NoTeX Template

Marco Tallone

2024

Contents

1 Introduction

Welcome to NoTeX [2], a powerful LATEX template for all your document needs. Whether you're writing notes for your next class, a side project report, or just enjoy the look of this template, NoTeX has you covered.

Figure 1.1: Importing images from the `images/` folder is easy with the `graphicspath` command.

With NoTeX, you can easily organize your content into sections, subsections, and more. The template takes care of all the formatting, so you can focus on the content itself. It also supports importing images from a separate folder as shown in figure 1.1, making it convenient to include graphics in your document.

To get started, simply replace this placeholder text with your own introduction. Feel free to explore the other sections and customize the template to suit your needs.

If you want to add more sections, it's possible to simply create a new `.tex` file in the `sections/` folder and import it in the `main.tex` file as showed for these examples sections.

If you're new to LATEX, don't worry! The template is designed to be easy to use. Just look at the example and try to replicate it. If you need help, there are plenty of resources online to help you get started [1].

If you like this template please visit my repository on GitHub [2] and give it a star ⭐.

Happy writing!

2   Usage Examples

This section presents simple examples on how to use the NoTeX template and its features. The template is designed to be easy to use and customize, so you can focus on the content of your document. But if you don't like something mind that it's always possible to customize every aspect by messing with the files in the `settings/` folder.

In the following subsections you will find examples of how to use the template's features.

2.1   Boxes

You can use different types of boxes to highlight important information. A simple box can be created with the `cbox` environment, which stands for "*colored box*".

> This is a simple, not really colorful, box.
> You can use many types of environments inside it, such as lists, equations, images, and more.
> This has been made using the `tcolorbox` package, therefore floats are not welcome inside these boxes and the next ones.

You can also change the color of the box by passing an optional argument to the `cbox` environment and of course use already implemented LaTeX enviromnets inside them such as equations, double columns, lists and more... For example, to create an orange box, you can use the following code:

`\ begin{cbox}[color]`

> This is an orange box. Maybe it's a little bit more colorful, so let's add a formula:
> $$i\hbar \frac{d}{dt}\psi(t) = \hat{\mathcal{H}}\psi(t)$$

Remember to always close all the environments you open with the corresponding end command. Notice that you might need to change the color inside the box to make the text more readable. You can either use the command `\ textcolor` or pick one of the predefined colors commands such as:

- \red
- \orange
- \yellow
- \green
- \azure
- \blue
- \purple

2.2 Definitions

Colored boxes are fun but sometimes, especially in a scientific document, you need to add some definitions. To do so, the `definition` environment provides a simple way to create numbered definitions. The general syntax wants to have a definitions block being started with:

`\begin{definition}[<title>][<label>]`

> 📖 *def* 2.1   **Triangle**
>
> A triangle is a polygon with three edges and three vertices. Triangles are classified in:
>
> - Equilateral: all sides are equal.
>
> - Isosceles: two sides are equal.
>
> - Scalene: no sides are equal.

You can then reference the definition by referencing the label you gave to it, preceded by `def:`. For example look in the source code how I defined the definition 2.1.

It's not mandatory to give a title, neither the label, but if you have to assign a label to a non-titled definition you still need to pass one set of empty brackets `\begin{definition}[][label]` .

2.3 Theorems, Propositions, Lemmas, and Corollaries

In a scientific document, you might also need to add theorems, propositions, lemmas, and corollaries. The template provides environments for each of these, which are defined as follows:

```
\begin{theorem}[title][label]
```

Notice that, in order to be referenced, all three of these environments share the counter prefix `th:` and hence the numbering is shared among them.

THEOREM 2.1 **Pythagorean Theorem**

In a right triangle, the square of the length of the hypotenuse is equal to the sum of the squares of the lengths of the other two sides.

$$a^2 + b^2 = c^2$$

Then also:

PROPOSITION 2.2 **Proposition Name**

This is a nice proposition, with practical implications.

LEMMA 2.3 **Lemma Name**

This is a lemma, because it missed something to be a theorem.

And finally:

COROLLARY 2.4 **Corollary Name**

This is a corollary, whose proof is left as an exercise to the reader. 😂

Of course you can always leave the title and label empty if you don't need them.

2.4 Warnings

In some cases you might want to highlight important warnings, i.e. something that you want to immediately see when reading a particular page. For this it's possible to use the `warning` environment with:

```
\begin{warning}[color]
```

⚠️ Be careful: in order to compile this document you need the `LuaLaTeX` compiler.

If you really need to alert the reader about something important, you might thing of changing the color of the warning box like this.

⚠️ If you don't read this the world will end in 5 minutes. 😱 🌋

2.5 Informative Blocks

Observations, Notes and informations can be highlighted using the `info` and `blueinfo` environments:

```
\begin{info}
```

ℹ️ **Note**: The `LuaLaTeX` compiler is a modern version of the `LaTeX` compiler that allows you to use modern fonts and features.

Or the **blue version**:

```
\begin{blueinfo}
```

ℹ️ **Eyecatching Note**: hey, look what I can do with my hands: 🖖

2.6 Examples

Examples are what you sometimes need to make your reader understand better what you're talking about. The `example` environment is designed to help you with this. The general syntax is:

$$\verb|\begin{example}[title][label]|$$

where, again, you might omit eihter the title or the label if you don't need them.

Let's see... an **example** of this! 😁

> ✏️ **EXAMPLE 2.1** **How to use the example environment**
>
> Here is an example of how to use the example environment.
> You can write whatever you want here, and it will be highlighted in a box.
> The color it's alittle bit different from the main text so it's not confused with definitions, theorems or infos which looks similar.
> You can still use other blocks inside but you have to change the color manually to `\gray` as this won't be done automatically.
>
> > ℹ️ **Look:** an info block inside an example block.

2.7 Code

We've finally arrived to the section dedicated to code, one of the most complex one, as you might want to write code in many different ways depending on the context.
Before starting, however, a warning about this!

> ⚠️ **Warning**: the code environments are based on the `listings` package, which is not perfect and might not always work as expected. In particular, not all coding languages are included out of the box. I've added a few of the ones I usually use in the `settings/codestyles.sty` file, but you might need to add more if you use different languages. There you also will be able to add keywords or change any other option relative to the code language. If you want to change something about the code blocks, then I suggest looking at the `settings/code.sty` file.

2.7.1 Inline Code

You can write `inline code` as we've already seen using the `\cc{}` command.

2.7.2 Code Blocks

A simple code block can be created using the environment:

<div align="center">

`\begin{code}{<language>}`

</div>

For example:

```python
def hello(name):
    """Greets the user by name."""
    print(f"Hello, {name}!")
```

You then might want to add a title and this can be done by passing the title as a **keyword argument** to the `code` environment before the mandatory programming language argument:

> **</> My hello name function**
>
> ```python
> def hello(name):
>     """Greets the user by name."""
>     print(f"Hello, {name}!")
> ```

> ⚠️ Just as a general warning, **make sure the first keyword argument is on the same line as the opening '[' bracket, otherwise it won't work**.

In a similar way you can also pass independently input and output arguments:

> **</> My hello name function**
>
> INPUT: A name
> OUTPUT: Hello <name>!
>
> ```python
> def hello(name):
>     """Greets the user by name."""
>     print(f"Hello, {name}!")
> ```

> ℹ️ Just be careful that if you want to pass math symbols to the input or output arguments you have to "protext" them to avoid strange behaviors. This can be done with the `\protect` command.

The previous situation for instance usually happens when writing in Pseudocode which is an actual defined language in notex but you can modify it to your liking in the `codestyles.sty` file.

2.8 Tables

Tables are a common way to present data in a structured way. LaTeX already provides many tabular environments, here I just give you a nicely formatted one for this template that you can copy and paste. You also find this template commented in the `settings/notex.sty` file under the "Tables" section.

For normal tables I usually use this style:

Column1	Column2	Column3
Row1	Row1	Row1

Table 2.1: Normal table

While for tables with a lot of text I usually use the `tabularx` environment with the `X` column type. This is useful when you have a lot of text in a column and you want it to wrap around. This table can also extend to the full textwidth Here is an example:

Environment	Command	Description
Box	`\begin{cbox}[color]`	Creates a colored box.
Definition	`\begin{definition}[title][label]`	Defines something new.
Theorem	`\begin{theorem}[title][label]`	States a theorem.
Proposition	`\begin{proposition}[title][label]`	States a proposition.
Lemma	`\begin{lemma}[title][label]`	States a lemma.
Corollary	`\begin{corollary}[title][label]`	States a corollary.
Warning	`\begin{warning}[color]`	Highlights a warning.
Info	`\begin{info}`	Provides information.
Blue Info	`\begin{blueinfo}`	Provides informatione.
Example	`\begin{example}[title][label]`	Provides an example.
Code Block	`\begin{code}[title=…,…]{language}`	Formal code block.

Table 2.2: Summary of NoTeX environments

3   Update: Version 2.0! 🚀

Hi, version 2.0 is here! This is a major update that adds the `minted` package for new coding features. In particular now text highlighting in coding blocks is done through the `pygments` library 🔗 . This means that you need to install it in your system to use the new features. An easy way to do so is through `pip`:

```
pip install pygments
```

This version is retro-compatible with the previous one, so you can still use the previous coding environments as before (indeed the previous sections are just as they were).
Additionally I've added a new theme to celebrate this new version, it's called `Tokyo`. I hope you like it!
Here I will show how the new coding environments look like. These will be easier to use as you just need to include the language between a `begin` and `end` command. Check that your required language is included in the `settings/code.sty` file and you're ready to go.

```python
1  def main():
2          print("Hello, World!")
3          print("This is a Python code block.")
4          print("I hope you like it!")
5          print("Goodbye!")
```

The also support the usual input/output commands. Here's an example:

</> Python code block

INPUT: input.txt

OUTPUT: The file extension

```python
1  def read_extension(filename):
2          return filename.split('.')[-1]
```

Pygments supports a lot of languages, like YAML:

```yaml
1   name: "Pippo"
2   surname: "Pallo"
3   age: 24
4   list:
5       - item1
6       - item2
7       - item3
8
9   # This is a comment
10  nested:
11      key1: value1
12      key2: value2
13      key3: value3
```

Additionally I added an `\inline{}` command to have inline code snippets with correct syntax highlighting. For example, `print("Hello, World!")`.

References

[1] Overleaf. Overleaf. URL: https://www.overleaf.com.

[2] Marco Tallone. Latex templates. URL: https://github.com/marcotallone/notex.