Premier document

I. L'apprentissage par la pratique

A. Écrire un fichier source

Dans la pratique, ce que l'on écrit avec TeXstudio se nomme le *fichier source* (ou encore le *code source* ou tout simplement la *source*). Ce fichier est compilé par LaTeX (plusieurs fois si nécessaire) et les différentes lignes de code sont interprétées pour obtenir en bout de chaîne le document final, celui qui sera imprimé.

Ouvrir un nouveau document avec TeXstudio et écrire le code suivant en respectant scrupuleusement les espaces et saut de lignes.

> Le symbole \ s'obtient avec la combinaison de touche `AltGr-8`. Les crochets s'obtiennent avec `AltGr-5` et `AltGr-°` et les accolades avec `AltGr-4` et `AltGr-+`

Premier document

```
1  \documentclass[12pt,french]{article}
2  \usepackage[utf8]{inputenc}
3  \usepackage[T1]{fontenc}
4  \usepackage{kpfonts}
5  \usepackage[a4paper,margin=2cm]{geometry}
6  \usepackage{mathtools,amssymb}
7  \usepackage{babel}
8
9  \begin{document}
10 \textbf{Définition 1.} Pour tous réels $a$, $b$ et $c \neq 0$, on a :\par
11 $\frac{a}{c} + \frac b c = \frac{a+b}{c}$. % Facile !
12
13 \textbf{Définition 2.} Soit $x \geqslant 0$ et $A \geqslant 0$ :
14 $\sqrt x = A \Leftrightarrow A^2 = x$. % Evident !
15
16
17 $\sqrt[3]{x}    =   A \Leftrightarrow A^3 = x$. % marche aussi avec x < 0 !
18
19 \'Evidemment,
20 ce
21 n'est
22 pas                bien                  compliqué.
23 \end{document}
24 C'est vraiment trop bien !!
```

Code I.1

Sauvegarder le document en l'appelant par exemple `Premier-Document.tex` puis compiler à l'aide de `F1`. Observer le résultat.

> En règle générale, dans les noms des documents, on évitera d'utiliser des espaces et des lettres accentuées.

B. Premières questions

1°) À quoi sert le symbole $?

2°) À quoi sert le symbole % ?

3°) À quoi sert la commande \par de la ligne 10 ?

4°) Que fait la commande \' de la ligne 19 ? Peut-on avoir \'P ?

5°) À quoi correspond un simple changement de ligne dans le *code source* ?

6°) Dans le *code source*, lorsque plusieurs espaces séparent deux mots, que se passe-t-il dans le document final ?

7°) Que fait une ligne laissée vide dans le *code source* ? Et deux lignes laissées vides ?

8°) Quel pourrait être l'intérêt de tout cela ?

II. Explication rapide du code source

A. Structure du code source

Le code source est divisé en plusieurs parties :

La définition du document : la première ligne permet de déterminer quel type de document est réalisé : on parle de la *classe* du document (d'où le nom `\documentclass`). Ici, il s'agit d'un document de type article. Les *options globales* sont également déterminées à ce moment : elles sont valables pour tout le document sauf indication contraire.

Le préambule : il s'agit des lignes entre `\documentclass` et `\begin{document}` (lignes 2 à 6). Le préambule contient tous les *packages* utilisés ainsi que différentes *commandes* définies par l'utilisateur ou spécifiquement utilisées dans le préambule.

Le corps du document : situé entre `\begin{document}` et `\end{document}`, il s'agit du contenu même du document qui sera alors formaté en fonction du contenu du préambule et des commandes utilisées dans le texte.

Après : les lignes après `\end{document}` ne sont pas interprétées par LaTeX et peuvent donc contenir ce que l'on veut : des commentaires, des notes, des parties mises de côté…

B. Explication du préambule

`\documentclass[12pt,french]{article}` : cette commande indique que le document est de classe article et sera donc assez court. Il existe, en comparaison, la classe book pour écrire des documents plus longs. Bien d'autres classes existent (letter, beamer,…).

Ce document respectera la typographie française et la taille des fontes sera de `12pt` (`10pt` étant la taille par défaut).

`\usepackage[utf8]{inputenc}` : cette commande permet de charger le *package* inputenc avec l'option `utf8`. Nous n'expliquerons rien en détails ici mais cela gère le *codage* d'entrée des caractères du *code source* (d'où l'intérêt d'avoir configuré TeXstudio en `utf8`).

`\usepackage[T1]{fontenc}` : cette commande permet de charger le *package* fontenc avec l'option `T1` permettant de gérer, entre autre, les caractères accentués ainsi les « copiés-collés » à partir de fichiers PDF.

`\usepackage{kpfonts}` : charge un ensemble de fontes de la police *Kp-Fonts*. D'autres sont disponibles comme par exemple mathpazo pour la police *Palatino* ou bien mathptmx pour la police *Times*.

`\usepackage[a4paper,margin=2cm]{geometry}` : charge le package geometry avec différentes options de mise en page. Nous détaillerons certaines possibilités dans une prochaine fiche.

`\usepackage{mathtools,amssymb}` : charge le *package* mathtools qui est essentiel dans un document destiné à composer des textes scientifiques, avec un formalisme et donc une mise en page particulière. Le *package* amssymb regroupe quant à lui quantité de symboles utilisés notamment en mathématiques et en physiques.

`\usepackage{babel}` : obligatoirement le dernier de la liste. Le *package* babel permet d'assurer au rédacteur que le texte sera composé en respectant les usages propres à la langue de composition du document (ici en français). La langue peut être spécifiée en option de ce *package* en écrivant `\usepackage[french]{babel}` mais il est préférable d'indiquer l'option de langue avec la *classe* du document (comme nous l'avons fait). Ainsi, cette langue sera utilisée de façon globale par tous les *packages* en ayant besoin.

III. Commandes

A. Arguments d'une commande

Les *commandes* permettent de structurer et de mettre en forme le document. Elles sont reconnaissables car elles commencent par le caractère \ suivi du nom de la commande (plus ou moins explicite). La commande se termine par tout caractère autre qu'une lettre (accolade, crochet, chiffre, espace, ponctuation…).

Différents types de commandes existent :

Sans *paramètres* : elles exécutent simplement une action : `\neq`, `\par`, `\Leftrightarrow`

Avec *paramètres* : il existe alors deux types de *paramètres*. Ils peuvent être :

Obligatoires : ceux-là sont notés entre accolades et il peut y avoir plusieurs paramètres par commande (une paire d'accolades pour chacun d'entre eux). Par exemple, la commande `\textbf` ne possède qu'un paramètre alors que la commande `\frac` en possède deux. Il a été dit que la commande s'arrêtait par tout caractère autre qu'une lettre. Dans ce cas, il n'est pas toujours nécessaire d'utiliser les accolades. Ainsi, `\frac{2}{3}` ⇔ `\frac23` ⇔ `\frac 2 3` : cela donne la fraction $\frac{2}{3}$. En revanche, écrire `\fracab` ne donnera pas $\frac{a}{b}$. Pourquoi ?

Facultatifs : ceux-là sont notés entre crochets avant le premier argument obligatoire (qui n'existe pas toujours d'ailleurs). Les *arguments* facultatifs ou *optionnels* permettent de modifier localement l'action d'une commande. Par exemple `\sqrt[3]{x}`.

En résumé, une commande peut avoir une des trois formes suivantes :
`\commande`
`\commande{`⟨*argument*$_1$⟩`}{`⟨*argument*$_2$⟩`}`...
`\commande[`⟨*argument optionnel*⟩`]{`⟨*argument*$_1$⟩`}{`⟨*argument*$_2$⟩`}`...

> Les majuscules dans les noms de commandes sont importantes : ainsi `\frac` n'est pas identique à `\Frac`.

B. Commandes semi-globales

Les *commandes locales* permettent de modifier l'aspect du texte de façon locale.

Les *commandes semi-globales* n'ont pas d'argument et modifient tout le texte qui suit jusqu'à ce qu'une autre commande semi-globale ou qu'une commande locale ne modifie encore la mise en forme. On peut limiter l'action des commandes semi-globales à l'aide d'une paire d'accolades englobant le texte mais également la commande. Autrement dit :

Commande locale : `\commande{`⟨*du texte*⟩`}`

Commande semi-globale : `{\commande` ⟨*du texte*⟩`}`

La plupart du temps, on utilise les commandes locales sur des textes courts sans changement de paragraphes alors que les commandes semi-globales sont appliquées à des textes plus longs et acceptent les changements de paragraphes. Voici un exemple :

Voici un **exemple :** tout va bien mais on peut vouloir continuer avec un texte plus petit jusqu'au bout.
Cela est important !
Comprenez-vous ? C'est bien !

```
1 Voici un \textbf{exemple :}
2 tout va bien mais on peut vouloir
3 continuer avec un texte
4 \tiny plus petit jusqu'au bout.\par
5 {\bfseries
6     Cela est important !\par
7     Comprenez-vous ?
8 }
9 C'est bien !
```

C. Les packages

Les *packages* sont chargés à l'aide de la commande `\usepackage`. Chaque *package* est une *extension* de LaTeX et c'est la création de ces milliers de *packages* qui fait que LaTeX évolue de jours en jours. Le nom du *package* est l'argument obligatoire et les options sont spécifiées entre crochets si nécessaire :

`\usepackage[`⟨*options*⟩`]{`⟨*nom du package*⟩`}`

Si plusieurs *packages* doivent être appelés sans option particulière (ou avec la même option), alors on peut les lister au sein de la même commande `\usepackage`. C'est le cas par exemple de la ligne 4 : `\usepackage{mathtools,amssymb}` fait appel à deux *packages* liés aux mathématiques.

> **Rappel :** Le *package* babel est le *package* qui permet la gestion de la langue dans laquelle est écrite le document. C'est d'ailleurs la fonctionnalité de `french` signalée dans la *commande* `\documentclass`.
> Le *package* babel doit être le dernier de la liste des *packages* utilisés (sauf exception).

IV. Les caractères spéciaux

On a vu que certains caractères avaient une utilisation spécifique dans le code source : c'est le cas par exemple des caractères `%` et `$` qui permettent respectivement d'entrer un commentaire et une formule mathématique. Comment faire cependant pour écrire -20% sur une veste à 50$ ou bien S = { 1 ; 3 } ?

Le tableau ci-dessous résume tous les caractères spéciaux réservés par LaTeX, leur rôle et la syntaxe nécessaire pour les utiliser dans un texte classique.

Caractère	Rôle spécifique	Syntaxe	Résultat	Exemple
`\`	commande	`\textbackslash`	\	C :\programs
`{`	délimiteur ouvrant	`{`	{	S = { 1 ; 3 }
`}`	délimiteur fermant	`}`	}	S = { 1 ; 3 }
`%`	commentaire	`\%`	%	50 %
`$`	mode mathématique	`\$`	$	50 $
`^`	exposant	`\^{}`	^	p̂
`_`	indice	`\_`	_	mon_mail
`&`	tableau	`\&`	&	Bob & Co.
`#`	commande personnel	`\#`	#	C#m
`~`	espace insécable	`\~{}`	~	p̃

> Le caractère `@` n'est pas un caractère spécial et permet d'obtenir simplement @. Cependant, dans certains cas, il est utilisé de façon particulière : pour les tableaux, les commandes personnelles...

V. Exercice

Écrire un *fichier source* complet permettant d'obtenir le document encadré ci-dessous.

Indications. Le mot Exercice est écrit en gras à l'aide de la *commande locale* `\textbf` et a été agrandi à l'aide de la *commande semi-globale* `\Large`.

La numérotation des questions est en italique à l'aide de la *commande locale* `\textit`.

Exercice

1. Soit f_1 la fonction définie pour tout réel x par $f_1(x) = 3x^2 + 2x - 1$.
Calculer $f_1(-2)$.

2. Écrire la forme simplifiée de l'expression suivante : $\frac{3+\frac{1}{2}}{\frac{3}{4}-1}$.