

Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

Introduction to L^AT_EX

Lecture IV: Graphs, Tables and Code

Liu Yihao

SJTU-UMJI Technology Department

June 22, 2021

Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

- 1 Graphs
 - Include Graphs
 - Figures
 - Draw Graphs

2 Tables

3 Code

Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

Include Graphs

Before all, you need the `graphics` or `graphicx` package, where `graphicx` is an extended and enhanced one. So you are recommended to insert the command in the preamble of your document.

Command

```
\usepackage{graphicx}
```

Then you can use the command `\includegraphics` to insert images of many formats, including `jpg`, `png` images and even other `pdf` files. `eps` images should be supported by most modern L^AT_EX distributions as well.

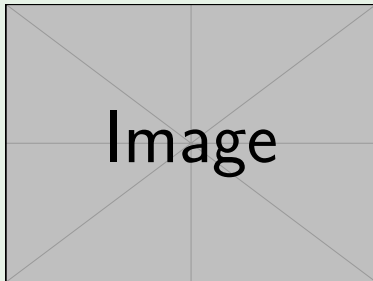
Command

```
\includegraphics[options]{filename}
```

There are some example images defined, you can insert them if the figure is not yet ready when writing L^AT_EX code. They are `example-image`, `example-image-golden`, `example-image-a`, `example-image-b` and etc.

Example

```
1 \includegraphics[width=0.4\textwidth]{example-image}
```



We usually use the `width` option to adjust the size of the image, according to a ratio of `\textwidth`, which means the maximum width of text here.

Options of Include Graphs

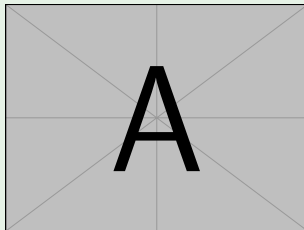
Here some useful **options** are listed:

- **height** - use any L^AT_EX measuring unit.
- **width** - use any L^AT_EX measuring unit.
- **scale** - scale the graph to this proportion
- **angle** - rotate the graph in anti-clockwise by this angle

L^AT_EX measuring unit can be `\textwidth`, `\linewidth`, `\textheight`, `\lineheight`, cm, pt, em, and etc..

Example

```
1 \includegraphics[width=4cm] %  
2 {example-image-a}
```



Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

- 1 Graphs
 - Include Graphs
 - **Figures**
 - Draw Graphs

2 Tables

3 Code

The `figure` Environment

The `figure` environment provides a wrapper of image inserted by `\includegraphics`, which add caption and label (reference) to an image. They are especially useful in report and paper writing, here is a template of how to use the environment.

Command

```
1 \begin{figure}[position]
2   \centering
3   \includegraphics[options]{filename}
4   \caption{caption}
5   \label{fig:label}
6 \end{figure}
```

- `filename` - the filename or relative path of the graph you want to insert, usually placed in the same or child directory as the tex file
- `position` - we usually use `!htbp` or `!H` here, which will be introduced later in this chapter
- `caption` - the caption displayed above/under the graph
- `label` - used for references in a document (will be introduced later)

Labels and References

You can use `\ref` to have a reference of a figure by its label. The figures will be automatically numbered (like equations), and the reference is also a hyperlink.

Example

```
1 \begin{figure}[!htbp]
2   \centering
3   \includegraphics[
4     width=0.8\textwidth,
5     angle=90
6   ]{example-image-b}
7   \caption{Example Image B rotated by 90
8     ↪ degree.}
9   \label{fig:img-b}
10 \end{figure}
11 B was shown in Figure
12 \ref{fig:img-b}.
```

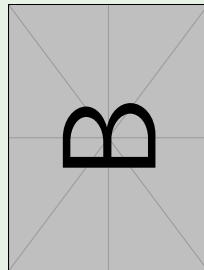


Figure 1: Example Image B rotated by 90 degree.

B was shown in Figure 1.

Floats and Positions

Floats are containers for things in a document that cannot be broken over a page. L^AT_EX by default recognizes `figure` and `table` (will be introduced later) floats.

If you don't provide the `position` option, L^AT_EX will try to help you find a place to set the figure. However, the position is often not ideal, so you need to add some specifiers yourselves.

- `h` - Place the float `here`, i.e., approximately at the same point it occurs in the source text (however, not exactly at the spot)
- `t` - Position at the `top` of the page.
- `b` - Position at the `bottom` of the page.
- `p` - Put on a special `page` for floats only.
- `!` - Override internal parameters L^AT_EX uses for determining “good” float positions.
- `H` - Places the float at precisely the location in the L^AT_EX code. Requires the float package, i.e., `\usepackage{float}`.

Include Multiple Graphs

Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

A useful extension is the `subcaption` package, which provides a `subfigure` environment to add multiple subfigures in a figure.

Note that there is also a package called `subfigure`, but it has been deprecated (not maintained), please do not use it. Another package called `subfig` provides the same commands as that of `subfigure` package. However, they can't be used together.

In simplicity, if there is some compatibility problem with your template after you tried the `subcaption` package, choose the `subfig` package.

Here is an example with the `subcaption` package.

Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

Example

```
1 \begin{figure}
2   \centering
3   \begin{subfigure}[0.3\textwidth]
4     \includegraphics[width=\textwidth]{example-image-a}
5     \caption{Example Image A.}
6     \label{fig:subcaption-a}
7   \end{subfigure}
8   ~
9   \begin{subfigure}[0.3\textwidth]
10    \includegraphics[width=\textwidth]{example-image-b}
11    \caption{Example Image B.}
12    \label{fig:subcaption-b}
13  \end{subfigure}
14
15  \begin{subfigure}[0.3\textwidth]
16    \includegraphics[width=\textwidth]{example-image-c}
17    \caption{Example Image C.}
18    \label{fig:subcaption-c}
19  \end{subfigure}
20  \caption{Example Images}\label{fig:subcaption}
21 \end{figure}
```

Graphs

Include Graphs

Figures

Draw Graphs

Tables

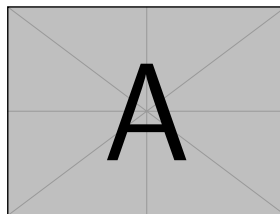
Tabulars

Tables

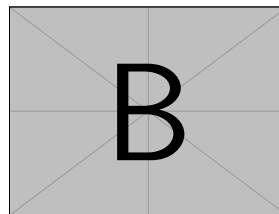
Code

Pseudo Code

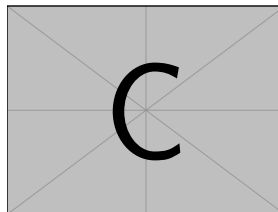
Code Listing



(a) Example Image A.



(b) Example Image B.



(c) Example Image C.

Figure 2: Example Images

Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

As shown in Figure 2, the figures can be arranged in columns and rows.

Between Figure 2a and Figure 2b, a `~` was added. You can add desired spacing between images, e. g. `~`, `\quad`, `\qquad`, `\hfill` (fill all rest horizontal spaces) and etc..

Between Figure 2b and Figure 2c, a newline was added. It will force the subfigure onto a new line.

The references of subfigures can be used by their `\label` as well. For example, above references are generated by these commands:

Example

```
1 \ref{fig:subcaption}  
2 \ref{fig:subcaption-a}  
3 \ref{fig:subcaption-b}  
4 \ref{fig:subcaption-c}
```

Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

- 1 Graphs
 - Include Graphs
 - Figures
 - Draw Graphs

2 Tables

3 Code

The `tikz` and `pgf` packages

The `tikz` and `pgf` packages can help you draw graphs in L^AT_EX for example:

Example

```
1 \begin{tikzpicture}[scale=2, bend angle=22.5]
2 \tikzstyle{every node}=[draw,shape=circle];
3 \foreach \i in {1,...,8}
4 {
5 \path (45*\i-45:1cm) node (v\i) {$v\_i$};
6 }
7 \draw
8 (v1) -- (v2) (v3) -- (v4) (v5) -- (v6) (v7) -- (v8)
9 (v1) -- (v3) (v3) -- (v5) (v5) -- (v7) (v7) -- (v1)
10 (v2) -- (v5) (v4) -- (v7) (v6) -- (v1) (v8) -- (v3)
11 (v1) -- (v5) (v3) -- (v7);
12 \end{tikzpicture}
```

Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

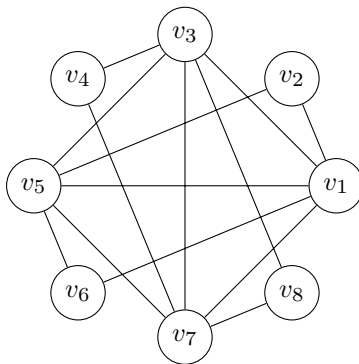
Tables

Code

Pseudo Code

Code Listing

This will generate a simple graph which consists of eight nodes:



There may be a lecture about [tikz](#) and [pgf](#) in the future. If you are now interested in it, please refer to the [pgf manual](#) by `texdoc tikz` or `texdoc pgf`.

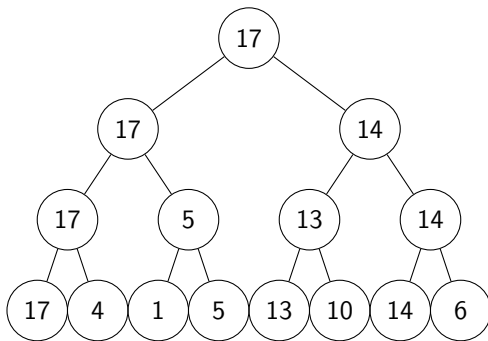
Another example:

Example

```
1 \begin{tikzpicture}[scale=0.8]
2 \tikzstyle{every node}=[draw,shape=circle,minimum size=0.8cm];
3 \node {17}[sibling distance=4cm]
4   child { node {17}[sibling distance=2cm]
5     child {
6       node {17}[sibling distance=1cm]
7       child { node {17} }
8       child { node {4} }
9     }
10    child {
11      node {5}[sibling distance=1cm]
12      child { node {1} }
13      child { node {5} }
14    }
15  }
16  child { node {14}[sibling distance=2cm]
17    child {
18      node {13}[sibling distance=1cm]
19      child { node {13} }
20      child { node {10} }
```

```
21     }  
22     child {  
23         node {14}[sibling distance=1cm]  
24         child { node {14} }  
25         child { node {6} }  
26     }  
27 };  
28 \end{tikzpicture}
```

This will generate a binary tree:



Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

1 Graphs

2 Tables

- Tabulars
- Tables

3 Code

The `tabular` Environment

Table is another common element in L^AT_EX, usually you will need the `array` package for enhanced functions of tables. You can insert the command in the preamble of your document.

Command

```
\usepackage{array}
```

Example

```
1 \begin{tabular}{|l|c|r|}
2 \hline
3 Title 1 & Title 2 & Title 3 \\
4 \hline
5 1 & 2 & 3 \\
6 \hline
7 \end{tabular}
```

Title 1	Title 2	Title 3
1	2	3

The syntax is similar to the `align` environment in maths. `&` is used to split the columns are `\\` is used to split the rows.

Column Format

Command

```
1 \begin{tabular}{format}  
2 ...  
3 \end{tabular}
```

`format` can be set as follow:

- `|` - represents a vertical separate line between two columns
- `l` - align left in this column
- `c` - align center in this column
- `r` - align right in this column

Example

`|l|l|l|`

Title 1	Title 2	Title 3
1	2	3

`||c|cc||`

Title 1	Title 2	Title 3
1	2	3

Graphs

Include Graphs
Figures
Draw Graphs

Tables

Tabulars
Tables

Code

Pseudo Code
Code Listing

With the help of the `array` package, more formats are available:

- `p{width}` - Equivalent to `\parbox[t]{width}`, vertically aligned **bottom**
- `b{width}` - Equivalent to `\parbox[b]{width}`, vertically aligned **top**
- `m{width}` - Equivalent to `\parbox{width}`, vertically aligned middle
- `>{decl.}` - Can be used before a letter option, inserts `decl` before the column.
- `<{decl.}` - Can be used after a letter option, inserts `decl` after the column.

`t` and `b` may be very confusing, but that's how they work in `\parbox`. With these new formats, the columns can be defined more flexibly.

Example

```
1 \begin{tabular}  
2 { |p{1.2cm}|b{1.2cm}|m{1.2cm}| }  
3 \hline  
4 Aligned Bottom & Aligned Top &  
5 Aligned Middle \\  
6 \hline  
7 1 & 2 & 3 \\  
8 \hline  
9 \end{tabular}
```

Aligned Bottom	Aligned Top	Aligned Middle
1	2	3

`t`, `b` and `m` only affect the vertical alignment. If you want to control the width and make the text horizontally centered as well, you can use `>\centering` to insert a `\centering` before the text in that column. You can also insert `>{$}` and `<{$}` to generate a column in math mode.

Example

```
1 \begin{tabular}{|>\centering m{2cm}|>{$} b{2cm}<{$}|}  
2 \hline  
3 Row of Text &  
4 \text{Row of Maths} \\\br/>5 \hline  
6 First & x \\\br/>7 Second & x^2 \\\br/>8 \hline  
9 \end{tabular}
```

Row of Text	Row of Maths
First	x
Second	x^2

If a column type will be used many times, and also very long, you can define a new column type by yourselves. You can use

Command

```
\newcolumntype{new type}{>{some declarations}{old type}<{some more declarations}}
```

If you want to repeat a format for multiple times, you can use `*{num}{format}`. Here's an example of the usage of `\newcolumntype` with multiple columns form.

Example

```
1 \newcolumntype{C}{>{$}c<{$}}
2 \newcolumntype{L}{>{$}l<{$}}
3 \newcolumntype{R}{>{$}r<{$}}
4
5 \begin{tabular}{|L| *{2}{C|} R|}
6 \hline
7 \text{First} & \text{Second} &
8 \text{Second} & \text{Third} \\
9 \hline
10 x & x^2 & x^2 & x^3 \\
11 \hline
12 y & y^2 & y^2 & y^3 \\
13 \hline
14 \end{tabular}
```

First	Second	Second	Third
x	x^2	x^2	x^3
y	y^2	y^2	y^3

Horizontal Lines

We usually need horizontal lines in tables. As shown in the examples above, you can add a `\hline` at the beginning of a row.

If you only want to draw a partial line, use `\cline[start-end]`.

Example

```
1 \begin{tabular}{c|l|c|r}
2   \hline\hline
3   & Title 1 & Title 2 & Title 3 \\
4   \cline{2-4}
5   Table & 1 & 2 & 3 \\
6   \cline{2-4}
7   & 4 & 5 & 6 \\
8   \hline\hline
9 \end{tabular}
```

Table	Title 1	Title 2	Title 3
	1	2	3
	4	5	6

Here we draw a table with a multirow, but it only works with multirows of odd row number. A more convenient method of drawing multirows will be introduced.

Combining Rows and Columns

There are two commands being used to combine rows and columns

Command

```
\multicolumn{ncols}{format}{text}
```

- `ncols` - the number of columns to be merged
- `format` - the format of the merged column, excluding the left | (eg. `c|`)
- `text` - the text in the merged column

```
\multirow{nrows}{width}[fixup]{text}
```

- `nrows` - the number of rows to be merged
- `width` - the width of the merged rows (use `*` for auto)
- `fixup` - the vertical position of the text (optional, default in the center)
- `text` - the text in the merged row

To use the `\multirow` command, you need to insert the package `multirow` in the preamble of your document.

Example

```
1 \centering
2 \begin{tabular}{|c|c|c|c|c|}
3 \hline
4 \multirow{4}{*}{Table} & Title 1 & Title 2 & Title 3 & Title 4 \\
5 \cline{2-5}
6 & \multicolumn{2}{c|}{Text 1} & & \\
7 \multicolumn{2}{c|}{\multirow{3}{*}{Text 3}} & & & \\
8 \cline{2-3}
9 & \multicolumn{2}{c|}{Text 2} & \multicolumn{2}{c|}{} \\
10 \cline{2-3}
11 & Text 4 & Text 5 & \multicolumn{2}{c|}{} \\
12 \hline
13 \end{tabular}
```

Table	Title 1	Title 2	Title 3	Title 4
	Text 1		Text 3	
	Text 2			
	Text 4	Text 5		

Just leave blank in the rest rows of `\multirow`.

Coloring Rows and Columns

The rows and columns can also be colored, with the `colortbl` package. You may also need the `xcolor` package to define new colors.

Command

```
1 \usepackage{xcolor}
2 \usepackage{colortbl}
```

Some commands are provided by these packages

Command

```
1 \definecolor{name}{system}{definition}
2 \rowcolor{color}
3 \columncolor{color}
```

Here `system` can be `rgb/hsb/cmyk/gray`. Please refer the lecture about defining colors.

Example

```
1 \definecolor{mygray}{gray}{.9}
2 \definecolor{barblue}{RGB}{153,204,254}
3 \centering
4 \begin{tabular}{|*2{>{\columncolor{mygray}}c|c|}{}}
5 \hline\rowcolor{barblue}
6 Title 1 & Title 2 & Title 3 & Title 4 \\
7 \hline
8 Text 1 & Text 2 & Text 3 & Text 4 \\
9 \hline
10 Text 5 & Text 6 & Text 7 & Text 8 \\
11 \hline
12 \end{tabular}
```

Title 1	Title 2	Title 3	Title 4
Text 1	Text 2	Text 3	Text 4
Text 5	Text 6	Text 7	Text 8

Note that the `\rowcolor` overwrites the `\columncolor`.

Styling Rows and Columns

Styling columns can be easily achieved by prepending styles in the `>\decl.` introduced before.

However, styling rows is much more complicated. You should only consider using this method when your table is really too large to style them one by one.

First, you may define these in the preamble of your document:

Command

```
1 \newcolumntype{+}{>\global\let\currentrowstyle\relax}}
2 \newcolumntype{^}{>\currentrowstyle}}
3 \newcommand{\rowstyle}[1]{\gdef\currentrowstyle{#1}%
4   #1\ignorespaces
5 }
```

Then you should add a `+` before the first column definition and a `^` before any other column definitions. (You can change the symbols `+` and `^` in the definition above.)

Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

Example

```
1 \newcolumntype{+}{>{\global\let\currentrowstyle\relax}}
2 \newcolumntype{^}{>{\currentrowstyle}}
3 \newcommand{\rowstyle}[1]{\gdef\currentrowstyle{#1}%
4   #1\ignorespaces
5 }
6 \centering
7 \begin{tabular}{|+>{\ttfamily}c|^c|^>{\ttfamily}c|^c|}
8   \hline\rowstyle{\bfseries\sffamily}
9   Title 1 & Title 2 & Title 3 & Title 4 \\
10  \hline
11  Text 1 & Text 2 & Text 3 & Text 4 \\
12  \hline
13  Text 5 & Text 6 & Text 7 & Text 8 \\
14  \hline
15 \end{tabular}
```

Title 1	Title 2	Title 3	Title 4
Text 1	Text 2	Text 3	Text 4
Text 5	Text 6	Text 7	Text 8

Note that the `\rowstyle` also overwrites the column style set in the formats.

Table Generators

Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

With `\multirow` and `\multicolumn`, we can almost draw tables of any style, but this coding process can never be as easy as the graphic one, like making tables in Word or Excel. Is there any ways to convert graphic tables into L^AT_EX codes directly?

- Use L^AT_EX Table Generator: <http://www.tablesgenerator.com/>
- L^AT_EX Complex Table Editor: <https://www.latex-tables.com/>
- Excel2latex: <https://ctan.org/tex-archive/support/excel2latex/>

Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

1 Graphs

2 Tables

- Tabulars

- **Tables**

3 Code

The `table` Environment

The `table` environment is used to arrange the place of a tabular, similar to the `figure` environment. Here is a template of how to use the environment.

Command

```
1 \begin{table}[position]
2   \centering
3   \begin{tabular}{format}
4     ...
5   \end{tabular}
6   \caption{caption}
7   \label{table:label}
8 \end{table}
```

The `position`, `caption`, `label` are same as those in the `figure` environment.

Recall the Positions

Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

We usually want to place the graphs or tables just below or above the content where we mention them, but even when we type `[h]` in position, you can not ensure that it will appear at the ideal position, and there are several methods to make up for this. You can try them one by one:

- 1 Change `[h]` to `[!h]`
- 2 Change `[!h]` to `[!H]`
- 3 Use `\newpage` to move the following content to the next page

Usually you don't need to pay too much attention about where the figures and tables are exactly are because you can use `\ref` to reference them. And the numbering of figures and tables will strictly follow the order of their code.

figure and table in Two-column Documents

If you are writing a document using two columns (i.e. you started your document with something like `\documentclass[twocolumn]{article}`), you might have noticed that you can't use floating elements that are wider than the width of a column (using a L^AT_EX notation, wider than `0.5\textwidth`), otherwise you will see the figure or table overlapping with text.

If you really have to use such wide elements, the only solution is to use the “starred” variants of the floating environments:

Command

```
1 \begin{figure*}[position]
2   ...
3 \end{figure*}

1 \begin{table*}[position]
2   ...
3 \end{table*}
```

Those “starred” versions work like the standard ones, but they will be as wide as the page, so you will get no overlapping.

The `array` Environment

When you use `tabular` in maths environment, the text format in the `tabular` won't be italic. However, there is a replacement of `tabular`, which is the `array` environment.

Command

```
1 \begin{array}{format}  
2 ...  
3 \end{array}
```

The options and usages of these two environment are exactly the same.

Though the environment is not provided by the `array` package (it's built-in one), you are also recommended to use this package for enhancements.

List of Floats

The figures and tables are all called floats. Captions can be listed at the beginning of a paper or report in a “List of Figures” or a “List of Tables” section with the commands:

Command

- 1 `\listoffigures`
- 2 `\listoftables`

The caption used for each figure will appear in these lists, along with the figure numbers, and page numbers that they appear on.

The `\caption` command also has an optional parameter, which is used for the List of Tables or List of Figures.

Command

`\caption[short]{long}`

Typically the `short` description is for the caption listing, and the `long` description will be placed beside the figure or table.

Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

1 Graphs

2 Tables

3 Code

- Pseudo Code

- Code Listing

The `algorithm` Environment

L^AT_EX has several packages for typesetting algorithms in form of “pseudocode”. They provide stylistic enhancements over a uniform style (i.e., all in typewriter font) so that constructs such as loops or conditionals are visually separated from other text. The pseudocode is usually put in an `algorithm` environment. Include it by adding the command to your document’s preamble.

Command

```
\usepackage{algorithm}
```

Then you can use the `algorithm` environment, which acts similar as the `figure` and `table` environments.

Command

```
1 \begin{algorithm}[position]
2   \caption{caption}
3   \label{algorithm:label}
4   <the actual pseudocode environment>
5 \end{algorithm}
```


The `algorithmic` Package

One of the packages, the `algorithmic`, defines the `algorithmic` environment. Include it by adding the command to your document's preamble.

Command

```
\usepackage{algorithmic}
```

The basic commands are:

Command

```

1  \STATE <text>
2  \IF{<condition>} \STATE {<text>} \ELSE \STATE{<text>} \ENDIF
3  \IF{<condition>} \STATE {<text>} \ELSIF{<condition>} \STATE{<text>} \ENDIF
4  \FOR{<condition>} \STATE {<text>} \ENDFOR
5  \FOR{<condition>} \TO <condition> } \STATE {<text>} \ENDFOR
6  \FORALL{<condition>} \STATE{<text>} \ENDFOR
7  \WHILE{<condition>} \STATE{<text>} \ENDWHILE
8  \REPEAT \STATE{<text>} \UNTIL{<condition>}
9  \LOOP \STATE{<text>} \ENDLOOP
10 \REQUIRE <text>, \ENSURE <text>, \RETURN <text>, \PRINT <text>
11 \AND, \OR, \XOR, \NOT, \TO, \TRUE, \FALSE, \COMMENT{<text>}

```

Example

```

1  \begin{algorithm}[H]
2    \caption{Calculate  $y = x^n$ }
3    \label{algorithm:n-square}
4    \begin{algorithmic}
5      \REQUIRE  $n \geq 0 \vee x \neq 0$ 
6      \ENSURE  $y = x^n$ 
7      \STATE  $y \leftarrow 1$ 
8      \IF{ $n < 0$ }
9        \STATE  $X \leftarrow 1 / x$ 
10       \STATE  $N \leftarrow -n$ 
11     \ELSE
12       \STATE  $X \leftarrow x$ 
13       \STATE  $N \leftarrow n$ 
14     \ENDIF
15     \WHILE{ $N \neq 0$ }
16       \IF{ $N$  is even}
17         \STATE  $X \leftarrow X \times X$ 
18         \STATE  $N \leftarrow N / 2$ 
19       \ELSE[ $N$  is odd]
20         \STATE  $y \leftarrow y \times X$ 
21         \STATE  $N \leftarrow N - 1$ 
22       \ENDIF
23     \ENDWHILE
24   \end{algorithmic}
25 \end{algorithm}

```

Algorithm 1 Calculate $y = x^n$

Require: $n \geq 0 \vee x \neq 0$ **Ensure:** $y = x^n$ $y \leftarrow 1$ **if** $n < 0$ **then** $X \leftarrow 1/x$ $N \leftarrow -n$ **else** $X \leftarrow x$ $N \leftarrow n$ **end if****while** $N \neq 0$ **do****if** N is even **then** $X \leftarrow X \times X$ $N \leftarrow N/2$ **else**[N is odd] $y \leftarrow y \times X$ $N \leftarrow N - 1$ **end if****end while**

The `algorithmicx` Package

Another package `algorithmicx` provides more functionalities, but it is not compatible with the `algorithmic` package. Include it by adding the command to your document's preamble.

Command

```
\usepackage{algpseudocode}
```

Note that `\usepackage{algorithmicx}` only defines some common macros and it is not enough. Don't insert `\usepackage{algorithmic}` in this situation.

The main difference of these two packages is that all of the command name are changed, so that only the first letter in a word is capital. For example, `\STATE` is changed to `\State` and `\ENDFOR` is changed to `\EndFor`.

The command `\begin{algorithmic}` can be given the optional argument of a positive integer, which if given will cause line numbering to occur at multiples of that integer. E.g. `\begin{algorithmic}[5]` will enter the `algorithmic` environment and number every fifth line.

Example

```
1 \begin{algorithm}[H]
2   \caption{Euclids algorithm}
3   \label{algorithm:euclid}
4   \begin{algorithmic}[1]
5     \Procedure{Euclid}{$a,b$}\Comment{The g.c.d. of a and b}
6       \State $r\gets a\bmod b$
7       \While{$r\neq 0$}\Comment{We have the answer if r is 0}
8         \State $a\gets b$
9         \State $b\gets r$
10        \State $r\gets a\bmod b$
11      \EndWhile\label{euclidendwhile}
12      \State \textbf{return} $b$\Comment{The gcd is b}
13    \EndProcedure
14  \end{algorithmic}
15 \end{algorithm}
```

Algorithm 2 Euclids algorithm

1: **procedure** EUCLID(a, b)▷ The g.c.d. of a and b 2: $r \leftarrow a \bmod b$ 3: **while** $r \neq 0$ **do**▷ We have the answer if r is 04: $a \leftarrow b$ 5: $b \leftarrow r$ 6: $r \leftarrow a \bmod b$ 7: **end while**8: **return** b ▷ The gcd is b 9: **end procedure**

Algorithms can also be listed like figures and tables, by the command:

Command`\listofalgorithms`

Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

1 Graphs

2 Tables

3 Code

- Pseudo Code

- Code Listing

The `verbatim` Environment

The default tool to display code in L^AT_EX is `verbatim`, which generates an output in monospaced font.

Example

```

1  \begin{verbatim}
2  Text enclosed inside \texttt{verbatim} environment
3  is printed directly
4  and all \LaTeX{} commands are ignored.
5  \end{verbatim}

```

Text enclosed inside `\texttt{verbatim}` environment
is printed directly
and all `\LaTeX{}` commands are ignored.

There's a starred version (`verbatim*`) whose output is slightly different.

Text enclosed inside `\texttt{verbatim}` environment
is printed directly
and all `\LaTeX{}` commands are ignored.

The `\verb` Command

Verbatim-like text can also be used inline with the command `\verb`

Example

```
1 In the directory \verb|C:\Windows\system32| you can find a lot of Windows
2 system applications.
3
4 The \verb+\ldots+ command produces \ldots
```

In the directory C:\Windows\system32 you can find a lot of Windows system applications.

The `\ldots` command produces ...

The command `\verb|C:\Windows\system32|` prints the text inside the delimiters `|` in verbatim format. Any character, except letters and `*`, can be used as delimiter. For instance `\verb+\ldots+` uses `+` as delimiter.

The listings Package

A better form of code listing can be done by the `listings` package. To use it, simply insert the command in the preamble of your document.

Command

```
\usepackage{listings}
```

It provides a `lstlisting` environment.

Command

```
1 \begin{lstlisting}[language=name]
2   ...
3 \end{lstlisting}
```

You can also input source code from file.

Command

```
\lstinputlisting[language=name]{filename}
```

Example

```
1 \begin{lstlisting}[language=Python]
2 import numpy as np
3
4 def incmatrix(genl1,genl2):
5     m = len(genl1)
6     n = len(genl2)
7 \end{lstlisting}
```

```
import numpy as np
```

```
def incmatrix(genl1 , genl2 ):
    m = len(genl1)
    n = len(genl2)
```

You can add code coloring and styling by some complicated configurations, see the Overleaf tutorial [▶ Link](#).

The documentation of the `listings` package can be found in [▶ Link](#).

The `minted` Package

All of the code in this lecture are highlighted by the `minted` package. To use it, simply insert the command in the preamble of your document.

Command

```
\usepackage{minted}
```

This is a very special package, it depends a program out of L^AT_EX called `pygmentize`, which is a code highlighting package written in `Python`.

You can install the package through `pip` (assuming you have `Python` 2 or 3 and `pip` installed) in your terminal:

Command

```
pip install Pygments
```

And then you can examine in your terminal whether `pygmentize` is your `PATH` by directly running it. You also need to add an option `-shell-escape` to your L^AT_EX compiler because L^AT_EX need this permission to run other programs on shell.

The `minted` Environment

You can use the `minted` environment to insert a block of code in the specific language.

Command

```
1 \begin{minted}[options]{language}
2   ...
3 \end{minted}
```

You can use the command in the terminal to find the supported languages.

Command

```
pygmentize -L lexers
```

There is also a list of languages on the online document [▶ Link](#). Note that if you want to insert plain text, use the `text` language which doesn't have any highlight.

Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

The Inline `minted`

For a single line of source code, you can alternatively use a shorthand notation:

Command

```
\mint[options]{language}|...|
```

Here we use a pair of `|`, same as the usage of the `\verb` command, which is also an inline verbatim command.

Or you can also use

Command

```
\mintinline[options]{language}|...|
```

Here `|` can also be replaced with `{}`, a pair of `+`, etc., the key is there should not exist the same delimiter inside the code.

Input File with `minted`

When you have a source code file alone, you can use the command to input the file.

Command

```
\inputminted[options]{language}{filename}
```

There are some commonly used options (not only for this command):

- `linenos` - Turn on line numbers
- `breaklines` - Automatically break long lines in `minted` environment and `\mint`, and wrap longer lines in `\mintinline`.
- `fontsize` - The size of the font to use, as a size command, e.g.
`fontsize=\footnotesize`.
- `tabsize` - The number of spaces a tab is equivalent to. (default is 8, but often set to 4)
- `firstline` - The first line to show. (default is 1, useful when showing part of a file)
- `lastline` - The last line to show. (default is the last line of the input)

Using Different Styles

You can use various styles of highlighting scheme provided by [pygmentize](#).

Command

```
\usemintedstyle{name}
```

You can use the command in the terminal to find the supported styles.

Command

```
pygmentize -L styles
```

There is also a demo of languages and styles on the online demo [▶ Link](#). The [autumn](#) style is used in this lecture.

In the end, [XeLaTeX](#) might be the best L^AT_EX compiler working with the [minted](#) package. It also supports typesetting with Chinese, if you meet problems when using the default [pdflatex](#) compiler, switch into [XeLaTeX](#) may solve your issues.

The documentation of the [minted](#) package can be found in [▶ Link](#).

Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

Custom Floats

If tables and figures are not adequate for your needs, then you always have the option to create your own! Examples of such instances could be source code examples, or maps. For a program float example, one might therefore wish to create a `float` named program. The package `float` is your friend for this task. All commands to set up the new float must be placed in the preamble, and not within the document.

Command

```
1 \usepackage{float}
2 \floatstyle{style}
3 \newfloat{type}{placement}{ext}[outer counter]
4 \floatname{type}{floatname}
```

The default name that appears at the start of the caption is the type. If you wish to alter this, use `\floatname{type}{floatname}`.

For the `\floatstyle` command, `style` can be:

- `plain` - the normal style for L^AT_EX floats, but the caption is always below the content.
- `plaintop` - the normal style for L^AT_EX floats, but the caption is always above the content.
- `boxed` - a box is drawn that surrounds the float, and the caption is printed below.
- `ruled` - the caption appears above the float, with rules immediately above and below. Then the float contents, followed by a final horizontal rule.

For the `\newfloat` command,

- `type` - the new name you wish to call your float, in the example, “program”.
- `placement` - t, b, p, or h (as previously described in Placement), where letters enumerate permitted placements.
- `ext` - the file name extension of an auxiliary file for the list of figures (or whatever). L^AT_EX writes the captions to this file.
- `outer counter` - the presence of this parameter indicates that the counter associated with this new float should depend on outer counter, for example “chapter”.

Graphs

Include Graphs

Figures

Draw Graphs

Tables

Tabulars

Tables

Code

Pseudo Code

Code Listing

Example

```
1 \documentclass{article}
2 \usepackage{float}
3 \floatstyle{ruled}
4 \newfloat{program}{thp}{lop}
5 \floatname{program}{Program}
6
7 \begin{document}
8
9 \begin{program}[H]
10 \begin{minted}{java}
11 class HelloWorldApp {
12     public static void main(String[] args) {
13         //Display the string
14         System.out.println("Hello World!");
15     }
16 }
17 \end{minted}
18 \caption{The Hello World! program in Java.}
19 \label{program:hello-world}
20 \end{program}
21
22 \end{document}
```

Program 1 The Hello World! program in Java.

```
1  class HelloWorldApp {
2      public static void main(String[] args) {
3          //Display the string
4          System.out.println("Hello World!");
5      }
6  }
```

You can also reference the custom floats.

Example

```
1  Program \ref{program:hello-world} is the Hello World! program in Java.
```

Program 1 is the Hello World! program in Java.