Advanced LaTeX

Rwitaban Goswami and Mihir Vahanwala

July 18, 2020

Contents

1   Introduction

Now that you get the hang of LaTeX more or less, you will be able to do fine and apply your LaTeX knowledge wherever required. But there still are some advanced things left to cover in LaTeX. This week we will be briefly going over the basics of some advanced LaTeX and how to proceed further.

2   Some other document classes

Till now you have only written in `\documentclass{article}` exclusively. But there is a whole world out there regarding other document types, and we will cover a few of those

2.1   Beamer

Ever wondered how the MA105 slides were made? Why do they all look so uniform (and bland)? They are made using the Beamer class. While they look bland, they are dead easy and fast to generate, and are easily customizable.

2.1.1   Basics

To use beamer you need, obviously, `\documentclass{beamer}` at the top of your preamble.
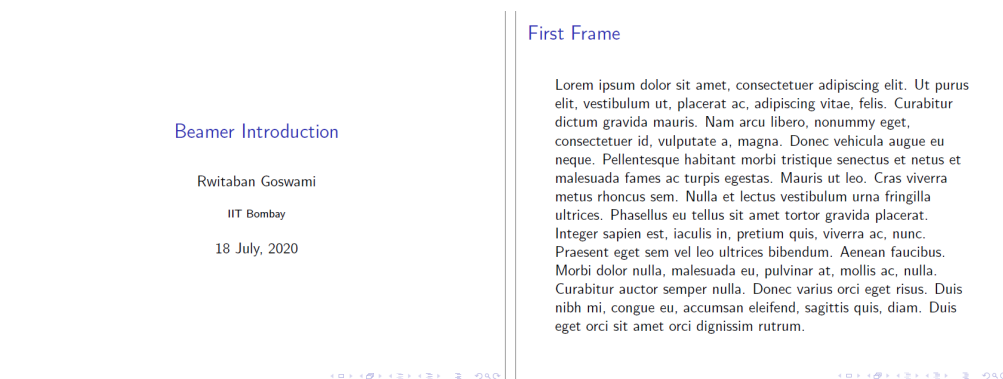
A minimal working example looks like this:

```
1  \documentclass{beamer}
2
3  \usepackage{lipsum}
4  \usepackage[utf8]{inputenc}
5
6
7  %Information to be included in the title page:
8  \title{Beamer Introduction}
9  \author{Rwitaban Goswami}
10 \institute{IIT Bombay}
11 \date{18 July, 2020}
12
```

```
13
14
15  \begin{document}
16
17  \frame{\titlepage}
18
19  \begin{frame}
20  \frametitle{First Frame}
21  \lipsum[1]
22  \end{frame}
23
24  \end{document}
```

This generates the following two slides:

(a) Slide 1

(b) Slide 2

Slides are analogous to frames in Beamer. To add new slides you need to add `\begin{frame}`, and give it an appropriate title with `\frametitle{}`

But it is not necessary that each frame will produce one slide. Each frame may produce multiple pages of pdf. For example, a frame with several bullet points can be set up to produce a new slide to reveal each consecutive bullet point.

2.1.2 Better titles

You can setup more comprehensive titles in Beamer.

```latex
1  \title[About Beamer]
2  {About the Beamer class in presentation making}
3
4  \subtitle{Beamer Intro}
5
6  \author[Rwitaban, Goswami]
7  {R.~Goswami\inst{1} \and M.~Vahanwala\inst{2}}
8
9  \institute[IITB]
10 {
11    \inst{1}%
12    Learner's Space instructor\\
13    IIT Bombay
14    \and
15    \inst{2}%
16    Learner's Space instructor\\
17    IIT Bombay
18 }
19
20 \date[LS\LaTeX{}]
21 {Learner's Space \LaTeX{}, 18 July 2020}
22
23 \logo{\includegraphics[height=1.5cm]{ugac.jpg}}
```

This generates the following title:

(a) Slide 1

2.1.3 TOC

You can even have a frame with a table of contents

```
1  \begin{frame}
2          \frametitle{Table of Contents}
3          \tableofcontents
4  \end{frame}
```
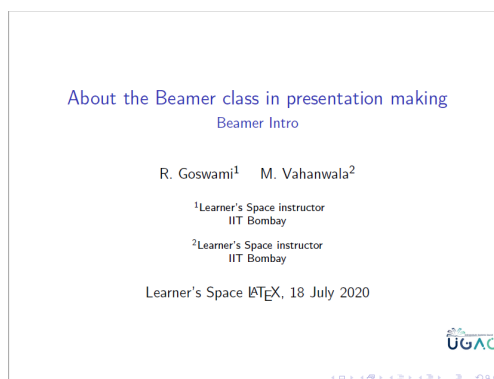
This generates: Keep in mind that frames are automatically not included in

(a) Slide 1

the TOC, the same rules as article class apply, and sections appear in the TOC

2.1.4 Effects

As we mentioned earlier, one frame may contribute to several slides. This is achieved through effects, or slideshows.
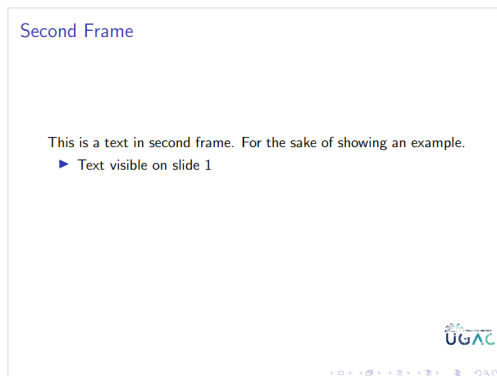
```
1  \begin{frame}
2          \frametitle{Second Frame}
3          This is a text in second frame. \pause
4          For the sake of showing an example.
5
6          \begin{itemize}
7                  \item<1-> Text visible on slide 1
8                  \item<2-> Text visible on slide 2
9                  \item<3> Text visible on slide 3
```

```
10                          \item<4–> Text  visible  on  slide  4
11               \end{itemize}
12
13  \end{frame}
```
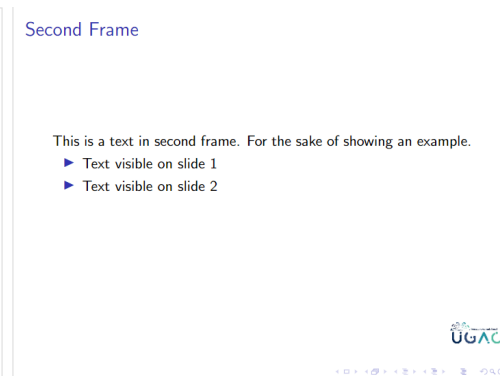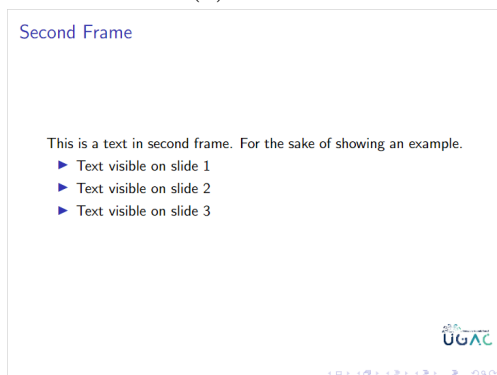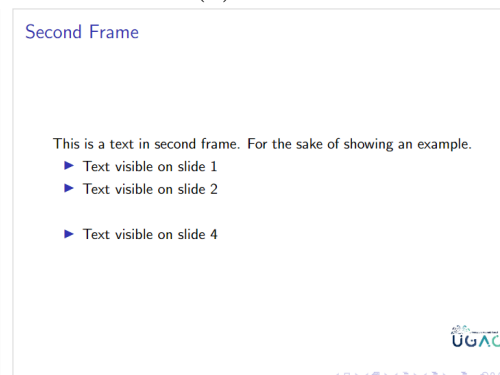
This single frame generates 4 slides:

(a) Slide 1

(b) Slide 2

(c) Slide 3

(d) Slide 4

As you can see, \item<1-> means it will be on every slide from 1 and till end of the frame. \item<3> will mean it will be visible only on slide 3.

You can also insert the \pause command in between text *outside* the itemize environment to generate a similar effect.

2.1.5   Themes

You can use premade themes for your slides. For example, just write `\usetheme{Madrid}` in your preamble to get this. Looks familiar?
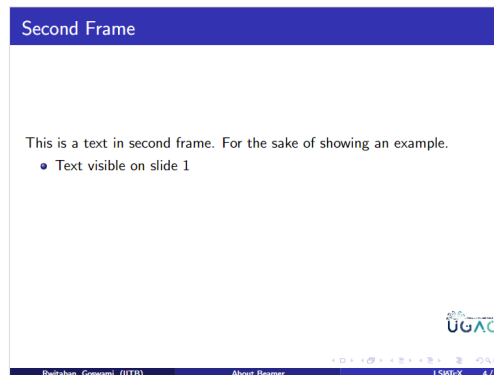
(a) Slide 1

2.1.6   Further

We cannot possibly even think of showing you all the commands there are in the beamer class, neither do we need to. You can use the references we have provided, search the internet, or look at the documentation to get the command and its syntax as you need.

Other classes which are extremely useful which you can read about are:

- PowerDot (For more powerful professional looking ppts)
- Poster (For making highly customizable posters)

3   Custom classes and packages

3.1   What are classes and packages really?

Till now we have assumed that whatever command we want to include, we can simply add `\usepackage{}` in your preamble and it will work like magic! What if we want to create our own? What if we have written a few very handy LaTeX macros, and we want to package them into a package, and we can `\usepackage{}` that wherever we want to?

In essence, the default formatting in LATEX documents is determined by the class used by that document. This default look can be changed and more functionalities can be added by means of a package. The class file names have the .cls extension, the package file names have the .sty extension.

3.1.1 Writing your own package

The first thing to do before coding a new package is to determine whether you really need a new package or not. It's recommended to search on CTAN (Comprehensive TEX Archive Network) `https://ctan.org/ctan-portal/search/` and see if someone already created something similar to what you need.

3.1.2 General Structure

Identification The file declares itself as a package written with the LATEX 2_ε syntax.

Preliminary declarations Here the external packages needed are imported. Also in this part of the file the commands and definitions needed by the declared options are coded.

Options The package declares and processes the options.

More declarations The main body of the package. Almost everything a package does is defined here.

3.1.3 Identification

There are two simple commands that all packages must have:

```
1  \NeedsTeXFormat{LaTeX2e}
2  \ProvidesPackage{examplepackage}[2014/08/24 Example
     LaTeX package]
```

The command `\NeedsTeXFormat{LaTeX2e}` sets the LATEX version for the package to work. Additionally, a date can be added within brackets to specify the minimal release date required.

The command `ProvidesPackage{examplepackage}[...]` identifies this package as examplepackage and, inside the brackets, the release date and some ad-

ditional information is included. The date should be in the form YYYY/MM/DD

3.1.4 Preliminary declarations

Most of the packages extend and customize existing ones, and also need some external packages to work. Below, some more code is added to the sample package "examplepackage.st".

```
1  \RequirePackage{imakeidx}
2  \RequirePackage{xstring}
3  \RequirePackage{xcolor}
4  \definecolor{greycolour}{HTML}{525252}
5  \definecolor{sharelatexcolour}{HTML}{882B21}
6  \definecolor{mybluecolour}{HTML}{394773}
7  \newcommand{\wordcolour}{greycolour}
```

The command `\RequirePackage` is very similar to the well-known `\usepackage`, adding optional parameters within brackets will also work. The only difference is that the `\usepackage` can not be used before `\documentclass` command. It's strongly recommended to use `\RequirePackage` when writing new packages or classes.

3.1.5 Options

To allow some flexibility in the packages a few additional options are very useful. The next part in the file "examplepackage.sty" handles the parameters passed to the package-importing statement.

```
1  \DeclareOption{red}{\renewcommand{\wordcolour}{
       sharelatexcolour}}
2  \DeclareOption{blue}{\renewcommand{\wordcolour}{
       mybluecolour}}
3  \DeclareOption*{\PackageWarning{examplepackage}{Unknown
       '\CurrentOption'}}
4  \ProcessOptions\relax
```

Below a description of the main commands that can handle the options passed to the package.

The command `\DeclareOption{}{}` handles a given option. It takes two parameters, the first one is the name of the option and the second one is the code to execute if the option is passed.

The command `\OptionNotUsed` will print a message in the compiler and the logs, the option won't be used.

The command `\Declareoption*{}` handles every option not explicitly defined. It takes only one parameter, the code to execute when an unknown option is passed. In this case it will print a warning by means of the next command:

`\PackageWarning{}{}`. This handles the errors which might occur in using the package. Lookup `https://www.overleaf.com/learn/latex/Writing_your_own_package` for a more detailed description on how to do that

`\CurrentOption` stores the name of the package option being handled at a determined moment.

The command `\ProcessOptions\relax` executes the code fore each option and must be inserted after all the option-handling commands were typed. There's a starred version of this command that will execute the options in the exact order specified by the calling commands.

In the example, if the options red or blue are passed to the `\usepackage` command within the document, the command `\wordcolor` is redefined. Both colours and the default grey colour were defined in the preliminary declarations after importing the xcolor package.

3.1.6   More declarations

In this part most of the commands will appear. In "examplepackage.sty".

```
1  %%Numbered environment
2  \newcounter{example}[section]
3  \newenvironment{example}[1][]{\refstepcounter{example}\
       par\medskip
4  \noindent  \textbf{My~environment~\theexample.  #1}  \
       rmfamily}{\medskip}
5
6  %%Important words are added to the index and printed in
        different colour
```

```
7   \newcommand{\important}[1]
8   {\IfSubStr{#1}{!}
9       {\textcolor{\wordcolour}{\textbf{\StrBefore
            {#1}{!}~\StrBehind{#1}{!}}}\index{#1}}
10      {\textcolor{\wordcolour}{\textbf{#1}}\index{#1}\
            kern−1pt}
11  }
```

This package defines the new environment example, and a new command
`\important`, that prints the words in a special colour and adds them to the
index.

3.1.7 Usage

To use the package, the sty file must be in the same location as your tex file,
or the relative/absolute path may be specified in `\usepackage{}`

3.2 Writing your own Class

This is pretty similar to writing your own package, and we encourage you
check out the steps here `https://www.overleaf.com/learn/latex/Writing_`
`your_own_class`

4   Some cool LaTeX tricks

Apart from custom classes and packages, there are some very handy packages
which come into use in some very field specific usages. Some examples are
given as follows. We cannot list out how to use every command in each
of these packages, so we encourage you to check out more examples in the
documentation or on the internet.

4.1 Chemistry formulae

There are a few LaTeXpackages to create chemistry formulae: chemfig, ochem,
streetex, and xymtex. The most intuitive is probably the chemfig pack-
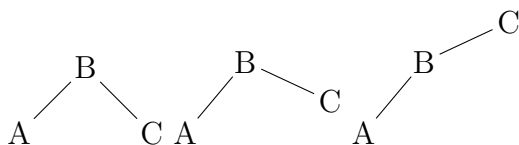age.

4.1.1 Introduction

Drawing a molecule consists mainly of connecting groups of atoms with lines. Simple linear formulas can be easily drawn with chemfig, `\chemfig{O=H}` generates O═══H. The command `\chemfig{O=H}` the draws the molecule. The symbol = determines the type of bond.

4.1.2 Angles

There are several ways to define angles to draw the bonds between molecules.

```
1  \chemfig{A−[1]B−[7]C}
2  \chemfig{A−[:50]B−[:−25]C}
3  \chemfig{A−[::50]B−[::−25]C}
```


Each one of the three commands in the example above uses a different method to determine the angle between bonds.

default units In the command `\chemfig{A-[1]B-[7]C}` the parameters inside brackets set the angle in special units, each unit equals 45°. Hence in the example the angles are $45^{c}irc$ and 315°.

absolute units The angles can be set in absolute units, in the command `\chemfig{A-[:50]B-[:-25]C}` the parameter inside the brackets represent the angle, in degrees, measured from the horizontal baseline. Negative angles are allowed.
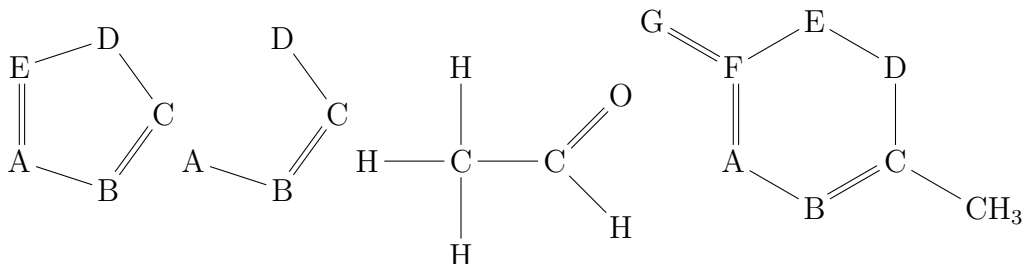
relative angles In the third example `\chemfig{A-[::50]B-[::-25]C}` the angles are measured from the previous bond, instead of the baseline.

4.1.3 Rings and Branches

You can even draw rings or branches using this package

```
1  \chemfig{A∗5(−B═C−D−E═)}
2  \chemfig{A∗5(−B═C−D)}
3  \chemfig{H−C(−[2]H)(−[6]H)−C(=[1]O)−[7]H}
```

```
4   \chemfig{A*6(−B=C(−CH_3)−D−E−F(=G)=)}
```

D, E, A, B, C, H, O, F, G, CH₃ — chemical structure diagrams

4.1.4 Formatting

You can customize the formatting using several parameters. Check `https://www.overleaf.com/learn/latex/Chemistry_formulae` for a reference

4.2 MO Diagrams

4.2.1 Introduction

Molecular diagrams are created using the package modiagram

```
1   \begin{MOdiagram}
2           \atom{left}{1s, 2s, 2p}
3   \end{MOdiagram}
```
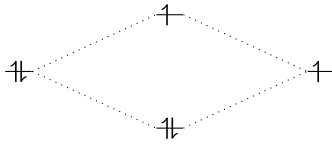
The basic command to draw MO diagrams is `\atom`. This command has two parameter in the example:

left The alignment of the atom.

1s, 2s, 2p The energy sub-levels to be drawn.

You can pass some extra information about the atomic orbitals to the command presented in the introductory example.

```
1  \begin{MOdiagram}
2   \atom{right}{
3      1s = {  0;  pair}  ,
4      2s = {  1;  pair}  ,
5      2p = {1.5;  up,  down }
6   }
7
8   \atom{left}{
9      1s = {  0;  pair}  ,
10     2s = {  1;  pair}  ,
11     2p = {1.5;  up,  down }
12  }
13 \end{MOdiagram}
```

The generic syntax to create atoms is: `1s = {energy; specifications}`

4.2.2   Molecules

The syntax for molecules is very similar to that of the `\atom`. The energy sub-levels 1s, 2s and 2p become 1sMO, 2sMO and 2pMO respectively.

```
1  \begin{MOdiagram}
2   \atom{left}{1s}
3   \atom{right}{1s={;up}}
4   \molecule{
5      1sMO={0.75;pair,up}
6   }
7  \end{MOdiagram}
```

4.3 Diagrams

One of the most extensive and useful packages in LaTeX is probably the Tikz package, used to draw any kinds of diagrams imaginable. We will just show you a few possibilites that tikz opens up for you.
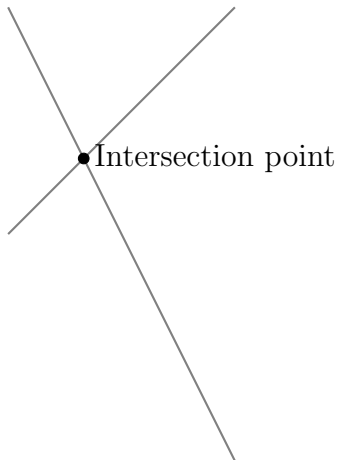
4.3.1 Introduction

First, you declare a tikzpicture environment, before this you must include the line `\usepackage{tikz}` in the preamble of your document.

```
\begin{tikzpicture}
    \draw[gray, thick] (-1,2) -- (2,-4);
    \draw[gray, thick] (-1,-1) -- (2,2);
    \filldraw[black] (0,0) circle (2pt) node[anchor=
        west] {Intersection point};
\end{tikzpicture}
```

See how powerful it is? Even with a few commands you can make extremely flexible graphics.

4.3.2 Basic elements: points, lines and paths

In this section is explained how to create basic graphic elements. These elements can be combined to create more elaborated figures.

```latex
1  \begin{tikzpicture}
2
3      \draw (-2,0) -- (2,0);
4      \filldraw [gray] (0,0) circle (2pt);
5      \draw (-2,-2) .. controls (0,0) .. (2,-2);
6      \draw (-2,2) .. controls (-1,0) and (1,0) .. (2,2);
7
8  \end{tikzpicture}
```
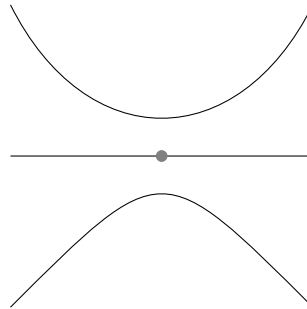
There are three basic commands in this example:

`\draw (-2,0) -- (2,0);` This defines a line whose endpoint are (-2,0) and (2,0). `\filldraw [gray] (0,0) circle (2pt);` The point is created as a very small gray circle centred at (0,0) and whose radius is (2pt). The `\filldraw` command is used in to draw elements and fill them with some specific colour. See the next section for more examples. `\draw (-2,2) .. controls (-1,0) and (1,` Draws a Bézier curve, is a bit tricky at first. There are 4 points defining it: (-2,2) and (2,2) are its endpoints, (-1,0) and (1,0) are control points (can be equal) that determine 'how curved' it is. You can think of these two points as "attractor points".

Note that all tikz commands end in a semicolon

4.3.3 Basic geometric shapes: Circles, ellipses and polygons

Geometric figures can be made up from simpler elements or created by an special command. Let's start with circles, ellipses and arcs.

17

```
1  \begin{tikzpicture}
2      \filldraw[color=red!60, fill=red!5, very thick
          ](-1,0) circle (1.5);
3      \fill[blue!50] (2.5,0) ellipse (1.5 and 0.5);
4      \draw[ultra thick, ->] (6.5,0) arc (0:220:1);
5  \end{tikzpicture}
```
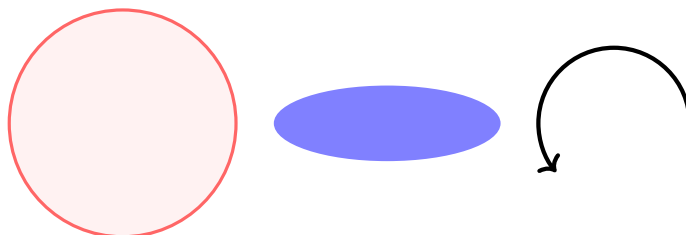
4.3.4   Nodes

The nodes are probably the most versatile elements in Tikz. We've already used one node in the introduction to add some text to the figure. In the next example nodes will be used to create a diagram.
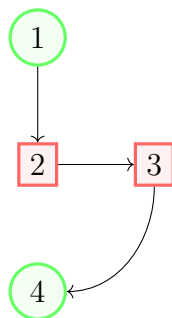
```
1  \begin{tikzpicture}[
2      roundnode/.style={circle, draw=green!60, fill=green
          !5, very thick, minimum size=7mm},
3      squarednode/.style={rectangle, draw=red!60, fill=
          red!5, very thick, minimum size=5mm},
4      ]
5      %Nodes
6      \node[squarednode]        (maintopic)
                                          {2};
7      \node[roundnode]          (uppercircle)        [above=
          of maintopic] {1};
8      \node[squarednode]        (rightsquare)        [right=
          of maintopic] {3};
9      \node[roundnode]          (lowercircle)        [below=
          of maintopic] {4};
10
11     %Lines
12     \draw[->] (uppercircle.south) -- (maintopic.north);
```

```
13        \draw[->] (maintopic.east) --- (rightsquare.west);
14        \draw[->] (rightsquare.south) .. controls +(down:7
            mm) and +(right:7mm) .. (lowercircle.east);
15  \end{tikzpicture}
```


There are essentially three commands in this figure: A node definition, a node declaration and lines that join two nodes.

```
roundnode/.style={circle, draw=green!60,
fill=green!5, very thick, minimum size=7mm}
```
Passed as a parameter to the tikzpicture environment defines a node that will be referred as roundnode, this node will be a circle whose outer ring will be green!60 and will be coloured with green!5, the stroke will be very thick and its minimum size is 7mm. The line below this defines a second rectangle-shaped node called squarednode with similar parameters.

```
\node[squarednode] (maintopic) {2};
```
This will create a squarednode, as defined in the previous command. This node will have an id, maintopic and will contain the number 2, if you leave an empty space inside the braces no text will be displayed.

```
[above=of maintopic]
```
Notice that all but the first node have an additional parameter, this parameter determines its position relative to other node. For instance [above=of maintopic] means that this node should appear above the node named maintopic. For this positioning system to work you have to add \usetikzlibrary{positioning} to your preamble. Without the positioning library, you can use the syntax above of=maintopic instead, but the positioning syntax is more flexible and powerful: you can extend it to write above=3cm of maintopic i.e. control the actual distance from maintopic.

```
\draw[->] (uppercircle.south) -- (maintopic.north);
```
An arrow-like straight line will be drawn. The syntax has been already explained at the basic elements section. The only thing special is the manner we write the endpoints of the line, by referencing a node (this is why we named them) and a position relative to the node.

4.4   Further

There are a lot more packages out there, here are a few we encourage you to go check out (some of them are requirements for completing the assignment)

tikz-3dplot Like tikz but lets you draw in 3D!

skak Useful for drawing chess notation as well as chessboards

knittingpatterns A useful package to make your document look better by using knitting patterns to embed your paragraphs in

CircuiTikz Versatile like tikz, has similar syntax but specializes in drawing electrical circuit diagrams

exam Useful for typesetting exams