Computer science

LaTeX
2020 – 2021

Lama TARSISSI

1.Introduction

What is $\TeX$?

Donald Ervin Knuth is an American computer scientist, mathematician, and professor emeritus at Stanford University. He is the 1974 recipient of the ACM Turing Award, informally considered the Nobel Prize of computer science.

What is TEX?

Knuth has been called the "father of the analysis of algorithms".

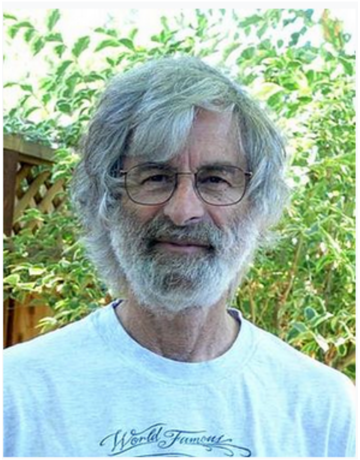
What is TEX?

Knuth is the creator of the TeX computer typesetting system, in 1977, the related METAFONT font definition language and rendering system, and the Computer Modern family of typefaces.

What is TEX?

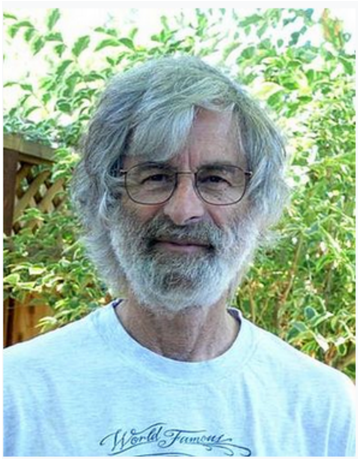
TeX is a popular means of typesetting complex mathematical formulae; it has been noted as one of the most sophisticated digital typographical systems and it was released in 1978.

Now $\LaTeX$?



Few years later, in 1984, **Leslie Lamport** who is an American computer scientist, mathematician, and Microsoft Research in Mountain View, California. He is the 2013 recipient of the ACM Turing Award.

Now LaTeX?



Lamport - due to his personal need of writing a book - also began working on a set of macros based on it, hoping that it would later become its standard macro package.

What is LaTeX?

1. LaTeX is a software system for document preparation.

What is LaTeX?

1. LaTeX is a software system for document preparation.
2. When writing, the writer uses **plain text** as opposed to the formatted text found in "What You See Is What You Get", WYSIWYG, word processors like Microsoft Word, LibreOffice Writer and Apple Pages.

What is LaTeX?

1. LaTeX is a software system for document preparation.
2. When writing, the writer uses **plain text** as opposed to the formatted text found in "What You See Is What You Get", WYSIWYG, word processors like Microsoft Word, LibreOffice Writer and Apple Pages.
3. LaTeX is widely used in academia for the communication and publication of scientific documents in many fields, including mathematics, statistics, computer science, engineering, physics, economics, linguistics, quantitative psychology, philosophy, and political science.

More

4. **LaTeX** is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation.

More

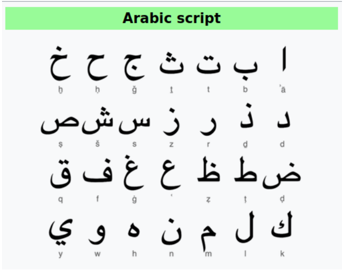
4. $\LaTeX$ is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation.
5. $\LaTeX$ is the de facto standard for the communication and publication of scientific documents.

More

4. $\LaTeX$ is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation.
5. $\LaTeX$ is the de facto standard for the communication and publication of scientific documents.
6. $\LaTeX$ started as a writing tool for mathematicians and computer scientists, but even from early in its development, it has also been taken up by scholars who needed to write documents that include complex math expressions or non-Latin scripts, such as Arabic,

More

4. LaTeX is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation.
5. LaTeX is the de facto standard for the communication and publication of scientific documents.
6. LaTeX started as a writing tool for mathematicians and computer scientists, but even from early in its development, it has also been taken up by scholars who needed to write documents that include complex math expressions or non-Latin scripts, such as Arabic,


Arabic script
ا 'ā
ب b
ت t
ث ṯ
ج ǧ
ح ḥ
خ ḫ
د d
ذ ḏ
ر r
ز z
س s
ش š
ص ṣ
ض ḍ
ط ṭ
ظ ẓ
ع ʿ
غ ġ
ف f
ق q
ك k
ل l
م m
ن n
ه h
و w
ي y

More

4. LaTeX is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation.
5. LaTeX is the de facto standard for the communication and publication of scientific documents.
6. LaTeX started as a writing tool for mathematicians and computer scientists, but even from early in its development, it has also been taken up by scholars who needed to write documents that include complex math expressions or non-Latin scripts, such as Arabic,Devanagari

Devanagari
देवनागरी
Chandas
अ आ इ ई उ ऊ
ऋ ॠ ऌ ॡ ए ऐ
ओ औ
ॐ
तत् त्वम् असि
क ख ग घ ङ च छ ज झ ञ ट
ठ ड ढ ण त थ द ध न प फ
ब भ म य र ल ळ व श ष स ह

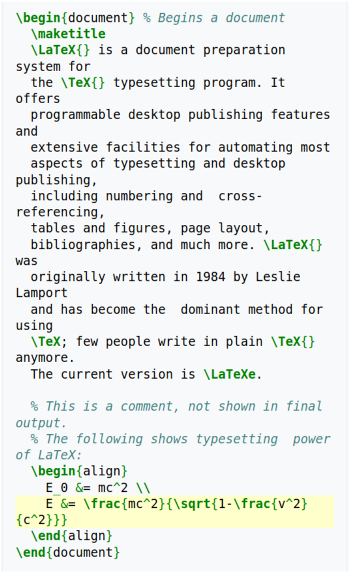
More

4. LaTeX is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation.
5. LaTeX is the de facto standard for the communication and publication of scientific documents.
6. LaTeX started as a writing tool for mathematicians and computer scientists, but even from early in its development, it has also been taken up by scholars who needed to write documents that include complex math expressions or non-Latin scripts, such as Arabic,Devanagari and Chinese.

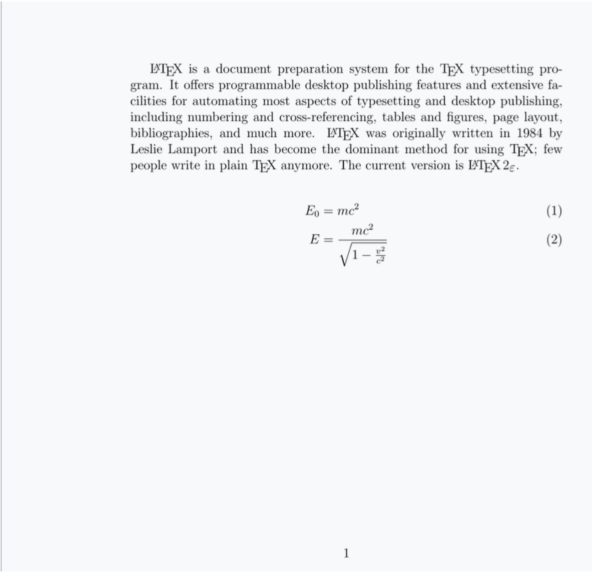
More

4. LaTeX is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation.
5. LaTeX is the de facto standard for the communication and publication of scientific documents.
6. LaTeX started as a writing tool for mathematicians and computer scientists, but even from early in its development, it has also been taken up by scholars who needed to write documents that include complex math expressions or non-Latin scripts, such as Arabic,Devanagari and Chinese.
7. LaTeX is available as free software.

Example



```latex
\begin{document} % Begins a document
  \maketitle
  \LaTeX{} is a document preparation
system for
  the \TeX{} typesetting program. It
offers
  programmable desktop publishing features
and
  extensive facilities for automating most
  aspects of typesetting and desktop
publishing,
  including numbering and  cross-
referencing,
  tables and figures, page layout,
  bibliographies, and much more. \LaTeX{}
was
  originally written in 1984 by Leslie
Lamport
  and has become the  dominant method for
using
  \TeX; few people write in plain \TeX{}
anymore.
  The current version is \LaTeXe.

  % This is a comment, not shown in final
output.
  % The following shows typesetting  power
of LaTeX:
  \begin{align}
    E_0 &= mc^2 \\
    E &= \frac{mc^2}{\sqrt{1-\frac{v^2}
{c^2}}}
  \end{align}
\end{document}
```



LaTeX is a document preparation system for the TeX typesetting program. It offers programmable desktop publishing features and extensive facilities for automating most aspects of typesetting and desktop publishing, including numbering and cross-referencing, tables and figures, page layout, bibliographies, and much more. LaTeX was originally written in 1984 by Leslie Lamport and has become the dominant method for using TeX; few people write in plain TeX anymore. The current version is LaTeX 2_ε.

$$E_0 = mc^2 \tag{1}$$

$$E = \frac{mc^2}{\sqrt{1-\frac{v^2}{c^2}}} \tag{2}$$

1

2.Why I write with LaTeX

Note

LaTeX is not exactly a word processor in the traditional sense.

Note

LaTeX is not exactly a word processor in the traditional sense.
When we use, for example, Google Docs or Microsoft Word, we see both the content of the document as well as its layout and formatting. This is sometimes called WYSIWYG - what you see is what you get.

Note

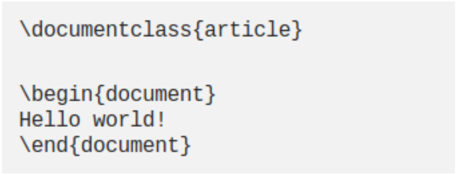
LaTeX is not exactly a word processor in the traditional sense.
When we use, for example, Google Docs or Microsoft Word, we see both the content of the document as well as its layout and formatting. This is sometimes called WYSIWYG - what you see is what you get.
LaTeX is different: Instead of WYSIWYG, LaTeX operates with a "source code"view that consists of unformatted text and markup commands that tell **LaTeX** what to do with that text in your final, **compiled** document.

Hello world!

Let's say you wanted to type the phrase "Hello world!" in a $\LaTeX$ document. In addition to typing "Hello world!", you'd have to give **$\LaTeX$** some instructions. A minimal example looks like this:

Hello world!

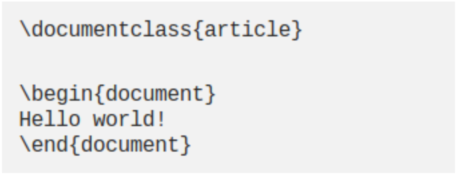
Let's say you wanted to type the phrase "Hello world!" in a LaTeX document. In addition to typing "Hello world!", you'd have to give **LaTeX** some instructions. A minimal example looks like this:



```
\documentclass{article}

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

Hello world!

Let's say you wanted to type the phrase "Hello world!" in a LaTeX document. In addition to typing "Hello world!", you'd have to give **LaTeX** some instructions. A minimal example looks like this:

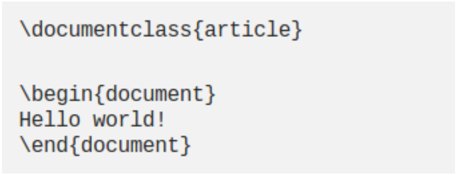


```
\documentclass{article}

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

This might seem weird at first: Why would you want to do additional work?

Hello world!

Let's say you wanted to type the phrase "Hello world!" in a LaTeX document. In addition to typing "Hello world!", you'd have to give **LaTeX** some instructions. A minimal example looks like this:

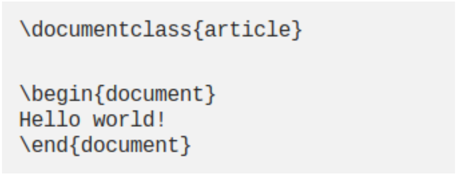


```
\documentclass{article}

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

This might seem weird at first: Why would you want to do additional work? When we use a word processor like Google Docs or Microsoft Word, we simply write what we want to write and we're done.

Hello world!

Let's say you wanted to type the phrase "Hello world!" in a LaTeX document. In addition to typing "Hello world!", you'd have to give **LaTeX** some instructions. A minimal example looks like this:



```
\documentclass{article}

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

This might seem weird at first: Why would you want to do additional work? When we use a word processor like Google Docs or Microsoft Word, we simply write what we want to write and we're done.
With LaTeX we have to write what we want to write and, in addition, we have to tell LaTeX exactly what we want it to do with the things we have written in order to produce a legible document.

The benefits of LaTeX

1. **Separating thinking and layouting:**

   1. The layout and visual formatting of your writing certainly matter a lot - depending on what your text looks like, it will be more or less legible and visually attractive.

The benefits of LaTeX

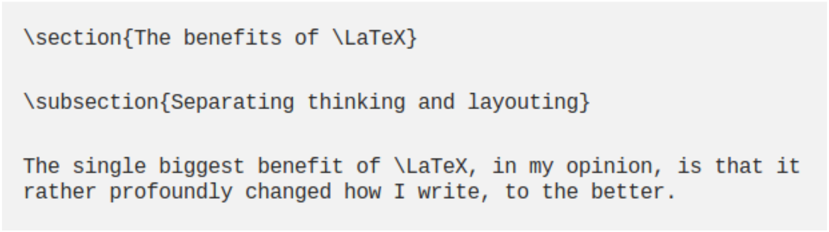
1. **Separating thinking and layouting:**

   1. The layout and visual formatting of your writing certainly matter a lot - depending on what your text looks like, it will be more or less legible and visually attractive.
   2. But writing and layouting really are two separate processes: Thinking thoughts and expressing them in writing is the core challenge of writing; changing what those thoughts that you have successfully written down look like is a second and separate step.

The benefits of LaTeX

1. Separating thinking and layouting:

1. The layout and visual formatting of your writing certainly matter a lot - depending on what your text looks like, it will be more or less legible and visually attractive.
2. But writing and layouting really are two separate processes: Thinking thoughts and expressing them in writing is the core challenge of writing; changing what those thoughts that you have successfully written down look like is a second and separate step.



```
\section{The benefits of \LaTeX}

\subsection{Separating thinking and layouting}

The single biggest benefit of \LaTeX, in my opinion, is that it
rather profoundly changed how I write, to the better.
```

2. No weird accidental layouting and formatting problems

2. No weird accidental layouting and formatting problems

When using something like Microsoft Word or LibreOffice for documents that are long-ish (let's say five or more A4 pages of text), layouting can become really weird.

2. No weird accidental layouting and formatting problems

When using something like Microsoft Word or LibreOffice for documents that are long-ish (let's say five or more A4 pages of text), layouting can become really weird.

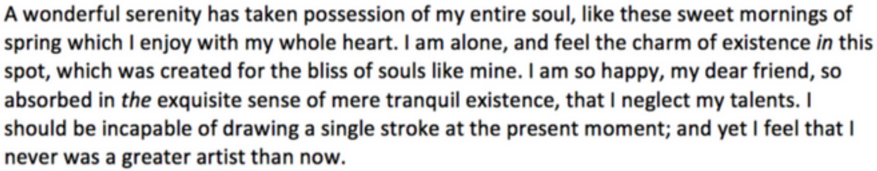
For example, you have an image on one of your pages, and more or less suddenly, the position of the image changes slightly. Then, when you try to drag the image back to where you want it, something else changes in a paragraph below the image. Then, as you try drag everything back to normal, more and more things change.

2. No weird accidental layouting and formatting problems

When using something like Microsoft Word or LibreOffice for documents that are long-ish (let's say five or more A4 pages of text), layouting can become really weird.

For example, you have an image on one of your pages, and more or less suddenly, the position of the image changes slightly. Then, when you try to drag the image back to where you want it, something else changes in a paragraph below the image. Then, as you try drag everything back to normal, more and more things change.

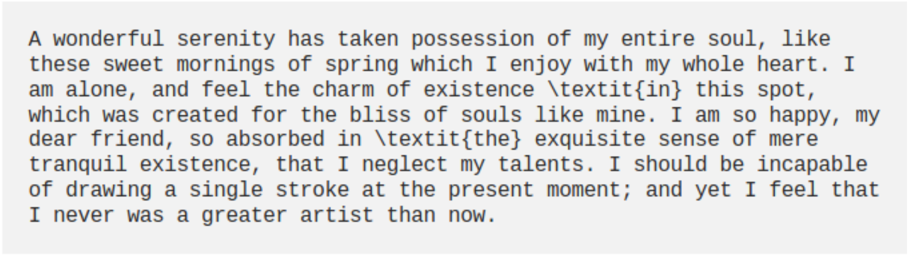
Here's a bunch of text in Microsoft Word with two italicized words:



A wonderful serenity has taken possession of my entire soul, like these sweet mornings of spring which I enjoy with my whole heart. I am alone, and feel the charm of existence *in* this spot, which was created for the bliss of souls like mine. I am so happy, my dear friend, so absorbed in *the* exquisite sense of mere tranquil existence, that I neglect my talents. I should be incapable of drawing a single stroke at the present moment; and yet I feel that I never was a greater artist than now.

2. No weird accidental layouting and formatting problems

When using something like Microsoft Word or LibreOffice for documents that are long-ish (let's say five or more A4 pages of text), layouting can become really weird.
For example, you have an image on one of your pages, and more or less suddenly, the position of the image changes slightly. Then, when you try to drag the image back to where you want it, something else changes in a paragraph below the image. Then, as you try drag everything back to normal, more and more things change.
Here's a bunch of text in Microsoft Word with two italicized words:



```
A wonderful serenity has taken possession of my entire soul, like
these sweet mornings of spring which I enjoy with my whole heart. I
am alone, and feel the charm of existence \textit{in} this spot,
which was created for the bliss of souls like mine. I am so happy, my
dear friend, so absorbed in \textit{the} exquisite sense of mere
tranquil existence, that I neglect my talents. I should be incapable
of drawing a single stroke at the present moment; and yet I feel that
I never was a greater artist than now.
```

3. **It's easy!**
4. **Great "scientific" features**
5. **Free and open source = many different editors**
6. **Nice output**

Word:

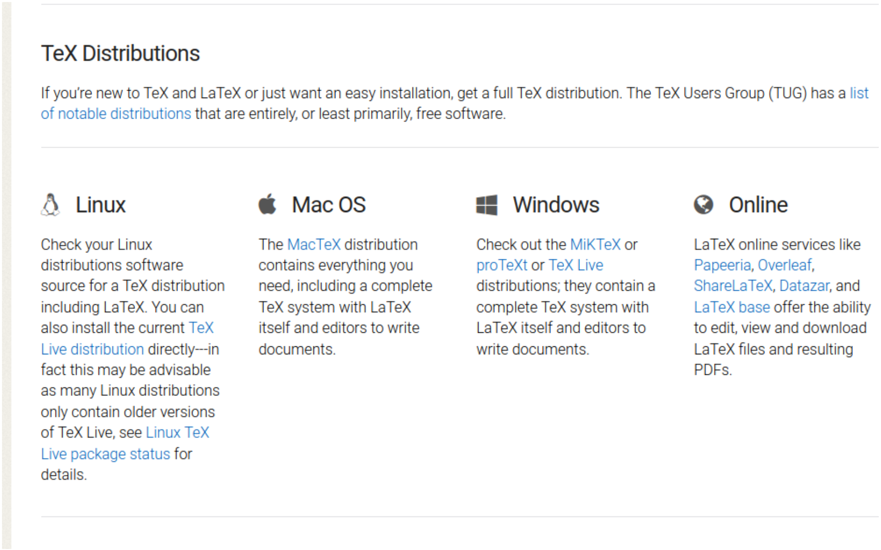
$$\iiint_G \left[u\nabla^2 v + (\nabla u, \nabla v)\right] d^3V = \oiint_S \left(u\frac{\partial v}{\partial n} + v\frac{\partial u}{\partial n}\right) d^2A$$

LaTeX:

$$\iiint_G \left[u\nabla^2 v - v\nabla^2 u\right] d^3V = \oiint_S \left(u\frac{\partial v}{\partial n} - v\frac{\partial u}{\partial n}\right) d^2A$$

3.Installation and configuration

Installing a TeX distribution on your computer:
https://www.latex-project.org/get/



TeX Distributions

If you're new to TeX and LaTeX or just want an easy installation, get a full TeX distribution. The TeX Users Group (TUG) has a list of notable distributions that are entirely, or least primarily, free software.

Linux

Check your Linux distributions software source for a TeX distribution including LaTeX. You can also install the current TeX Live distribution directly---in fact this may be advisable as many Linux distributions only contain older versions of TeX Live, see Linux TeX Live package status for details.

Mac OS

The MacTeX distribution contains everything you need, including a complete TeX system with LaTeX itself and editors to write documents.

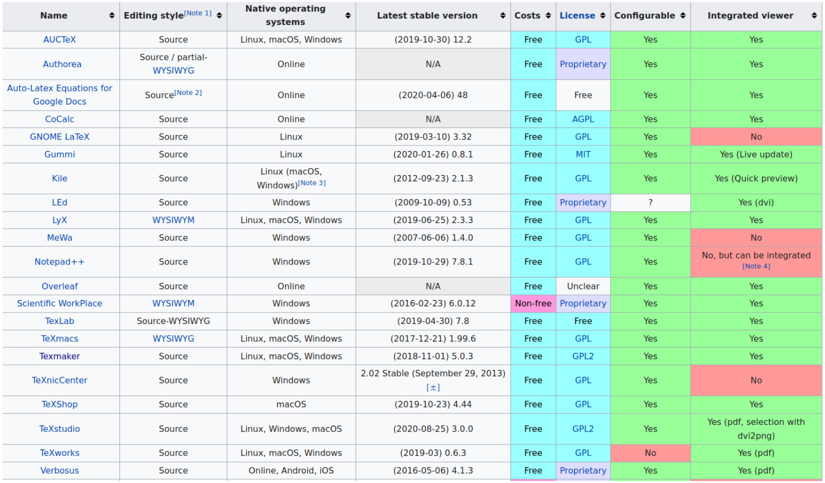
Windows

Check out the MiKTeX or proTeXt or TeX Live distributions; they contain a complete TeX system with LaTeX itself and editors to write documents.

Online

LaTeX online services like Papeeria, Overleaf, ShareLaTeX, Datazar, and LaTeX base offer the ability to edit, view and download LaTeX files and resulting PDFs.

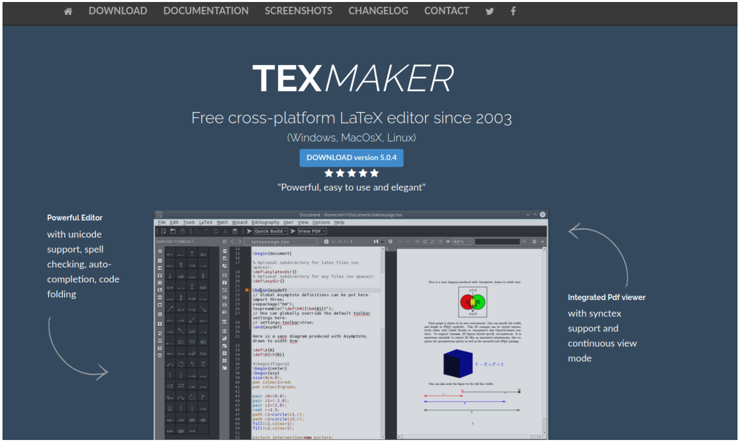
List of LaTeX editors:



Name	Editing style[Note 1]	Native operating systems	Latest stable version	Costs	License	Configurable	Integrated viewer
AUCTeX	Source	Linux, macOS, Windows	(2019-10-30) 12.2	Free	GPL	Yes	Yes
Authorea	Source / partial-WYSIWYG	Online	N/A	Free	Proprietary	Yes	Yes
Auto-Latex Equations for Google Docs	Source[Note 2]	Online	(2020-04-06) 48	Free	Free	Yes	Yes
CoCalc	Source	Online	N/A	Free	AGPL	Yes	Yes
GNOME LaTeX	Source	Linux	(2019-03-10) 3.32	Free	GPL	Yes	No
Gummi	Source	Linux	(2020-01-26) 0.8.1	Free	MIT	Yes	Yes (Live update)
Kile	Source	Linux (macOS, Windows)[Note 3]	(2012-09-23) 2.1.3	Free	GPL	Yes	Yes (Quick preview)
LEd	Source	Windows	(2009-10-09) 0.53	Free	Proprietary	?	Yes (dvi)
LyX	WYSIWYM	Linux, macOS, Windows	(2019-06-25) 2.3.3	Free	GPL	Yes	Yes
MeWa	Source	Windows	(2007-06-06) 1.4.0	Free	GPL	Yes	No
Notepad++	Source	Windows	(2019-10-29) 7.8.1	Free	GPL	Yes	No, but can be integrated [Note 4]
Overleaf	Source	Online	N/A	Free	Unclear	Yes	Yes
Scientific WorkPlace	WYSIWYM	Windows	(2016-02-23) 6.0.12	Non-free	Proprietary	Yes	Yes
TexLab	Source-WYSIWYG	Windows	(2019-04-30) 7.8	Free	Free	Yes	Yes
TeXmacs	WYSIWYG	Linux, macOS, Windows	(2017-12-21) 1.99.6	Free	GPL	Yes	Yes
Texmaker	Source	Linux, macOS, Windows	(2018-11-01) 5.0.3	Free	GPL2	Yes	Yes
TeXnicCenter	Source	Windows	2.02 Stable (September 29, 2013) [±]	Free	GPL	Yes	No
TeXShop	Source	macOS	(2019-10-23) 4.44	Free	GPL	Yes	Yes
TeXstudio	Source	Linux, Windows, macOS	(2020-08-25) 3.0.0	Free	GPL2	Yes	Yes (pdf, selection with dvi2png)
TeXworks	Source	Linux, macOS, Windows	(2019-03) 0.6.3	Free	GPL	No	Yes (pdf)
Verbosus	Source	Online, Android, iOS	(2016-05-06) 4.1.3	Free	Proprietary	Yes	Yes (pdf)

TEXMAKER

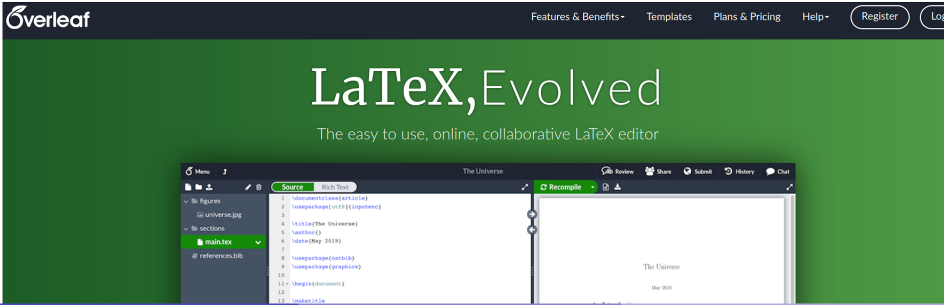
https://www.xm1math.net/texmaker/


DOWNLOAD
DOCUMENTATION
SCREENSHOTS
CHANGELOG
CONTACT
TEXMAKER
Free cross-platform LaTeX editor since 2003
(Windows, MacOsX, Linux)
DOWNLOAD version 5.0.4
★★★★★
"Powerful, easy to use and elegant"
Powerful Editor
with unicode support, spell checking, auto-completion, code folding
Integrated Pdf viewer
with synctex support and continuous view mode

Online LaTeX editors

ShareLaTeX, Overleaf

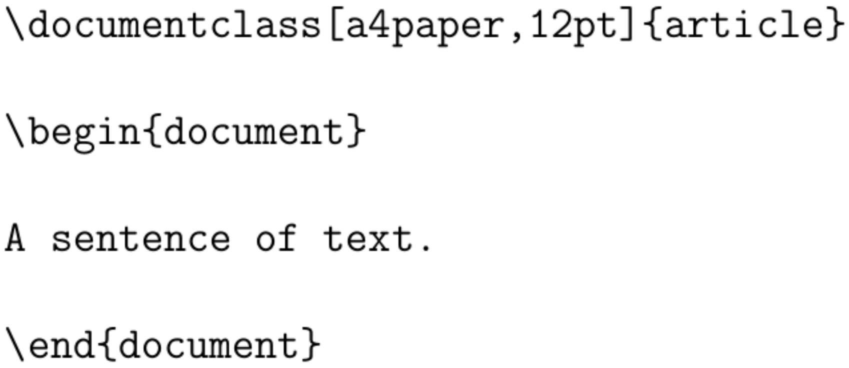
ShareLaTeX



```
\documentclass{article}
\usepackage[utf8]{inputenc}

\title{The Universe}
\author{}
\date{May 2019}

\usepackage{natbib}
\usepackage{graphicx}

\begin{document}

\maketitle
```

First LaTeX file

Type the following:



```
\documentclass[a4paper,12pt]{article}

\begin{document}

A sentence of text.

\end{document}
```

First LaTeX file

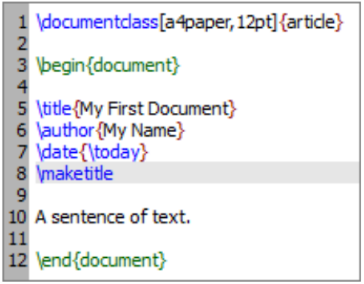
Type the following:

- Click on the **Save** button.
- Create a new folder called **LaTeX course** in **Libraries>Documents**.
- Name your document **Doc1** and save it as a **TeX document** in this folder.

Type the following directly after the \begin{document} command:

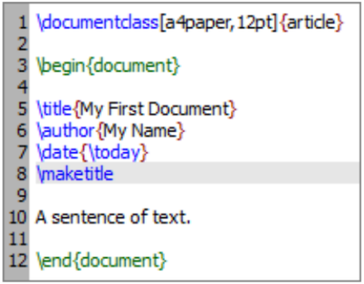
```
\title{My First Document}
\author{My Name}
\date{\today}
\maketitle
```

Type the following directly after the \begin{document} command:



```
\documentclass[a4paper,12pt]{article}

\begin{document}

\title{My First Document}
\author{My Name}
\date{\today}
\maketitle

A sentence of text.

\end{document}
```

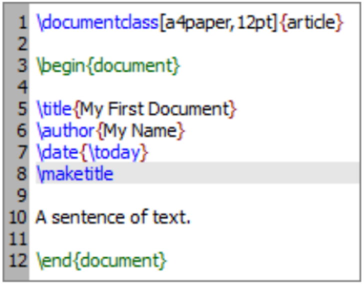
Type the following directly after the \begin{document} command:



```
\documentclass[a4paper,12pt]{article}

\begin{document}

\title{My First Document}
\author{My Name}
\date{\today}
\maketitle

A sentence of text.

\end{document}
```

1. \today is a command that inserts today's date. You can also type in a different date, for example \date{November 2020}.

Type the following directly after the \begin{document} command:



```
\documentclass[a4paper,12pt]{article}

\begin{document}

\title{My First Document}
\author{My Name}
\date{\today}
\maketitle

A sentence of text.

\end{document}
```

1. \today is a command that inserts today's date. You can also type in a different date, for example \date{November 2020}.
2. **Article** documents start the text immediately below the title on the same page. **Reports** put the title on a separate page.

1.Table of contents

Table of contents

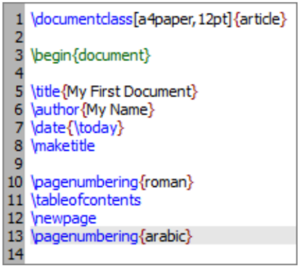
- If you use sectioning commands it is very easy to generate a table of contents. Type \tableofcontents where you want the table of contents to appear in your document. (often directly after the title page).

Table of contents

- If you use sectioning commands it is very easy to generate a table of contents. Type \tableofcontents where you want the table of contents to appear in your document. (often directly after the title page).
- You may also want to change the page numbering so that roman numerals (i, ii, iii)are used for pages before the main document starts. This will also ensure that the main document starts on page 1.

Table of contents

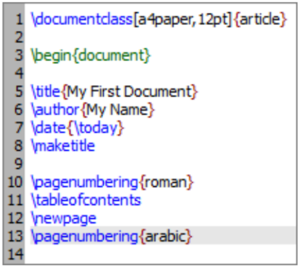
- If you use sectioning commands it is very easy to generate a table of contents. Type \tableofcontents where you want the table of contents to appear in your document. (often directly after the title page).
- You may also want to change the page numbering so that roman numerals (*i*, *ii*, *iii*)are used for pages before the main document starts. This will also ensure that the main document starts on page 1. Page numbering can be switched between **arabic** and **roman** using \pagenumbering{...}.



```latex
\documentclass[a4paper,12pt]{article}

\begin{document}

\title{My First Document}
\author{My Name}
\date{\today}
\maketitle

\pagenumbering{roman}
\tableofcontents
\newpage
\pagenumbering{arabic}

```

Table of contents

- If you use sectioning commands it is very easy to generate a table of contents. Type \tableofcontents where you want the table of contents to appear in your document. (often directly after the title page).
- You may also want to change the page numbering so that roman numerals (*i*, *ii*, *iii*)are used for pages before the main document starts. This will also ensure that the main document starts on page 1. Page numbering can be switched between **arabic** and **roman** using \pagenumbering{...}.

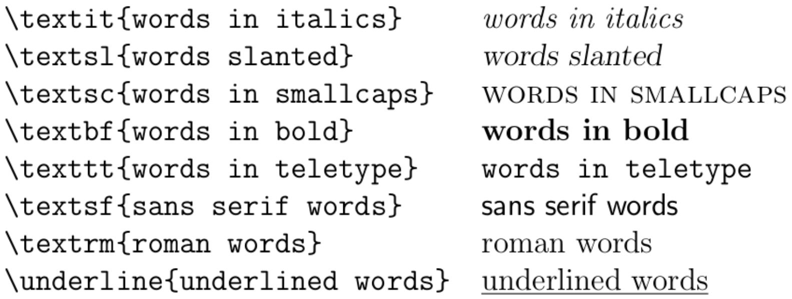


```
\documentclass[a4paper,12pt]{article}

\begin{document}

\title{My First Document}
\author{My Name}
\date{\today}
\maketitle

\pagenumbering{roman}
\tableofcontents
\newpage
\pagenumbering{arabic}

```

- The \newpage command inserts a page break

2.Typesetting Text

Font Effects



`\textit{words in italics}`	*words in italics*
`\textsl{words slanted}`	*words slanted*
`\textsc{words in smallcaps}`	WORDS IN SMALLCAPS
`\textbf{words in bold}`	**words in bold**
`\texttt{words in teletype}`	`words in teletype`
`\textsf{sans serif words}`	sans serif words
`\textrm{roman words}`	roman words
`\underline{underlined words}`	$\underline{\text{underlined words}}$

Coloured Text

1. To put coloured text in your document you need to use a **package**.

Coloured Text

1. To put coloured text in your document you need to use a **package**.
2. There are many packages that can be used with LaTeXto enhance its functionality.

Coloured Text

1. To put coloured text in your document you need to use a **package**.
2. There are many packages that can be used with LaTeXto enhance its functionality.
3. Packages are included in the preamble (before the \begin{document} command).

Coloured Text

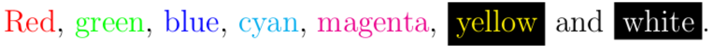
1. To put coloured text in your document you need to use a **package**.
2. There are many packages that can be used with LaTeXto enhance its functionality.
3. Packages are included in the preamble (before the \begin{document} command).
4. Packages are **activated** using the \usepackage[options]{package}command.

Coloured Text

1. To put coloured text in your document you need to use a **package**.
2. There are many packages that can be used with LaTeXto enhance its functionality.
3. Packages are included in the preamble (before the \begin{document} command).
4. Packages are **activated** using the \usepackage[options]{package}command.
5. **package** is the name of the package and **options** is an optional list of keywords that trigger special features in the package.

Coloured Text

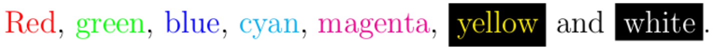
1. To put coloured text in your document you need to use a **package**.
2. There are many packages that can be used with LaTeXto enhance its functionality.
3. Packages are included in the preamble (before the \begin{document} command).
4. Packages are **activated** using the \usepackage[options]{package}command.
5. **package** is the name of the package and **options** is an optional list of keywords that trigger special features in the package.
6. The basic colour names that\usepackage{color} knows about are black, red, green, blue, cyan, magenta, yellow and white:



Red, green, blue, cyan, magenta, yellow and white.

Coloured Text

1. To put coloured text in your document you need to use a **package**.
2. There are many packages that can be used with LaTeXto enhance its functionality.
3. Packages are included in the preamble (before the \begin{document} command).
4. Packages are **activated** using the \usepackage[options]{package}command.
5. **package** is the name of the package and **options** is an optional list of keywords that trigger special features in the package.
6. The basic colour names that\usepackage{color} knows about are black, red, green, blue, cyan, magenta, yellow and white:

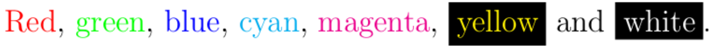


Red, green, blue, cyan, magenta, yellow and white.

7. The following code to produces coloured text: {\color{colour_name}text}

Coloured Text

1. To put coloured text in your document you need to use a **package**.
2. There are many packages that can be used with LaTeXto enhance its functionality.
3. Packages are included in the preamble (before the \begin{document} command).
4. Packages are **activated** using the \usepackage[options]{package}command.
5. **package** is the name of the package and **options** is an optional list of keywords that trigger special features in the package.
6. The basic colour names that\usepackage{color} knows about are black, red, green, blue, cyan, magenta, yellow and white:



Red, green, blue, cyan, magenta, yellow and white.

7. The following code to produces coloured text: {\color{colour_name}text}
8. Where colour_name is the name of the colour you want, and text is the text you want to be coloured.

Font Sizes

There are LaTeXcommands for a range of font sizes:

`{\tiny tiny words}`	tiny words
`{\scriptsize scriptsize words}`	scriptsize words
`{\footnotesize footnotesize words}`	footnotesize words
`{\small small words}`	small words
`{\normalsize normalsize words}`	normalsize words
`{\large large words}`	large words
`{\Large Large words}`	Large words
`{\LARGE LARGE words}`	LARGE words
`{\huge huge words}`	huge words

Lists

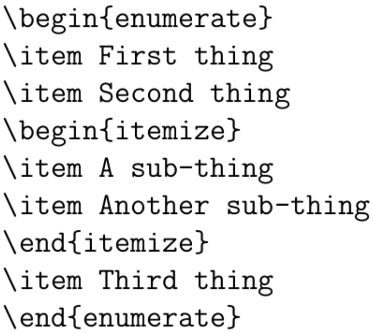
- LaTeXsupports two types of lists:
    1. enumerate produces numbered lists.

Lists

- LaTeXsupports two types of lists:
  1. enumerate produces numbered lists.
  2. itemize is for bulleted lists.
- Each list item is defined by \item.

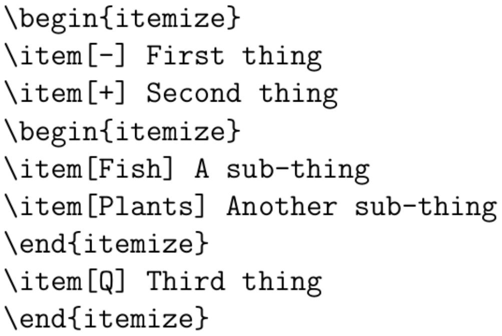
Lists

- LaTeXsupports two types of lists:
  1. enumerate produces numbered lists.
  2. itemize is for bulleted lists.
- Each list item is defined by \item.
- Lists can be nested to produce **sub-lists**.



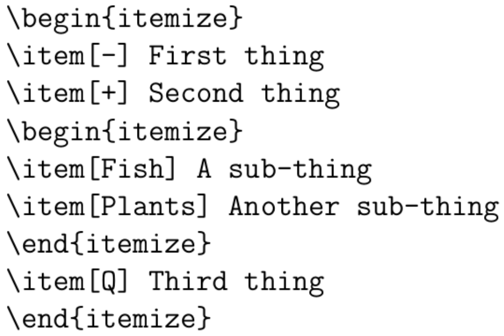
```
\begin{enumerate}
\item First thing
\item Second thing
\begin{itemize}
\item A sub-thing
\item Another sub-thing
\end{itemize}
\item Third thing
\end{enumerate}
```

It is easy to change the bullet symbol using square brackets after the **item**



```
\begin{itemize}
\item[-] First thing
\item[+] Second thing
\begin{itemize}
\item[Fish] A sub-thing
\item[Plants] Another sub-thing
\end{itemize}
\item[Q] Third thing
\end{itemize}
```

It is easy to change the bullet symbol using square brackets after the **item**

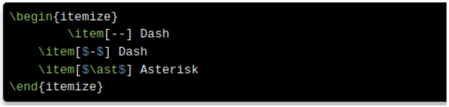


```
\begin{itemize}
\item[-] First thing
\item[+] Second thing
\begin{itemize}
\item[Fish] A sub-thing
\item[Plants] Another sub-thing
\end{itemize}
\item[Q] Third thing
\end{itemize}
```

Think of checking: \addtocounter{enumi}{n}

Changing the numbering / bullets

You can easily modify the output of the list. You can make the following changes easily without loading a package:



```
\begin{itemize}
        \item[--] Dash
    \item[$-$] Dash
    \item[$\ast$] Asterisk
\end{itemize}
```

Comments and Spacing

1. Comments are created using %. When LaTeX encounters a% character while processing a .tex file, it ignores the rest of the line (until the [Return] key has been pressed to start a new line.

Comments and Spacing

1. Comments are created using %. When LaTeX encounters a% character while processing a .tex file, it ignores the rest of the line (until the [Return] key has been pressed to start a new line.

2. Two backslashes (\\) can be used to start a new line.

Comments and Spacing

1. Comments are created using %. When LaTeX encounters a% character while processing a .tex file, it ignores the rest of the line (until the [Return] key has been pressed to start a new line.
2. Two backslashes (\\) can be used to start a new line.
3. If you want to add blank space into your document use the\vspace{...} or \hspace{...} commands.

Comments and Spacing

1. Comments are created using %. When LaTeX encounters a% character while processing a .tex file, it ignores the rest of the line (until the [Return] key has been pressed to start a new line.
2. Two backslashes (\\) can be used to start a new line.
3. If you want to add blank space into your document use the\vspace{...} or \hspace{...} commands.
4. We can also use the \bigskip command.

Special Characters

The following symbols are reserved characters which have a special meaning in LaTeX:

```
#  $  %  ^  &  _  {  }  ~  \
```

Special Characters

The following symbols are reserved characters which have a special meaning in LaTeX:

```
#   $   %   ^   &   _   {   }   ~   \
```

All of these apart from the backslash \ can be inserted as characters in your document by adding a prefix backslash:

```
\#  \$  \%  \^{} \&  \_  \{  \}  \~{}
```

Special Characters

The following symbols are reserved characters which have a special meaning in LaTeX:

```
#  $  %  ^  &  _  {  }  ~  \
```

All of these apart from the backslash \ can be inserted as characters in your document by adding a prefix backslash:

```
\#  \$  \%  \^{}  \&  \_  \{  \}  \~{}
```

The backslash character \ can not be entered by adding a prefix backslash, \\, as this is used for line breaking. Use the `\textbackslash` command instead.

1.Tables

Tables

The `tabular` command is used to typeset tables. By default, LaTeX tables are drawn without horizontal and vertical lines — you need to specify if you want lines drawn. LaTeX determines the width of the columns automatically.

Tables

The `tabular` command is used to typeset tables. By default, LaTeX tables are drawn without horizontal and vertical lines — you need to specify if you want lines drawn. LaTeX determines the width of the columns automatically.

This code starts a table:

```
\begin{tabular}{...}
```

Tables

The `tabular` command is used to typeset tables. By default, LaTeX tables are drawn without horizontal and vertical lines — you need to specify if you want lines drawn. LaTeX determines the width of the columns automatically.

This code starts a table:

```
\begin{tabular}{...}
```

Where the dots between the curly brackets are replaced by code defining the columns:

- `l` for a column of **left**-aligned text (letter el, *not* number one).
- `r` for a column of **right**-aligned text.
- `c` for a column of **centre**-aligned text.
- `|` for a vertical line.

The table data follows the `\begin` command:

- `&` is placed between columns.
- `\\` is placed at the end of a row (to start a new one).

The table data follows the `\begin` command:

- `&` is placed between columns.
- `\\` is placed at the end of a row (to start a new one).

- `\hline` inserts a horizontal line.
- `\cline{1-2}` inserts a partial horizontal line between column 1 and column 2.

The table data follows the `\begin` command:

- `&` is placed between columns.
- `\\` is placed at the end of a row (to start a new one).

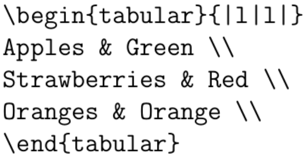
- `\hline` inserts a horizontal line.
- `\cline{1-2}` inserts a partial horizontal line between column 1 and column 2.

The command\end{tabular} finishes the table.

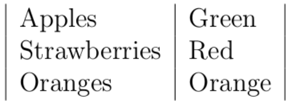
Examples

```
\begin{tabular}{|l|l|}
Apples & Green \\
Strawberries & Red \\
Oranges & Orange \\
\end{tabular}
```

Examples



```
\begin{tabular}{|l|l|}
Apples & Green \\
Strawberries & Red \\
Oranges & Orange \\
\end{tabular}
```



Apples	Green
Strawberries	Red
Oranges	Orange

Examples

```
\begin{tabular}{rc}
Apples & Green \\
\hline
Strawberries & Red \\
\cline{1-1}
Oranges & Orange \\
\end{tabular}
```

Examples

```
\begin{tabular}{rc}
Apples & Green \\
\hline
Strawberries & Red \\
\cline{1-1}
Oranges & Orange \\
\end{tabular}
```

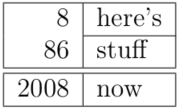
Apples	Green
Strawberries	Red
Oranges	Orange

Examples

```
\begin{tabular}{|r|l|}
\hline
8 & here's \\
\cline{2-2}
86 & stuff \\
\hline \hline
2008 & now \\
\hline
\end{tabular}
```

Examples

```
\begin{tabular}{|r|l|}
\hline
8 & here's \\
\cline{2-2}
86 & stuff \\
\hline \hline
2008 & now \\
\hline
\end{tabular}
```



8	here's
86	stuff
2008	now

Exercise

➲ Write code to produce the following tables:

Item	Quantity	Price ($)
Nails	500	0.34
Wooden boards	100	4.00
Bricks	240	11.50

Exercise

➲ Write code to produce the following tables:

Item	Quantity	Price ($)
Nails	500	0.34
Wooden boards	100	4.00
Bricks	240	11.50

	Year		
City	2006	2007	2008
London	45789	46551	51298
Berlin	34549	32543	29870
Paris	49835	51009	51970

2.Figures

To insert an image in to your LaTeXdocument, which requires the graphicx package. Images should be PDF, PNG, JPEGor GIF files. The following code will insert an image called **myimage**:

To insert an image in to your LaTeXdocument, which requires the graphicx package. Images should be PDF, PNG, JPEGor GIF files. The following code will insert an image called **myimage**:

```
\begin{figure}[h]
\centering
\includegraphics[width=1\textwidth]{myimage}
\caption{Here is my image}
\label{image-myimage}
\end{figure}
```

To insert an image in to your LaTeXdocument, which requires the graphicx package. Images should be PDF, PNG, JPEGor GIF files. The following code will insert an image called **myimage**:

```
\begin{figure}[h]
\centering
\includegraphics[width=1\textwidth]{myimage}
\caption{Here is my image}
\label{image-myimage}
\end{figure}
```

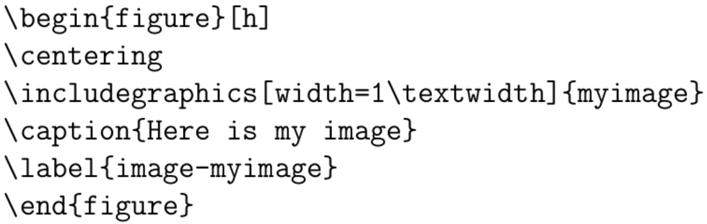
- **[h]** is the placement specifier. h means put the figure approximately here.

To insert an image in to your LaTeXdocument, which requires the graphicx package. Images should be PDF, PNG, JPEGor GIF files. The following code will insert an image called **myimage**:

```
\begin{figure}[h]
\centering
\includegraphics[width=1\textwidth]{myimage}
\caption{Here is my image}
\label{image-myimage}
\end{figure}
```

- **[h]** is the placement specifier. h means put the figure approximately here.
- Other options are **t** (at the top of the page),

To insert an image in to your LaTeXdocument, which requires the graphicx package. Images should be PDF, PNG, JPEGor GIF files. The following code will insert an image called **myimage**:



```
\begin{figure}[h]
\centering
\includegraphics[width=1\textwidth]{myimage}
\caption{Here is my image}
\label{image-myimage}
\end{figure}
```

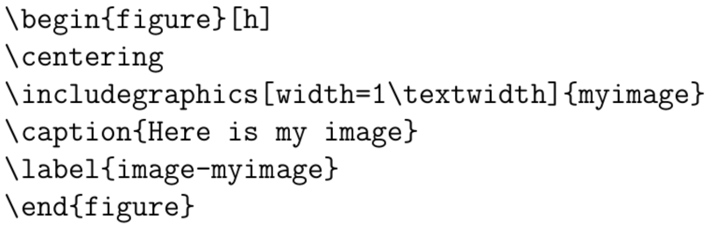
- **[h]** is the placement specifier. h means put the figure approximately here.
- Other options are **t** (at the top of the page),**b** (at the bottom of the page)

To insert an image in to your LATEXdocument, which requires the graphicx package. Images should be PDF, PNG, JPEGor GIF files. The following code will insert an image called **myimage**:

```
\begin{figure}[h]
\centering
\includegraphics[width=1\textwidth]{myimage}
\caption{Here is my image}
\label{image-myimage}
\end{figure}
```

- **[h]** is the placement specifier. h means put the figure approximately here.
- Other options are **t** (at the top of the page),**b** (at the bottom of the page)and **p** (on a separate page for figures).

To insert an image in to your LaTeXdocument, which requires the graphicx package. Images should be PDF, PNG, JPEGor GIF files. The following code will insert an image called **myimage**:



```
\begin{figure}[h]
\centering
\includegraphics[width=1\textwidth]{myimage}
\caption{Here is my image}
\label{image-myimage}
\end{figure}
```

- **[h]** is the placement specifier. h means put the figure approximately here.
- Other options are **t** (at the top of the page),**b** (at the bottom of the page)and **p** (on a separate page for figures).
- \centering centres the image on the page, if not used images are left-aligned by default.

To insert an image in to your LaTeXdocument, which requires the graphicx package. Images should be PDF, PNG, JPEGor GIF files. The following code will insert an image called **myimage**:

```
\begin{figure}[h]
\centering
\includegraphics[width=1\textwidth]{myimage}
\caption{Here is my image}
\label{image-myimage}
\end{figure}
```

- **[h]** is the placement specifier. h means put the figure approximately here.
- Other options are **t** (at the top of the page),**b** (at the bottom of the page)and **p** (on a separate page for figures).
- \centering centres the image on the page, if not used images are left-aligned by default.It's a good idea to use this as the figure captions are centred.

To insert an image in to your LaTeXdocument, which requires the graphicx package. Images should be PDF, PNG, JPEGor GIF files. The following code will insert an image called **myimage**:

```
\begin{figure}[h]
\centering
\includegraphics[width=1\textwidth]{myimage}
\caption{Here is my image}
\label{image-myimage}
\end{figure}
```

- **[h]** is the placement specifier. h means put the figure approximately here.
- Other options are **t** (at the top of the page),**b** (at the bottom of the page)and **p** (on a separate page for figures).
- \centering centres the image on the page, if not used images are left-aligned by default.It's a good idea to use this as the figure captions are centred.
- \includegraphics{...}is the command that actually puts the image in your document.

To insert an image in to your LaTeXdocument, which requires the graphicx package. Images should be PDF, PNG, JPEGor GIF files. The following code will insert an image called **myimage**:

```
\begin{figure}[h]
\centering
\includegraphics[width=1\textwidth]{myimage}
\caption{Here is my image}
\label{image-myimage}
\end{figure}
```

- **[h]** is the placement specifier. h means put the figure approximately here.
- Other options are **t** (at the top of the page),**b** (at the bottom of the page)and **p** (on a separate page for figures).
- \centering centres the image on the page, if not used images are left-aligned by default.It's a good idea to use this as the figure captions are centred.
- \includegraphics{...}is the command that actually puts the image in your document.The image file should be **saved** in the same folder as the *.tex* file.

- The command [width=1\textwidth]is optional, it specifies the width of the picture.

- The command [width=1\textwidth]is optional, it specifies the width of the picture.You could also use [scale=0.5].

- The command [width=1\textwidth]is optional, it specifies the width of the picture.You could also use [scale=0.5].
- \caption{...}defines a caption for the figure. It will add "Figure" and a number before the caption.

- The command [width=1\textwidth]is optional, it specifies the width of the picture.You could also use [scale=0.5].
- \caption{...}defines a caption for the figure. It will add "Figure" and a number before the caption.
- \label{...}creates a label to allow you to refer to the table or figure in your text.

- The command [width=1\textwidth]is optional, it specifies the width of the picture.You could also use [scale=0.5].
- \caption{...}defines a caption for the figure. It will add "Figure" and a number before the caption.
- \label{...}creates a label to allow you to refer to the table or figure in your text.

Try with an example by choosing a picture on your own.

3.Equations

Inserting Equations

1. You can enter math mode with an opening and closing dollar sign $.

Inserting Equations

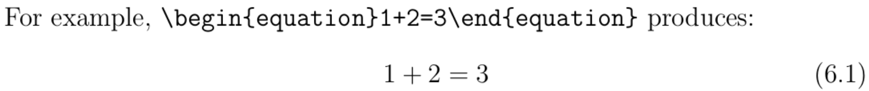
1. You can enter math mode with an opening and closing dollar sign \$.
2. If you want a "displayed" equation on its own line use \$\$...\$\$.

Inserting Equations

1. You can enter math mode with an opening and closing dollar sign $.
2. If you want a "displayed" equation on its own line use $$...$$.
3. For a numbered displayed equation, use \begin{equation}...\end{equation}.

Inserting Equations

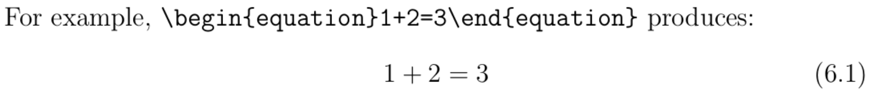
1. You can enter math mode with an opening and closing dollar sign $.
2. If you want a "displayed" equation on its own line use $$...$$.
3. For a numbered displayed equation, use \begin{equation}...\end{equation}.


For example, `\begin{equation}1+2=3\end{equation}` produces:

$$1+2=3 \tag{6.1}$$

Inserting Equations

1. You can enter math mode with an opening and closing dollar sign \$.
2. If you want a "displayed" equation on its own line use \$\$...\$\$.
3. For a numbered displayed equation, use \begin{equation}...\end{equation}.



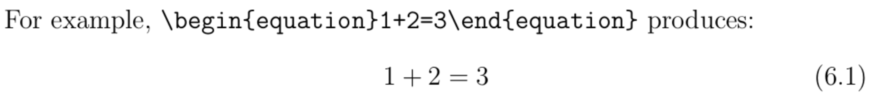
For example, `\begin{equation}1+2=3\end{equation}` produces:

$$1 + 2 = 3 \tag{6.1}$$

4. Use \begin{eqnarray}...\end{eqnarray} to write equation arrays for a series of equations/inequalities.

Inserting Equations

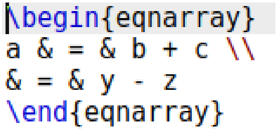
1. You can enter math mode with an opening and closing dollar sign $.
2. If you want a "displayed" equation on its own line use $$...$$.
3. For a numbered displayed equation, use \begin{equation}...\end{equation}.



For example, `\begin{equation}1+2=3\end{equation}` produces:

$$1 + 2 = 3 \tag{6.1}$$

4. Use \begin{eqnarray}...\end{eqnarray} to write equation arrays for a series of equations/inequalities.



```
\begin{eqnarray}
a & = & b + c \\
& = & y - z
\end{eqnarray}
```

Inserting Equations

1. You can enter math mode with an opening and closing dollar sign \$.
2. If you want a "displayed" equation on its own line use \$\$...\$\$.
3. For a numbered displayed equation, use \begin{equation}...\end{equation}.

For example, `\begin{equation}1+2=3\end{equation}` produces:

$$1 + 2 = 3 \tag{6.1}$$

4. Use \begin{eqnarray}...\end{eqnarray} to write equation arrays for a series of equations/inequalities.

Produces:

$$\begin{aligned} a &= b + c \qquad (6.2) \\ &= y - z \qquad (6.3) \end{aligned}$$

Examples

`$$\sum_{x=1}^5 y^z$$` produces:

$$\sum_{x=1}^5 y^z$$

Examples

`$$\sum_{x=1}^5 y^z$$` produces:

$$\sum_{x=1}^5 y^z$$

`$$\int_a^b f(x)$$` produces:

$$\int_a^b f(x)$$

Examples

`$$\sum_{x=1}^5 y^z$$` produces:

$$\sum_{x=1}^{5} y^z$$

`$$\int_a^b f(x)$$` produces:

$$\int_a^b f(x)$$

`$$\frac{a}{3}$$` produces:

$$\frac{a}{3}$$

Examples

`$$\sum_{x=1}^5 y^z$$` produces:

$$\sum_{x=1}^5 y^z$$

`$$\int_a^b f(x)$$` produces:

$$\int_a^b f(x)$$

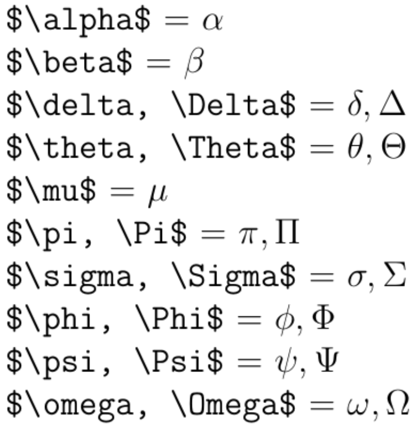
`$$\frac{a}{3}$$` produces:

$$\frac{a}{3}$$

`$$\frac{y}{\frac{3}{x}+b}$$` produces:

$$\frac{y}{\frac{3}{x}+b}$$

Greek symbols



`$\alpha$` $= \alpha$
`$\beta$` $= \beta$
`$\delta, \Delta$` $= \delta, \Delta$
`$\theta, \Theta$` $= \theta, \Theta$
`$\mu$` $= \mu$
`$\pi, \Pi$` $= \pi, \Pi$
`$\sigma, \Sigma$` $= \sigma, \Sigma$
`$\phi, \Phi$` $= \phi, \Phi$
`$\psi, \Psi$` $= \psi, \Psi$
`$\omega, \Omega$` $= \omega, \Omega$

Exercises

➲ Write code to produce the following equations:

$$e = mc^2 \tag{6.1}$$

$$\pi = \frac{c}{d} \tag{6.2}$$

$$\frac{d}{dx}e^x = e^x \tag{6.3}$$

$$\frac{d}{dx}\int_0^\infty f(s)ds = f(x) \tag{6.4}$$

$$f(x) = \sum_i = 0^\infty \frac{f^{(i)}(0)}{i!}x^i \tag{6.5}$$

$$x = \sqrt{\frac{x_i}{z}y} \tag{6.6}$$

1.Index and several hints

Index table

The standard subject index is created using the following procedure:

Index table

The standard subject index is created using the following procedure:

1. Include \index{entry} commands wherever you want an index entry.

Index table

The standard subject index is created using the following procedure:

1. Include \index{entry} commands wherever you want an index entry.
2. Include \usepackage{makeidx}and \makeindex in the preamble.

Index table

The standard subject index is created using the following procedure:

1. Include \index{entry} commands wherever you want an index entry.
2. Include \usepackage{makeidx}and \makeindex in the preamble.
3. Put a \printindex command where the index is to appear, normally before the \end{document} command.

Footnote

- **Footnotes** are a very useful way of providing extra information to the reader.

Footnote

- **Footnotes** are a very useful way of providing extra information to the reader.
- Usually, it is non-essential information which can be placed at the bottom of the page.This keeps the main body of text concise.

Footnote

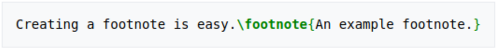
- **Footnotes** are a very useful way of providing extra information to the reader.
- Usually, it is non-essential information which can be placed at the bottom of the page.This keeps the main body of text concise.
- The command you need is: \footnote{text}.

Footnote

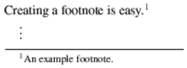
- **Footnotes** are a very useful way of providing extra information to the reader.
- Usually, it is non-essential information which can be placed at the bottom of the page.This keeps the main body of text concise.
- The footnote facility is easy to use. The command you need is: \footnote{text}.
- Do not leave a **space** between the command and the word where you wish the footnote marker to appear, otherwise LATEXwill process that space and will leave the output not looking as intended.

Footnote

- **Footnotes** are a very useful way of providing extra information to the reader.
- Usually, it is non-essential information which can be placed at the bottom of the page.This keeps the main body of text concise.
- The footnote facility is easy to use. The command you need is: \footnote{text}.
- Do not leave a **space** between the command and the word where you wish the footnote marker to appear, otherwise LATEXwill process that space and will leave the output not looking as intended.



```
Creating a footnote is easy.\footnote{An example footnote.}
```



Creating a footnote is easy.[1]

⋮

[1]An example footnote.

List of Tables and Figures

A list of the tables and figures keep the **information organized** and provide **easy access** to a specific element.

List of Tables and Figures

A list of the tables and figures keep the **information organized** and provide **easy access** to a specific element.

- Include \graphicspath{{figures/} } in the preamble.

List of Tables and Figures

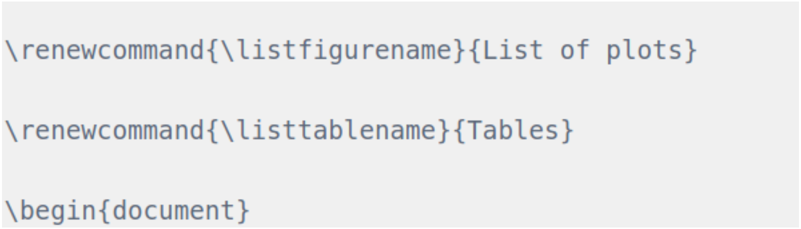
A list of the tables and figures keep the **information organized** and provide **easy access** to a specific element.

- Include \graphicspath{{figures/} } in the preamble.
- Include \listoffigures and \listoftables commands where the lists will appear, normally before the \end{document} command.

List of Tables and Figures

A list of the tables and figures keep the **information organized** and provide **easy access** to a specific element.

- Include \graphicspath{{figures/} } in the preamble.
- Include \listoffigures and \listoftables commands where the lists will appear, normally before the \end{document} command.
- You can **personalize** the name of theses lists as follows:



```
\renewcommand{\listfigurename}{List of plots}

\renewcommand{\listtablename}{Tables}

\begin{document}
```

2.References

Bibliograhy

- **LaTeX** includes features that allow you to easily cite references and create bibliographies in your document.

Bibliograhy

- **LaTeX** includes features that allow you to easily cite references and create bibliographies in your document.
- We need to use a separate BibTeX file to store the details of your references.

Bibliograhy

- **LaTeX** includes features that allow you to easily cite references and create bibliographies in your document.
- We need to use a separate BibTeX file to store the details of your references.
- Your **BibTeX** file contains all the references you want to cite in your document.

Bibliograhy

- **LaTeX** includes features that allow you to easily cite references and create bibliographies in your document.
- We need to use a separate BibTeX file to store the details of your references.
- Your **BibTeX** file contains all the references you want to cite in your document.
- It has the file extension .bib.

Bibliograhy

- **LaTeX** includes features that allow you to easily cite references and create bibliographies in your document.
- We need to use a separate BibTeX file to store the details of your references.
- Your **BibTeX** file contains all the references you want to cite in your document.
- It has the file extension .bib.
- It should have the same name as and kept in the same folder as your .tex file.

Bibliograhy

- **LaTeX** includes features that allow you to easily cite references and create bibliographies in your document.
- We need to use a separate BibTeX file to store the details of your references.
- Your **BibTeX** file contains all the references you want to cite in your document.
- It has the file extension .bib.
- It should have the same name as and kept in the same folder as your .tex file.

```
@article{
Birdetal2001,
   Author = {Bird, R. B. and Smith, E. A. and Bird, D. W.},
   Title = {The hunting handicap: costly signaling in human
   foraging strategies},
   Journal = {Behavioral Ecology and Sociobiology},
   Volume = {50},
   Pages = {9-19},
   Year = {2001} }
```

Inserting the bibliography

Type the following where you want the bibliography to appear in your doc- ument (usually at the end):

Inserting the bibliography

Type the following where you want the bibliography to appear in your doc- ument (usually at the end):

```
\bibliographystyle{plain}
\bibliography{Doc1}
```

Inserting the bibliography

Type the following where you want the bibliography to appear in your doc- ument (usually at the end):

```
\bibliographystyle{plain}
\bibliography{Doc1}
```

Where Doc1 is the name of your .bib file.

Citing references

- Type \cite{citationkey} where you want to cite a reference in your .tex document.
- If you donâĂŹt want an in text citation, but still want the reference to appear in the bibliography, use \nocite{citationkey}.
- To cite multiple references include all the citation keys within the curly brackets separated by commas: \cite{citation01,citation02,citation03}.

Outline

Session XXV

1. TiKz package, draw with LaTeX

1.IFigures, Grid, axis, and graph of functions.

Declare and use TiKz.

TikZ is a package in LaTeX, then it has to be declare in the preambule by adding the following instruction:

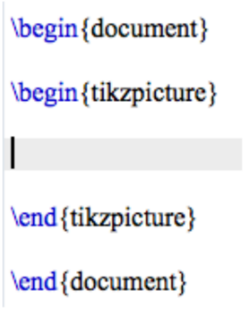
```
\usepackage{pgf, tikz}
```

Declare and use TiKz.

TikZ is a package in LaTeX, then it has to be declare in the preambule by adding the following instruction:

```
\usepackage{pgf, tikz}
```

The future instructions will be written inside the domain of TiKz as we can see:



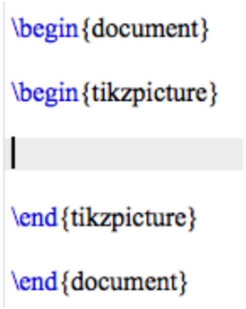
```
\begin{document}

\begin{tikzpicture}



\end{tikzpicture}

\end{document}
```

Declare and use TiKz.

TikZ is a package in LATEX, then it has to be declare in the preambule by adding the following instruction:

```
\usepackage{pgf, tikz}
```

The future instructions will be written inside the domain of TiKz as we can see:



```
\begin{document}

\begin{tikzpicture}



\end{tikzpicture}

\end{document}
```

If you are working with TexMaker, you can find some shortcuts on the left by using the button: **TI**.

The instruction \draw

TiKz works with cartesian or polar coordinates.

The instruction \draw

TiKz works with cartesian or polar coordinates.

Start with the first drawing:

```
\draw (0, 0) -- (4, 0);
```

The instruction \draw

TiKz works with cartesian or polar coordinates.

Start with the first drawing:

```
\draw (0, 0) -- (4, 0);
```

This will give you a line segment that joins the coordinates $(0, 0)$ and $(4, 0)$.

The instruction \draw

TiKz works with cartesian or polar coordinates.

Start with the first drawing:

```
\draw (0, 0) -- (4, 0);
```

This will give you a line segment that joins the coordinates $(0, 0)$ and $(4, 0)$.

Note that each instruction must finish with ;

Square

To draw a square, the following instruction is used:

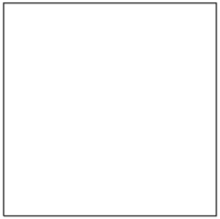
```
\draw (0, 0) -- (4, 0) -- (4, 4) -- (0, 4) -- (0, 0);
```

Square

To draw a square, the following instruction is used:

```
\draw (0, 0) -- (4, 0) -- (4, 4) -- (0, 4) -- (0, 0);
```

This will give us:

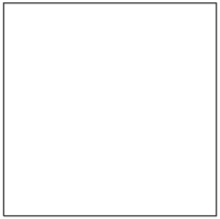


Square

To draw a square, the following instruction is used:

```
\draw (0, 0) -- (4, 0) -- (4, 4) -- (0, 4) -- (0, 0);
```

This will give us:



The following command is equivalent to the previous one and gives the same result:

```
\draw (0, 0) -- (4, 0) -- (4, 4) -- (0, 4) -- cycle;
```

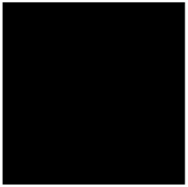
The advantages of the instruction cycle is not in reducing the number of words but in being able to use the instruction \fill and color the inner surface.

The advantages of the instruction cycle is not in reducing the number of words but in being able to use the instruction \fill and color the inner surface.

```
\fill (0, 0) -- (4, 0) -- (4, 4) -- (0, 4) -- cycle;
```

The advantages of the instruction cycle is not in reducing the number of words but in being able to use the instruction \fill and color the inner surface.

```
\fill (0, 0) -- (4, 0) -- (4, 4) -- (0, 4) -- cycle;
```



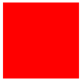
The advantages of the instruction cycle is not in reducing the number of words but in being able to use the instruction \fill and color the inner surface.

```
\fill (0, 0) -- (4, 0) -- (4, 4) -- (0, 4) -- cycle;
```

To change the color, it is sufficient to add [ ] after the command \fill and to specify the color.

```
\fill [red] (0,0)--(1,0)--(1,1)-- (0,1)--cycle;
```

and the result is the following:

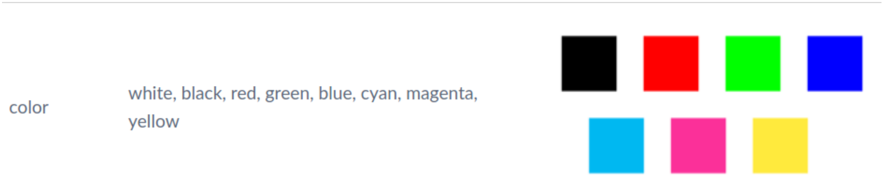


and the result is the following:

Here is a list of the possible colors:

and the result is the following:

Here is a list of the possible colors:



color	white, black, red, green, blue, cyan, magenta, yellow	⬛ 🟥 🟩 🟦 🟦 🟪 🟨

More ideas

It is possible to draw several figures in the same domain as we can see:

More ideas

It is possible to draw several figures in the same domain as we can see:

```
\draw (0, 0) rectangle (8, 6);
\draw (0, 0) parabola (8, 6);
```

More ideas

It is possible to draw several figures in the same domain as we can see:

```
\draw (0, 0) rectangle (8, 6);
\draw (0, 0) parabola (8, 6);
```

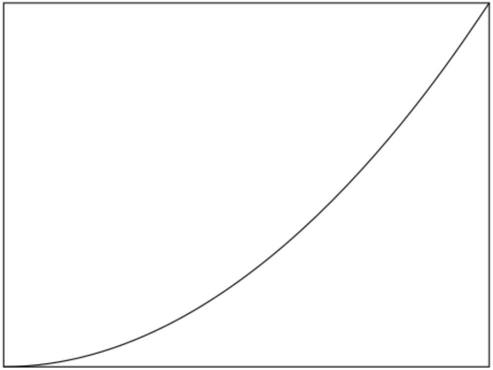
and we get:

More ideas

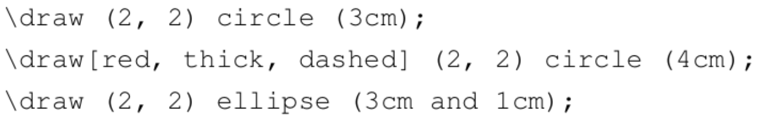
It is possible to draw several figures in the same domain as we can see:

```
\draw (0, 0) rectangle (8, 6);
\draw (0, 0) parabola (8, 6);
```

and we get:



Color, thickness and style



```
\draw (2, 2) circle (3cm);
\draw[red, thick, dashed] (2, 2) circle (4cm);
\draw (2, 2) ellipse (3cm and 1cm);
```

This can be a good example to show you how can we modify the color, the thickness and the styles.

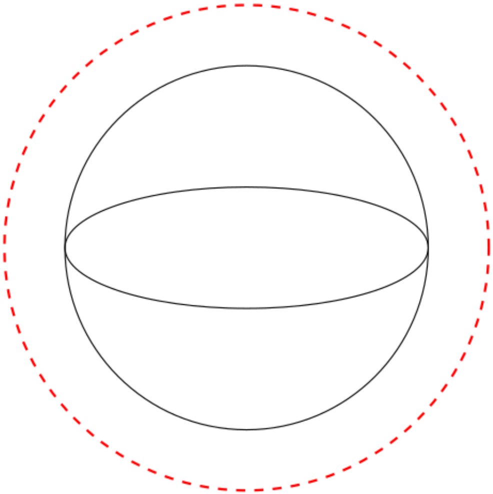
Color, thickness and style

```
\draw (2, 2) circle (3cm);
\draw[red, thick, dashed] (2, 2) circle (4cm);
\draw (2, 2) ellipse (3cm and 1cm);
```

This can be a good example to show you how can we modify the color, the thickness and the styles.

1. The first command draws a circle of center $(2, 2)$ and radius 3.
2. The second gives a red dashed circle with same center but of radius 4.
3. The third is an ellipse with big axis made of 3cm and the small one of 1.

This is the result:



Try to modify the colors using this list:

$$red \mid green \mid blue \mid cyan \mid yellow \mid magenta \mid black \mid white \mid gray$$

Try to modify the colors using this list:

$$red \mid green \mid blue \mid cyan \mid yellow \mid magenta \mid black \mid white \mid gray$$

the thickness:

$$ultrathin \mid verythin \mid thin \mid thick \mid verythick \mid ultrathick$$

Try to modify the colors using this list:

$$red \mid green \mid blue \mid cyan \mid yellow \mid magenta \mid black \mid white \mid gray$$

the thickness:

$$ultrathin \mid verythin \mid thin \mid thick \mid verythick \mid ultrathick$$

and the style.

$$\begin{array}{c|c|c} dotted & looselydotted & denselydotted \\ dashed & looselydashed & denselydashed \end{array}$$

Arcs

The following command shows us how to create an arc:

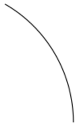
```
\draw (3, 0) arc (0: 60: 3cm);
```

Arcs

The following command shows us how to create an arc:

```
\draw (3, 0) arc (0: 60: 3cm);
```

It gives us:



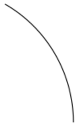
which is an arc starting from point$(3, 0)$ at angle 0 to 60 in a circle of radius 3.

Arcs

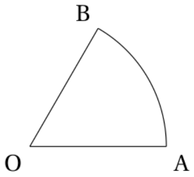
The following command shows us how to create an arc:

```
\draw (3, 0) arc (0: 60: 3cm);
```

It gives us:



which is an arc starting from point$(3, 0)$ at angle 0 to 60 in a circle of radius 3.
In furthest section we wil be able to draw:



Grid

The instruction

```
\draw[step=1cm, gray, very thin] (-2, -2) grid (6, 6);
```

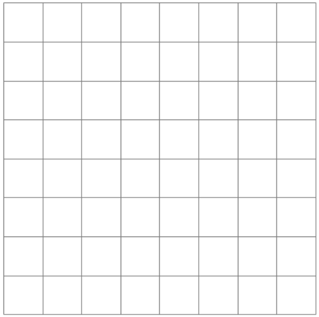
gives a grid that starts at the point $(-2, -2)$ on the bottom leftmost side, and reaches the point $(6, 6)$ on the top rightmost side as we can see in this figure:

Grid

The instruction

```
\draw[step=1cm, gray, very thin] (-2, -2) grid (6, 6);
```

gives a grid that starts at the point $(-2,-2)$ on the bottom leftmost side, and reachs the point $(6,6)$ on the top rightmost side as we can see in this figure:



Grid

The instruction

```
\draw[step=1cm, gray, very thin] (-2, -2) grid (6, 6);
```

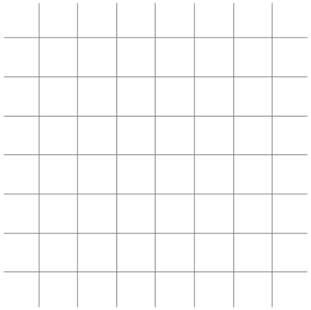
gives a grid that starts at the point $(-2,-2)$ on the bottom leftmost side, and reachs the point $(6,6)$ on the top rightmost side as we can see in this figure:
If we replace $(-2,-2)$ by $(-1.9,1.9)$ and $(6,6)$ by $(5.9,5.9)$, we obtain the following grid:

Grid

The instruction

```
\draw[step=1cm, gray, very thin] (-2, -2) grid (6, 6);
```

gives a grid that starts at the point $(-2, -2)$ on the bottom leftmost side, and reaches the point $(6, 6)$ on the top rightmost side as we can see in this figure:
If we replace $(-2, -2)$ by $(-1.9, 1.9)$ and $(6, 6)$ by $(5.9, 5.9)$, we obtain the following grid:



Axes

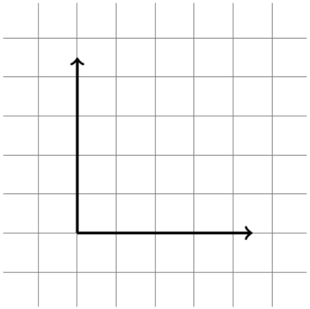
By adding to the previous command this new instruction:

```
\draw[very thick, ->] (0, 0) -- (4.5, 0);
```

and

```
\draw[very thick, ->] (0, 0) -- (0, 4.5);
```

, we get the traditional coordinates axis.



We can also annotate the axis at a certain position by using one of the following keywords:

above	*below*	*right*	*left*
aboveleft	*aboveright*	*belowleft*	*belowright*

We can also annotate the axis at a certain position by using one of the following keywords:

above	*below*	*right*	*left*
aboveleft	*aboveright*	*belowleft*	*belowright*

For example if we write:

```
\draw[very thick, ->] (0, 0) -- (4.5, 0) node[below]{axe x};
\draw[very thick, ->] (0, 0) -- (0, 4.5) node[left]{axe y};
```

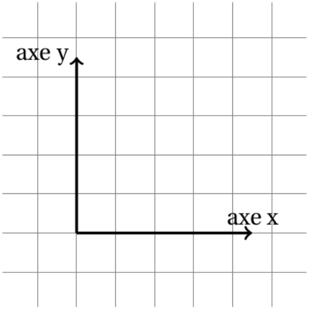
We can also annotate the axis at a certain position by using one of the following keywords:

above	*below*	*right*	*left*
aboveleft	*aboveright*	*belowleft*	*belowright*

For example if we write:

```
\draw[very thick, ->] (0, 0) -- (4.5, 0) node[below]{axe x};
\draw[very thick, ->] (0, 0) -- (0, 4.5) node[left]{axe y};
```

we get



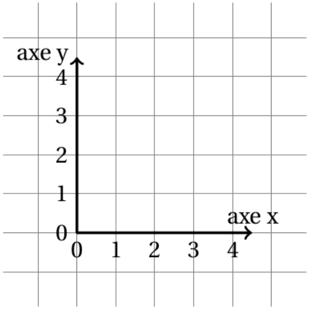
We can also annotate the axis at a certain position by using one of the following keywords:

above	*below*	*right*	*left*
aboveleft	*aboveright*	*belowleft*	*belowright*

For example if we write:

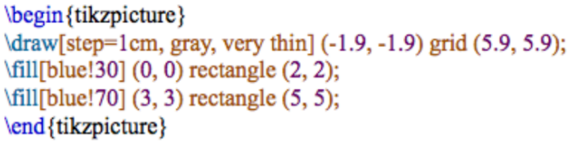
```
\draw[very thick, ->] (0, 0) -- (4.5, 0) node[below]{axe x};
\draw[very thick, ->] (0, 0) -- (0, 4.5) node[left]{axe y};
```

What is left and we will se it later on is how to graduate our axis:

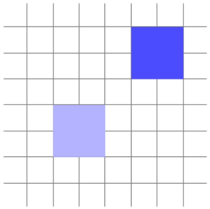


Management of color

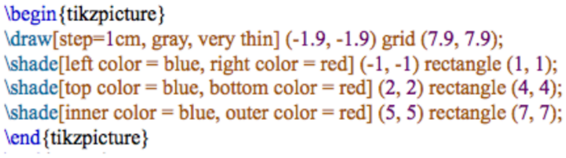
The instruction \fill[options] helps specify the position of the square, the degree of its color using (). Note that the degree can go from $1 - - > 100$ as we can see:



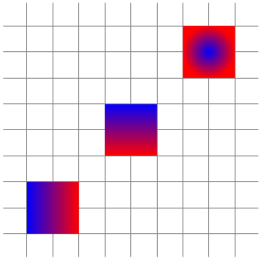
```
\begin{tikzpicture}
\draw[step=1cm, gray, very thin] (-1.9, -1.9) grid (5.9, 5.9);
\fill[blue!30] (0, 0) rectangle (2, 2);
\fill[blue!70] (3, 3) rectangle (5, 5);
\end{tikzpicture}
```



We can also do some geometric shading:



```
\begin{tikzpicture}
\draw[step=1cm, gray, very thin] (-1.9, -1.9) grid (7.9, 7.9);
\shade[left color = blue, right color = red] (-1, -1) rectangle (1, 1);
\shade[top color = blue, bottom color = red] (2, 2) rectangle (4, 4);
\shade[inner color = blue, outer color = red] (5, 5) rectangle (7, 7);
\end{tikzpicture}
```



1.Graphs of functions.

Functions

As in Pythpon, in order to draw functions, we need to start to write them in a parametric function.
For example, in order to draw the functions: $f(x) = x$, $g(x) = sin(x)$ and $h(x) = cos(x)$ we must write them as:

$$\begin{cases} x = & x \\ y = & 2x \end{cases}$$

,

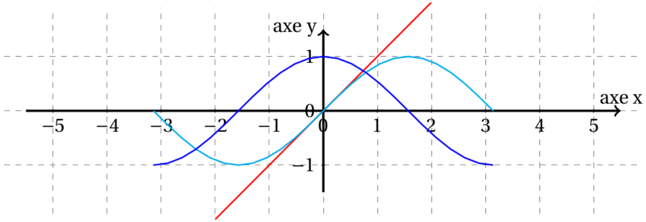
$$\begin{cases} x = & t \\ y = & sin(t) \end{cases}$$

and

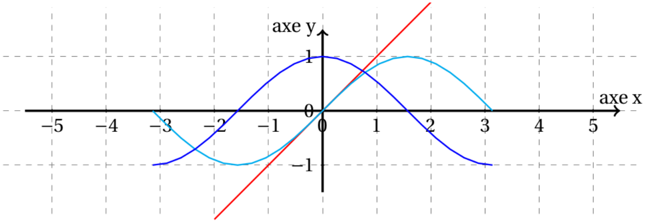
$$\begin{cases} x = & t \\ y = & cos(t) \end{cases}$$

Try to write the code in python to get these functions and we will see how to code it with TiKz.

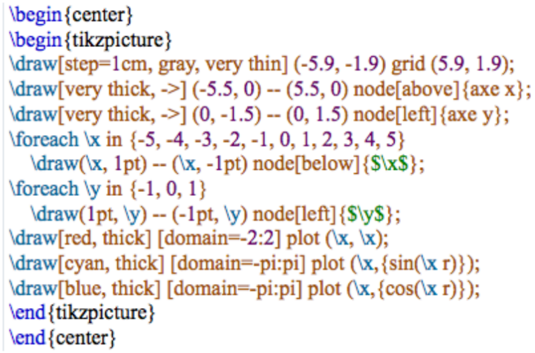
In order to get this graph:



In order to get this graph:



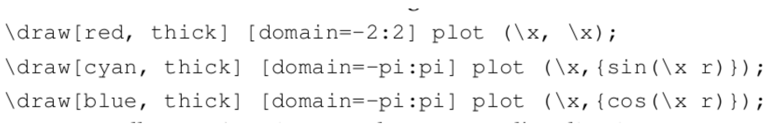
We need this code



```
\begin{center}
\begin{tikzpicture}
\draw[step=1cm, gray, very thin] (-5.9, -1.9) grid (5.9, 1.9);
\draw[very thick, ->] (-5.5, 0) -- (5.5, 0) node[above]{axe x};
\draw[very thick, ->] (0, -1.5) -- (0, 1.5) node[left]{axe y};
\foreach \x in {-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5}
   \draw(\x, 1pt) -- (\x, -1pt) node[below]{$\x$};
\foreach \y in {-1, 0, 1}
   \draw(1pt, \y) -- (-1pt, \y) node[left]{$\y$};
\draw[red, thick] [domain=-2:2] plot (\x, \x);
\draw[cyan, thick] [domain=-pi:pi] plot (\x,{sin(\x r)});
\draw[blue, thick] [domain=-pi:pi] plot (\x,{cos(\x r)});
\end{tikzpicture}
\end{center}
```

Explanation

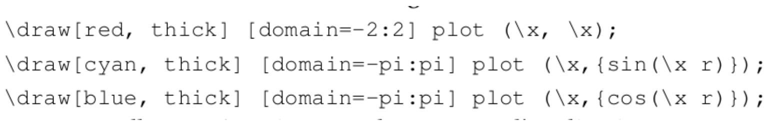
These three lines need to be explained



```
\draw[red, thick] [domain=-2:2] plot (\x, \x);
\draw[cyan, thick] [domain=-pi:pi] plot (\x,{sin(\x r)});
\draw[blue, thick] [domain=-pi:pi] plot (\x,{cos(\x r)});
```

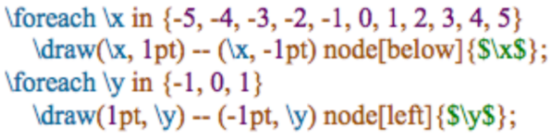
Explanation

These three lines need to be explained



```
\draw[red, thick] [domain=-2:2] plot (\x, \x);
\draw[cyan, thick] [domain=-pi:pi] plot (\x,{sin(\x r)});
\draw[blue, thick] [domain=-pi:pi] plot (\x,{cos(\x r)});
```

Same thing for these two



```
\foreach \x in {-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5}
   \draw(\x, 1pt) -- (\x, -1pt) node[below]{$\x$};
\foreach \y in {-1, 0, 1}
   \draw(1pt, \y) -- (-1pt, \y) node[left]{$\y$};
```

1. The part $[domain = -2 : 2]$ gives the domain of definition of the function defined.

1. The part $[domain = -2 : 2]$ gives the domain of definition of the function defined.
2. The command $plot(\backslash x, \backslash x)$ is used to draw the function $f(x) = x$. Same thing for the other functions.

1. The part $[domain = -2 : 2]$ gives the domain of definition of the function defined.

2. The command $plot(\backslash x, \backslash x)$ is used to draw the function $f(x) = x$. Same thing for the other functions.

3. While for the command $cos(\backslash x\ r)$, the letter r is needed because normally the trigonometrical functions expect values in degree, while we normally work with radian. That's why we add the letter r to transform from radian to degree so we can obtain $sin(\pi/2\ r) = 1$.

1. The part $[domain = -2 : 2]$ gives the domain of definition of the function defined.

2. The command $plot(\backslash x, \backslash x)$ is used to draw the function $f(x) = x$. Same thing for the other functions.

3. While for the command $cos(\backslash x\ r)$, the letter r is needed because normally the trigonometrical functions expect values in degree, while we normally work with radian. That's why we add the letter r to transform from radian to degree so we can obtain $sin(\pi/2\ r) = 1$.

4. The command $\backslash foreach$ makes a for loop to put all the coordinates over the x and $y-$ axis in one time, instead of doing it step by step.

Note that!!

The power and exponential are mathematical functions that can not be computed using LaTeX.

Note that!!

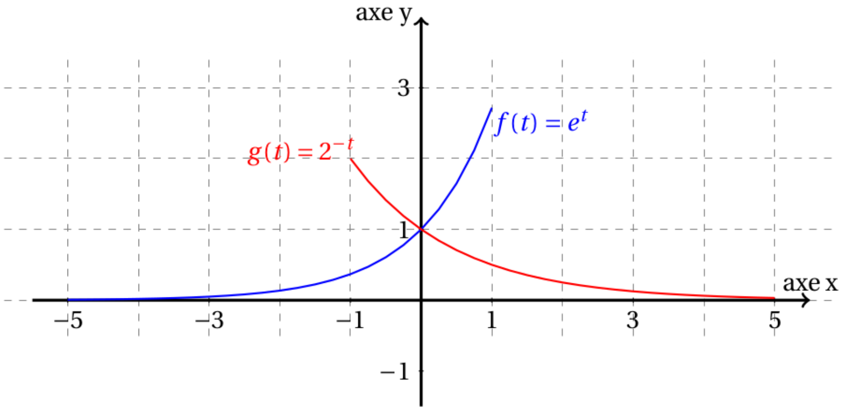
The power and exponential are mathematical functions that can not be computed using LaTeX.
To get x^2, we express it by typing $\backslash x * \backslash x$.

Note that!!

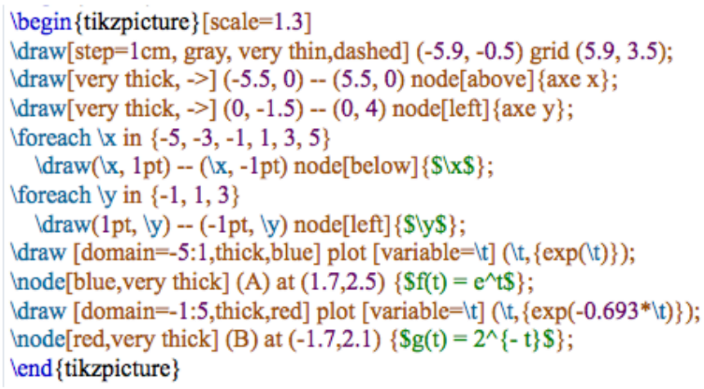
The power and exponential are mathematical functions that can not be computed using LaTeX.
To get x^2, we express it by typing $\backslash x * \backslash x$.
While e^x is expressed by $exp(\backslash x)$.

Note that!!

The power and exponential are mathematical functions that can not be computed using LaTeX.
To get x^2, we express it by typing $\backslash x * \backslash x$.
While e^x is expressed by $exp(\backslash x)$.Try to write the code in order to get the following functions/



Solution



```
\begin{tikzpicture}[scale=1.3]
\draw[step=1cm, gray, very thin,dashed] (-5.9, -0.5) grid (5.9, 3.5);
\draw[very thick, ->] (-5.5, 0) -- (5.5, 0) node[above]{axe x};
\draw[very thick, ->] (0, -1.5) -- (0, 4) node[left]{axe y};
\foreach \x in {-5, -3, -1, 1, 3, 5}
   \draw(\x, 1pt) -- (\x, -1pt) node[below]{$\x$};
\foreach \y in {-1, 1, 3}
   \draw(1pt, \y) -- (-1pt, \y) node[left]{$\y$};
\draw [domain=-5:1,thick,blue] plot [variable=\t] (\t,{exp(\t)});
\node[blue,very thick] (A) at (1.7,2.5) {$f(t) = e^t$};
\draw [domain=-1:5,thick,red] plot [variable=\t] (\t,{exp(-0.693*\t)});
\node[red,very thick] (B) at (-1.7,2.1) {$g(t) = 2^{- t}$};
\end{tikzpicture}
```

This is a list of several mathematical functions:

$$abs(x), exp(x), ln(x), sqrt(x), sin(x), cos(x), tan(x), cot(x), sec(x), cosec(x), asin(x), acos(x), atan(x).$$

This is a list of several mathematical functions:

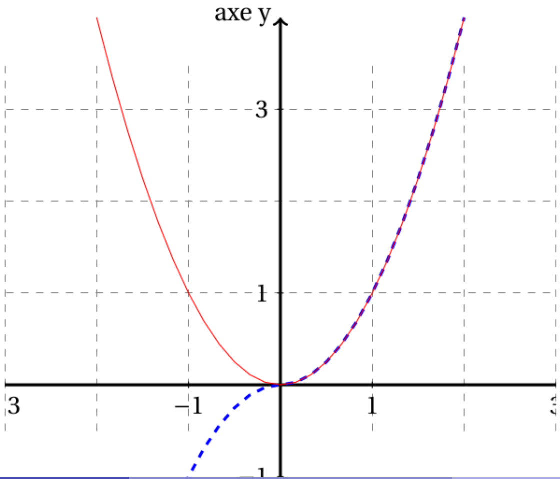
$$abs(x), exp(x), ln(x), sqrt(x), sin(x), cos(x), tan(x), cot(x), sec(x), cosec(x), asin(x), acos(x), atan(x).$$

Try to draw the function t^2 in dotted blue line and $\backslash t * \backslash t$ in red to see the difference.

This is a list of several mathematical functions:

$$abs(x), exp(x), ln(x), sqrt(x), sin(x), cos(x), tan(x), cot(x), sec(x), cosec(x), asin(x), acos(x), atan(x).$$

Try to draw the function t^2 in dotted blue line and $\backslash t * \backslash t$ in red to see the difference.

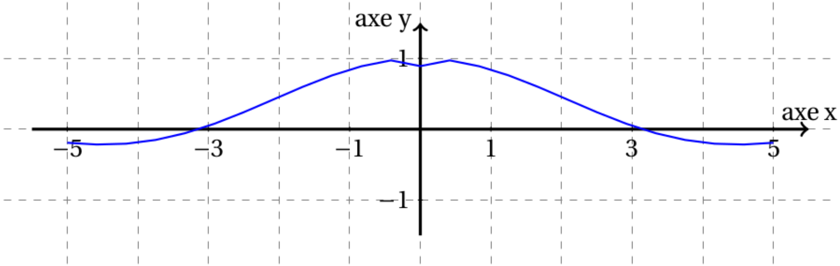


Exercise

Try to plot the function $\frac{sin(x)}{x}$ in the domain $[-5, 5]$.

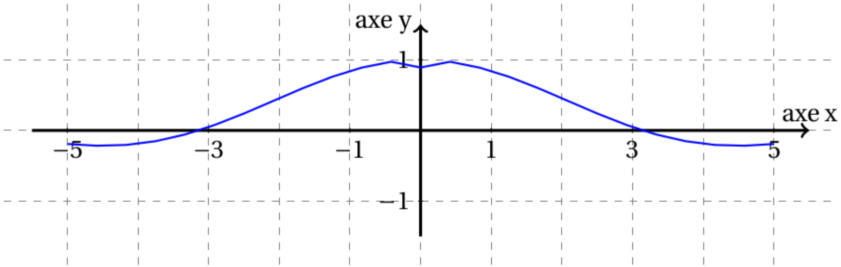
Exercise

Try to plot the function $\frac{sin(x)}{x}$ in the domain $[-5, 5]$.



Exercise

Try to plot the function $\frac{sin(x)}{x}$ in the domain $[-5, 5]$.



There is a problem at point 0 since the number of points studied are not enough to get a precise graph.

If we cahnge the line:

```
\draw [domain=-5:5] plot (\x,{sin(\x r)/\x});
```

by

```
\draw [domain=-5:5,samples=200] plot (\x,{sin(\x r)/\x});
```

,

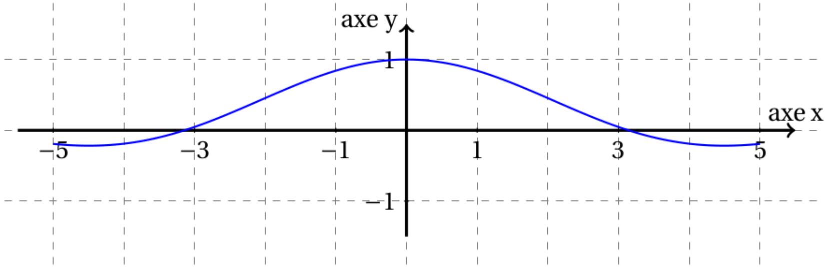
If we cahnge the line:

```
\draw [domain=-5:5] plot (\x,{sin(\x r)/\x});
```

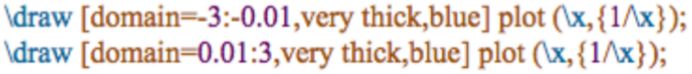
by

```
\draw [domain=-5:5,samples=200] plot (\x,{sin(\x r)/\x});
```

,we get:



Draw the function $f(x) = \frac{1}{x}$ by dividing the domain into:

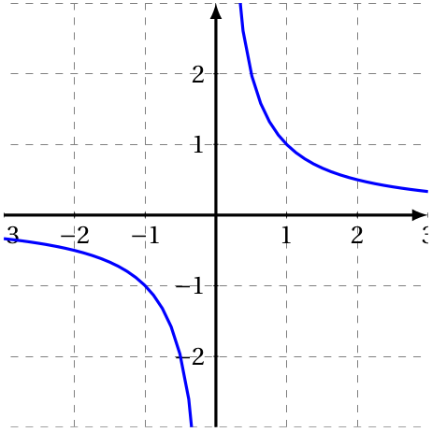


```
\draw [domain=-3:-0.01,very thick,blue] plot (\x,{1/\x});
\draw [domain=0.01:3,very thick,blue] plot (\x,{1/\x});
```

Draw the function $f(x) = \frac{1}{x}$ by dividing the domain into:

```
\draw [domain=-3:-0.01,very thick,blue] plot (\x,{1/\x});
\draw [domain=0.01:3,very thick,blue] plot (\x,{1/\x});
```

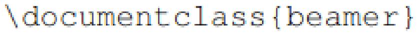
in order to obtain:



2.Beamer

The Header

All you really need in you header is a line like this:



```
\documentclass{beamer}
```

The Header

All you really need in you header is a line like this:

```
\documentclass{beamer}
```

However that will leave it looking very bare so you may want toalso add the line:

```
\usetheme{Malmoe}
```

The Header

All you really need in you header is a line like this:

```
\documentclass{beamer}
```

However that will leave it looking very bare so you may want toalso add the line:

```
\usetheme{Malmoe}
```

Also you can add most any other packages and macros youwould usually use in the standard way:

```
\usepackage{amsthm}
\usepackage{times}
\usepackage{graphicx}
...
```

Change the theme!

$$\backslash usetheme\{\ldots\}$$
$$\backslash usecolortheme\{\ldots\}$$

All the possible themes and color themes can be found in:

Beamer Theme Matrix

Contextual Information in the Header

You will probably also want to add in some biographic notes inthe header like

$\backslash title\{A\ Banquet\ of\ \{\backslash sc\ Beamer\} Basics\}$
$\backslash author\{Lama Tarsissi\}$
$\backslash date\{23/11/2020\}$

The frame

The main structure of a presentation is just the slide which is called a frame in Beamer.The simplest way to create a frame is just:

$$\backslash begin\{frame\}\{FRAMETITLE\}$$

$$content$$

$$\backslash end\{frame\}$$

The frame

The main structure of a presentation is just the slide which is called a frame in Beamer.The simplest way to create a frame is just:

$$\backslash begin\{frame\}\{FRAMETITLE\}$$

$$content$$

$$\backslash end\{frame\}$$

With in a frame you can put almost any regular latex that you like.

The frame

The main structure of a presentation is just the slide which is called a frame in Beamer.The simplest way to create a frame is just:

$$\backslash begin\{frame\}\{FRAMETITLE\}$$

$$content$$

$$\backslash end\{frame\}$$

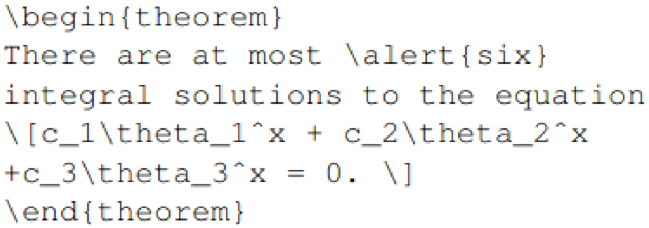
With in a frame you can put almost any regular latex that you like.

Simple Commands

BEAMER has included useful environments like theorem,lemma, proof, definition, corollary and example. Note that anenvironment has to be ended in the same frame it was started.Also included is the alert command to bring extra attention to aword.

More Example

For example the code



```
\begin{theorem}
There are at most \alert{six}
integral solutions to the equation
\[c_1\theta_1^x + c_2\theta_2^x
+c_3\theta_3^x = 0. \]
\end{theorem}
```

More Example

For example the code

```
\begin{theorem}
There are at most \alert{six}
integral solutions to the equation
\[c_1\theta_1^x + c_2\theta_2^x
+c_3\theta_3^x = 0. \]
\end{theorem}
```

looks like

Theorem

There are at most six integral solutions to the equation

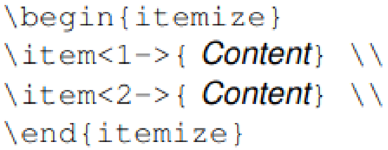
$$c_1\theta_1 x + c_2\theta_2 x + c_3\theta_3 x = 0.$$

More Itemizing

You can create a list where each point shows up separately(this is called showing up in separate overlays) just by using the code.

More Itemizing

You can create a list where each point shows up separately(this is called showing up in separate overlays) just by using the code.



```
\begin{itemize}
\item<1->{ Content}  \\
\item<2->{ Content}  \\
\end{itemize}
```

Pause

However there is really nothing special about an itemize list.
You can stop a frame anywhere you like (almost) with the command:

\pause

Pause

However there is really nothing special about an itemize list.
You can stop a frame anywhere you like (almost) with the command:

$\backslash$*pause*

Note that the pause command does not put in any carriage return or spaces so if you want extra space you had better addit yourself.

Pause

However there is really nothing special about an itemize list.
You can stop a frame anywhere you like (almost) with the command:

$$\backslash pause$$

Note that the pause command does not put in any carriage return or spaces so if you want extra space you had better addit yourself. Be warned that you can not stick a pause in an align environment. When you do funny things start to happen withthe slide.

Pause

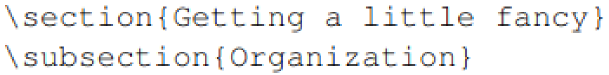
However there is really nothing special about an itemize list.
You can stop a frame anywhere you like (almost) with the command:

\pause

Note that the pause command does not put in any carriage return or spaces so if you want extra space you had better addit yourself. Be warned that you can not stick a pause in an align environment. When you do funny things start to happen withthe slide.
For most presentations these tools will be plenty. However BEAMER can do much more than this so let's explore a few other things we can do.

Section and subsection

You can add the section structure that will be illustrated in the frames. This is simply done by adding section and subsection tags between the frames like so:



```
\section{Getting a little fancy}
\subsection{Organization}
```

Columns

Define a table with two columns

$\backslash begin\{tabular\}\{cc\}$
Content of my first column
&
Content of my second column
$\backslash end\{tabular\}$

Second method

Define two minipages next to each other

```
\begin{minipage}[c]{0.45 \ linewidth}
    Content of my first column
        \ end{minipage}
\ begin{minipage}[c]{0.45 \ linewidth}
   Content of my second column
                   \ end{minipage}
```

1.Beamer.

Columns

Define a table with two columns

$\backslash begin\{tabular\}\{cc\}$

$Content\ of\ my\ first\ column$

$\&$

$Content\ of\ my\ second\ column$

$\backslash\ end\{tabular\}$

Second method

Define two minipages next to each other

```
\begin{minipage}[c]{0.45 \ linewidth}
    Content of my first column
        \ end{minipage}
\ begin{minipage}[c]{0.45 \ linewidth}
   Content of my second column
                    \ end{minipage}
```

Other method!!

```
\begin{columns}
\begin{column}{6cm}
Content of my first column
\end{column}
\begin{column}{6cm}
Content of my second column
\end{column}
\end{columns}
```

Layers and Overlay

Beamer is able to overlay different layers while showing. Here is an example :

Layers and Overlay

Beamer is able to overlay different layers while showing. Here is an example :

- My first element
- Another element that remains

Layers and Overlay

Beamer is able to overlay different layers while showing. Here is an example :

- My first element
- Another element that remains
- A third element that will become bold

Layers and Overlay

Beamer is able to overlay different layers while showing. Here is an example :

- My first element
- Another element that remains
- **A third element that will become bold**
- The end.

The code that gave this, is the following:

```
\begin{itemize}
\item<1> My first element
\item<2-> Another element that remains
\item<3-> \textbf<4>{A third element
   that will become bold}
\item<4> The end.
\end{itemize}
```

The code that gave this, is the following:

```
\begin{itemize}
\item<1> My first element
\item<2-> Another element that remains
\item<3-> \textbf<4>{A third element
   that will become bold}
\item<4> The end.
\end{itemize}
```

Add this sentence to your preambule and check what will happen:
\setbeamercovered{transparent}

One more trick!!!

Instead of showing the elements in several slides, one after the other, we can show them by erasing each element and replacing it. For that we use the command $\backslash only < k > \{command\}$, where k is the number of the slide on which you will get the image. This will give you the following:

One more trick!!!

Instead of showing the elements in several slides, one after the other, we can show them by erasing each element and replacing it. For that we use the command $\backslash only < k > \{command\}$, where k is the number of the slide on which you will get the image. This will give you the following:

One more trick!!!

Instead of showing the elements in several slides, one after the other, we can show them by erasing each element and replacing it. For that we use the command $\backslash only < k > \{command\}$, where k is the number of the slide on which you will get the image. This will give you the following:

One more trick!!!

Instead of showing the elements in several slides, one after the other, we can show them by erasing each element and replacing it. For that we use the command $\backslash only < k > \{command\}$, where k is the number of the slide on which you will get the image. This will give you the following:

The previous code:

```
\only<2>{\begin{center}
\includegraphics[width=0.75\textwidth]{f1}
\end{center}} \pause
\only<3>{\begin{center}
\includegraphics[width=0.65\textwidth]{f2}
\end{center}} \pause
\only<4>{\begin{center}
\includegraphics[width=0.75\textwidth]{f3}
\end{center}}
```

Using Onslide

We can use, with the same syntax, $\backslash$*onslide* $<> \{\}$, that reserves the place of the image when removing it. This gives:

Using Onslide

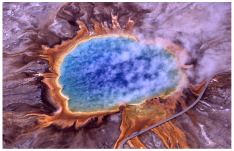
We can use, with the same syntax, $\backslash$*onslide* $<> \{\}$, that reserves the place of the image when removing it. This gives:

Using Onslide

We can use, with the same syntax, $\backslash$*onslide* $<>$ $\{\}$, that reserves the place of the image when removing it. This gives:

Using Onslide

We can use, with the same syntax, *\onslide* <> {}, that reserves the place of the image when removing it. This gives:



The code is the following:

```
\onslide<2>{\begin{center}
\includegraphics[width=0.25\textwidth]{f1}
\end{center}} \pause
\onslide<3>{\begin{center}
\includegraphics[width=0.25\textwidth]{f2}
\end{center}} \pause
\onslide<4>{\begin{center}
\includegraphics[width=0.25\textwidth]{f3}
\end{center}}
```

2.Animation for the presentations.

Animation

- There exists several animations that can be used between two slides.

Animation

- There exists several animations that can be used between two slides.
- In order to use them, we need to use the \trans something... inside of the slide

Animation

- There exists several animations that can be used between two slides.
- In order to use them, we need to use the \trans something... inside of the slide
- You can add several options inside the brackets, like duration= time in seconds, and direction=angle

Animation

- There exists several animations that can be used between two slides.
- In order to use them, we need to use the \trans something... inside of the slide
- You can add several options inside the brackets, like duration= time in seconds, and direction=angle

Normal dissolvation

transdissolve

Fast dissolvation

transdissolve[duration=0.1]

Slow dissolvation

transdissolve[duration=5]

Wiping

transwipe

Wiping in different angle

transwipe[direction=90]

Wiping in inverse direction

transwipe[direction=180]

Here are the different options that we can use with the animation:

Here are the different options that we can use with the animation:

- transblindhorizontal
- transblindvertical
- transboxin
- transboxout
- transglitter
- transsplitverticalin
- transsplitverticalout
- transsplithorizontalin
- transsplithorizontalout

Here are the different options that we can use with the animation:

- transblindhorizontal
- transblindvertical
- transboxin
- transboxout
- transglitter
- transsplitverticalin
- transsplitverticalout
- transsplithorizontalin
- transsplithorizontalout

Finally, the command \tranduration{*timeinsecondes*} allows to make the transition after a specific duration given in seconds. Pay attention You must be very careful while using it! It is so impressive but also delicate.

Outline

Session XXIX - PythonTex

Before using pythontex

Before using pythontex

Create a user command: User ->User Commands->Edit User Commands

Before using pythontex

Create a user command: User ->User Commands->Edit User Commands

On Windows:

```
pdflatex --shell-escape -synctex=1 -interaction=nonstopmode %.tex|
python C:\Users\lama\AppData\Local\Programs\MiKTeX\scripts\pythontex
\pythontex.py %.tex|
pdflatex --shell-escape -synctex=1 -interaction=nonstopmode %.tex|
"C:/Program Files (x86)/Adobe/Acrobat 11.0/Acrobat/Acrobat.exe" %.pdf
```

On Mac OS:

```
pdflatex --shell-escape -synctex=1 -interaction=nonstopmode %.tex|
pythontex %.tex|
pdflatex --shell-escape -synctex=1 -interaction=nonstopmode %.tex|
open %.pdf
```

Using the python console

Let us make a python variable, raise it to the power 2, and show the result in Latex. To do that, create the following LaTeX document.

```
\documentclass[11pt]{article}%
\usepackage{pythontex}
\usepackage{nopageno}
\begin{document}
\begin{pyconsole}
x = 987.27
x = x**2
\end{pyconsole}

The variable is $x=\pycon{x}$
\end{document}
```

Using the python console

Let us make a python variable, raise it to the power 2, and show the result in Latex. To do that, create the following LaTeX document.

```
\documentclass[11pt]{article}%
\usepackage{pythontex}
\usepackage{nopageno}
\begin{document}
\begin{pyconsole}
x = 987.27
x = x**2
\end{pyconsole}

The variable is $x=\pycon{x}$
\end{document}
```

When compiled, we get the following

```
>>> x =987.27
>>> x = x**2
```

The variable is $x = 974702.0529$

Using a python variable inside latex

Let us make a python variable, raise it to the power 2, and show the result in Latex. To do that, write the following document.

```
\documentclass[11pt]{article}%
\usepackage{pythontex}
\usepackage{nopageno}
\begin{document}
\begin{pycode}
x = 987.27
x = x**2
\end{pycode}

The variable is $x=\py{x}$
\end{document}
```

When compiled, we get the following:

Using a python variable inside latex

Let us make a python variable, raise it to the power 2, and show the result in Latex. To do that, write the following document.

```
\documentclass[11pt]{article}%
\usepackage{pythontex}
\usepackage{nopageno}
\begin{document}
\begin{pycode}
x = 987.27
x = x**2
\end{pycode}

The variable is $x=\py{x}$
\end{document}
```

When compiled, we get the following:
The variable is $x = 974702.0529$

Defining a python function

```
\documentclass[11pt]{article}%
\usepackage{pythontex}
\usepackage{nopageno}
\begin{document}
\begin{pycode}

def fib(n):    # nth Fibonacci value
    a, b = 0, 1
    for i in range(n):
        a, b = b, a + b
    return a

\end{pycode}

Did you know that $F_{10} = \py{fib(10)}$?
\end{document}
```

Defining a python function

```
\documentclass[11pt]{article}%
\usepackage{pythontex}
\usepackage{nopageno}
\begin{document}
\begin{pycode}

def fib(n):   # nth Fibonacci value
    a, b = 0, 1
    for i in range(n):
        a, b = b, a + b
    return a

\end{pycode}

Did you know that $F_{10} = \py{fib(10)}$?
\end{document}
```

Did you know that $F_{10} = 55$?

Generating Tables with pycode

```
\documentclass[11pt]{article}%
\usepackage{pythontex}
\usepackage{nopageno}
\begin{document}
\begin{center}
\begin{pycode}

print(r"\begin{tabular}{c|c}")
print(r"$m$ & $2^m$ \\ \hline")
print(r"%d & %d \\" % (1, 2**1))
print(r"%d & %d \\" % (2, 2**2))
print(r"%d & %d \\" % (3, 2**3))
print(r"%d & %d \\" % (4, 2**4))
print(r"\end{tabular}")

\end{pycode}
\end{center}
\end{document}
```

m	2^m
1	2
2	4
3	8
4	16

Generating Tables with a loop

```
\documentclass[11pt]{article}%
\usepackage{pythontex}
\usepackage{nopageno}
\begin{document}
\begin{center}
\begin{pycode}

lo, hi = 1, 6
print(r"\begin{tabular}{c|c}")
print(r"$m$ & $2^m$ \\ \hline")
for m in range(lo, hi + 1):
    print(r"%d & %d \\" % (m, 2**m))
print(r"\end{tabular}")

\end{pycode}
\end{center}
\end{document}
```

m	2^m
1	2
2	4
3	8
4	16
5	32
6	64

Symbolic computation

In this example we will use sympy to do symbolic computation, which is integrating a function, then obtain the latex back of the result.

Symbolic computation

In this example we will use sympy to do symbolic computation, which is integrating a function, then obtain the latex back of the result.

```
\documentclass[11pt]{article}%
\usepackage{pythontex}
\usepackage{nopageno}
\begin{document}
\begin{pycode}
from sympy import *
x=symbols('x')
value=integrate("(1+x)**(1/2)",x)
result = latex(value)
\end{pycode}

The result of integrating $\int \sqrt{ 1+x } dx$ is
   given by $\py{result}$
\end{document}
```

Symbolic computation

In this example we will use sympy to do symbolic computation, which is integrating a function, then obtain the latex back of the result.

```
\documentclass[11pt]{article}%
\usepackage{pythontex}
\usepackage{nopageno}
\begin{document}
\begin{pycode}
from sympy import *
x=symbols('x')
value=integrate("(1+x)**(1/2)",x)
result = latex(value)
\end{pycode}

The result of integrating $\int \sqrt{ 1+x } dx$ is
   given by $\py{result}$
\end{document}
```

The result of integrating $\int \sqrt{1+x} dx$ is given by $\frac{2}{3}(x+1)^{\frac{3}{2}}$

Symbolic computation - Using a function

Another example, this one uses a function:

```
from sympy import *

def int(theIntegrand,var):
    var  = symbols(var)
    anti = integrate(theIntegrand,var)
    return latex(anti)
```

```
 The result of integrating $\int \frac{1}{\sqrt{ 1+x
    }} \, dx$ is given by $\py{int("1/(1+x)**(1/2)","x
    ")}$
```

The result of integrating $\int \frac{1}{\sqrt{1+x}}\, dx$ is given by

Symbolic computation - Try it yourself

Here is some list of integrations to do

$$\int \frac{1}{\sqrt{1+x}}\, dx = 2\sqrt{x+1}$$
$$\int \sin x\, dx = -\cos(x)$$
$$\int x\sin x\, dx = -x\cos(x) + \sin(x)$$
$$\int x^2\sin x\, dx = -x^2\cos(x) + 2x\sin(x) + 2\cos(x)$$
$$\int xe^{2x}\, dx = \frac{e^{2x}}{4}(2x-1)$$
$$\int \frac{1}{1+u}\, du = \log(u+1)$$

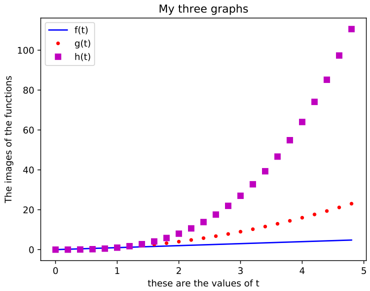
Symbolic computation - Solution

```
\begin{pycode}
from sympy import *
def int(theIntegrand,var):
    var  = symbols(var)
    return latex(integrate(theIntegrand,var))
\end{pycode}

Here is some list of integrations to do
\begin{align*}
\int \frac{1}{\sqrt{ 1+x }} \, dx &=  \py{int("1/(1+x)
   **(1/2)","x")} \\
\int \sin x \, dx &=  \py{int("sin(x)","x")} \\
\int x \sin x \, dx &=  \py{int("x*sin(x)","x")} \\
\int x^2 \sin x \, dx &=  \py{int("x**2 * sin(x)","x")} \\
\int x e^{2 x} \, dx &=  \py{int("x*exp(2*x)","x")} \\
\int \frac{1}{1+u} \, du &=  \py{int("1/(1+u)","u")} \\
\end{align*}
```

This example should be somehow familiar

Draw the following functions:

$$f(t) = t$$
$$g(t) = t^2$$
$$h(t) = t^3$$



Outline

Session XXX - Dicttionary

What is a Collection?

- A collection is nice because we can put more than one value in them and carry them all around in one convenient package.
- We have a bunch of values in a single "variable"
- We do this by having more than one place "in" the variable.
- We have ways of finding the different places in the variable

What is not a "Collection"?

Most of our "variables" have one value in them - when we put a new value in the variable - the old value is over written.

```
>>> x = 2
>>> x = 4
>>> print(x)
```

A story of two Collections

1. List: A linear collection of values that stay in order
2. Dictionary: A "bag" of values, each with its own label

Dictionaries

- Dictionaries are Python's most powerful data collection
- Dictionaries allow us to do fast database-like operations in Python

Dictionaries

- Lists index their entries based on the position in the list
- Dictionaries are like bags - no order
- So we index the things we put in the dictionary with a "lookup tag"

Dictionaries

- Lists index their entries based on the position in the list
- Dictionaries are like bags - no order
- So we index the things we put in the dictionary with a "lookup tag"

```
>>> purse = dict()
>>> purse['money'] = 12
>>> purse['candy'] = 3
>>> purse['tissues'] = 75
>>> print (purse)
{'money': 12, 'tissues': 75, 'candy': 3}
>>> print (purse['candy'])
3
>>> purse['candy'] = purse['candy'] + 2
>>> print (purse)
{'money': 12, 'tissues': 75, 'candy': 5}
```

Comparing Lists and Dictionaries

Dictionaries are like Lists except that they use keys instead of numbers to look up values

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print (lst)
[21, 183]
>>> lst[0] = 23
>>> print (lst)
[23, 183]
```

```
>>> ddd = dict()
>>> ddd['age'] = 21
>>> ddd['course'] = 182
>>> print (ddd)
{'course': 182, 'age': 21}
>>> ddd['age'] = 23
>>> print (ddd)
{'course': 182, 'age': 23}
```

Dictionary Literals (Constants)

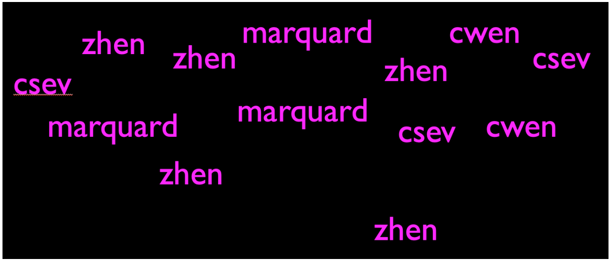
- Dictionary literals use curly braces and have a list of key : value pairs
- You can make an empty dictionary using empty curly braces

Dictionary Literals (Constants)

- Dictionary literals use curly braces and have a list of key : value pairs
- You can make an empty dictionary using empty curly braces

```
>>> jjj = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}
>>> print (jjj)
{'jan': 100, 'chuck': 1, 'fred': 42}

>>> ooo = {}
>>> print (ooo)
>>> {}
```

Most Common Name?


zhen

zhen

marquard

cwen

csev

zhen

csev

marquard

marquard

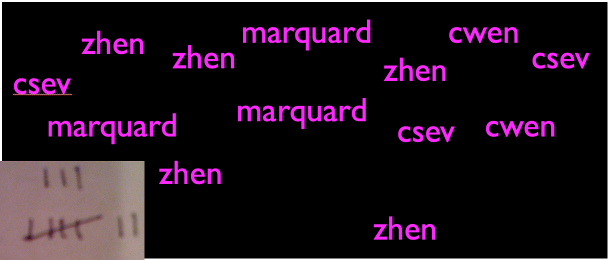
csev

cwen

zhen

zhen

Most Common Name?



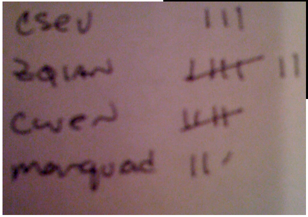
zhen zhen marquard cwen csev

csev zhen

marquard marquard csev cwen

zhen

zhen

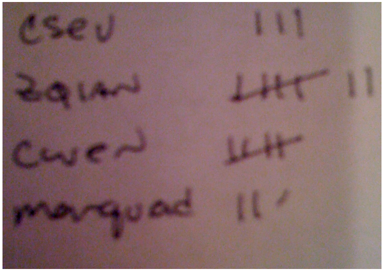


csev	|||
zqian	~~||||~~ ||
cwen	~~||||~~
marquad	||'

Many counters with a dictionary

One common use of dictionary is counting how often we "see" something

```
>>> ccc = dict()
>>> ccc['csev'] = 1
>>> ccc['cwen'] = 1
>>> print ccc
{'csev': 1, 'cwen': 1}
>>> ccc['cwen'] = ccc['cwen'] + 1
>>> print ccc
{'csev': 1, 'cwen': 2}
```



Dictionary Tracebacks

- It is an error to reference a key which is not in the dictionary
- We can use the in operator to see if a key is in the dictionary

```
>>> ccc = dict()
>>> print (ccc['csev'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'csev'
>>> print ('csev' in ccc)
False
```

When we see a new name

When we encounter a new name, we need to add a new entry in the dictionary and if this the second or later time we have seen the name, we simply add one to the count in the dictionary under that name

```
counts = dict()
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for name in names :
    if name not in counts:
        counts[name] = 1
    else :
         counts[name] = counts[name] + 1
print(counts)

{'csev': 2, 'zqian': 1, 'cwen': 2}
```

The get method for dictionaries

This pattern of checking to see if a key is already in a dictionary and assuming a default value if the key is not there is so common, that there is a method called get() that does this for us

```
    if name in counts:
        x = counts[name]
    else :
        x = 0

x = counts.get(name, 0) # Default value if key does not exist (and
    no Traceback).

{'csev': 2, 'zqian': 1, 'cwen': 2}
```

Simplified counting with get()

```
counts = dict()
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for name in names :
    counts[name] = counts.get(name, 0) + 1
print (counts)

{'csev': 2, 'zqian': 1, 'cwen': 2}
```

Counting Pattern

The general pattern to count the words in a line of text is to split the line into words, then loop through the words and use a dictionary to track the count of each word independently.

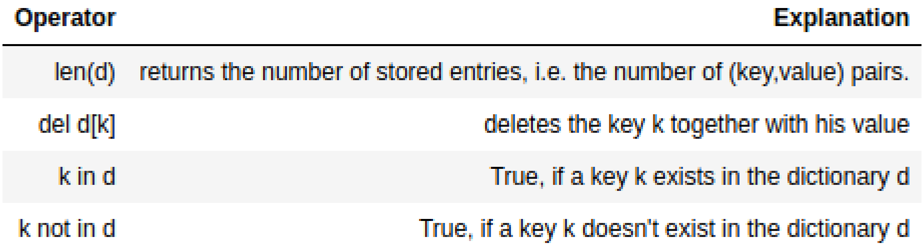
```
counts = dict()
print ('Enter a line of text:')
line = input('')

words = line.split()

print 'Words:', words

print ('Counting...')
for word in words:
    counts[word] = counts.get(word,0) + 1
print ('Counts', counts)
```

Operators in Dictionnary

Let d be a dictionnary, we have:

Operators in Dictionnary

Let d be a dictionnary, we have:



Operator	Explanation
len(d)	returns the number of stored entries, i.e. the number of (key,value) pairs.
del d[k]	deletes the key k together with his value
k in d	True, if a key k exists in the dictionary d
k not in d	True, if a key k doesn't exist in the dictionary d

Morse code-Example

The following dictionary contains a mapping from latin characters to morsecode.

```python
morse = {
"A" : ".-", "B" : "-...", "C" : "-.-.", "D" : "-..", "E" : ".", "F" : "..-.",
"G" : "--.", "H" : "....", "I" : "..", "J" : ".---", "K" : "-.-", "L" : ".-..",
"M" : "--", "N" : "-.", "O" : "---", "P" : ".--.", "Q" : "--.-", "R" : ".-.",
"S" : "...", "T" : "-", "U" : "..-", "V" : "...-", "W" : ".--", "X" : "-..-",
"Y" : "-.--", "Z" : "--..", "0" : "-----", "1" : ".----", "2" : "..---",
"3" : "...--", "4" : "....-", "5" : ".....", "6" : "-....", "7" : "--...",
"8" : "---..", "9" : "----.", "." : ".-.-.-", "," : "--..--"
}
```

Answer the following questions:

Answer the following questions:

1. What is the length of morsecode?

Answer the following questions:

1. What is the length of morsecode?
2. Is the letter "a" in morse?

Answer the following questions:

1. What is the length of morsecode?
2. Is the letter "a" in morse?
3. Is the letter "A" in morse?

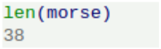
Answer the following questions:

1. What is the length of morsecode?
2. Is the letter "a" in morse?
3. Is the letter "A" in morse?
4. Give a word and transform it into a morsecode.

Answer the following questions:

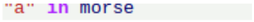
1. What is the length of morsecode?
2. Is the letter "a" in morse?
3. Is the letter "A" in morse?
4. Give a word and transform it into a morsecode.

Solution



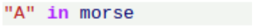
```
len(morse)
38
```

Output: : 38



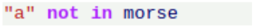
```
"a" in morse
```

Output: : False



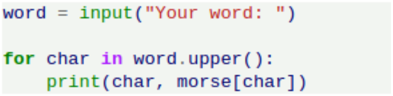
```
"A" in morse
```

Output: : True



```
"a" not in morse
```

Output: : True



```
word = input("Your word: ")

for char in word.upper():
    print(char, morse[char])
```