LearnLaTeX

Domipoke

July 29, 2026

Contents

1 Prefazione **5**

1.1 Semplicità e accuratezza 5

1.2 LaTeX: vita morte e miracoli 6

1.3 Ai 6

1.4 Dove scrivere in LaTeX 7

1.4.1 Overleaf 7

1.4.2 Overleaf Toolkit 7

1.4.3 Download locale 8

2 LaTeX **11**

2.1 Le basi 11

2.2 \documentclass{} 12

2.3 Importare librerie 13

2.4 Metadati 14

2.5 document 15

2.5.1 Comandi 15

2.5.2 Ambienti 16

2.5.3 Parentesi 17

3 Formattazione **19**

3.1 Bold, Italic, Underline 19

3.2 Font 20

3.2.1 Definire un font 21

3.2.2 Usare un font 21

3.3 Colori 22

3.3.1 Definire un colore 23

3.3.2 Uso di un colore 24

4 Liste **27**

5 Tabelle **29**

6 Immagini **31**

6.1 Figure 31

6.2 Tikz 31

7 Equazioni **33**

Chapter 1

Prefazione

1.1 Semplicità e accuratezza

Questa è una guida che scrivo per un pubblico di non programmatori in quanto penso che molto spesso si tende a voler usare tecnicismi per creare una bolla di conoscenze non "alla portata di tutti". Mi rendo conto spiegando spesso a "persone che non sanno programmare"[1] che ci sono tantissimi termini loro sconosciuti che dò per scontato. In questa guida tratterò questi termini con piccole note a piè di pagina che spiegano nel modo più semplice e chiaro[2] a cosa faccio riferimento. Per essere più chiari e diretti, il commento sulle spiegazioni "Mi dicono che in realtà non funziona proprio così" o "Questa spiegazione è un po' riduttiva rispetto a quello che accade realmente" sono precisazioni talvolta inutili altre un po' sconsiderate. Sono ben felice di darvi qualsiasi tipo di spiegazione su ognuna delle cose che conosco e sulle quali metto mano quotidianamente ma spiegazioni del genere richiedono tempo e parole. Fidatevi di questi due punti, quando leggerete in questa prefazione non è altro che la sintesi estremamente accorciata di una registrazione audio di 40

[1] modo semplice e un po' uno scusante per indicare quelle persone che non sono "avezze" nell'uso del computer nonostante siano "nativi digitali"

[2] Per chiarezza intendo lo speigare qualcosa a qualcuno nel migliore dei modi che ho di associare quel concetto a qualcosa che questa persona conosce anche se la mia spiegazione dovesse risultare non tecnicamente accurata.

minuti che ho fatto per ipotizzare ciò che avrei dovuto scrivere, 30 di questi ho parlato di git e Github, programmi che conosco e utilizzo da sempre[3].

1.2 LaTeX: vita morte e miracoli

Prima di inizia poniamo delle premesse, LaTeX è un linguaggio di marcatura il che vuol dire che nasce col fine di formattare testo. Nasce nel 1994 e si evolve in una miriade di sottolinguaggi come il KaTeX di notion che verticalizza e isola l'uso di LaTeX rendendolo una comoda implementazione per scrivere equazioni. Un'altra evoluzione seguita da LaTeX è stata quella di evolvere i propri compilatori[4], li vedremo più nello specifico in futuro ma vi basti sapere che ho scelto di usare XeLaTeX che è un implementazione più nuova e ottimizzata del compilatore base denominato PdfLaTeX.

1.3 Ai

Una premessa anche sull'uso dell'intelligenza artificiale è opportuna. Gemini, in quanto probabilmente sarà la vostra scelta, nella sua modalità pro in particolare, è brava a capire come usare piccoli e singoli blocchi di LaTeX, tipo "come si crea una tabella su LaTeX?", "come creo una lista?", "come aggiungo un'immagine". Più informazioni passiamo a Gemini, più questa risposta sarà precisa. Un'accortezza che dovrete avere o quantomeno un punto da tener in considerazione nell'uso di Gemini è lo specificare le librerie che volete usi o informarmi, anche conversando con Gemini e prendendo le sue informazioni con le pinze, di quali siano le librerie compatibili con la compilazione che abbiamo scelto. Attenzione però che Gemini usato senza controllo sbaglia, sbaglia spesso e porta a peggiorare gli errori commessi consigliando di apportare diverse modifiche, magari giuste in generale ma non per

[3] la data di creazione del mio account Github risale al 2018-08-11T14:13:22Z

[4] Un compilatore è un programma che si occupa di tradurre da codice a programma finito, nel nostro caso da .tex a pdf

il vostro progetto. Il mio consiglio è di dare due possibilità a Gemini, se due modifiche di Gemini non sistemano il codice allora ritornate indietro a prima che Gemini ci mettesse mano. Un'altra alternativa sarebbe usare Claude per il quale valgono pur sempre le avvertenze ma è molto migliore di Gemini. Claude però ha un piano base con limiti piuttosto stringenti quindi usatelo "in caso di emergenza". Un uso utile di un'intelligenza artificiale è quando il vostro lavoro non compila e come errore vi dà un log immenso da leggere, copia incollatelo interamente su un ai e cercate di vedere che spiegazione vi dà, valutatene la correttezza delle sue affermazioni e corregete.

1.4 Dove scrivere in LaTeX

L'ambiente in cui lavorare è alla fine una vostra scelta, posso al più darvi dei consigli e sconsigliarvi altri metodi.

1.4.1 Overleaf

Il tool web più usato si chiama Overleaf, è una piattaforma comoda da usare ma ha un piano base molto limitato che generalmente vieta di lavorare su grandi file o su file che contengono tante strutture più complesse del semplice testo. Altro punto negativo è che essendo un sito web si può accedere ai file solo se connessi a una rete wifi.

1.4.2 Overleaf Toolkit

È la versione selfhosted[5] di Overleaf. Permette di alzare i limiti ma essendo il mio server non particolarmente potente non so che carico potrà effettivamente sostenere.

[5] Il termine selfhosted vuol dire che si può installare localmente su un server o computer, nel mio caso userò selfhosted per indicare i programmi che posso scaricare sul mio server e rendere a voi accessibili

Il carico inoltre scala in funzione di quanti di voi decideranno di utilizzarlo.

1.4.3 Download locale

Il download locale di LaTeX necessita di 2 componenti:

- Ambiente di sviluppo
- Compilatore locale o remoto

Ambiente di sviluppo

Visual Studio Code L'ambiente di sviluppo locale che consiglio è Visual Studio Code. Mi rendo conto possa essere un troppo per quel poco di LaTeX che dobbiamo fare in quanto nasce come ambiente per la programmazione ma proprio per questo risulta essere il più completo editor di testo che puoi desiderare. Possiede una miriade di estensioni installabili per migliorarne l'uso e permette di visualizzare nella stessa applicazione sia il codice che il pdf compilato. Visual Studio Code possiede una funzione denominata IntelliSense che suggerisce la completazione automatica del codice (non del testo) che stiamo scrivendo in funzione di ciò che abbiamo scritto. IntelliSense non è basata su intelligenza artificiale.

TeXStudio L'opzione più vecchia è altrimenti usare TeXStudio che nativamente supporta LaTeX e che forse possiede un compilatore inegrato. Non l'ho mai usato. Sarà sicuramente comodo quanto vecchio. Lo penso perchè non sono nati molti altri programmi del genere.

Compilatore locale

Windows Per quanto riguarda Windows abbiamo due possibilità: **TeX Live** e **MiKTeX**.

TeX Live TeX Live installa la libreria completa di LaTeX con tutte librerie e i compilatori ad esso associate. Si parla di 1.6GB ma il problema non è tanto il peso quanto il fatto di essere distribuito da un server vecchio e che il download non è centralizzato, ovvero ogni singolo file ha una posizione differente, ciò rallenta maggiormente il download di questi file. A me ha impiegato circa 6 ore. È inoltre un download che non dovrebbe essere interrotto.

MiKTeX MiKTeX è l'alternativa più veloce e più nuova, installa una minima parte delle librerie e dei compilatori necessari, necessita del download di una libreria in più denominata Pearl, un linguaggio di programmazione ($200MB$), per poter essere usata con l'estensione di Visual Studio Code per LaTeX. MiKTeX scarica le ulteriori librerie solo quando compiliamo un progetto che le usa.

MacOsX L'opzione per MacOsX è una sola: MacTeX. Non l'ho mai usata.

Compilatore remoto

Questa è una soluzione che sto testando con i lavori degli ultimi giorni. Consiste nell'usare git insieme a Gitea (versione selfhosted del più famoso Github). Git gestisce il `push` (upload) e `pull` (download) dei progetti in una cartella remota definita come `repository`. Possiede una funzione detta `Action` che permette di eseguire in modo automatico su un docker[6] remoto delle azioni di compilazione e aggiungere al nostro upload il risultato finito. Gitea essendo selfhosted è priva di tempi di attesa a meno di più richieste simultanee e generalmente risolve la compilazione entro i due minuti. È possibile scaricare il pdf in locale effettuando una `pull`. Altro vantaggio di git il quale lo rende estremamente usato sia dai programmatori che più in generale per qualsiasi progetto di gruppo è la sua comoda gestione del lavoro simultaneo di più persone anche sullo stesso file.

[6] Un particolare programma di macchine virtuali

Chapter 2

LaTeX

2.1 Le basi

La base di un progetto in LaTeX è un file con estensione[1] `.tex`. La generale struttura di un file `.tex` è composto da:

- `\commands[optionArgs]{Args}`
- `\begin{ambiente}...\end{ambiente}`: immaginatelo come una parentesi che si apre e si chiude e che ha un contenuto.
- Testo normale
- `%` Commento: ciò che segue `%` nella stessa riga viene ignorato

Lo scheletro di un file `.tex` è il seguente:

```
\documentclass{article} % Definizione del tipo di documento
% Lista di librerie
...

% Inizio del documento
\begin{document}
```

[1] Ogni file possiede un'estensione che si divide dal nome del file utilizzando un punto (`.`). L'estensione generalmente è una sigla di massimo 4 caratteri. Esempi famosi sono: .exe, .mp3, .mp4, .png, .pdf.

```
    % Testo del nostro documento
\end{document}
```

La riga `\documentclass` è necessaria per identificare il file e dovrebbe essere la prima chiamata. La lista delle librerie è buona norma elencarla all'inizio, subito dopo `\documentclass` per assicurarci che il compilatore abbia già caricato tutte le librerie correttamente prima di iniziare a compilare il testo effettivamente. Tutto ciò che appartiene al nostro documento va inserito all'interno dell'ambiente `document`.

2.2 \documentclass{}

La prima riga di un file `.tex`, se rappresenta un root file, ovvero il file che dovrà essere compilato in pdf, deve delineare la classe del documento. Ognuna delle classi del documento delinea delle generali regole di formattazione che successivamente elencherò che possono subire leggere modifiche attraverso degli argomenti opzionali. La generica forma è `\documentclass{classe}` ma è possibile modificare alcuni parametri utilizzando la forma `\documentclass[key1=value1, key2=value2, key3=value3]{classe}`. Tra le classi standard vi sono le seguenti:

article Generalmente utilizzata per la scrittura di piccoli articoli. Non possiede una divisione in capitoli.

report Generalmente utilizzato per la scrittura di documenti di media dimensione. Possiede una divisione in capitoli.

book Ideale per la stesura di lunghi testi. Supporta una struttura articolata in capitoli, sezioni e sottosezioni. Generalmente il miglior formato per sistemare gli appunti in un unico file.

letter Generalmente utilizzato per scrivere nel formato di lettera (circa A5). Formato utile quando bisogna scrivere corti testi o testi di cui c'è la necessità di fruire su un dispositivo mobile o su un ereader.

Altre classi sono:

scrbook Come book ma con opzioni più avanzate, è quello che sto utilizzando per questo libro.

beamer Il formato per la realizzazione di slide.

Le classi standard possiedono tra i loro argomenti opzionali i seguenti:

fontsize Dimensione generica del font utilizzato. Es. `fontsize=10pt`

pagesize Dimensione della pagina. Es. `pagesize=a4paper` o `pagesize=letterpaper`. Si può avere una letter impaginata in foglio a4 o un libro inpaginato in un foglio lettera

pagelayout Layout delle pagine. Es. `pagelayout=onecolumn` o `pagelayout=twocolumn` o nel caso di article `pagelayout=oneside` o `pagelayout=twoside`

2.3 Importare librerie

Una libreria è un insieme di comandi di LaTeX che vengono importati mediante l'uso di un determinato comando e che ha lo scopo di cambiare o introdurre nuove regole e istruzioni nel nostro progetto. Quando vogliamo importare una libreria necessitiamo di scrivere sotto `\documentclass` l'istruzione `\usepackage{libreria}`. È possibile anche in questo caso passare degli argomenti opzionali mediante l'uso di `\usepackage[key1=value1, key2=value2, key3=value3]{libreria}`.

Nota

Quando scriverò "dalla libreria `nome`", implico la necessità di avere nella parte iniziale del documento, dopo `\documentclass` ma prima delle successive istruzioni una riga `\usepackage{nome}`

È possibile creare proprie librerie da importare comodamente in diversi progetti creando un file `nome.sty` e chiamandolo nel file `.text` usando il comando `\usepackage{nome}`. Sarà tra le ultime cose che vi spiegherò, ve lo dico in quanto ne faccio uso in questa guida.

2.4 Metadati

L'aggiunta di metadati[2] avviene dopo l'importazione delle librerie e prima dell'inizio del documento. È possibile creare una pagina nel documento utilizzando il comando `\maketitle` generata in automatico dalle informazioni definite come metadati.

```
\usepackage{book}
% Librerie
...
% Metadati
\title{Titolo}
\author{Nome}
\date{Data}
% Documento
\begin{document}
\maketitle
\end{document}
```

Al posto di `Titolo`, `Nome`, `Data` è possibile inserire qualsiasi stringa[3] e i comandi che hanno come risultato una stringa. Un esempio particolare è l'uso del comando `\today` per la data.

[2] Un metadato è un dato aggiuntivo di un file. Un esempio di metadato è la posizione gps o le informazioni sulla telecamera nelle foto.

[3] Per stringa si intende un insieme di caratteri e spazi che formano del testo

2.5 document

Il documento è il vero contenuto del nostro pdf. Qui ogni cosa che scriveremo sarà effettivamente visibile nel nostro pdf e formattato seguendo i comandi e gli ambienti che useremo.

2.5.1 Comandi

Un comando è sempre introdotto dal carattere /. Qualora volessimo inserire proprio il carattere /nel nostro testo possiamo usare il comando `\slash` Un comando può avere una delle seguenti strutture:

Senza argomenti `\comando`

Argomenti obbligatori `\comando{}` sono espressi all'interno di una coppia di parentesi graffe. Se il comando necessita di un numero n di argomenti allora necessitiamo di inserire n coppie di parentesi graffe concatenate. Esempio $n = 3$ `\comando{arg1}{arg2}{arg3}`

Argomenti facoltativi `\comando[]{}` sono espressi all'interno di una coppia di parentesi quadre. Se il comando necessita di un numero m di argomenti allora necessitiamo di inserire m coppie di parentesi quadre concatenate. Esempio $m = 3, n = 1$ `\comando[arg1][arg2][arg3]{arg4}` o esempio $m = 3, n = 3$ `\comando[arg1][arg2][arg3]{arg4}{arg5}{arg6}`.

Nota

Un comando termina all'inserimento di uno spazio al di fuori di una coppia di parentesi. `\comando{arg1} {arg2}` è errato in quanto lo spazio spezza la definizione del comando. Scrivere nella forma precedente è come scrivere `\comando{arg1} arg2` dove arg2 è testo normale visualizzato e non passato come argomento del comando.

A causa di questa regola capita spesso di avere errori di formattazione dovuti a comandi che necessitano di un doppio spazio, uno per essere interrotti e uno per separarli di uno spazio dalla lettera successiva. Un esempio di errore è il seguente: `\LaTeX è un linguaggio di marcatura` (con un solo spazio dopo il comando) fornisce questo risultato LaTeXè un linguaggio di marcatura.
Un escamotage che si può utilzzare è quello di scrivere il comando all'interno di una coppia di parentesi graffe `{\LaTeX}` ma è un'operazione che può rallentare molto la scrittura.
Non tutti i comandi possiedono questa problematica, i comandi delle librerie più nuove possiedono una serie di regole che valutano quando lo spazio posto dopo deve essere effettivamente visualizzato come uno spazio.

2.5.2 Ambienti

Un ambiente è un blocco che modifica la formattazione del testo per creare blocchi visuali. Si utilizzano anche per la creazione di tabelle, blocchi formattati, blocchi di codice trascritto o l'inserimento di immagini. Un ambiente si inizializza con begin e si termina con end passando come primo argomento obbligatorio per entrambi il nome dell'ambiente. Alcuni ambienti nel blocco di begin possono richiedere argomenti opzionali passati all'interno di parentesi quadre e più di un argomento obbligatorio passato dopo il nome dell'ambiente in parentesi graffe.

```
\begin{ambiente}
...
\end{ambiente}

\begin{ambiente}[ArgomentoOpzionale]
...
\end{ambiente}
\begin{ambiente}{ArgomentoObbligatorio}
...
\end{ambiente}
\begin{ambiente}[ArgomentoOpzionale]{ArgomentoObbligatorio}
...
\end{ambiente}
```

Nota

A differenza dei comandi è necessario prima sempre esplicare il tipo di ambiente, seguono gli argomenti opzionali, seguono gli argomenti obbligatori.

In generale un ambiente può contenere un altro ambiente in un rapporto gerarchico infinito[4]

2.5.3 Parentesi

Le parentesi graffe ({}) si utilizzano per creare un blocco o un insieme. Ciò che avviene in un blocco è isolato dal resto del testo, se invochiamo dei comandi che apportano modifiche questi esisteranno solo nel blocco. Un blocco può contenere un sottoblocco. Raggruppare un comando aiuta a renderizzarlo correttamente in caso di errori di visualizzazione.

[4]Può esistere un ambiente dentro un ambiente dentro un ambiente dentro un ambiente dentro un ambiente...

Chapter 3

Formattazione

3.1 Bold, Italic, Underline

Il testo scritto all'interno di un ambiente documento viene normalmente visualizzato come testo regolare. Per decorare il testo trasformandolo in grassetto (bold), corsivo (italic) o sottolinearlo (underline) è richiesto includere il testo in uno dei rispettivi comandi.

Bold Per tasformare il testo in grassetto è necessario l'uso del comando `\textbf{testo}`. Come dice il comando scrive il testo in "bold font". Si ottiene un effetto del genere **testo**.

Italic Per tasformare il testo in corsivo è necessario l'uso del comando `\textit{testo}`. Si ottiene un effetto del genere *testo*.

Underline Per sottolineare il testo si utilizza in LaTeX base il comando `\underline{testo}`. Si ottiene un effetto del genere <u>testo</u>. Underline però ha un problema, se vogliamo sottolineare un testo particolarmente lungo allora underline non supporterà l'andata a capo automatica del testo. Ciò si risolve utilizzando la libreria ulem e utilizzando al posto di underline, `\uline{a}`. Con una conoscenza più avanzata di LaTeX potete

usare il comando `\renewcommand{cmd}{def}` per rinominare un comando. In questa guida io utilizzo queste due istruzioni. La prima per importare la libreria (necessario), la seconda per ridefinire il comando underline in modo che si comporti come uline.

```
\usepackage[normalem]{ulem}

% se pensate vi possa tornare utile, copia incollatelo nel vostro progetto
\renewcommand{\underline}[1]{\uline{#1}}
```

3.2 Font

Un font è un file che definisce lo stile generico di un insieme di caratteri. Esistono vari file descrittivi di un font, i più utilizzati sono i `ttf` e i `otf`. I `ttf` sono `otf` raggruppati. Un `ttf` possiede generalmente il raggruppamento degli stili Regular, Bold, Italic e BoldItalic, `ttf` più complessi possiedono varianti anche sul peso che possiedono associate a un valore generalmente da 1 a 1000.

Per ognuno dei pesi è generalmente necessario specificare le 4 varianti: Regular, Bold, italic e BoldItalic. Non è necessario che un font definisca tutti i pesi.

Name	**Weight**
Thin / Hairline	100
Ultra/Extra light	200
Light	300
Normal / regular	400
Medium	500
Semi/Demi bold	600
Bold	700
Extra/Ultra bold	800
Heavy / Black	900
Extra/Ultra black	950

3.2.1 Definire un font

Usando il compilatore XeLaTeX è possibile utilizzare la libreria fontspec per definire un font da un file. Importata la libreria si utilizza `\newfontfamily` che prende come primo argomento il nome del comando che lo identificherà e come secondo argomento il nome del font o il suo percorso e come terzo argomento (opzionale) un listato di opzioni in forma `key=value`. Tra le opzioni che possiamo passare vi è la specifica dei file `otf` che compongono il font. Nell'esempio cerca il file nella cartella `./fonts/` (è importante lo \finale per la cartella) con il nome che finisce[1] in `-Regular` per il regular (UprightFont), che finisce con `-Bold` per il bold e così via. Il comando che creiamo con `\newfontfamily` necessita di iniziare con un solo \. Il comando cambia il font da lì in poi quindi è utile usarlo includendo il comando e il testo in una coppia di parentesi graffe che crea un'insieme isolato dal resto del lavoro.

```
\usepackage{fontspec}
\newfontfamily{\simplecustomfont}{./path/to/font}
\newfontfamily{\extendedcustomfont}{nomefont}[
    Path = ./fonts/,
    Extension = .otf,
    UprightFont = *-Regular,
    BoldFont = *-Bold,
    ItalicFont = *-Italic,
    BoldItalicFont = *-BoldItalic,
]
\begin{document}
{\simplecustomfont Regular text}
{\extendedcustomfont Regular text}
\end{document}
```

3.2.2 Usare un font

Il metodo base per usare un font è quello visto nella sezione precedente che consiste nel chiamarlo utilizzando il comando definito. Il comando cambia il font in uso da lì in avanti ed è spesso utile porlo all'interno di un blocco di parentesi

[1] Il simbolo `*` è un wildcard, generalmente in un pattern indica una serie di qualsiasi carattere, scrivere `*testo` vuol dire che si cerca qualcosa che finisce con "testo"

graffe. Per impostare un font che vale come default per l'intero documento usiamo `\setmainfont{path/to/font}` o

```
\setmainfont{OpenDyslexic}[
  Path = fonts/OpenDyslexic/,
  Extension = .otf,
  UprightFont = *-Regular,
  BoldFont = *-Bold,
  ItalicFont = *-Italic,
  BoldItalicFont = *-BoldItalic
]
```

Inoltre la libreria fontspec ha una lunga lista di font preinstallati. Potete scaricare i file `ttf` o `otf` da siti come Google Fonts, DaFont e 1001fonts oppure ricercare font simili a quello che desiderate usando siti come WhatTheFont

3.3 Colori

Il miglior modo di usare dei colori in LaTeX è importare la libreria `xcolor`. Gli argomenti opzionali comma-separated[2] di questa libreria si possono raggruppare in due tipi: "nomi dei colori" e "tipo di colore". La libreria `xcolor` di base presenta 19 colori, aggiungendo un argomento opzionale del tipo "nomi dei colori", forziamo la libreria ad ampliare la lista di colori disponibili. Esempi di liste disponibili sono `svgnames`[3], `dvipsnames`[4] e `table`. Con l'argomento di "tipo di colore" definiamo come la libreria `xcolor` gestisce i colori. I "tipi di colore" disponibili sono: natural (non esegue alcuna modifica), rgb[5], cmyk[6], gray (trasforma tutto in toni di grigio).

[2] Gli argomenti sono passati come una lista di parole divise da una virgola. es. [a, b, c, d]

[3] SVG è un formato di file molto usato nei disegni vettoriali. Un disegno vettoriale ha la peculiarietà di essere facilmente scalabile in dimensione ed è la scelta preferita per la rappresentazione delle icone

[4] dvi è un formato intermedio di latex, ps fa riferimento a un vecchio linguaggio di Adobe per la stampa di alcuni file. Alcune librerie potrebberlo richiederlo, tuttavia non è un obbligo importarlo

[5] Red Green Blue è il formato usato dai display. Ottimo per i pdf in formato digitale

[6] Cyan Magenta Yellow Key è il formato usato dalle stampanti. Ottimo per i pdf da stampare

```
\usepackage[svgnames, dvipsnames, table, rgb]{xcolor}
```

3.3.1 Definire un colore

La libreria `xcolor` permette di definire un colore personalizzato nel formato che si desidera.

RGB Lo standard RGB prevede il definire un colore mediante la specifica di 3 valori interi da 0 a 255 rispettivamente indicanti le intensità del rosso, del verde e del blu. Il comando da utilizzare è `\definecolor{nome}{RGB}{0,100,200}`.

rgb Lo standard è lo stesso del RGB ma prevede l'uso di 3 da 0 a 1 ottenuti come $valore_rgb = valore_RGB/255$. Il comando da utilizzare è `\definecolor{nome}{RGB}{0,0.39,0.78}`.

HTML Lo standard di HTML[7] è una rappresentazione RGB in formato HEX[8]. Ognuno dei 3 valori è scritto come una coppia di cifre esadecimali[9] da $0_{10} = 00_{16}$ a $255_{10} = FF_{16}$. Il comando da utilizzare è `\definecolor{nome}{HTML}{0064C8}`

cmyk Lo standard cmyk è quello delle stampanti, prevede l'uso di 4 valori da 0 a 1 facenti riferimento alla quantità di colore ciano, magenta e giallo e, relativamente al parametro key, all'intesità dell'insieme. Non è molto usato per la definizione di colori. Il comando da utilizzare è `\definecolor{nome}{cmyk}{1.000, 0.500, 0.000, 0.216}`

gray Lo standard gray è il monocromatico, prevede l'uso di un solo valore da 0 a 1 facente riferimento all'intensità del grigio. Il comando da utilizzare è `\definecolor{nome}{cmyk}{0.39}`

[7] Linguaggio di marcatura per la creazione di pagine web statiche, ovvero prive di animazioni o interazioni

[8] Esadecimale, ovvero base16. Base16 vuol dire che le cifre che abbiamo a disposizione non sono 10 bensì 16. Le cifre sono 0123456789ABCDEF. A volte si presenta con un hashtag iniziale o preceduto dalla sigla 0x, per latex né l'una né l'altra sono necessarie

[9] 0123456789ABCDEF

Mix di colori La libreria `xcolor` permette la definizione di un colore sulla base della miscelazione di colori esistenti. Il comando da utilizzare è `\colorlet{nome}{nomecolore!percentuale!n` Come potete notare si alternano un nome del colore e una percentuale divisi da punti esclamativi, l'ultimo colore può non avere percentuale e verrà per differenza dal totale calcolata in automatico. Potete miscelare tra loro $n = 1, 2, 3 \ldots$ colori. La percentuale è solo un numero, privo del simbolo `%`.

Nota

Definire un colore secondo un formato è non correlato all'imposizione data nell'importazione della libreria. Qualora il tipo definito nell'importazione della libreria non dovesse coincidere con quello della definizione, sarà la libreria stessa a riadattare verso quanto definito dal formato definito nell'importazione. Ciò vuol dire che siete liberi di definire i colori come preferite, preferendo uno o l'altro formato o anche di definire differenti colori con differenti formati. Immaginate di avere un documento inizialmente pensato per solo formato digitale e di doverlo stampare, sarebbe impensabile dover andare a ridefinire tutti i colori usati ed infatti ciò non avviene, basterà invece modificare la sola specifica nell'argomento opzionale della libreria e dunque ricompilare.

3.3.2 Uso di un colore

La libreria `xcolor` permette di cambiare colore al testo, allo sfondo o di creare un riquadro intorno a quest'ultimo. Inoltre i colori definiti con la libreria `xcolor` sono compatibili con la maggiorparte delle librerie di LaTeX.

Cambiare colore al testo Per scrivere del testo in un colore particolare si preferisce usare il comando `\textcolor{nomecolore}{testo}`. Un'altra opzione è usare il comando `\color{nomecolore}` che cambia il colore di default del testo dalla sua esecuzione in poi. È possibile usare il comando `\color{nomecolore}` all'interno di un blocco di parentesi graffe in modo da effettuare la modifica per il solo contenuto del blocco. La differenza tra il comando `{\color{nomecolore} testo}` e `\textcolor{nomecolore}{testo}` è

che mentre il primo non dovrebbe avere problemi, il secondo si aspetta di avere come argomento ”testo” perlappunto solo testo e potrebbe avere problemi di incompatibilità quando usato insieme a comandi più particolari come la creazione di una tabella, di una lista.

Cambiare colore allo sfondo

Chapter 4

Liste

Chapter
5

Tabelle

Chapter 6

Immagini

6.1 Figure

6.2 Tikz

Chapter 7

Equazioni