

Customising L^AT_EX

Up to now we have been using standard L^AT_EX commands and packages to create documents.

However, L^AT_EX is highly customisable, and if we find that we need to do something that is not covered by an existing command, environment or package, we can write our own!

Custom L^AT_EX Commands/Macros

We specify our own commands using the `\newcommand` command:

```
\newcommand{name}[num]{definition}
```

It requires two arguments, the *name* of the new command, and its *definition*. It also has an optional argument (num) allowing you to specify how many arguments it takes (up to 9).

Custom L^AT_EX Commands - example

```
\newcommand{\ual}{UGC Course on Advanced \LaTeX}
```

This is the \ual.

This is the UGC Course on Advanced L^AT_EX.

Custom L^AT_EX Commands - arguments

```
\newcommand{\ualarg}[1]{UGC Course on Advanced \LaTeX\, presented  
by #1}
```

```
\newcommand{\ualargtwo}[2]{UGC Course on Advanced \LaTeX\,  
presented by #1 and #2}
```

This is the `\ualarg{Martin}`.

This is the `\ualargtwo{Martin}{Someone}`.

This is the UGC Course on Advanced L^AT_EX presented by Martin.
This is the UGC Course on Advanced L^AT_EX presented by Martin and Someone.

Custom L^AT_EX Commands/Macros

You cannot use digits when naming commands/macros, only letters.

L^AT_EX will not allow you to create new commands with the same name as existing commands.

Custom L^AT_EX Commands - changing commands

If you want to re-use the name of an existing command, or change the definition of an existing command, use the `\renewcommand` option:

```
\renewcommand{name}[num]{definition}
```

So, for example, to change Chapter headings from 'Chapter' to 'Bigger Section':

```
\renewcommand{\chaptertitle}{Bigger Section}
```

Custom L^AT_EX Environments

We can specify our own environments using the `\newenvironment` command:

```
\newenvironment{name}[num]{before}{after}
```

Any code given in the before block is processed after the `\begin{name}`.
Any code given in the after block is processed at the `\end{name}`.

Custom L^AT_EX Environments - example

```
\newenvironment{dotty}{\noindent\textbullet}{\dotfill}  
  
\begin{dotty}  
Here is some text  
\end{dotty}
```

- Here is some text

Custom L^AT_EX Environments - counters

We can create environments with counters included, much like the Figure or Table environments, by declaring a counter with the `\newcounter` command:

```
\newcounter{examplecounter}  
\newenvironment{numberedexample}{\refstepcounter{  
examplecounter}\textbf{Example \arabic{examplecounter}  
}}\qquad{}
```

Custom L^AT_EX Environments - counters

```
\newcounter{examplecounter}  
\newenvironment{numberedexample}{\refstepcounter{examplecounter}\  
textbf{Example \arabic{examplecounter}}\quad}{}  
  
\begin{numberedexample}  
An example  
\end{numberedexample}
```

Example 1	An example
------------------	------------

Source Code in L^AT_EX

The `listings` package provides us with an easy way to include source code within our L^AT_EX documents.

It allows you to use the `lstlistings` environment to add formatted source code into your document, including features such as line numbers and syntax highlighting.

The listings package

Most simply, we use `listings` by first including the package in our document:

```
\usepackage{listings}
```

and then by including our source code within a `lstlistings` environment.

The listings package - code input

Alternatively, we can input source code directly from the source itself:

```
\lstinputlisting[language=Java]{source_filename.java}
```

or import just part of a file:

```
\lstinputlisting[language=Java, firstline=23, lastline=31]{source_filename.java}
```

The listings package - options

The listings package supports many different languages and has many different options to control how the code is displayed, including:

Type	Use
<code>backgroundcolor</code>	set the background colour
<code>basicstyle</code>	set the code font size
<code>captionpos</code>	set the caption position
<code>commentstyle</code>	comment style
<code>frame</code>	add a frame around source code
<code>morekeywords={...}</code>	add extra keywords
<code>numbers=left</code>	where to add line numbers
<code>numbersep</code>	how much space between numbers and code
<code>numberstyle</code>	set the style of the line numbers
<code>showspaces</code>	add underscores to show spaces
<code>showtabs</code>	add underscores to show tabs
<code>stepnumber</code>	how many lines between line numbers

The listings package - options

So, for example, we could define some options for our code as:

```
\lstset{
  language=C
  backgroundcolor=\color{gray}
  frame=single,
  numbers=left,
  numbersep=6pt,
  numberstyle=\tiny\color{green},
  stepnumber=2,
  breaklines=true
}
```

The listings package - styles

We can also define styles - or groups of options - so we can format individual pieces of code separately:

```
\lstdefinestyle{python}{  
    language=Python  
    backgroundcolor=\color{gray}  
    frame=single,  
    numbers=left,  
    numberstyle=\tiny\color{green},  
    stepnumber=2,  
}  
  
\lstdefinestyle{java}{  
    language=Java  
    backgroundcolor=\color{blue}  
    numbers=right,  
    numberstyle=\tiny\color{red},  
    stepnumber=1,  
}
```


The listings package - styles

We can then use our different styles by specifying the style as an option on the listing environment:

L^AT_EX Tips and Tricks

Use `\thispagestyle{empty}` to suppress page numbers on a page.

Use a starred version of a sectioning command to suppress numbering (`\section*{section_name}`).

Use `\marginpar{notes in margin}` to add some notes in the margin of a document - useful when commenting or giving feedback.

L^AT_EX Tips and Tricks

Use the `hyperref` package to add hyperlinks and navigation in your .pdf output

```
\usepackage[pdftex,  
  pdfauthor={Martin Chorley},  
  pdftitle={Advanced LaTeX},  
  pdfpagelayout=TwoColumnRight,  
  pdfborder=0  
{hyperref}
```

L^AT_EX Tips and Tricks

Use the `\input` command to include multiple L^AT_EX files within the same document.

This allows you to separate your document across multiple files.

```
\begin{document}  
  \input{introduction}  
  \input{results}  
  \input{conclusion}  
\end{document}
```

L^AT_EX Tips and Tricks

Use the `\rotating` package to rotate items that are too wide to fit on a page.

```
\usepackage{rotating}
\begin{document}
  \begin{sideways}
    % table or item that is too wide goes here
  \end{sideways}
\end{document}
```

Page headers and footers are not affected, only the content within the `sideways` environment