inline, with the rest of the text. Say that you want to talk about a Taylor series:

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \dots \quad (1)$$

More Maths!

- There's even more you can do with the maths mode.
- Sometimes you have some big long formula that's too long to write inline, with the rest of the text. Say that you want to talk about a Taylor series:

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \dots \quad (1)$$

- It's not really feasible to fit that in one line. Thankfully, you can use the `equation` environment to typeset large equations on their own lines.

More Maths!

- There's even more you can do with the maths mode.
- Sometimes you have some big long formula that's too long to write inline, with the rest of the text. Say that you want to talk about a Taylor series:

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \dots \quad (1)$$

- It's not really feasible to fit that in one line. Thankfully, you can use the `equation` environment to typeset large equations on their own lines.
- Just type `\begin{equation} ...\end{equation}`, with your maths going between the beginning and the end, and it'll get its own line, with a little reference number on the right to boot.

More Maths!

- There's even more you can do with the maths mode.
- Sometimes you have some big long formula that's too long to write inline, with the rest of the text. Say that you want to talk about a Taylor series:

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \dots \quad (1)$$

- It's not really feasible to fit that in one line. Thankfully, you can use the `equation` environment to typeset large equations on their own lines.
- Just type `\begin{equation} ...\end{equation}`, with your maths going between the beginning and the end, and it'll get its own line, with a little reference number on the right to boot.
- If you don't want the number, write `equation*` instead.

More Maths!

- There's even more you can do with the maths mode.
- Sometimes you have some big long formula that's too long to write inline, with the rest of the text. Say that you want to talk about a Taylor series:

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \dots$$

- It's not really feasible to fit that in one line. Thankfully, you can use the `equation` environment to typeset large equations on their own lines.
- Just type `\begin{equation} ...\end{equation}`, with your maths going between the beginning and the end, and it'll get its own line, with a little reference number on the right to boot.
- If you don't want the number, write `equation*` instead.

Aligning Maths

- One limitation of the above is that you can't write two long equations close to each other; you have to use two separate environments. What if we want to align two equations?

Aligning Maths

- One limitation of the above is that you can't write two long equations close to each other; you have to use two separate environments. What if we want to align two equations?
- LaTeX (more specifically the `amsmath` package) provides for this. We can create an `align` environment by writing the following:

Aligning Maths

- One limitation of the above is that you can't write two long equations close to each other; you have to use two separate environments. What if we want to align two equations?
- LaTeX (more specifically the `amsmath` package) provides for this. We can create an `align` environment by writing the following:

```
\begin{align}

\end{align}
```

Aligning Maths

- One limitation of the above is that you can't write two long equations close to each other; you have to use two separate environments. What if we want to align two equations?
- LaTeX (more specifically the `amsmath` package) provides for this. We can create an `align` environment by writing the following:
- And then add the maths we want. We use the `&` character to tell LaTeX what character we're aligning about, and `\\` to start a new line.

```
\begin{align}

\end{align}
```

Aligning Maths

- One limitation of the above is that you can't write two long equations close to each other; you have to use two separate environments. What if we want to align two equations?
- LaTeX (more specifically the `amsmath` package) provides for this. We can create an `align` environment by writing the following:
- And then add the maths we want. We use the `&` character to tell LaTeX what character we're aligning about, and `\\` to start a new line.

```
\begin{align}
  x &= \pi^2 - 3 \\
  y &= 6x
\end{align}
```

Aligning Maths

- One limitation of the above is that you can't write two long equations close to each other; you have to use two separate environments. What if we want to align two equations?
- LaTeX (more specifically the `amsmath` package) provides for this. We can create an `align` environment by writing the following:
- And then add the maths we want. We use the `&` character to tell LaTeX what character we're aligning about, and `\\` to start a new line.
- Typesetting this gives us:

```
\begin{align}
  x &= \pi^2 - 3 \\
  y &= 6x
\end{align}
```

Aligning Maths

- One limitation of the above is that you can't write two long equations close to each other; you have to use two separate environments. What if we want to align two equations?
- LaTeX (more specifically the `amsmath` package) provides for this. We can create an `align` environment by writing the following:
- And then add the maths we want. We use the `&` character to tell LaTeX what character we're aligning about, and `\\` to start a new line.
- Typesetting this gives us:

$$x = \pi^2 - 3 \tag{1}$$
$$y = 6x \tag{2}$$

Aligning Maths

- One limitation of the above is that you can't write two long equations close to each other; you have to use two separate environments. What if we want to align two equations?
- LaTeX (more specifically the `amsmath` package) provides for this. We can create an `align` environment by writing the following:
- And then add the maths we want. We use the `&` character to tell LaTeX what character we're aligning about, and `\\` to start a new line.
- Typesetting this gives us:
- Again, to get rid of the reference numbers, just write `align*` instead:

$$x = \pi^2 - 3 \tag{1}$$
$$y = 6x \tag{2}$$

Aligning Maths

- One limitation of the above is that you can't write two long equations close to each other; you have to use two separate environments. What if we want to align two equations?
- LaTeX (more specifically the `amsmath` package) provides for this. We can create an `align` environment by writing the following:
- And then add the maths we want. We use the `&` character to tell LaTeX what character we're aligning about, and `\\` to start a new line.
- Typesetting this gives us:
- Again, to get rid of the reference numbers, just write `align*` instead:

$$x = \pi^2 - 3$$
$$y = 6x$$

Typesetting Code

- In the computer science degree, you'll want to be writing down code a lot of the time in your problem sheets. It's definitely possible (you've been seeing it in this presentation!), but from what we've seen so far, LaTeX doesn't get the indentation right, and has trouble with the symbols.

Typesetting Code

- In the computer science degree, you'll want to be writing down code a lot of the time in your problem sheets. It's definitely possible (you've been seeing it in this presentation!), but from what we've seen so far, LaTeX doesn't get the indentation right, and has trouble with the symbols.
- The answer is the `tabbing` environment. The `tabbing` environment lets you control the indentation of your code when it's being typeset. First we'd set up the environment:

Typesetting Code

- In the computer science degree, you'll want to be writing down code a lot of the time in your problem sheets. It's definitely possible (you've been seeing it in this presentation!), but from what we've seen so far, LaTeX doesn't get the indentation right, and has trouble with the symbols.
- The answer is the `tabbing` environment. The `tabbing` environment lets you control the indentation of your code when it's being typeset. First we'd set up the environment:

```
\begin{tabbing}

\end{tabbing}
```

Typesetting Code

- In the computer science degree, you'll want to be writing down code a lot of the time in your problem sheets. It's definitely possible (you've been seeing it in this presentation!), but from what we've seen so far, LaTeX doesn't get the indentation right, and has trouble with the symbols.
- The answer is the `tabbing` environment. The `tabbing` environment lets you control the indentation of your code when it's being typeset. First we'd set up the environment:
- And now we set the tab-stops. Tab-stops tell LaTeX how large you want your indentations to be. Here we'll set them to be one `em`, which is about the width of "M" on the page.

```
\begin{tabbing}

\end{tabbing}
```

Typesetting Code

- In the computer science degree, you'll want to be writing down code a lot of the time in your problem sheets. It's definitely possible (you've been seeing it in this presentation!), but from what we've seen so far, LaTeX doesn't get the indentation right, and has trouble with the symbols.
- The answer is the `tabbing` environment. The `tabbing` environment lets you control the indentation of your code when it's being typeset. First we'd set up the environment:
- And now we set the tab-stops. Tab-stops tell LaTeX how large you want your indentations to be. Here we'll set them to be one `em`, which is about the width of "M" on the page.

```
\begin{tabbing}
  \hspace{1em}\=\hspace{1em}\=\kill

\end{tabbing}
```

Typesetting Code

- Now we'll want to add some code. Here you can just type in whatever you want typeset, so long as it's valid LaTeX.

```
\begin{tabbing}
  \hspace{1em}\=\hspace{1em}\=\kill

\end{tabbing}
```

Typesetting Code

- Now we'll want to add some code. Here you can just type in whatever you want typeset, so long as it's valid LaTeX.
  For indentation, you'll want to add \> to the start of each line you want an indent; add multiple for multiple indents.
  You'll also want to add \\ to end your lines, as with the `align` environment.

```
\begin{tabbing}
  \hspace{1em}\=\hspace{1em}\=\kill

\end{tabbing}
```

Typesetting Code

- Now we'll want to add some code. Here you can just type in whatever you want typeset, so long as it's valid LaTeX.
  For indentation, you'll want to add \> to the start of each line you want an indent; add multiple for multiple indents.
  You'll also want to add \\ to end your lines, as with the `align` environment.

```
\begin{tabbing}
  \hspace{1em}\=\hspace{1em}\=\kill
  int x = 1; \\
  while(x $<$ len)\{\\
    \> xs = x:xs\\
  \}
\end{tabbing}
```

Typesetting Code

- Now we'll want to add some code. Here you can just type in whatever you want typeset, so long as it's valid LaTeX.
  For indentation, you'll want to add \> to the start of each line you want an indent; add multiple for multiple indents.
  You'll also want to add \\ to end your lines, as with the `align` environment.
- This ends up looking like:

```
\begin{tabbing}
  \hspace{1em}\=\hspace{1em}\=\kill
  int x = 1; \\
  while(x $<$ len)\{\\
    \> xs = x:xs\\
  \}
\end{tabbing}
```

Typesetting Code

- Now we'll want to add some code. Here you can just type in whatever you want typeset, so long as it's valid LaTeX.
  For indentation, you'll want to add \> to the start of each line you want an indent; add multiple for multiple indents.
  You'll also want to add \\ to end your lines, as with the `align` environment.
- This ends up looking like:

```
int x = 1;
while (x < len) {
    xs = x:xs
}
```

Typesetting Code

- Now we'll want to add some code. Here you can just type in whatever you want typeset, so long as it's valid LaTeX.
  For indentation, you'll want to add \> to the start of each line you want an indent; add multiple for multiple indents.
  You'll also want to add \\ to end your lines, as with the `align` environment.
- This ends up looking like:
- If you want the typewriter look, add `\tt` at the start of each line (but after any \>'s); this gets us:

```
int x = 1;
while (x < len) {
    xs = x:xs
}
```

Typesetting Code

- Now we'll want to add some code. Here you can just type in whatever you want typeset, so long as it's valid LaTeX.
  For indentation, you'll want to add \> to the start of each line you want an indent; add multiple for multiple indents.
  You'll also want to add \\ to end your lines, as with the `align` environment.
- This ends up looking like:
- If you want the typewriter look, add `\tt` at the start of each line (but after any \>'s); this gets us:

```
int x = 1;
while (x < len) {
  xs = x:xs
}
```

Typesetting Code

- Now we'll want to add some code. Here you can just type in whatever you want typeset, so long as it's valid LaTeX.
  For indentation, you'll want to add \> to the start of each line you want an indent; add multiple for multiple indents.
  You'll also want to add \\ to end your lines, as with the `align` environment.
- This ends up looking like:
- If you want the typewriter look, add `\tt` at the start of each line (but after any \>'s); this gets us:
  Which is just how the code examples in these slides were made.

```
int x = 1;
while (x < len) {
  xs = x:xs
}
```

A Final Word

What have we covered today? Well, we've looked at:

A Final Word

What have we covered today? Well, we've looked at:

- How to set up a document that'll typeset successfully,

A Final Word

What have we covered today? Well, we've looked at:

- How to set up a document that'll typeset successfully,
- How to organise a document into semantic sections,

A Final Word

What have we covered today? Well, we've looked at:

- How to set up a document that'll typeset successfully,
- How to organise a document into semantic sections,
- How to typeset common maths symbols, including sub- and superscripts,

A Final Word

What have we covered today? Well, we've looked at:

- How to set up a document that'll typeset successfully,
- How to organise a document into semantic sections,
- How to typeset common maths symbols, including sub- and superscripts,
- How to define our own macros, to save on energy,

A Final Word

What have we covered today? Well, we've looked at:

- How to set up a document that'll typeset successfully,
- How to organise a document into semantic sections,
- How to typeset common maths symbols, including sub- and superscripts,
- How to define our own macros, to save on energy,
- How to typeset more complex mathematical formulae, and

A Final Word

What have we covered today? Well, we've looked at:

- How to set up a document that'll typeset successfully,
- How to organise a document into semantic sections,
- How to typeset common maths symbols, including sub- and superscripts,
- How to define our own macros, to save on energy,
- How to typeset more complex mathematical formulae, and
- How to typeset code.

A Final Word

What have we covered today? Well, we've looked at:

- How to set up a document that'll typeset successfully,
- How to organise a document into semantic sections,
- How to typeset common maths symbols, including sub- and superscripts,
- How to define our own macros, to save on energy,
- How to typeset more complex mathematical formulae, and
- How to typeset code.

But there's so much more that LaTeX can do...

A Final Word

What have we covered today? Well, we've looked at:

- How to set up a document that'll typeset successfully,
- How to organise a document into semantic sections,
- How to typeset common maths symbols, including sub- and superscripts,
- How to define our own macros, to save on energy,
- How to typeset more complex mathematical formulae, and
- How to typeset code.

But there's so much more that LaTeX can do...

- Typeset graphs, automata, diagrams, and just about anything you can think of, using Ti*k*Z,

A Final Word

What have we covered today? Well, we've looked at:

- How to set up a document that'll typeset successfully,
- How to organise a document into semantic sections,
- How to typeset common maths symbols, including sub- and superscripts,
- How to define our own macros, to save on energy,
- How to typeset more complex mathematical formulae, and
- How to typeset code.

But there's so much more that LaTeX can do...

- Typeset graphs, automata, diagrams, and just about anything you can think of, using Ti*k*Z,
- Organise larger documents, using footnotes, references, and typesetting your own theorems and lemmata,

A Final Word

What have we covered today? Well, we've looked at:

- How to set up a document that'll typeset successfully,
- How to organise a document into semantic sections,
- How to typeset common maths symbols, including sub- and superscripts,
- How to define our own macros, to save on energy,
- How to typeset more complex mathematical formulae, and
- How to typeset code.

But there's so much more that LaTeX can do...

- Typeset graphs, automata, diagrams, and just about anything you can think of, using Ti*k*Z,
- Organise larger documents, using footnotes, references, and typesetting your own theorems and lemmata,
- Make slides for presentations (like these that you're reading now!),

A Final Word

What have we covered today? Well, we've looked at:

- How to set up a document that'll typeset successfully,
- How to organise a document into semantic sections,
- How to typeset common maths symbols, including sub- and superscripts,
- How to define our own macros, to save on energy,
- How to typeset more complex mathematical formulae, and
- How to typeset code.

But there's so much more that LaTeX can do...

- Typeset graphs, automata, diagrams, and just about anything you can think of, using Ti*k*Z,
- Organise larger documents, using footnotes, references, and typesetting your own theorems and lemmata,
- Make slides for presentations (like these that you're reading now!),
- Run actual code — TeX is Turing complete!

A Final Word

What have we covered today? Well, we've looked at:

- How to set up a document that'll typeset successfully,
- How to organise a document into semantic sections,
- How to typeset common maths symbols, including sub- and superscripts,
- How to define our own macros, to save on energy,
- How to typeset more complex mathematical formulae, and
- How to typeset code.

But there's so much more that LaTeX can do...

- Typeset graphs, automata, diagrams, and just about anything you can think of, using Ti*k*Z,
- Organise larger documents, using footnotes, references, and typesetting your own theorems and lemmata,
- Make slides for presentations (like these that you're reading now!),
- Run actual code — TeX is Turing complete!

And so much more.

A Final Word

Hopefully that's given you an idea of the power and extensibility of LaTeX! It really is a wonderful tool, and I hope you're feeling confident enough to start typing up some of your problem sheets and projects.

A Final Word

Hopefully that's given you an idea of the power and extensibility of LaTeX! It really is a wonderful tool, and I hope you're feeling confident enough to start typing up some of your problem sheets and projects.
If you ever need help with something LaTeX-related, fire me a message! I'd be happy to help out.

Thanks for listening!

Any questions?