Lists, tables, figures

It is perfectly possible to write whole documents using nothing but section headings and paragraphs. As mentioned in section 2.6 on page 47, novels, for example, usually consist just of chapters divided into paragraphs. However, it’s more common to need other features as well, especially if the document is technical in nature or complex in structure.

In Chapter 2 starting on page 33 we saw how to create a hierarchical document structure with chapters and sections and paragraphs; this chapter covers the other building-blocks which you need within your structure: lists, tables, figures (including images), boxes like sidebars and panels, block quotations, and verbatim text (computer program listings). In Chapter 5 starting on page 107 we will cover the textual tools that you need inside text: footnotes, marginal notes, cross-references, citations, indexes, and glossaries.

---

1 It’s worth pointing out that ‘technical’ doesn’t necessarily mean ‘computer technical’ or ‘engineering technical’, least of all ‘mathematical technical’: it just means it contains a lot of $\varepsilon$, Greek for specialist material or artistry. A literary analysis such as *La Textualisation de Madame Bovary* (on the marginal notes in the manuscripts of Flaubert’s novel) is every bit as technical in the literary or linguistic field as the maintenance manual for the Airbus 380 is in the aircraft engineering field.
4.1 Lists

Lists are useful tools for arranging thoughts in a digestible format, usually a small piece of information at a time. There are four basic types of list, shown in Table 4.1.

<table>
<thead>
<tr>
<th>Type of List</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Random or itemized lists</strong></td>
<td>(sometimes called ‘arbitrary’ or ‘bulleted’ lists) where the order of items is unimportant. The items are often prefixed with a bullet or other symbol for clarity or decoration, but are sometimes simply left blank, looking like miniature paragraphs (when they are known as ‘simple’ or ‘trivial’ lists).</td>
</tr>
<tr>
<td><strong>Enumerated or ordered lists</strong></td>
<td>(sometimes called ‘sequential’ or ‘numbered’ lists) where the order of items is critical, such as sequences of instructions or rankings of importance. The enumeration can be numeric (Arabic or Roman), or lettered (uppercase or lowercase), and can be programmed to be hierarchical (1.a.viii, 2.3.6, etc).</td>
</tr>
<tr>
<td><strong>Descriptive or labelled lists</strong></td>
<td>(sometimes called ‘discussion’ lists), which are composed of subheadings or topic labels (usually unnumbered but typographically distinct), each followed by one or more indented paragraphs of discussion or explanation.</td>
</tr>
<tr>
<td><strong>Inline lists</strong></td>
<td>which are sequential in nature, just like enumerated lists, but are a) formatted within their paragraph; b) usually labelled with letters like this example; and c) often mutually inclusive or exclusive, with the final item prefixed by ‘and’ or ‘or’ respectively.</td>
</tr>
</tbody>
</table>

There are actually two other types, segmented lists and reference lists, but these are much rarer, and outside the scope of this document.

The structure of lists in \LaTeX{} is identical for each type, but with a different environment name. Lists are another example of this \LaTeX{} technique (environments), where a pair of matched commands surrounds some text which needs special treatment.

Within a list environment, list items are always identified by the command `\item` (followed by an item label in [square brackets] in the case of labelled lists). You don’t type the bullet or the number or the formatting, it’s all automated.
4.1 Lists

4.1.1 Itemized lists

To create an itemized list, use the \texttt{itemize} environment:

\begin{itemize}
\item Itemized lists usually have a bullet;
\item Long items use ‘hanging’ indentation, whereby the text is wrapped with a margin which brings it clear of the bullet used in the first line of each item;
\item The bullet can be changed for any other symbol, for example from the \texttt{bbding} or \texttt{pifont} package.
\end{itemize}

\begin{itemize}
\item Itemized lists usually have a bullet;
\item Long items use ‘hanging indentation’, whereby the text is wrapped with a margin which brings it clear of the bullet used in the first line of each item;
\item The bullet can be changed for any other symbol, for example from the \texttt{bbding} or \texttt{pifont} package.
\end{itemize}

The default list bullet is the normal round, solid one ($\bullet$), which is also available with the command \texttt{textbullet} if you load the \texttt{textcomp} package. See section 7.6.1 on page 171 for details of how to change the settings for list item bullets.

Nested itemized lists (see section 4.1.6 on page 77) used differing symbols for their bullets as well as more indentation and less spacing.

4.1.2 Enumerated lists

To create an enumerated list, use the \texttt{enumerate} environment:

\begin{enumerate}
\item\texttt{Formatting Information}
\end{enumerate}
\begin{enumerate}
\item Enumerated lists use numbering on each item (can also be letters or roman numerals);
\item Long items use ‘hanging indentation’ in just the same way that itemized lists do;
\item The numbering system can be changed for any level.
\end{enumerate}

1. Enumerated lists use numbering on each item (can also be letters or roman numerals);

2. Long items use ‘hanging indentation’, in just the same way that itemized lists do;

3. The numbering system can be changed for any level.

See section 4.1.6 on page 77 for details of how to change the numbering schemes for each level.

In standard \LaTeX{} document classes, the vertical spacing between items, and above and below the lists as a whole, is more than between paragraphs. If you want tightly-packed lists, use the \texttt{enumerate} package, which provides an environment option \texttt{nosep} for the three main list environments (there is also a \texttt{nosep} option for even more compact spacing). Both these options come \texttt{after} the environment name, not before; eg \texttt{\begin{itemize}[nosep]}

4.1.3 Description lists

To create a description list, use the \texttt{description} environment:
4.1 LISTS

\begin{description}
\item[Identification:] description lists require a topic for each item given in square brackets;
\item[Hanging indentation:] Long items use this in the same way as all other lists;
\item[Reformatting:] Long topic labels can be reprogrammed to fold onto multiple lines.
\end{description}

I very strongly recommend using the \texttt{enumitem} package with its \texttt{unboxed} environment option for description lists, which avoids the spacing problems with \LaTeX's default handling of long labels. This package has so many good features I tend to load it for virtually every document I create.

All three of these types of lists can have multiple paragraphs per item: just type the additional paragraphs in the normal way, with a blank line between each. So long as they are still contained within the enclosing environment, they will automatically be indented to follow underneath their item.

4.1.4 Inline lists

Inline lists are a special case, as they require the use of the \texttt{enumitem} or \texttt{paralist} packages.

The \texttt{enumitem} package with the \texttt{inline} option provides ‘starred’ versions of the three standard list types to do this: enumerate*, itemize*, and description*. It uses a specification in the optional argument for formatting the labels (for example, italic letters and an upright parenthesis), and it also provides extensive support for the punctuation and conjunction between items, making it unnecessary to type it separately for each item (and differently for the last-but-one).
\usepackage[inline]{enumitem} 

... 
\textbf{Inline lists}, which are sequential in nature, just like enumerated lists, but are 
\begin{enumerate}[label={\textit{\textsc{alpha}}*},
  itemjoin={\{}; \}},itemjoin*=\{}; and \}]
\item formatted within their \textbf{paragraph} 
\item usually labelled with letters 
\item usually have the final item prefixed with ‘and’ or ‘or’ 
\end{enumerate}, like this.

The \texttt{paralist} package provides an \texttt{inparaenum} environment, again with the optional formatting specification in square brackets:

\usepackage{paralist} 

... 
\textbf{Inline lists}, which are sequential in nature, just like enumerated lists, but are 
\begin{inparaenum}[	extit{a}]
\item formatted within their \textbf{paragraph}; 
\item usually labelled with letters; and 
\item usually have the final item prefixed with ‘and’ or ‘or’ 
\end{inparaenum}, like this.

See Chapter 6 starting on page 129 for details of the font-changing commands used in the optional arguments to the \texttt{enumerate*} and \texttt{inparaenum} shown in these examples.
4.1 LISTS

Exercise 11. List practice

Add some lists to your document. Pick any two of the ones described here to practice with.

Try the `enumitem` package or the `paralist` package (read the documentation).

4.1.5 Reference lists and segmented lists

Reference lists are visually indistinguishable from numbered or lettered lists, but the numbering or lettering does not imply a sequence. The numbers or letters are just used as labels so that the items can be referred to from elsewhere in the text (as in ‘see item 501(c)3’). In this sense they are really a kind of sub-sectional division, and \TeX’s \texttt{paragraph} or \texttt{subparagraph} commands (with appropriate renumbering) would probably be a far better solution than using a list. Label them and refer to them with \texttt{label} and \texttt{ref} as for any other cross-reference (see section 5.3 on page 109).

Segmented lists are a highly specialised structure and outside the scope of this document. For details of their usage, see the ‘TEI Guidelines’ (Burnard & Sperberg-McQueen, 1995).

4.1.6 Lists within lists

You can start a new list environment within the item of an existing list, so you can embed one list inside another up to four deep. The lists can be of any type, so you can have a description list containing an item in which there is a numbered sub-list, within which there is an item containing a bulleted sub-sub-list.

1. by default an outer enumerated list uses Arabic numerals;
   (a) an embedded enumerated list is lettered in lowercase;
     i. a third level is numbered in lowercase Roman numerals;
     A. the fourth level uses uppercase alphabetic letters.

Multiple embedded lists automatically change the bullet or numbering scheme so that the levels don’t get confused, and the spacing
between levels is adjusted to become slightly tighter for more deeply nested levels.

- by default the outer itemized list item has a bullet;
  - an embedded itemized list uses a dash;
    * a third level uses an asterisk;
    · the fourth level uses a small bullet.

These are only defaults and can easily be changed by redefining the relevant set of values. You could also add a fifth and further levels, although I suspect that would mean your document structure needed some careful analysis, as lists embedded five deep will probably confuse your readers.

The values for lists come in pairs: for each level there is a counter to count the items and a command to produce the label:\footnote{In fact, any time you define a counter in \LaTeX, you automatically get a command to reproduce its value. So if you defined a new counter \texttt{example} to use in a teaching book, by saying \texttt{\newcounter{example}}, that automatically makes available the command \texttt{\theexample} for use when you want to display the current value of \texttt{example}.}

Note that each counter and command ends with the Roman numeral value of its level (this is to overcome the rule that \LaTeX commands can only be made of letters — digits wouldn’t work here). To change the format of a numbered list item counter, just renew the meaning of its label:

\begin{verbatim}
\renewcommand{\theenumi}{\Alph{enumi}}
\renewcommand{\theenumii}{\roman{enumii}}
\renewcommand{\theenumiii}{\arabic{enumiii}}
\end{verbatim}

\footnote{In fact, any time you define a counter in \LaTeX, you automatically get a command to reproduce its value. So if you defined a new counter \texttt{example} to use in a teaching book, by saying \texttt{\newcounter{example}}, that automatically makes available the command \texttt{\theexample} for use when you want to display the current value of \texttt{example}.}

\hspace{1cm}

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|}
\hline
\textbf{Level} & \textbf{Default} & \textbf{Counter} & \textbf{Label command} \\
\hline
1 & digit. & \texttt{enumi} & \texttt{\theenumi} \\
2 & (letter) & \texttt{enumii} & \texttt{\theenumii} \\
3 & roman. & \texttt{enumiii} & \texttt{\theenumiii} \\
4 & LETTER. & \texttt{enumiv} & \texttt{\theenumiv} \\
\hline
\end{tabular}
\caption{Default numbering for nested numbered lists}
\end{table}
4.2 TABLES

This would make the outermost list use uppercase letters, the second level use lowercase roman, and the third level use ordinary Arabic numerals. The fourth level would remain unaffected.

Exercise 12. Nesting

Extend your use of lists by nesting one type inside a different one.

4.2 Tables

Tabular typesetting is the most complex and time-consuming of all textual features to get right. This holds true whether you are typing in plaintext form, using a wordprocessor, using \texttt{\LaTeX}, using HTML or XML, using a DTP system, or some other text-handling package.

Fortunately, \texttt{\LaTeX} provides a table model with a mixture of defaults and configurability to let it produce very high quality tables with a minimum of effort.

4.2.1 Floats

Tables and Figures (and several other features of documents like sidebars) are what printers and publishers refer to as ‘floats’. This means they are not part of the normal stream of sentences of text, but separate freestanding entities, positioned in a part of the page to themselves (top, middle, bottom, left, right, or wherever the designer specifies). They

Lists and Tables: a caution to the unwary

Treat lists with care: people sometimes use tables for labelled information which is really a list and would be better handled as such. They often do this because their wordprocessor has no way to do what they want (usually to place the item label level with the description or explanation) except by using a table, hence they are misled into believing that their text is really a table when it’s actually not.
CHAPTER 4. LISTS, TABLES, FIGURES

Terminology

\LaTeX, in common with standard typesetting practice, uses the word ‘Table’ to mean a formal textual feature, numbered, with a caption, and containing an aligned grid of numbers or text, referred to from the surrounding document (as in ‘See Table 5’). A Table is the whole thing, not just the grid, and — critically — it floats (see section 4.2.1 on the previous page).

The grid arrangement of information in rows and columns within either of these structures is called a ‘tabulation’ or ‘tabular matter’.

It is important to keep this distinction firmly in mind for this section.

You can also have ‘informal’ tables, which simply grids occurring between two paragraphs, without caption or number or reference: there’s one in section 1.4.1 on page 12.

always have a caption describing them and they are always numbered so they can be referred to from elsewhere in the text.

\LaTeX automatically floats Tables and Figures, depending on how much space is left on the page at the point that they are processed. If there is not enough room on the current page, the float is moved to the top of the next page. This can be changed by moving the Table or Figure to an earlier or later point in the text, or by adjusting some of the parameters which control automatic floating.

Authors sometimes want many floats occurring in rapid succession, which raises the problem of how they are going to fit on the page and still leave room for text. In this case, \LaTeX stacks them all up and prints them together if possible, or leaves them to the end of the chapter in protest.

The skill is to space them out within your text so that they intrude neither on the thread of your argument or discussion, nor on the visual balance of the typeset pages. But this is a skill few authors have, and it’s one point at which professional typographic advice or manual intervention may be needed.

There is a float package which lets you create new classes of floating object (sidebars, examples, exercises, etc), and it also implements a
method of forcing a float not to float (that is, to appear where it occurs in the text, even if that breaks the page layout).

Please now read from section 4.2.1 on page 79 up to here a second time.

4.2.2 Normal tables

To create a \LaTeX Table, use the \texttt{table} environment containing a \texttt{\caption} command followed by a \texttt{\label} command (the label can be used to refer to the table: see section 5.3.1 on page 110).

\begin{table}
\caption{Project expenditure to year-end 2016}
\label{ye2016exp}
...
\end{table}

Numbering is automatic, but the \texttt{\label} command \textbf{MUST} \textit{follow} the \texttt{\caption} command, not precede it. The numbering automatically includes the chapter number in document classes where this is appropriate (but this can of course be overridden). The \texttt{\caption} command has an optional argument to provide a short caption if the full caption would be too long for the \texttt{List of Tables}:

\begin{table}
\caption[Something short]{Some very long caption that will only look reasonable in the full figure.}
\end{table}

4.2.3 Simple tabular matter

Within a Table, there are four ways to enter the data:

\textbf{By hand}: you can enter the tabular matter (cell data) by typing it in, which is perhaps the most common method;

\textbf{In a grid tool}: many \LaTeX editors come with a pop-up grid tool like a miniature spreadsheet, which makes creating tabular matter easier, at the cost of some loss of control;

\textbf{With a package}: if the quantity of data is very large and is already in a spreadsheet, or if it is spreadsheet data which will change frequently before you are finished, you can use the \texttt{datatool}
package to read the data from a spreadsheet CSV export file (see section 4.2.4 on page 85);

**As an image**: it is also possible to include a ‘table’ which has actually been captured as an image from elsewhere, such as a screenshot from a spreadsheet (but then this isn’t really a table). We will see how to include images in section 4.3 on page 89 on Figures, where they are more common.

In Table 4.3 we have a table which we’ll use as an example. It’s got a number, a caption, three columns with headings and some ruled lines, and a comment afterwards.

**Table 4.3**: Project expenditure to year-end 2016

<table>
<thead>
<tr>
<th>Item</th>
<th>€ Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Salaries (2 part-time research assistants)</td>
<td>28,000</td>
</tr>
<tr>
<td>Conference fees and travel expenses</td>
<td>14,228</td>
</tr>
<tr>
<td>Computer equipment (5 workstations)</td>
<td>17,493</td>
</tr>
<tr>
<td>Software</td>
<td>3,562</td>
</tr>
<tr>
<td>b) Rent, light, heat, etc</td>
<td>1,500</td>
</tr>
<tr>
<td>Total</td>
<td>64,783</td>
</tr>
</tbody>
</table>

The Institute also contributes to (a) and (b).

To typeset tabular matter, within a table environment or elsewhere, you use the tabular environment. This takes one compulsory argument which specifies how many columns and what type they are. You follow the \begin{tabular} command with a pair of curly braces giving the alignment of the columns.

**Column alignment**: You give one letter for each column using one of l, c, and r for a left-aligned, centered, or right-aligned column. The number of letters **MUST** be the same as the number of columns you are putting in the table.

\begin{tabular}{clr} ... \end{tabular}

In the example in Table 4.3, the tabular setting has three columns, the first one centered, the second left-aligned, and the third one right-aligned, so it is specified as \{clr\}. 

*Format setting*
4.2. TABLES

**Cell and row division:** You can then type in each row, making sure each cell’s data in the row is separated with an & character, and each row ends with a double backslash (\). You don’t need to add any extra spaces or do any manual formatting, although you can if you want: \LaTeX just uses the column specification to know how to format it.

| Salaries (2 part-time research assistants) | \textbackslash
|-------------------------------------------|--------------------------|
| \textbackslash| 28,000\textbackslash
|

If a cell has nothing to go in it, you just don’t type anything, but the ampersand must still be there:

| Total | 64,783\textbackslash
|-------|--------------------------|

**Column headings:** These are often set in **bold type**, as in the example (see ‘Cell formatting’ below).

<table>
<thead>
<tr>
<th>Item</th>
<th>EUR Amount</th>
</tr>
</thead>
</table>
|\textbf{Item} & \textbf{EUR \ Amount} & {\texttt{6pt}}\textbackslash
|----------------|----------------|---|
|hline

In this case there is also some extra space (6pt, see ‘Row spacing’ below) and a horizontal line across the table (see ‘Tables rules’ below).

All the data for a row may be longer than a line in your editor, but it can take up as many lines as needed; the end of the row is signalled by the double backslash, so \LaTeX knows when it’s time for the next row.

**Cell formatting:** Font changes can be done within a cell (bold, italic, etc; we’ll come on to these later, see section 6.2.6 on page 149) and these changes are limited to the cell in which they occur: they do not ‘bleed’ across cells (in the example, the column headings have each been made bold separately).

**Row spacing:** Additional vertical white-space **below** a row (but above a rule) can be specified by giving a dimension in [square brackets] immediately after the double backslash which ends the row (3pt in the case of the last row before the totals in the example). A negative value will decrease the spacing below that row.
If the line below a horizontal rule looks too close, it can be optically spaced by adding a \textbf{strut} at the start of the next line (that is, after the \hline). A 'strut' is hidden vertical rule a little bit higher than the row-height; hidden because its width is zero, making it invisible, as in the example code. The \rule command can be used for this, with a width of 0pt and height of 1.2em, just a fraction higher than the text, forcing the lines apart by 0.2em.

\begin{table}
\caption{Project expenditure to year-end 2016}
\label{ye2016exp}
\centering\smallskip
\begin{tabular}{|l|r|}
\hline
Item & EUR Amount \\
\hline
\textbf{a)} Salaries (2 part-time research assistants) & 28,000 \\
 & Conference fees and travel expenses & 14,228 \\
 & Computer equipment (5 workstations) & 17,493 \\
 & Software & 3,562 \\
\textbf{b)} Rent, light, heat, etc. & 1,500 \\
\hline
\textbf{Total} & 64,783 \\
\end{tabular}
\par\medskip\footnotesize
The Institute also contributes to (a) and (b).
\end{table}

Table rules: A line across the whole table is done with the \hline command after the double-backslash which ends a row.

For a line which only covers some of the columns, use the \cline command (in the same place), with the column range to be ruled in curly braces. If only one column needs a rule, it must still be given as a range (eg in the example, \{3-3\}).

Vertical rules (between columns) can be specified in the column specifications with the vertical bar character (\|) before, after, or between the l, c, r letters. This character creates rules which extend the whole height of a table: it is not necessary to repeat them every row.

I have indented the code example given just to make the elements of the table clearer to read: this is for editorial convenience, and has no effect on the formatted result (see Table 4.3 on page 82). If you copy and paste this into your example document, you will need to add the marvosym package to your Preamble, which will let you
use the official CEC-conformant Euro symbol command \EUR\ (€ as distinct from €).

4.2.4 More complex tabular formatting

\TeX’s original tabular settings were designed for classical numerical grids, where each cell contains a single value. If you need a cell to contain multiline text, like a miniature paragraph, you can use the letter \emph{p} followed by a width in curly braces instead of an \emph{l}, \emph{c}, or \emph{r}. Thus \emph{p\{3.5cm\}} would mean a column in which each cell would be 3.5cm wide, and could contain more than one line of text or values.

\begin{tabular}{cp{3.5cm}r}

These \emph{p} column specifications are \emph{not} multi-row (row-spanned) entries: they are single cells which contain multiple lines of typesetting: the distinction is extremely important. These paragraphic cells are typeset justified (two parallel margins) and the baseline of the top line of text is aligned with the baseline of neighbouring cells in the row.

The \emph{array} package provides some important enhancements which overcome the limitations of the \emph{p} cells:

**Vertical alignment:** In addition to the \emph{p}, whose vertical alignment baseline is the the top line of text, the \emph{array} package provides the \emph{m} and \emph{b} letters. These work the same way as \emph{p} (followed by a width in curly braces), but their vertical alignment baseline is the middle or bottom of the cell respectively.

**Prefixes and suffixes:** With the \emph{array} package, any column specification letter can be preceded by > and some \LaTeX\ commands in curly braces. These commands are applied to every cell in that column, so to make a \emph{p} column typeset ragged-right you would say, for example, >{\raggedright}p\{3.5cm\} (or \raggedleft, or \centering).

Note that if you do this, the final column specification must include a prefix or suffix containing the \texttt{arraybackslash} command, to revert the meaning of the double-backslash, which gets redefined by horizontal formatting commands like \raggedright.
There is a suffix format as well: you can follow a column letter with < and code in curly braces (often used to turn off math mode started in a prefix).

The \texttt{colortbl} package lets you colour rows, columns, and cells; and the \texttt{dcolumn} package provides decimal-aligned column specifications for scientific or financial tabulations. Multi-column (column-spanning) is built into \LaTeX{} tables with the \texttt{\%multicolumn} command; but for multi-row (row-spanning) cells you need to add the \texttt{multirow} package. Multi-page and rotated (landscape format) tables can be done with the \texttt{longtable}, \texttt{rotating}, and \texttt{landscape} packages.

The \LaTeX{} table model is very different from the HTML auto-adjusting model used in web pages; it’s closer to the CALS model used in technical documentation. However, auto-adjusting column widths are possible with the \texttt{tabularx} and \texttt{tabulary} packages, offering different approaches to dynamic table formatting.

You do not need to format the tabular data in your editor: \LaTeX{} does this for you when it typesets the table, using the column specifications you provided. You can give the cell values all on one line, or split over many lines: it makes no difference so long as cells are separated with the \& and rows are ended with the double-backslash.

As mentioned earlier, some editors have a grid-like array editor for entering tabular data. Takaaki Ota provides an excellent \texttt{tables-mode} for \texttt{Emacs} which uses a spreadsheet-like interface and can generate \LaTeX{} table source code (see Figure 4.1 on the next page).

If your tabular data comes from outside \LaTeX{}, Nicola Talbot’s excellent \texttt{datatool} package allows the import of data from spreadsheet .csv files and other sources.

### 4.2.5 More on tabular spacing

Extra space, called a ‘shoulder’, is automatically added on both sides of all columns by default. The default value is 6pt, so you get that amount left and right of the tabulation; because it is added left and right of every cell, the space between columns is therefore 12pt by default. This can be adjusted by changing the value of the \texttt{\%tabcolsep} dimension \texttt{before} you begin the tabular environment.

\begin{verbatim}
\setlength{\tabcolsep}{3pt}
\end{verbatim}
Figure 4.1: Tables mode for Emacs

The shoulder can be omitted in specific locations by adding the code `@{}` in the appropriate places. For example to omit it at the left-hand and right-hand sides of a tabular setting, put it at the start and end of the column specifications (putting it between two column specifications will remove all space between those columns).

\begin{tabular}{@{}|c|c|c@{}}

To change the row-spacing in a tabular setting, you can redefine the `\arraystretch` command (using `\renewcommand` because it’s defined as a command, not a length). `\arraystretch` is actually a multiplier, preset to 1, so `\renewcommand{\arraystretch}{1.5}` would make the lines of your tabular settings one and a half times bigger than normal.

It is conventional to centre the tabular setting within a Table, using the `center` environment (note US spelling) or the `centering` command (as in the example), but this is an æsthetic decision (or...
perhaps one mandated by your publisher: some journals insist instead that all tabular material is set flush to the left-hand margin).

If there is no data for a cell, just don’t type anything — but you still need the & separating it from the next column’s data. The astute reader will already have deduced that for a table of \( n \) columns, there must always be \( n - 1 \) ampersands in each row. The exception to this is when the \texttt{\multicolumn} command is used to create cells which span multiple columns, when the ampersands of the spanned columns are omitted.

\subsection*{4.2.6 Tabular techniques for alignment}

As mentioned earlier, it’s also perfectly possible to typeset tabular matter outside a formal Table, where you want to lay out an informal tabulation where a fully floating formal Table would be unnecessary (these are usually quite short: there are several of them in this document).

By default, \LaTeX{} typesets tabular environments \textit{inline} to the surrounding text (that is, within the paragraph, as if the whole tabular environment was a character), so if you want your alignment displayed by itself, put it between paragraphs, inside a positioning environment like \texttt{center}, \texttt{flushright}, or \texttt{flushleft}, or leave a blank line or \texttt{par} before and after so it gets typeset separately.

One side-effect of this is that small and intricately constructed micro-tabulations can be used to good effect when creating special effects like logos, as they get treated like a character and can be typeset anywhere.

\begin{verbatim}
\begin{tabular}
  @{} & \texttt{\raggedright}p{.5\textwidth} @{} \\
  @{} & \texttt{\raggedleft\arraybackslash}p{.5\textwidth} @{} \\
\end{tabular}

left-hand material
&
right-hand material\\
\end{tabular}
\end{verbatim}

Tabular setting can also be used wherever you need to align material side by side, such as in designing letterheads, where you may want
your company logo and address on one side and some other information on the other side to line up with each other. One common way to implement ‘spring’ margins like this is to create two columns each half the width of the page, allowing for the extra space that would otherwise be added automatically between columns and at the edges:

As mentioned earlier, the @{} suppresses the inter-column gap (or the shoulder left or right) so that the total width available will be the full text width of the page.

### Exercise 13. Create a tabulation

Create one of the following in your document:

1. a formal Table with a caption showing the number of people in your class broken down by age and sex;
2. an informal tabulation showing the price for three products;
3. the logo \[ \text{YEAR} \ \text{2000} \] (hint: section 4.6.2 on page 101).

### 4.3 Figures

As explained in section 4.2.1 on page 79, Figures and Tables float to a vacant part of the page, as they are not part of your normal text, but illustrative objects that you refer to.

\begin{figure}
\caption{Total variable overhead variance (after \textcite{p.191}{bull})}
\label{workeff}
\centering
\fbox{\includegraphics[width=.75\columnwidth]{diagram}}
\end{figure}

To create a figure, use the \texttt{figure} environment. Like Tables, they automatically get numbered, and they MUST include a \texttt{caption} (with a \texttt{label} after it, if needed for cross-referencing). Like Tables, it is conventional to centre the material, but that is a personal choice.
You can see that the structure is very similar to the `table` environment, but in this case we have a graphic included with the `\includegraphics` command. Here, it's also enclosed in an `\fbox`, which places a frame box around it (see section 4.6.2 on page 101). Details of including graphics are in the next section: you need the `graphicx` package. Details of the bibliographic citation mechanism used in the caption are in section 5.3.2 on page 111.

Figures can contain text, diagrams, pictures, or any other kind of illustration, even a `tabular` environment — \LaTeX\ is agnostic on this point, so Tables can contain an image (of a table, presumably) and Figures can contain a tabulation. What matters is that you describe them properly.

Figure 4.2: Total variable overhead variance (after Bull (1972, p. 191))

The content of the Figure could of course also be textual, in the form of lists, paragraphs, or other blocks of text. For drawings, \LaTeX\ has a very simple drawing environment called `picture`, which lets you create a limited set of lines and curves, but for a diagram of any complexity, you should use a standard vector drawing program (see ??)
4.4. Images

Images (graphics) can be included anywhere in a \LaTeX{} document, although in most cases of formal documents they will occur in Figures (see preceding section). To use graphics, you need to use the graphicx package in your Preamble: \texttt{usepackage{graphicx}}. This package provides the command \texttt{includegraphics} which is used to insert an image in a document. The command is followed by the name of your graphics file \texttt{without the filetype} (we’ll see in a minute why you don’t normally need to include the filetype).

\begin{verbatim}
\includegraphics{myhouse}
\end{verbatim}

In most cases you should just make sure the image file is in the same folder (directory) as the document you use it in. This avoids a lot of messing around remembering where you put the files; but you could instead put them all in a single folder and include that as part of the filename you use in the command.

\begin{verbatim}
\includegraphics{images/myhouse}
\end{verbatim}

If you have images you want to use in several different documents in different places on your disk, there is a way to tell \LaTeX{} where to look (see section 4.4.4 on page 96).

4.4.1 Supported image file formats

The type of image file you use depends on \LaTeX{} processor you are using (see section 1.2 on page 4 for how to choose). The common file type are:

\begin{itemize}
  \item You may find a lot of old files which use a package called epsf. Don’t use it: It’s obsolete.
\end{itemize}
Joint Photographic Experts Group (JPG), used for photographs and scanned images;
Portable Network Graphic (PNG), used for photographs and scanned images;
Portable Document Format (PDF), used for vector graphics (drawings, diagrams) and typographic output from other programs;
Encapsulated Postscript (EPS), an old publishing industry standard for many years, and the forerunner of PDF, still used by some older programs that generate diagrammatic or typographic output.

See ?? on page ?? and ?? on page ?? for other file formats. For more details, see the answers to the question Which graphics formats can be included in documents processed by latex or pdflatex?

For X\LaTeX and PDFLaTeX (creating PDF output) Graphics files MAY be in PNG, PDF, or JPG (JPEG) format.

For the original \LaTeX (creating DVI output): Graphics MUST be in EPS format: no other format will work (see ?? on page ??).

1. PNG actually gets converted to the PDF internal format automatically (at a small penalty in terms of speed) so for lots of images, or very large images, use JPG format or preconvert them to PDF;

2. It is also of course possible to convert (repackage) your JPG pictures to PDF, using any of the standard graphics conversion/manipulation programs (see ?? on page ?? for details). Preconverting all your images to PDF makes them load into your document slightly faster.

3. X\LaTeX and PDFLaTeX will search for the graphic file by file type, in this order (check for the newest definition in your pdftex.def): .png, .pdf, .jpg, .mps, .jpeg, .jbig2, .jb2, .PNG, .PDF, .JPG, JPEG, JBIG2, and JB2. Thanks to Enrico Gregorio and Philipp Stephani on comp.text.tex for locating this for me.

4. See ?? on page ?? for more about how to create and manage your image files.
4.4. IMAGES

sam2p

Péter Szabó’s *sam2p* utility converts from several image formats to PDF (or EPS). It is available precompiled for Windows and Linux: Linux users may also find it in their distribution’s repositories. Due to the way the program is compiled it is not included in the TeX Live distributions.

### 4.4.1 Other file formats

Convert them to one of the supported formats using a graphics editing or conversion tool such as *GIMP* or *ImageMagick*.

Some commercial distributions of TeX systems allow other formats to be used, such as GIF, Microsoft Bitmap (BMP), or Hewlett-Packard’s Printer Control Language (PCL) files, and others, by using additional conversion software provided by the supplier; but you cannot send such documents to other TeX users and expect them to work if they don’t have the same distribution installed as you have.

It is in fact possible to tell TeX to generate the right file format by itself during processing, but this requires an external command-line graphics converter, and as it gets done afresh each time, it may slow things down rather a lot.

### 4.4.2 Postscript

Since TeX 2010, EPS files will be automatically converted to PDF in PDFLaTeX or XeLaTeX if you include the `epstopdf` package. This avoids need to keep your graphics in two formats, at the expense of a longer compile time while it converts every EPS image (not recommended).

All good graphics packages (eg *GIMP*, *PhotoShop*, *Corel Draw*, etc) can save images as EPS, but be very careful with other software such as statistics, engineering, mathematical, and numerical analysis packages, because some of them, especially on Microsoft Windows platforms, use a very poor quality driver, which in turn creates very poor quality EPS files. If in doubt, check with an expert. If you find an EPS graphic doesn’t print, the chances are it’s been badly made by the creating software. Downloading Adobe’s own Postscript driver from their Web site and using that instead may improve things, but the only real solution is to use software that creates decent output.
For these reasons, if you create vector EPS graphics, and convert them to PDF format, do not keep additional JPG or PNG copies of the same image in the same directory, because they risk being used first by PDFLaTeX instead of the PDF file, because of the order in which it searches.

EPS files, especially bitmaps, can be very large indeed, because they are stored in ASCII format. Staszek Wawrykiewicz has drawn my attention to a useful MS-DOS program to overcome this, called cep ('Compressed Encapsulated PostScript') available from CTAN archive in the support/pstools directory, which can compress EPS files to a fraction of their original size. The original file can be replaced by the new smaller version and still used directly with \includegraphics.

One final warning about using EPS files with \includegraphics: never try to specify an absolute path (one beginning with a slash) or one addressing a higher level of directory (one beginning with ../). The dvips driver will not accept these because they pose a security risk to PostScript documents. Unlike PDF, PostScript is a real programming language, capable of opening and deleting files, and the last thing you want is to create a document able to mess with your filesystem (or someone else's).

4.4.2 Resizing images

The \includegraphics command can take optional arguments within square brackets before the filename to specify the height or width, as in the example below. This will resize the image that prints; whichever dimension you specify (height or width) the other dimension will automatically be scaled in proportion to preserve the aspect ratio.

The file on disk does not get changed in any way, and nor does the copy included inside the PDF: what gets changed is just the size that it displays at in the finished document. So if you include a huge JPG but tell \LaTeX{} to print it at a small size, your PDF will still include the whole image file at full size — all that changes is the way it shows it. This is very inefficient: normally you should create images at the right size for the document.
4.4. IMAGES

If you specify both height and width, the image will be distorted to fit (not really useful except for special effects). You can scale an image by a factor (using the `scaled` option) instead of specifying height or width; clip it to specified coordinates; or rotate it in either direction. Multiple optional arguments are separated with commas.

For details of all the arguments, see the documentation on the `graphicx` package or a copy of the *Companion*. The package also includes commands to `rotate`, `mirror`, and `scale` text as well as images.

### 4.4.3 Making images

There are two types of image: bitmaps and vectors.

**Bitmaps**: Bitmap images are made of coloured dots, so if you enlarge them, they go jagged at the edges, and if you shrink them, they may go blurry. Bitmaps are fine for photographs, where every tiny dot is a different colour, and the eye won’t notice so long as you don’t shrink or enlarge too much. Bitmaps for diagrams and drawings, however, are almost always the wrong choice, and often disastrously bad.

**Vectors**: Vector drawings are made from instructions, just like TeX is, but using a different language (eg ‘draw this from here to here, using a line this thick’). They can be enlarged or reduced as much as you like, and never lose accuracy, because they get redrawn.
automatically at any size. You can’t do photographs as vectors, but vectors are the only acceptable method for drawings or diagrams.

Figure 4.3: The vector diagram from Figure 4.2 on page 90 shrunk and enlarged.

Vector graphic packages are also better suited for saving your image directly in EPS or PDF format (both of which use their own vector language internally). All the major graphics-generating packages in all disciplines output vector formats: AutoCAD, ChemDraw, MathCAD,
Maple, Mathematica, ArcInfo, and so on. EPS was for decades the universally-accepted format for creating vector graphics for publication, with PDF a close second. PDF is now the most common format, but most of the major graphics (drawing) packages can still save as EPS, such as PhotoShop, PaintShop Pro, Adobe Illustrator, Corel Draw, and GIMP. There are also some free vector plotting and diagramming packages available like InkScape, tkPaint, and GNUplot which do the same. Never, ever (except in the direst necessity) create any diagram as a bitmap.

Bitmap formats like JPG and PNG are ideal for photographs, as they are also able to compress the data substantially without too much loss of quality. However, compressed formats are bad for screenshots, if you are documenting computer tasks, because too much compression makes them blurry. The popular Graphics Interchange Format (GIF) is good for screenshots, but is not supported by \TeX: use PNG instead, with the compression turned down to minimum. Avoid uncompressible formats like BMP as they produce enormous and unmanageable files. The Tagged Image File Format (TIFF), popular with graphic designers, should also be avoided if possible, partly because it is even vaster than BMP, and partly because far too many companies have designed and implemented non-standard, conflicting, proprietary extensions to the format, making it virtually useless for transfer between different types of computers (except in faxes, where it’s still used in a much stricter version).

4.4.4 Graphics storage

I mentioned earlier that there was a way to tell \LaTeX where to look if you had stored images centrally for use in many different documents. The answer is in a command \texttt{\graphicspath} which you supply with an argument giving one or more names of additional directories you want searched when a file uses the \texttt{\includegraphics} command.

Put the path in an additional set of curly braces (this lets you add more paths later: each in their own subset of curly braces). I’ve used the ‘safe’ (MS-DOS) form of the Windows My Pictures folder in the example because you should never use directory or file names containing spaces (see the panel ‘Picking suitable filenames’ on p. 39).
Exercise 14. Adding pictures

Add `\usepackage{graphicx}` to the Preamble of your document, and copy or download an image you want to include. Make sure it is a JPG, PNG, or PDF image if you use X\LaTeX{} or PDF\LaTeX{}; or an EPS image if you use standard \LaTeX{}.

Add `\includegraphics` and the filename in curly braces (without the filetype), and process the document and preview or print it.

Make it into a figure following the example in section 4.3 on page 89.

Be aware that some DVI previewers are not able to display all types of graphics, and some cannot display colour. For best results, use PDF or PostScript previewers.

Using `\graphicspath` does make your file less portable, though, because file paths tend to be specific both to an operating system and to your computer, like the examples above.

```latex
\graphicspath{{c:/mypict-1/camera}{z:/corp/imagelib}}
\graphicspath{{/var/lib/images}{/home/peter/Pictures}}
```

If you use original \LaTeX{} and \texttt{dvips} to print or create PostScript files, be aware that some versions will not by default handle EPS files which are outside the current directory, and will issue the error message saying that it is ‘unable to find’ the image. As we mentioned above, this is because PostScript is a programming language, and it would theoretically be possible for a maliciously-made image to contain code which might compromise your system. The decision to restrict operation in this way has been widely criticised, but it seems unlikely to be changed. If you are certain that your EPS files are kosher, use the \texttt{R0} option in your command, eg `\$dvips -R0 ... dvifile`

4.5 Quotations

Direct speech and short quotes within a sentence ‘like this’ are done with simple quotation marks as described in section 1.7 on page 17. Sometimes, however, you may want longer quotations set as a separate
4.5. QUOTATIONS

paragraph. Typically these are indented from the surrounding text. \LaTeX{} has two environments for doing this.

Such quotations are often set in a smaller size of type, although this is not the default, but you can use one of the size commands like \texttt{\small} (see section 6.2.7 on page 151) as shown in the example.

\begin{quote}{\small
noindent
At the turn of the century William Davy, a Devonshire parson, finding errors in the first edition of his \citetitle{davy}, asked for a new edition to be printed. His publisher refused and Davy purchased a press, type, and paper. He harnessed his gardener to the press and apprenticed his housemaid to the typesetting. After twelve years work, a new edition of fourteen sets of twenty-six volumes was issued—which surely indicates that, when typomania is coupled with religious fervour, anything up to a miracle may be achieved.\hfill\textcite[p.76]{ryder}
\end{quote}

\begin{quote}
At the turn of the century William Davy, a Devonshire parson, finding errors in the first edition of his \textit{A System of Divinity}, asked for a new edition to be printed. His publisher refused and Davy purchased a press, type, and paper. He harnessed his gardener to the press and apprenticed his housemaid to the typesetting. After twelve years work, a new edition of fourteen sets of twenty-six volumes was issued—which surely indicates that, when typomania is coupled with religious fervour, anything up to a miracle may be achieved.

(Ryder, 1976, p. 76)
\end{quote}

The inclusion of a bibliographic citation at the end is optional but commonplace, especially in academic or research documents where it may be compulsory because of the need to cite everything you quote. It’s also possible for this to be tucked into the space at the end of the last line of the quotation, if there is room (if it’s too long, it obviously has to go on a line by itself).

The \texttt{quote}{} environment sets the whole block of text indented, and each paragraph of it also has its own indentation on the first line, even the first paragraph. This is rather unconventional as a default, so it is common to add a \texttt{noindent} command at the start.
of the quotation so that the first paragraph does not get indented (others still will).

4.6 Boxes, sidebars, and panels

\LaTeX, like most typesetting systems, works by setting text into boxes. Each character is a box, with a height and a width, just like it is in metal type; characters are assembled into lines, which are also boxes; and lines are assembled into pages, which are also boxes. The page-making mechanism also works like an old compositor’s galley (tray) from the days of metal type: the box accumulates lines of typeset text until it’s a bit longer than the height of the page. \TeX then works out how much of it really will fit on the page, cuts it off and ships it out to the DVI or PDF file, and puts the remainder back into the galley (box) at the top, ready to start accumulating more material for the following page.

4.6.1 Boxes of text

Because of this ‘box model’, \LaTeX can typeset any text into a box of any width. The simplest command for small amounts of text is \texttt{\parbox}. This command needs two arguments in curly braces: the first is the width you want the text set to, and the second is the text itself, as in the example shown.

\begin{quote}
\parbox{3in}{Please make sure you send in your completed forms by January 1st next year, or the penalty clause 2(a) will apply.}
\end{quote}

The text is typeset to the required width, and the box is extended downwards for as long as is required to fit the text. Note that the baseline of a \texttt{\parbox} is set to the midpoint of the box, so if you include a \texttt{\parbox} in mid-sentence, the centre of the box will be lined up with the line of type currently being set. You can specify that the
top or bottom should align differently with respect to any surrounding text instead by adding an optional \texttt{t} (top) or \texttt{b} (bottom) in square brackets before the width. For example, \texttt{parbox[t]{3in}{...}} will produce a box with the baseline aligned with the top line of the text in the box.

Where the contents is more complex, use the \texttt{minipage} environment.

\begin{minipage}{3in}
Please make sure you send in your completed forms by January 1st next year, or the penalty clause 2(a) will apply:
\begin{itemize}[noitemsep]
\item Incomplete forms will be returned to you unprocessed.
\item Forms must be accompanied by the correct fee.
\item There is no appeal. The adjudicators’ decision is final.
\end{itemize}
\end{minipage}

Notice that when setting very narrow measures with type that is too large, the words will not fit nicely and the spacing may become uneven or there may be too much hyphenation. Either use \texttt{raggedright} or reduce the type size, or (in extreme cases) reword the text or break each line by hand. Fortunately, it is rare for \LaTeX to need this level of attention.

Within a \texttt{minipage} you can use virtually everything that occurs in normal text (eg lists, paragraphs, tabulations, etc) with the exception of floats like Tables and Figures. The \texttt{minipage} environment takes a
compulsory argument just like \parbox does, and it means the same: the width you want the text set to.

Note that in both minipages and \parbox{}es, the paragraph indentation (\parindent) is reset to zero. If you need to change it, do so inside the minipage or \parbox{} using the \setlength command (see section 2.7 on page 49).

Because a minipage is typeset independently from the rest of your text, any footnotes inside a minipage will be typeset at the end of the minipage, not at the foot of the containing page. They will also be done using lowercase letters by default, to keep them separate from the normal footnotes. (We haven't done footnotes yet, but they're in the next chapter.

There are other ways of typesetting text to widths other than the normal text width: you can use a one-row, one-cell \texttt{tabular} environment with the \texttt{p} column type specification; or you can use the technique of setting the material into a special box that remembers it, and then emitting it where you want it (this is implemented by the standard \texttt{lrbox} environment or by the \texttt{Sbox} environment from the \texttt{fancybox} package, but these are advanced techniques).

### 4.6.2 Framed boxes

To put a frame round \texttt{some text}, use the \texttt{fbox} command:

\begin{quote}
\texttt{fbox}\{some text\}
\end{quote}

We already saw this used in the Quick Start document and also to frame an image in Figure 4.2 on page 90. For text, this works for a few words in mid-line, but the framed box and its contents won't break over the end of a line. To typeset multiline text in a box, put it in a \texttt{parbox}, or use a minipage or \texttt{tabular} environment as described above, and enclose the whole thing in a \texttt{fbox}.

\begin{quote}
\texttt{fbox}\{\texttt{parbox}\{3in\}\{Please make sure you send in your completed forms by January 1st next year, or the penalty clause 2(a) will apply.\}\}
\end{quote}

The spacing between text and box is controlled by the value of \texttt{fboxsep}, and the thickness of the line by \texttt{fboxrule}, both of which can be reset with the \texttt{setlength} command.
4.7 VERBATIM TEXT

If you are using colour, the xcolor package extends boxing to the \colorbox command, which takes two arguments: a colour name or code for the background colour, and the text (which will need a foreground colour if black would not be suitable):

\begin{verbatim}
\colorbox{green}{\textcolor{white}{some text}}
\end{verbatim}

The package also provides \fcolorbox which puts a frame around a coloured box; in this case the first argument is the frame colour, the second the background colour, and the third the contents.

4.6.3 Sidebars and panels

The fancybox package lets you extend the principle of \fbox with commands to surround text in square, oval (round-cornered), and drop-shadow boxes (eg \ovalbox, \shadowbox, etc: see the documentation for details).

You can create panels of any size with these borders by using the minipage environment to typeset the text inside a special Sbox environment which fancybox defines. The minipage formats the text but the Sbox ‘captures’ it, allowing you to delay putting the frame around until it is complete.

The printed version of this document uses this extensively and there is a worked example shown in section 7.5 on page 168.

4.7 Verbatim text

If you are documenting computer procedures, you probably need fixed-width type for examples of programming or data input or output. Even if you are writing about completely non-computer topics, you may often want to quote a URI, filename, or email address which needs to be typeset specially.

\TeX includes two features for handling fixed-format text: inline verbatim and display verbatim. There are many more variations available in other packages.

4.7.1 Inline verbatim

To specify a word or phrase as verbatim text in typewriter type within a sentence, use the special command \verb, followed by the word or
phrase surrounded by any suitable character which does not occur in
the word or phrase itself. This is a very rare exception to the rule that
arguments go in curly braces.

For example, you could use the plus sign to show a \LaTeX command in
a manual like this one:

\begin{verbatim}
You can typeset a phrase verbatim, even if it includes \LaTeX command characters, for example the command to
insert an image: \verb+\includegraphics[width=3in]{myhouse}+
\end{verbatim}

You can typeset a phrase verbatim, even if it includes \LaTeX command characters, for example the command to insert an
image: \includegraphics[width=3in]{myhouse}

The plus sign is ‘safe’ to use here because it doesn’t appear in
the code you want to typeset but you could use the grave accent or
backtick key ‘ or the vertical bar | if the phrase already had a
plus sign in it:

\begin{verbatim}
for example \verb|\(a=b+c\)| when illustrating the
\LaTeX equation \(a=b+c\).
\end{verbatim}

for example \verb|\(a=b+c\)| when illustrating the \LaTeX equation \(a=b+c\).

The \verb command has the advantage that it turns off all special
characters (see section 1.6 on page 15) except the one you use as
the delimiter, so you can easily quote sequences of characters in any
computer syntax — including \TeX. However, \LaTeX will never break the
argument of \verb at a line-end when formatting a paragraph, even
if it contains spaces, so if it happens to be long, and falls towards
the end of a line, it will stick out into the margin. See section 1.9.2
on page 26 for more information on line-ends and hyphenation. The
argument to \verb MUST NOT contain a linebreak in your editor: this
will cause it to fail. See also the warning about using \verb inside
\footnote.

4.7.1.1 Typesetting URIs

The \url package avoids this by performing a hyphenless break at
punctuation characters. It is particularly important in \textsc{uris} to avoid
adding a spurious hyphen if they have to break over a line-end, because a hyphen might be mistaken by the user as a part of the address. 4

**URIs** present another problem: it’s important for them to be visibly accurate, so they can be copied and retyped from print. It is therefore essential to use a typeface which distinguishes well between 1 (digit one), l (lowercase ell) and I (uppercase eye), and between 0 (zero) and O (uppercase oh). Monospaced ‘typewriter’ type usually makes this clear, but many sans-serif fonts do not. It is a common error by designers not to distinguish **URIs** in this way.

The `url` package provides the command \url which works in the same way as \verb, but uses the standard curly braces to enclose the address, eg \url{http://latex.silmaril.ie} — the command understands the syntax of a **URI** (Berners-Lee, Fielding & Masinter, 2005) and will never break mid-way through an unpunctuated word, only at slashes and full points (and never at embedded hyphens unless the **hyphen** package option is used). Bear in mind, however, that spaces and non-ASCII characters are (currently) forbidden in **URIs**, so using spaces in a \url argument will cause it to fail, as will using other non-URI-valid characters like accented letters.

### 4.7.1.2 Enhanced inline verbatim

The `listings` package, which we look at more below for display verbatim, also has an inline form. This can use colour to highlight your examples based on the language you are documenting — I am using it extensively in the PDF of this book.

The command \lstinline uses the same syntax as \verb (two matching but otherwise unused characters) to enclose the argument, but it provides for very extensive options to specify the language, font, size, style, and formatting. The most useful is the language, of which about 100 are predefined, from **ADA** to **Verilog**, and you can add new keywords and even whole new languages.

This is probably the most effective way to show computer-language examples inline, because it handles the syntax-based enhancement for you. It is, however, still subject to the same limitations as \verb, in

---

4 The original term Uniform Resource Locator (**URL**) is now deprecated in favour of the more accurate Uniform Resource Indicator (**URI**). For details see [www.w3.org/Addressing/](http://www.w3.org/Addressing/). Unfortunately the older term still persists, especially in this **LaTeX** package and its command, and in some **XML** markup vocabularies.
that the code must fit on the space available in the line, or it will stick out into the margin.

For example, you could use the plus sign to show a \LaTeX\ command: \verbatim{\includegraphics[width=3in]{myhouse}}+ in order to display \includegraphics[width=3in]{myhouse}, because the plus sign does not occur in the command, and is therefore free to be used.

For longer (multiline) chunks of fixed-format text like examples of programming, use the \texttt{verbatim} environment. Like \texttt{verb}, this turns off all special characters, so you can include anything at all in the verbatim text except the exact line \texttt{end{verbatim}}, which \textbf{MUST} occur on a line by itself.

\begin{verbatim}
\documentclass[11pt,a4paper,oneside]{report}
\begin{document}
\title{Practical Typesetting}
\author{Peter Flynn\ Silmaril Consultants}
\date{December 2004}
\maketitle
\end{document}
\end{verbatim}

For more control over formatting there are two useful packages: the \texttt{verbatim} package, which overcomes a few of the limitations of the built-in \texttt{verbatim} environment; and the \texttt{fancyvrb} package, which provides much greater flexibility with a \texttt{Verbatim} environment (note the capital letter).

However, as I mentioned above, for a much more powerful verbatim environment, I use the \texttt{listings} package for its ability to colour the
keywords of a program according to the language used. It can also add rules, interpret internal formatting, and include external files, and let you add your own language definitions for new languages. The penalty is a slightly more complex configuration, but if you are documenting any kind of computer code in significant quantities, the quality and usability of the result is well worth it.

Exercise 15. Try some fixed-format text

1. Add your email address and home page \verb|url| using the \verb and \url commands. You’ll need to \usepackage{url} for the latter.

2. Load the \texttt{listings} package and try the \lstinline command to do the same.