Mu Nu Chapter of IEEE-HKN

Politecnico di Torino - Honor Society

Guide to conventions and packages for the HKNotes project

Authors:

Erik Scolaro

Editors:

Version: 1.1

3 luglio 2025

#WeAreHKN

Il **Mu Nu Chapter** di **IEEE-Eta Kappa Nu (HKN)** è la Honor Society del Politecnico di Torino dedicata agli studenti di eccellenza nei campi dell'ingegneria elettrica, elettronica, informatica e delle telecomunicazioni. IEEE-HKN è la più antica e prestigiosa honor society internazionale per ingegneri elettrici ed elettronici, fondata nel 1904. Il nostro capitolo promuove l'eccellenza accademica, il servizio alla comunità e lo sviluppo professionale attraverso attività di tutoring, workshop e progetti collaborativi.

Progetto Gratuito per la Condivisione della Conoscenza

Questo è un progetto completamente gratuito volto alla distribuzione libera di materiale didattico di qualità. Crediamo nell'importanza di rendere accessibile a tutti una formazione di eccellenza e offriamo gratuitamente queste risorse per supportare il percorso di studio di ogni studente.

Scopri le nostre iniziative:

- **Sito web:** https://hknpolito.org/
- **Instagram:** @hknpolito
- **Repository GitHub:** https://github.com/MuNuChapterHKN/Eta-Kappa-Notes/

Segnalazioni ed Errori

Hai trovato un errore o vorresti contribuire al miglioramento di questo materiale? Consulta il documento CONTRIBUTING.md nella nostra repository GitHub oppure scrivici a: responsabile.tutoring@hknpolito.it

Licenza Creative Commons

Indice

1 Introduction **1**
1.1 Project Structure 1
1.2 Document Class Attributes 2

2 Graphs **4**

3 Advanced Mathematics **8**
3.1 Introduction 8
3.2 Assigning a Label to an Equation 8
3.3 Citing an Equation 8
3.4 Citing an Equation with a Prefix 9
3.5 Citations with the `align` Environment 9
3.6 Citing an Equation with the `\eqref` Command 10

4 Formatting Code **11**
4.1 Inserting Code 11
4.2 Formatting the Code 12
4.3 Currently Defined Custom Styles 15
4.4 Complete Example 15

5 Boxes **16**
5.1 Boxing examples 16

6 Tables with longtables **17**
6.1 Longtable Example 17
6.2 Syntax exaplanation 18

Capitolo 1

Introduction

In this file, which has the same structure as the notes you will have to write, I will show you the basic usage of some packages and explain the structure of the project.

1.1 Project Structure

The project contains several folders:

- `chapters`, where you will insert a .tex file for each chapter with the same name (you can also name them chapter 1, 2, but trust me, if you name them sensibly, it will be much easier to reorder them later);
- `res`, this folder will contain any source files of type `.gbb` from Geogebra and `.py` if you want to use Python for creating graphs. Every time you create a graph, you must save the source file in the corresponding folder. Additional folders will be added based on the future tools chosen for generating graphs or other resources.
  - `gbb`, contains the Geogebra source file. It must be saved to ensure you can modify the corresponding graph in the future.
  - `py`, I've prepared some .py scripts to evaluate the potential of using matplotlib for creating your graphs. Inside, you will also find a file named `run_all_scripts.py`, which allows you to run all other .py files in the folder and automatically generate all the graphs. Note that this solution avoids the issue of saving the graphs; in fact, the various examples you can take as base templates for your graphs automatically save in the `./res/svg` folder, with the same name as the .py file and in svg format.
  - `svg`: this is where all the resources used within your notes will be placed, such as any svg files generated from Geogebra or Python or draw.io. draw.io is a powerful tool that allows you to intuitively and graphically create diagrams and electrical circuits. In this specific case, you don't need to save both the source and the svg, just the file in .drawio.svg format, because it can be used both in LaTeX as svg and in draw.io for future modifications.

- `template.tex`, this is where the magic happens. Basically, you only need to focus on organizing the chapters in the order you prefer, and you may need to modify a few small things that will be indicated to you as the project takes shape, such as inserting your information if you want to be credited and a brief explanation of the changes made to the document.
- Other files you don't need to worry about are generated automatically by the compiler. The only significant one will be `main.pdf` or whatever you named the main file, which will be the compiled .pdf file and and it is located in the build folder in a folder named the same as the one that contains the project you are compiling.

1.2 Document Class Attributes

When creating a LaTeX document, you can specify various attributes in the `\documentclass` command to customize the appearance and behavior of your document. Here are the attributes you can use with the `HKNdocument` class:

1.2.1 Language

You can set the language of the document to either Italian or English. The default language is Italian. To set the language, use one of the following options:

- `italian` (default)
- `english`

1.2.2 Table of Contents (ToC) Depth

You can control the depth of the Table of Contents (ToC) by specifying one of the following options:

- `toc=chapters`: Shows only chapters in the ToC (tocdepth=0)
- `toc=sections`: Shows chapters and sections (tocdepth=1)
- `toc=subsections` (default): Shows up to subsections (tocdepth=2)
- `toc=subsubsections`: Shows up to sub-subsections (tocdepth=3)

1.2.3 Font Size

You can set the base font size of the document using one of the following options:

- `10pt`
- `11pt` (default)
- `12pt`

1.2.4 Example Usage

Here is an example of how to use the `\documentclass` command with the `HKNdocument` class and some of the attributes mentioned above:

```
\documentclass[english, toc=sections, 12pt]{HKNdocument}
```

This command sets the document language to English, includes chapters and sections in the Table of Contents, and sets the base font size to 12pt.

Capitolo 2

Graphs

Here, I will simply show you how to import graphs in .svg format, add captions to them, and resize them properly.

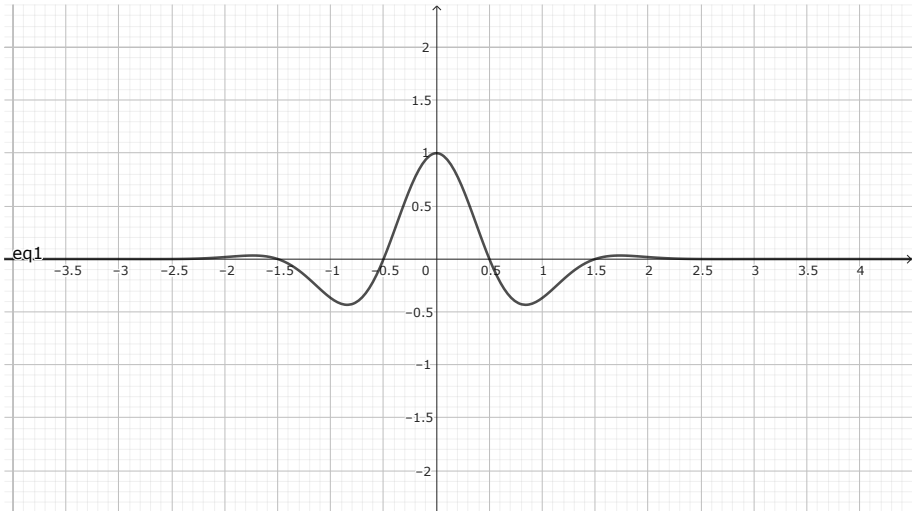

Figura 2.1: This is the example graph I created with Geogebra, with a caption and an associated label. To import SVG graphs, you need to install Inkscape, add it to the system path (watch out for the path, nothing works if you don't do this, it's usually `C:/Program Files/Inkscape/bin`), and modify the compiler flag by adding `-shell-escape`. If you are on Overleaf, it does this automatically (lucky you who didn't waste 1 hour figuring this out, as a revenge, I write the most important things in the captions xD).

If you open the .tex file, you will notice that I have inserted the `\href` command between each graph. This is to prevent LaTeXfrom optimizing the space by fragmen-

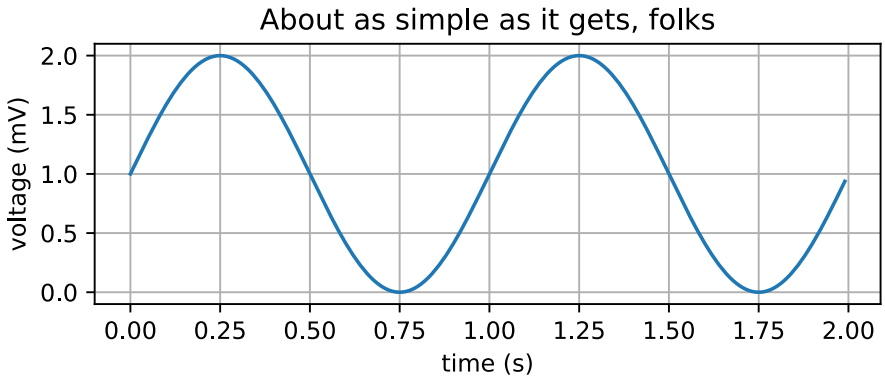

Figura 2.2: This is the 2d graph I created with matplotlib, see script.

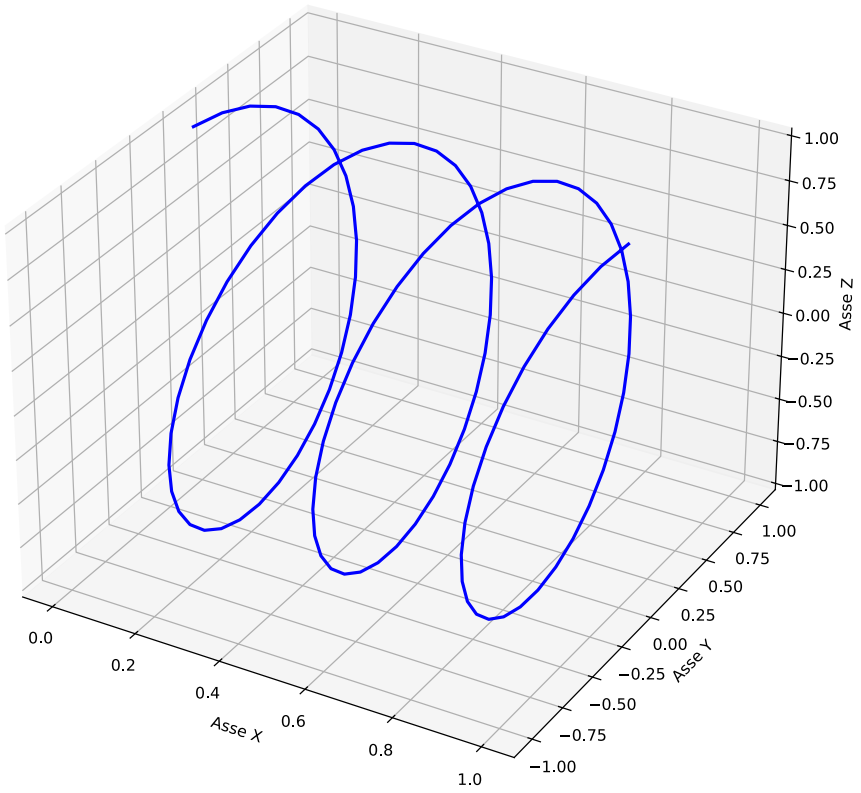

Figura 2.3: This is the continuous 3d graph I created with matplotlib, see script.

ting the bulleted list below between the graphs. Don't believe me? Try it and see what chaos it creates.
Labels can be associated with many objects in Matlab, make good use of them! Especially when you need to refer to formulas, tables, or graphs in your text. Here are all the ways you can reference an object with an associated label:

1. `\ref{graph:esempiografico}`: To refer to the number associated with the

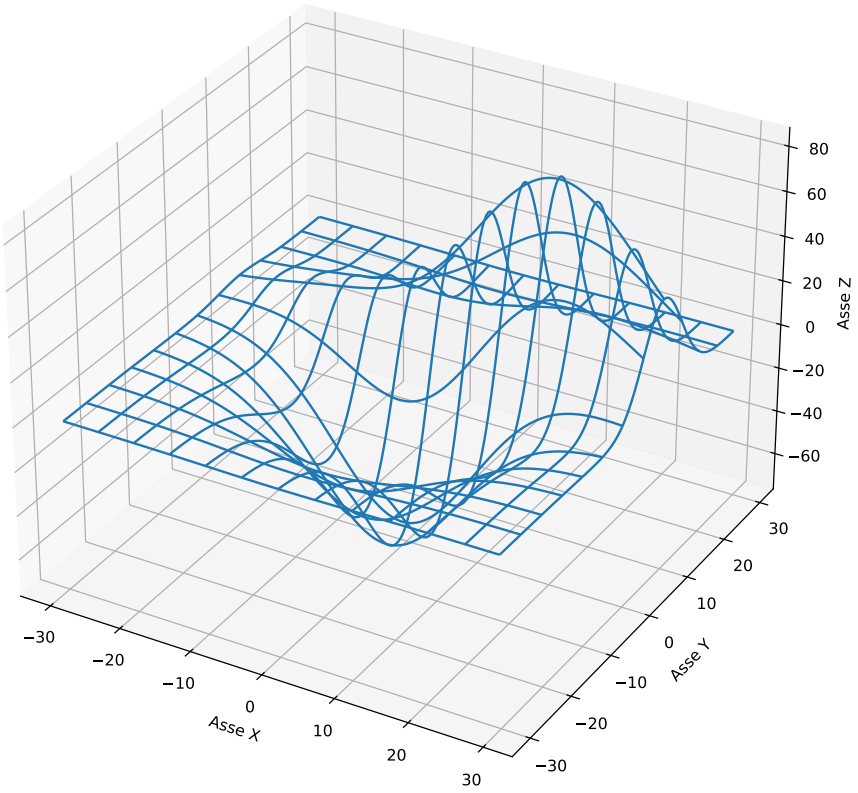

Figura 2.4: This is the continuous 3d graph I created with matplotlib, see script.

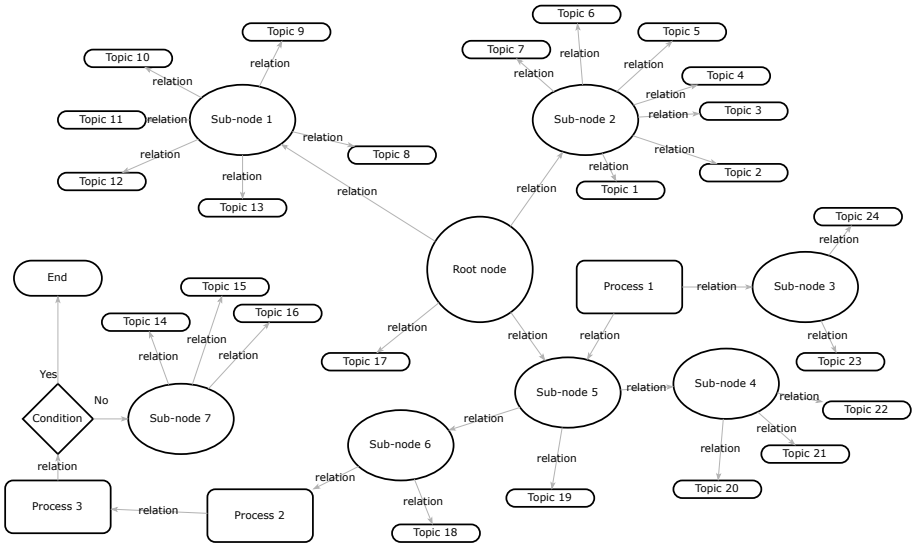

Figura 2.5: A dummy conceptual map created with the draw.io online editor.

label, such as a figure, table, section, etc. (example: Figure 2.1).

2. `\pageref{graph:esempiografico}`: To refer to the page number where the

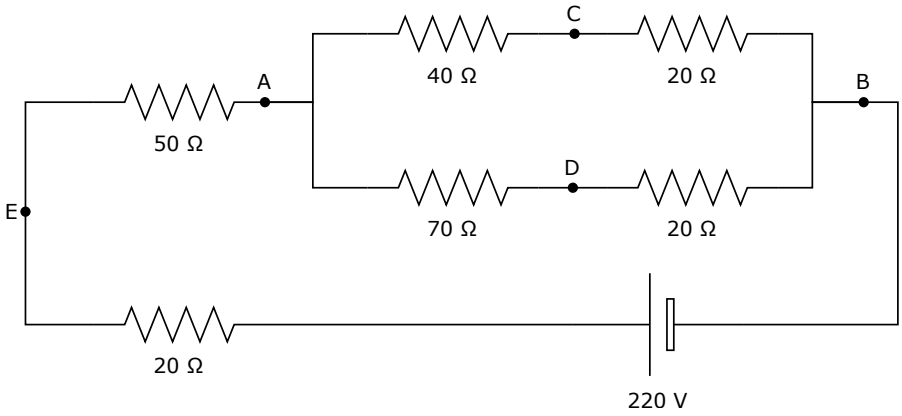

Figura 2.6: A basic circuit created with the draw.io online editor.

labeled object appears (example: See page 4).

3. `\hyperref{graph:esempiografico}`: If you use the `hyperref` package, you can click the reference to go directly to the figure, table, etc.

4. `\eqref{graph:esempiografico}`: To refer to an equation, if the label has been applied to an equation environment (example: as shown in equation (2.1)).

Capitolo 3

Assigning Labels to Equations in LaTeX and Citing Them Later

I don't want to talk to you about how to write mathematics, there are millions of guides on this topic. Instead, I want to talk about the ability to assign labels to equations in LaTeX so that they can be cited later.

3.1 Introduction

One of LaTeX's most powerful features is the ability to assign labels to equations and then easily cite them within the document. This is particularly useful in scientific and technical documents, where it's often necessary to refer to specific equations. In this chapter, we will see how to label equations and how to cite them automatically, ensuring that the equation number is updated correctly during the document compilation.

3.2 Assigning a Label to an Equation

To label an equation, you need to use the `\label{}` command within the environment where the equation is written (e.g., `equation`, `align`, etc.). The argument of the `\label{}` command is the name of the label, which must be unique in the document.
Here's an example of labeling an equation:

$$E = mc^2 \tag{3.1}$$

In this case, the equation $E = mc^2$ is labeled with `eqn:energy`. Now, LaTeX will automatically assign a number to this equation, which can be cited later in the document.

3.3 Citing an Equation

To reference the labeled equation, use the `\ref{}` command. The argument of the `\ref{}` command is the name of the label that was assigned to the equation.

Here's how to cite the previous equation:

> As shown in equation 3.1, Einstein's famous formula relates energy to mass.

The result will be:

> As shown in equation (1), Einstein's famous formula relates energy to mass.

LaTeX will automatically replace `\ref{eqn:energy}` with the equation number (e.g., (1)), which will update if other equations are added or removed from the document.

3.4 Citing an Equation with a Prefix

If you want to add a prefix to the equation number (like "Eq." or "Equation"), you can do so manually. For example:

> As shown in Eq. 3.1, Einstein's famous formula relates energy to mass.

The result will be:

> As shown in Eq. (1), Einstein's famous formula relates energy to mass.

3.5 Citations with the `align` Environment

When using environments like `align`, which allow you to write multiple equations on separate lines, you can label each individual equation. Here's an example:

$$a + b = c \tag{3.2}$$
$$x - y = z \tag{3.3}$$

To cite the individual equations:

> As shown in 3.2 and 3.3, the operations of addition and subtraction are defined respectively.

The result will be:

> As shown in (1) and (2), the operations of addition and subtraction are defined respectively.

3.6 Citing an Equation with the `\eqref` Command

If you want to cite an equation including the parentheses around the equation number automatically, you can use the `\eqref{}` command instead of `\ref{}`. Here's how:

As shown in (3.1), Einstein's formula expresses energy in terms of mass.

The result will be:

As shown in (1), Einstein's formula expresses energy in terms of mass.

Capitolo 4

Formatting Code

In this chapter, we will show how to use the `listings` package to format source code in LaTeX. This package allows you to highlight the syntax of various programming languages and customize the style to fit your document.

4.1 Inserting Code

After defining the style, you can use it to format code within the document. To do this, use the `\lstset` command to set the style and the `\lstinputlisting` or `\beginlstlisting...\endlstlisting` command to include the code. Here is an example:

4.1.1 Example with External File

You can also include an external file containing the code:

```
"""
Visualizzazione di un grafico 3D con stile wireframe.
Il file SVG viene salvato automaticamente nella
   cartella predestinata.
"""
import matplotlib.pyplot as plt
import numpy as np
import os

# Dimensioni immagine in cm
width = 15
height = 5

# Nome del file SVG basato sul nome dello script
script_name =
   os.path.splitext(os.path.basename(__file__))[0]
file_name = f"{script_name}.svg"

# Cartella di destinazione (relativa alla posizione
   dello script)
```

```
folder_path =
    os.path.join(os.path.dirname(__file__),
    "../../res/svg")

# Crea la cartella di destinazione se non esiste
os.makedirs(folder_path, exist_ok=True)

# Genera i dati
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2 * np.pi * t)

# Crea il grafico
fig, ax = plt.subplots()
ax.plot(t, s)

# Aggiungi etichette agli assi
ax.set(xlabel='time (s)', ylabel='voltage (mV)',
       title='About as simple as it gets, folks')
ax.grid()

# Imposta le dimensioni della figura in centimetri
    (1 cm = 0.393701 pollici)
cm_to_inch = 0.393701
fig.set_size_inches(width * cm_to_inch, height *
    cm_to_inch)

# Salva il grafico nella cartella di destinazione
output_path = os.path.join(folder_path, file_name)
plt.savefig(output_path, format="svg",
    bbox_inches="tight")

print(f"Grafico salvato in: {output_path}")

# Disabilitare quando compili tutti i file con
    run_all_scripts.py
# plt.show()
```

4.2 Formatting the Code

The `listings` package allows you to display source code within a LaTeX document while preserving formatting and syntax highlighting. In this section, we will explore how to use `listings` to include and format code in our document.

4.2.1 Predefined Styles

`listings` provides some predefined styles for syntax highlighting of various programming languages. These styles allow you to easily include code blocks with

correct syntax highlighting.
For example, to include Python code in the document, you can use the following command:

```
\begin{lstlisting}[language=Python]
        def hello_world():
        print("Hello, World!")
\end{lstlisting}
```

Result:

```
def hello_world():
print("Hello, World!")
```

In this example, the `listings` package automatically recognizes Python syntax and highlights it correctly.

4.2.2 Supported Programming Languages

`listings` supports many programming languages, so you can use automatic syntax highlighting for a wide range of languages. Just specify the language using the `language` option within the `lstlisting` environment. Some supported languages include:

- Python: `language=Python`
- C/C++: `language=C`
- Java: `language=Java`
- SQL: `language=SQL`
- HTML: `language=HTML`
- CSS: `language=CSS`
- JavaScript: `language=JavaScript`
- XML: `language=XML`

Example of SQL code:

```
SELECT * FROM users WHERE age > 18;
```

4.2.3 Customizing Styles

Although the package offers predefined styles, it is possible to customize the appearance of the code. If there is a need to create new styles, contact the responsible person. Some customizable parameters, listed here to highlight the potential of the styles, include:

- `language`: Defines the language of the code.

- `basicstyle`: Sets the base font for the code.
- `keywordstyle`: Modifies the style of keywords.
- `stringstyle`: Modifies the style of strings.
- `commentstyle`: Defines the style for comments.
- `numbers`: Displays line numbers.
- `frame`: Adds a border around the code.
- `backgroundcolor`: Sets the background color of the code.

This is the before and after applying the SQL style defined in HKNtools.tex:

```
-- Query to select user data
SELECT name, age
FROM users
WHERE age > 18
ORDER BY age DESC;
```

```sql
-- Query to select user data
SELECT name, age
FROM users
WHERE age > 18
ORDER BY age DESC;
```

4.2.4 Global Configuration

You can also configure `listings` globally to apply the same style settings throughout the document using the
`lstset` command. For example:

```
\lstset{
        language=Python,
        numbers=left,
        frame=single,
        backgroundcolor=\color{lightgray},
        stepnumber=1
}
```

This will set the options for all Python code blocks in the document. Here are the effects before and after using the instruction:

```
def hello_world():
print("Hello, World!")
```

```
1 def hello_world():
2 print("Hello, World!")
```

4.3 Currently Defined Custom Styles

Additional styles will be implemented as needed by collaborators. For now, the following styles are currently defined:

- SQL: sqlstyle

4.4 Complete Example

Below is an example of a C program highlighted with the `listings` package. This program prints a greeting on the screen. As you can see, I used a `label` to easily reference the code later.

Listing 4.1: Demo

```c
#include <stdio.h>

int x = 10;
uint32_t y = 10;

// Example program
int main() {
  printf("Hello, World!\n");
  return 0;
}
```

This example is labeled with `label=code:demo`, so we can refer to this code anywhere in the document using the command `4.1`. For example, we can reference the code 4.1 in the text.

Capitolo 5

Boxes

5.1 Boxing examples

Additionally, the numbering between different types of boxes (such as Definitions, Theorems, Exercises, etc.) is independent. This means that the numbering for Definitions will not affect the numbering for Theorems, and so on. The numbering of the various boxes (such as Definitions, Theorems, Exercises, etc.) is automatically reset at the beginning of each chapter. This means that every time a new chapter starts, the box counter resets to 1.

Definizione 1. An example of a definition

This is an example of a colored box with the title "Definition" in English.

Teorema 1. An example of a theorem

This is an example of a colored box with the title "Theorem" in English.

Corollario 1. An example of a corollary

This is an example of a colored box with the title "Corollary" in English.

Esercizio 1. An example of an exercise

This is an example of a colored box with the title "Exercise" in English.

Osservazione 1. An example of an observation

This is an example of a colored box with the title "Observation" in English.

Capitolo 6

Tables with longtables

6.1 Longtable Example

Tabella 6.1: Example of a Longtable with Caption and Label

Concetto	**Tipo**	**Volume**	**Motivazione**
Utente	E	30.000.000	Ipotizziamo una piattaforma in cui sono iscritte 30 milioni di utenti
Host	E	150.000	Ipotizziamo che sulla piattaforma si iscriveranno circa 150 mila host
Alloggio	E	169.000	Ipotizziamo che nella piattaforma verranno registrati circa 169 mila alloggi
Prenotazione	E	36.000.000	Ipotizziamo che sulla piattaforma siano state effettuate circa 36 milioni di prenotazioni
Soggiorno	E	34.920.000	Ipotizziamo che sulla piattaforma ci siano stati circa 35 milioni di soggiorni
Recensione	E	12.000.000	Ipotizziamo che sulla piattaforma vengano scritte circa 12 milioni di recensioni
Commento	E	16.000.000	Ipotizziamo che sulla piattaforma vengano scritti circa 16 milioni di commenti
Lista	E	45.000.000	Ipotizziamo che sulla piattaforma vengano create circa 45 milioni di liste di alloggi preferiti
Servizio	E	20	Ipotizziamo che sulla piattaforma vengano messi a disposizione circa 20 servizi differenti

Continua all pagina successiva

Concetto	Tipo	Volume	Motivazione
Possedimento	R	169.000	Ipotizziamo che nella piattaforma ogni host abbia almeno un alloggio, e che 1 host su 8 abbia 2-3 alloggi
Richiesta	R	36.000.000	Ipotizziamo che nella piattaforma 4 utenti registrati su 5 abbiano fatto almeno una prenotazione, e 1 su 5 ne abbia fatto almeno 3
Generazione	R	34.920.000	Ipotizziamo che sul totale delle prenotazioni, circa il 2% vengano cancellate. Tutte le altre diventano soggiorni effettivi
Elaborazione	R	12.000.000	Ipotizziamo che 1 utente su 3 che ha effettuato una prenotazione poi scriva una recensione
Contenuto	R	16.000.000	Ipotizziamo che circa 1 recensione su 3 abbia un thread con almeno 3 commenti e 1 su 3 abbia un solo commento
Creazione	R	45.000.000	Ipotizziamo che circa 6 utenti su 10 creino delle liste, con una media di 2-3 liste per ciascuno di questi utenti
Scritto	R	16.000.000	Ipotizziamo che circa 1 utente su 5 abbia scritto un commento, e di questi uno ne abbia scritto circa 2-3
Correlazione	R	12.000.000	Ipotizziamo che circa 1 soggiorno su 6 riceva una recensione da parte dell'utente o dell'host, e che 1 su 6 la riceva da parte di entrambi
Riserva	R	36.000.000	Ipotizziamo che tutti gli alloggi vengano riservati circa 36 milioni di volte, una volta per ogni prenotazione
Offerto	R	250.000	Ipotizziamo che ogni alloggio offra più di una decina di servizi
Valutazione	R	2.000.000	Ipotizziamo che circa 1 recensione su 3 viene scritta verso un alloggio

6.2 Syntax exaplanation

The syntax used in LaTeX for the table 6.1 with the `longtable` package is as follows:

- **Table Declaration**:

```
\begin{longtable}{|l|c|c|p{6.2cm}|}
```

Here, a table is declared with 4 columns. The first column is left-aligned (`l`), the second and third columns are centered (`c`), while the fourth column has a width of 6.2 cm and adjusts to the content (`p{6.2cm}`).

- **Table Header**:

```
\hline \textbf{Concept} & \textbf{Type} & \textbf{Volume}
& \textbf{Reason} \\\hline
```

These lines are used to define the header of the table and separate it from the subsequent rows with a horizontal line.

- **Commands for Different Table Sections**:
  - `endfirsthead`: Defines the header to be used on the first page of the table.
  - `endhead`: Defines the header to be used on the following pages of the table.
  - `endfoot`: Defines the footer for each page of the table.
  - `endlastfoot`: Defines the final footer of the table.
- **Data Rows**: Each data row is separated by
and ends with `hline` to add a horizontal line after each row. The data in each cell is separated by `&`.

  Example:

```
User & E & 30.000.000 & {Let's assume a platform with 30
million users} \\\hline
```

This structure allows the creation of tables that can span multiple pages and contains horizontal lines in both the headers and between the data, maintaining a clear and readable format.