Rédaction mathématique et informatique

Organisation du module

- 4 enseignants, soit 1 par groupe :

  MI1 : Valentin GUIEN
  MI2 : Diego PERDIGAO MARTINO
  MI3 : Valentin GUIEN
  MI4 : Diego PERDIGAO MARTINO
  MI5 : Bruno GUILLON *(responsable du cours)*
  MI6 : Diego PERDIGAO MARTINO
  MI7 : Anthony MEUNIER

Organisation du module

- 4 enseignants, soit 1 par groupe :

  MI1 : Valentin GUIEN
  MI2 : Diego PERDIGAO MARTINO
  MI3 : Valentin GUIEN
  MI4 : Diego PERDIGAO MARTINO
  MI5 : Bruno GUILLON *(responsable du cours)*
  MI6 : Diego PERDIGAO MARTINO
  MI7 : Anthony MEUNIER

- 5 TD et 5 TP (alternance)

Organisation du module

- 4 enseignants, soit 1 par groupe :

  MI1 : Valentin GUIEN
  MI2 : Diego PERDIGAO MARTINO
  MI3 : Valentin GUIEN
  MI4 : Diego PERDIGAO MARTINO
  MI5 : Bruno GUILLON *(responsable du cours)*
  MI6 : Diego PERDIGAO MARTINO
  MI7 : Anthony MEUNIER

- 5 TD et 5 TP (alternance) +1 TP noté +2 TD (rappels math)
- 100% contrôle continu
  - 1 TP noté, individuel – 50%
  - 1 projet (rendu asynchrone), en binôme – 50%

Organisation du module

- 4 enseignants, soit 1 par groupe :

  MI1 : Valentin GUIEN
  MI2 : Diego PERDIGAO MARTINO
  MI3 : Valentin GUIEN
  MI4 : Diego PERDIGAO MARTINO
  MI5 : Bruno GUILLON *(responsable du cours)*
  MI6 : Diego PERDIGAO MARTINO
  MI7 : Anthony MEUNIER

- 5 TD et 5 TP (alternance) +1 TP noté +2 TD (rappels math)
- 100% contrôle continu
  - 1 TP noté, individuel – 50%
  - 1 projet (rendu asynchrone), en binôme – 50%
- 1 espace Moodle (documentation, annonces, liens...)

Dans ce module

Manipulation de deux outils logiciels :

- LaTeX : un “moteur de formatage”, un format, un langage
- git : un gestionnaire de version (pas spécifique à la rédaction)

Dans ce module

Manipulation de deux outils logiciels :

- LaTeX : un “moteur de formatage”, un format, un langage
- git : un gestionnaire de version (pas spécifique à la rédaction)

Dans ce module

Manipulation de deux outils logiciels :

- LaTeX : un “moteur de formatage”, un format, un langage
- git : un gestionnaire de version (pas spécifique à la rédaction)

Dans ce module

Manipulation de deux outils logiciels :

- LaTeX : un “moteur de formatage”, un format, un langage
- git : un gestionnaire de version (pas spécifique à la rédaction)

Découvertes de concepts standards de l’informatique :

- séparation contenu/forme
- compilation, fichier source

Développement de compétences transverses :

- rédaction, écriture de preuve, raisonnement logique
- collaboration, synchronisation et archivage
- utilisation du terminal (Shell)

Dans ce module

Manipulation de deux outils logiciels :

- LaTeX : un “moteur de formatage”, un format, un langage
- git : un gestionnaire de version (pas spécifique à la rédaction)

Découvertes de concepts standards de l’informatique :

- séparation contenu/forme
- compilation, fichier source

Développement de compétences transverses :

- rédaction, écriture de preuve, raisonnement logique
- collaboration, synchronisation et archivage
- utilisation du terminal (Shell)

Suite du module

- l’UE projet (Python) :
  - utilisation de git (gestion du projet)
  - utilisation de LaTeX (rédaction du rapport)
- utilisation personnelle

Aujourd'hui

1. Présentation générale & organisation du module
2. Prérequis
3. Le traitement de texte
4. Initiation à LaTeX

Aujourd'hui

1. Présentation générale & organisation du module
2. Prérequis
3. Le traitement de texte
4. Initiation à LaTeX

Aujourd'hui

1. Présentation générale & organisation du module
2. Prérequis
3. Le traitement de texte
4. Initiation à LaTeX

Prérequis

- base du système de fichier
- interaction en terminal (*c.f.*, cours Shell)
- preuves mathématiques
- recherche et lecture de documentation

Prérequis

- base du système de fichier
- interaction en terminal (*c.f.*, cours Shell)
- preuves mathématiques exercices !
- recherche et lecture de documentation

Système de fichier

- **répertoire** (ou dossiers, en anglais *directory*)
  contient des répertoires et des fichiers $\Longrightarrow$ organisation

Système de fichier

- **répertoire** (ou dossiers, en anglais *directory*)
  contient des répertoires et des fichiers $\Longrightarrow$ organisation

  Conseil: *créez un répertoire pour chaque TP:* ‘`redacMI_tp1/`’.

Système de fichier

- **répertoire** (ou dossiers, en anglais *directory*)
  contient des répertoires et des fichiers $\Longrightarrow$ organisation

  Conseil: *créez un répertoire pour chaque TP:* ‘`redacMI/tp1/`’.

Système de fichier

- **répertoire** (ou dossiers, en anglais *directory*)
  contient des répertoires et des fichiers $\Longrightarrow$ organisation

  Conseil: *créez un répertoire pour chaque TP: '*`redacMI/tp1/`*'.*

- **fichier régulier**
  - **binaire :** s'exécute, ou bien s'ouvre avec des logiciels spécifiques
  - **texte :** contient du texte lisible, par exemple du code
    on l'ouvre avec des *éditeurs de texte*

Système de fichier

- **répertoire** (ou dossiers, en anglais *directory*)
  contient des répertoires et des fichiers $\Longrightarrow$ organisation

  Conseil: *créez un répertoire pour chaque TP: '*`redacMI/tp1/`*'.*

- **fichier régulier**
  - **binaire :** s'exécute, ou bien s'ouvre avec des logiciels spécifiques
  - **texte :** contient du texte lisible, par exemple du code
    on l'ouvre avec des *éditeurs de texte*

- **extension**
  partie finale d'un nom de fichier (suffixe) commençant par '`.`'
  e.g., `.txt`, `.ods`, `.doc`, `.pdf`, `.html`, `.conf`
  et indiquant :
  - quel type de données est contenu dans le fichier ?
  - quel logiciel utiliser pour l'ouvrir/l'interpréter ?

Utilisation du terminal

- le *répertoire courant* (ou *de travail*)

```
$ pwd
```

Utilisation du terminal

- le *répertoire courant* (ou *de travail*)

```
$ pwd
```

- les chemins relatifs et absolus

```
./file
file
../file
```

versus

```
/home/login/user/file
~/documents/file
$HOME/file
```

Utilisation du terminal

- le *répertoire courant* (ou *de travail*)

```
$ pwd
```

- les chemins relatifs et absolus

```
./file
file
../file
```

versus

```
/home/login/user/file
~/documents/file
$HOME/file
```

- la forme standard des lignes de commandes simples :

```
$ cmd [opts ...] [args ...]
```

Utilisation du terminal

- le *répertoire courant* (ou *de travail*) `$ pwd`
- les chemins relatifs et absolus

```
./file
file
../file
```

versus

```
/home/login/user/file
~/documents/file
$HOME/file
```

- la forme standard des lignes de commandes simples :

```
$ cmd [opts ...] [args ...]
```

Exemple

- `$ ls`
- `$ ls -A -l` ou `$ ls -Al`
- `$ cp -iv model.tex ../redacMI/tp1/`
- `$ cd ~/Téléchargements`
- `$ cat haut.txt bas.txt`

Utilisation du terminal

- le *répertoire courant* (ou *de travail*)

```
$ pwd
```

- les chemins relatifs et absolus

```
./file
file
../file
```

versus

```
/home/login/user/file
~/documents/file
$HOME/file
```

- la forme standard des lignes de commandes simples :

```
$ cmd [opts ...] [args ...]
```

on verra aussi

```
$ cmd subcmd [opts ... ] [args ...]
```

Utilisation du terminal

- le *répertoire courant* (ou *de travail*) `$ pwd`
- les chemins relatifs et absolus

```
./file
file
../file
```

versus

```
/home/login/user/file
~/documents/file
$HOME/file
```

- la forme standard des lignes de commandes simples :

```
$ cmd [opts ...] [args ...]
```

on verra aussi `$ cmd subcmd [opts ... ] [args ...]`

- les options communes (`--help`/`-h`, `--verbose`/`-v`, `--`, ...)

Utilisation du terminal

- le *répertoire courant* (ou *de travail*) `$ pwd`
- les chemins relatifs et absolus

```
./file
file
../file
```

versus

```
/home/login/user/file
~/documents/file
$HOME/file
```

- la forme standard des lignes de commandes simples :

```
$ cmd [opts ...] [args ...]
```

on verra aussi `$ cmd subcmd [opts ... ] [args ...]`

- les options communes (`--help`/`-h`, `--verbose`/`-v`, `--`, ...)
- les commandes communes *c.f.*, slide suivant

Les commandes Shell communes

- `cd` *(change directory)* : changer de répertoire courant
- `pwd` *(print working directory)* : voir où l'on est
- `ls` *(list)* : lister les fichiers d'un répertoire
- `cp` *(copy)* : copier
- `mv` *(move)* : bouger/renommer
- `rm` *(remove)* : supprimer
- `mkdir` *(make directory)* : créer un répertoire
- `rmdir` *(remove directory)* : supprimer un répertoire vide
- `cat` *(concatenate)* : afficher le contenu de fichiers (texte)
- `man` *(manual)* : documentation

Les commandes Shell communes

- `cd` *(change directory)* : changer de répertoire courant
- `pwd` *(print working directory)* : voir où l'on est
- `ls` *(list)* : lister les fichiers d'un répertoire
- `cp` *(copy)* : copier
- `mv` *(move)* : bouger/renommer
- `rm` *(remove)* : supprimer
- `mkdir` *(make directory)* : créer un répertoire
- `rmdir` *(remove directory)* : supprimer un répertoire vide
- `cat` *(concatenate)* : afficher le contenu de fichiers (texte)
- `man` *(manual)* : documentation

- `find` (trouver) : trouver un fichier
- `grep` (attraper) : rechercher une expression dans un fichier

Quantificateurs

$\forall$: *“pour tout”*

$\exists$: *“il existe”*

Quantificateurs

$\forall$: *"pour tout"* $\exists$: *"il existe"*

Exercice

Soit $\mathcal{P}$ une population
et $x \hookrightarrow y$ la relation de parenté (*"x est père de y"*) sur $\mathcal{P}$.

Quantificateurs

$\forall$: *"pour tout"* $\exists$: *"il existe"*

Exercice

Soit $\mathcal{P}$ une population

et $x \hookrightarrow y$ la relation de parenté (*"x est père de y"*) sur $\mathcal{P}$.

Que signifient les formules suivantes ?

1. $\forall x \in P, \exists y \in P, x \hookrightarrow y$.
2. $\forall x \in P, \exists y \in P, y \hookrightarrow x$.
3. $\exists x \in P, \exists y \in P, x \hookrightarrow y$.
4. $\exists x \in P, \forall y \in P, x \hookrightarrow y$.
5. $\exists x \in P, x \hookrightarrow x$.

Quantificateurs

$\forall$: *"pour tout"* $\exists$: *"il existe"*

Exercice

Soit $\mathcal{P}$ une population
et $x \hookrightarrow y$ la relation de parenté (*"x est père de y"*) sur $\mathcal{P}$.

Que signifient les formules suivantes ?

1. $\forall x \in P, \exists y \in P, x \hookrightarrow y$.
2. $\forall x \in P, \exists y \in P, y \hookrightarrow x$.
3. $\exists x \in P, \exists y \in P, x \hookrightarrow y$.
4. $\exists x \in P, \forall y \in P, x \hookrightarrow y$.
5. $\exists x \in P, x \hookrightarrow x$.

Comment exprimer les assertions suivantes ?

6. Toute personne admet un père.
7. Certaines personnes sont pères de plusieurs autres.
8. Aucune personne n'a deux pères.
9. Personne n'est le père de son père.
10. La négation de l'assertion **6.**.

Preuve

Exercice

Montrez que pour tout entier naturel $n \in \mathbb{N}$, $2^{2n} - 1$ est divisible par 3.

Preuve

autrement dit: $\forall n \in \mathbb{N}$, $\exists k \in \mathbb{N}$ tel que $2^{2n} - 1 = 3k$

Exercice

Montrez que pour tout entier naturel $n \in \mathbb{N}$, $2^{2n} - 1$ est divisible par 3.

Preuve

autrement dit: $\forall n \in \mathbb{N}, \exists k \in \mathbb{N}$ tel que $2^{2n} - 1 = 3k$

Exercice

Montrez que pour tout entier naturel $n \in \mathbb{N}$, $2^{2n} - 1$ est divisible par 3.

(à vous de jouer)

Preuve

autrement dit: $\forall n \in \mathbb{N}, \exists k \in \mathbb{N}$ tel que $2^{2n} - 1 = 3k$

Exercice

Montrez que pour tout entier naturel $n \in \mathbb{N}$, $2^{2n} - 1$ est divisible par 3.

Démonstration

Nous montrons cette propriété en procédant par récurrence sur n.

(à vous de jouer)

Preuve

autrement dit: $\forall n \in \mathbb{N}, \exists k \in \mathbb{N}$ tel que $2^{2n} - 1 = 3k$

Exercice

Montrez que pour tout entier naturel $n \in \mathbb{N}$, $2^{2n} - 1$ est divisible par 3.

Démonstration

Nous montrons cette propriété en procédant par récurrence sur n.

Initialisation :

(à vous de jouer)

Hérédité :

Preuve

autrement dit: $\forall n \in \mathbb{N}, \exists k \in \mathbb{N}$ tel que $2^{2n} - 1 = 3k$

Exercice

Montrez que pour tout entier naturel $n \in \mathbb{N}$, $2^{2n} - 1$ est divisible par 3.

Démonstration

Nous montrons cette propriété en procédant par récurrence sur n.

Initialisation : Considérons le cas $n = 0$.

Hérédité :

Preuve

autrement dit: $\forall n \in \mathbb{N}, \exists k \in \mathbb{N}$ tel que $2^{2n} - 1 = 3k$

Exercice

Montrez que pour tout entier naturel $n \in \mathbb{N}$, $2^{2n} - 1$ est divisible par 3.

Démonstration

Nous montrons cette propriété en procédant par récurrence sur n.

Initialisation : Considérons le cas $n = 0$. Puisque $2^{2n} = 2^0 = 1$, nous avons bien que $2^{2n} - 1 = 0 = 3 \times 0$ est divisible par 3.

Hérédité :

Preuve

autrement dit: $\forall n \in \mathbb{N}, \exists k \in \mathbb{N}$ tel que $2^{2n} - 1 = 3k$

Exercice

Montrez que pour tout entier naturel $n \in \mathbb{N}$, $2^{2n} - 1$ est divisible par 3.

Démonstration

Nous montrons cette propriété en procédant par récurrence sur n.

Initialisation : Considérons le cas $n = 0$. Puisque $2^{2n} = 2^0 = 1$, nous avons bien que $2^{2n} - 1 = 0 = 3 \times 0$ est divisible par 3.

Hérédité : Soit $n \in \mathbb{N}$. Supposons que $2^{2n} - 1$ est divisible par 3 et montrons qu'il en va de même pour $2^{2(n+1)} - 1$.

Preuve

autrement dit: $\forall n \in \mathbb{N}$, $\exists k \in \mathbb{N}$ tel que $2^{2n} - 1 = 3k$

Exercice

Montrez que pour tout entier naturel $n \in \mathbb{N}$, $2^{2n} - 1$ est divisible par 3.

Démonstration

Nous montrons cette propriété en procédant par récurrence sur n.

Initialisation : Considérons le cas $n = 0$. Puisque $2^{2n} = 2^0 = 1$, nous avons bien que $2^{2n} - 1 = 0 = 3 \times 0$ est divisible par 3.

Hérédité : Soit $n \in \mathbb{N}$. Supposons que $2^{2n} - 1$ est divisible par 3 et montrons qu'il en va de même pour $2^{2(n+1)} - 1$.

Nous avons $2^{2(n+1)} = 2^{2n+2} = 2^2 \times 2^{2n} = 4 \times 2^{2n}$. Or, par hypothèse de récurrence, nous pouvons fixer un entier $k \in \mathbb{N}$ tel que $2^{2n} - 1 = 3k$. D'où, $2^{2n} = 3k + 1$ et donc $2^{2(n+1)} = 4 \times (3k + 1) = 12k + 4$. Ainsi, nous obtenons $2^{2(n+1)} - 1 = 12k + 3 = 3 \times (4k + 1)$, qui est bien divisible par 3.

Preuve

autrement dit: $\forall n \in \mathbb{N}, \exists k \in \mathbb{N}$ tel que $2^{2n} - 1 = 3k$

Exercice

Montrez que pour tout entier naturel $n \in \mathbb{N}$, $2^{2n} - 1$ est divisible par 3.

Démonstration

Nous montrons cette propriété en procédant par récurrence sur n.

Initialisation : Considérons le cas $n = 0$. Puisque $2^{2n} = 2^0 = 1$, nous avons bien que $2^{2n} - 1 = 0 = 3 \times 0$ est divisible par 3.

Hérédité : Soit $n \in \mathbb{N}$. Supposons que $2^{2n} - 1$ est divisible par 3 et montrons qu'il en va de même pour $2^{2(n+1)} - 1$.

Nous avons $2^{2(n+1)} = 2^{2n+2} = 2^2 \times 2^{2n} = 4 \times 2^{2n}$. Or, par hypothèse de récurrence, nous pouvons fixer un entier $k \in \mathbb{N}$ tel que $2^{2n} - 1 = 3k$. D'où, $2^{2n} = 3k + 1$ et donc $2^{2(n+1)} = 4 \times (3k + 1) = 12k + 4$. Ainsi, nous obtenons $2^{2(n+1)} - 1 = 12k + 3 = 3 \times (4k + 1)$, qui est bien divisible par 3.

Ceci conclut la récurrence. □

Preuve

autrement dit: $\forall n \in \mathbb{N}, \exists k \in \mathbb{N}$ tel que $2^{2n} - 1 = 3k$

Exercice

Montrez que pour tout entier naturel $n \in \mathbb{N}$, $2^{2n} - 1$ est divisible par 3.

Démonstration

Nous montrons cette propriété en procédant par récurrence sur n.

Initialisation : Considérons le cas $n = 0$. Puisque $2^{2n} = 2^0 = 1$, nous avons bien que $2^{2n} - 1 = 0 = 3 \times 0$ est divisible par 3.

Hérédité : Soit $n \in \mathbb{N}$. Supposons que $2^{2n} - 1$ est divisible par 3 et montrons qu'il en va de même pour $2^{2(n+1)} - 1$.

Nous avons $2^{2(n+1)} = 2^{2n+2} = 2^2 \times 2^{2n} = 4 \times 2^{2n}$. Or, par hypothèse de récurrence, nous pouvons fixer un entier $k \in \mathbb{N}$ tel que $2^{2n} - 1 = 3k$. D'où, $2^{2n} = 3k + 1$ et donc $2^{2(n+1)} = 4 \times (3k + 1) = 12k + 4$. Ainsi, nous obtenons $2^{2(n+1)} - 1 = 12k + 3 = 3 \times (4k + 1)$, qui est bien divisible par 3.

Ceci conclut la récurrence. □

Aujourd'hui

1. Présentation générale & organisation du module
2. Prérequis
3. Le traitement de texte
4. Initiation à LaTeX

Traitement de texte

But : mettre en forme du texte

Traitement de texte

But : mettre en forme du texte

Il y a deux grandes familles de logiciels de traitement de texte :

1. WYSIWYG (*What You See Is What You Get*)
2. les *systèmes de composition*, qui font appel à un programme de rendu

Traitement de texte

But : mettre en forme du texte

Il y a deux grandes familles de logiciels de traitement de texte :

1. WYSIWYG (*What You See Is What You Get*)
2. les *systèmes de composition*, qui font appel à un programme de rendu

TEX fait parti de la seconde famille :

Traitement de texte

But : mettre en forme du texte

Il y a deux grandes familles de logiciels de traitement de texte :

1. WYSIWYG (*What You See Is What You Get*)
2. les *systèmes de composition*, qui font appel à un programme de rendu

TEX fait parti de la seconde famille :

on édite un fichier (*What You See*)
et on en obtient un autre (*What You Get*)

Traitement de texte

But : mettre en forme du texte

Il y a deux grandes familles de logiciels de traitement de texte :

1. WYSIWYG (*What You See Is What You Get*)
2. les *systèmes de composition*, qui font appel à un programme de rendu

TEX fait parti de la seconde famille :

on édite un fichier (*What You See*)
et on en obtient un autre (*What You Get*)
par le mécanisme de *compilation*.

Traitement de texte

But : mettre en forme du texte

Il y a deux grandes familles de logiciels de traitement de texte :

1. WYSIWYG (*What You See Is What You Get*)
2. les *systèmes de composition*, qui font appel à un programme de rendu

TEX fait parti de la seconde famille :

on édite un fichier (*What You See*)
et on en obtient un autre (*What You Get*)
par le mécanisme de *compilation*.

Donc, avec TEX, *What You See* ***IS NOT*** *What You Get*.

Compilation

compiler : transformer (ou traduire) un code source en code objet.

Compilation

compiler : transformer (ou traduire) un code source en code objet.

Dans le cas de TEX,

- le code source : fichier texte, d'extension `.tex`
- le code "objet" : fichier pdf, d'extension `.pdf`

Compilation

compiler : transformer (ou traduire) un code source en code objet.

Dans le cas de TEX,

- le code source : fichier texte, d'extension `.tex`
- le code "objet" : fichier pdf, d'extension `.pdf`

ou autre, *e.g.*, `.ps`, `.html`.

Pourquoi séparer ?

- un même fichier source (`.tex`) pour des rendus différents...
- automatisation de certaines tâches, par exemple :
  - table des matières générées
  - références dures $\Longrightarrow$ mises-à-jour automatiquement
  - utilisation de variables
  - styles cohérents (*e.g.*, tous les titres de sections sont large, gras, numérotés avec des lettres et soulignés ; élargir la taille de police du texte hors titre se fait facilement)
  - ...

Pourquoi séparer ?

- un même fichier source (`.tex`) pour des rendus différents...
- automatisation de certaines tâches, par exemple :
  - table des matières générées
  - références dures $\Longrightarrow$ mises-à-jour automatiquement
  - utilisation de variables
  - styles cohérents (*e.g.*, tous les titres de sections sont large, gras, numérotés avec des lettres et soulignés ; élargir la taille de police du texte hors titre se fait facilement)
  - ...
- accès facile à des caractères spéciaux via des commandes, notamment mathématiques (*e.g.*, $\frac{1}{2} \times \left(1 - \sum_{i=0}^{n} (\alpha_i + 1)^2\right)$).
- contrôle précis du rendu

Pourquoi séparer ?

- un même fichier source (`.tex`) pour des rendus différents...
- automatisation de certaines tâches, par exemple :
  - table des matières générées
  - références dures $\Longrightarrow$ mises-à-jour automatiquement
  - utilisation de variables
  - styles cohérents (*e.g.*, tous les titres de sections sont large, gras, numérotés avec des lettres et soulignés ; élargir la taille de police du texte hors titre se fait facilement)
  - ...
- accès facile à des caractères spéciaux via des commandes, notamment mathématiques (*e.g.*, $\frac{1}{2} \times \left(1 - \sum_{i=0}^{n} (\alpha_i + 1)^2\right)$).
- contrôle précis du rendu
- avec LaTeX, le rédacteur peut se concentrer sur l'essentiel, à savoir le contenu, et laisser le système gérer la forme

Contenu *vs* forme

Par exemple, je dis :

- “ceci est un titre de section”
- “ceci est mis-en-valeur”
- “ceci fait référence à l’élément identifié par `id_exemple`”
- “je veux la table des matières”

plutôt que :

- “ceci est en grand, en gras et numéroté”
- “ceci est en italique”
- “ceci fait référence à l’élément 3”
- “Partie 1 : LaTeX”, “Partie 2 : Git”...

Contenu *vs* forme

Par exemple, je dis :

- “ceci est un titre de section”
- “ceci est mis-en-valeur”
- “ceci fait référence à l’élément identifié par `id_exemple`”
- “je veux la table des matières”

plutôt que :

- “ceci est en grand, en gras et numéroté”
- “ceci est en italique”
- “ceci fait référence à l’élément 3”
- “Partie 1 : LaTeX”, “Partie 2 : Git”...

Je donne des indications supplémentaires sur le contenu :

- “ceci est une figure”, “un théorème”, “un exemple”
- “ceci est l’identifiant de cet élément référençable”
- “voici le nom de l’auteur du document”’
- ...

Aujourd'hui

1. Présentation générale & organisation du module
2. Prérequis
3. Le traitement de texte
4. Initiation à LaTeX

TEX ou LATEX ?

TEX ou LATEX ?

les deux.

TEX ou LATEX ?

les deux.

TEX : à la fois :

- un moteur (programme) de formatage
  (le programme qui réalise la compilation)
- un format (*plain-tex*) orienté forme
- un langage de programmation

TEX ou LATEX ?

les deux.

TEX : à la fois :

- un moteur (programme) de formatage
  (le programme qui réalise la compilation)
- un format (*plain-tex*) orienté forme
- un langage de programmation

LATEX : sur-couche de TEX

- un format **orienté contenu**
- un langage enrichi, plus accessible que TEX

TEX ou LATEX ?

les deux.

TEX : à la fois :

- un moteur (programme) de formatage
  (le programme qui réalise la compilation)
- un format (*plain-tex*) orienté forme
- un langage de programmation

LATEX : sur-couche de TEX

- un format **orienté contenu**
- un langage enrichi, plus accessible que TEX

dans ce module : LATEX

Concrètement

Un code LaTeX contient :

- du texte (bien entendu),
- des commandes, qui commencent par `\`
- et des caractères spéciaux (*e.g.*, `%`, `$`, `{`, `}`).

ceci est une barre oblique inversée (un *backslash*)

Concrètement

Un code LaTeX contient :

- du texte (bien entendu),
- des commandes, qui commencent par `\`
- et des caractères spéciaux (*e.g.*, `%`, `$`, `{`, `}`).

ceci est une barre oblique inversée (un *backslash*)

Exemple

code LaTeX	**rendu**
`J'aime \LaTeX.`	J'aime LaTeX.

Concrètement

Un code LaTeX contient :

- du texte (bien entendu),
- des commandes, qui commencent par `\`
- et des caractères spéciaux (*e.g.*, `%`, `$`, `{`, `}`).

ceci est une barre oblique inversée (un *backslash*)

Exemple

code LaTeX	**rendu**
`J'aime \LaTeX.`	J'aime LaTeX.
`Si $A$ alors $B$ s'écrit $A\implies B$.`	Si A alors B s'écrit $A \implies B$.

Concrètement

Un code LaTeX contient :

- du texte (bien entendu),
- des commandes, qui commencent par `\`
- et des caractères spéciaux (*e.g.*, `%`, `$`, `{`, `}`).

ceci est une barre oblique inversée (un *backslash*)

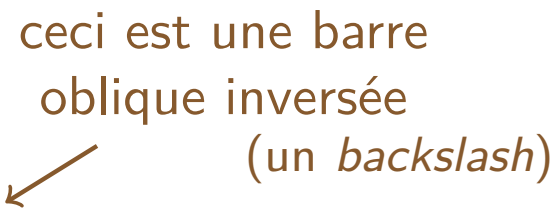
Exemple

code LaTeX	**rendu**
`J'aime \LaTeX.`	J'aime LaTeX.
`Si $A$ alors $B$ s'écrit $A\implies B$.`	Si A alors B s'écrit $A \implies B$.
`Visible. % invisible \LaTeX.`	Visible.

Concrètement

Un code LaTeX contient :

- du texte (bien entendu),
- des commandes, qui commencent par `\`
- et des caractères spéciaux (*e.g.*, `%`, `$`, `{`, `}`).



Exemple

code LaTeX	**rendu**
`J'aime \LaTeX.`	J'aime LaTeX.
`Si $A$ alors $B$ s'écrit $A\implies B$.`	Si A alors B s'écrit $A \implies B$.
`Visible. % invisible \LaTeX.`	Visible.
`Mettre en \emph{valeur}.`	Mettre en *valeur*.

Concrètement

Un code LaTeX contient :

- du texte (bien entendu),
- des commandes, qui commencent par `\`
- et des caractères spéciaux (*e.g.*, `%`, `$`, `{`, `}`).

ceci est une barre oblique inversée (un *backslash*)

Exemple

code LaTeX	**rendu**
`J'aime \LaTeX.`	J'aime LaTeX.
`Si $A$ alors $B$ s'écrit $A\implies B$.`	Si A alors B s'écrit $A \implies B$.
`Visible. % invisible \LaTeX.`	Visible.
`Mettre en \emph{valeur}.`	Mettre en *valeur*.

`\begin{example}`
`  Par exemple.`
`\end{example}`
	Exemple
	Par exemple.

Concrètement

Un code LaTeX contient :

- du texte (bien entendu),
- des commandes, qui commencent par `\`
- et des caractères spéciaux (*e.g.*, `%`, `$`, `{`, `}`).

ceci est une barre oblique inversée (un *backslash*)

Exemple

code LaTeX	**rendu**
`J'aime \LaTeX.`	J'aime LaTeX.
`Si $A$ alors $B$ s'écrit $A\implies B$.`	Si A alors B s'écrit $A \implies B$.
`Visible. % invisible \LaTeX.`	Visible.
`Mettre en \emph{valeur}.`	Mettre en *valeur*.

`\begin{example}`
`  Par exemple.`
`\end{example}`
	Exemple
	Par exemple.

Concrètement

Un code LaTeX contient :

- du texte (bien entendu),
- des commandes, qui commencent par `\`
- et des caractères spéciaux (*e.g.*, `%`, `$`, `{`, `}`).

ceci est une barre oblique inversée (un *backslash*)

Exemple

code LaTeX	**rendu**
`J'aime \LaTeX.`	J'aime LaTeX.
`Si $A$ alors $B$ s'écrit $A\implies B$.`	Si A alors B s'écrit $A \implies B$.
`Visible. % invisible \LaTeX.`	Visible.
`Mettre en \emph{valeur}.`	Mettre en *valeur*.

`\begin{example}`
`  Par exemple.`
`\end{example}`
	Exemple
	Par exemple.

Concrètement

Un code LaTeX contient :

- du texte (bien entendu),
- des commandes, qui commencent par `\`
- et des caractères spéciaux (*e.g.*, `%`, `$`, `{`, `}`).

ceci est une barre oblique inversée (un *backslash*)

Exemple

code LaTeX	**rendu**
`J'aime \LaTeX.`	J'aime LaTeX.
`Si $A$ alors $B$ s'écrit $A\implies B$.`	Si A alors B s'écrit $A \implies B$.
`Visible. % invisible \LaTeX.`	Visible.
`Mettre en \emph{valeur}.`	Mettre en *valeur*.

`\begin{example}`
`  Par exemple.`
`\end{example}`
	Exemple
	Par exemple.

Concrètement

Un code LaTeX contient :

- du texte (bien entendu),
- des commandes, qui commencent par `\`
- et des caractères spéciaux (*e.g.*, `%`, `$`, `{`, `}`).

ceci est une barre oblique inversée (un *backslash*)

Exemple

code LaTeX	**rendu**
`J'aime \LaTeX.`	J'aime LaTeX.
`Si $A$ alors $B$ s'écrit $A\implies B$.`	Si A alors B s'écrit $A \implies B$.
`Visible. % invisible \LaTeX.`	Visible.
`Mettre en \emph{valeur}.`	Mettre en *valeur*.

`\begin{example}`
`  Par exemple.`
`\end{example}`
	Exemple
	Par exemple.

Structure d'un document LaTeX

1. Un préambule

2. Un contenu

Structure d'un document LaTeX

1. Un préambule

2. Un contenu
   - commence par `\begin{document}`
   - termine par `\end{document}`

Structure d'un document LaTeX

1. Un préambule

2. Un contenu
   - commence par `\begin{document}`
   - contient le contenu du document (texte, *etc. . .*)
   - termine par `\end{document}`

Structure d'un document LaTeX

1. Un préambule
   - commence au début du fichier
   - se termine juste avant `\begin{document}`
2. Un contenu
   - commence par `\begin{document}`
   - contient le contenu du document (texte, *etc. . .*)
   - termine par `\end{document}`

Structure d'un document LaTeX

1. Un préambule
   - commence au début du fichier
   - indique `\documentclass[options]{nom-de-classe}`, généralement au début [a]
   - se termine juste avant `\begin{document}`
2. Un contenu
   - commence par `\begin{document}`
   - contient le contenu du document (texte, *etc. . .*)
   - termine par `\end{document}`

a. *classe LaTeX* : un format de base (*e.g.*, `article`, `report`, `book`, `slides`)

Structure d'un document LaTeX

1. Un préambule
   - commence au début du fichier
   - indique `\documentclass[options]{nom-de-classe}`, généralement au début [a]
   - contient des imports (`\usepackage{nom-de-librairie}`)
   - des définitions (de variable, de commande, de style, . . .)
   - des méta-données
   - se termine juste avant `\begin{document}`

   a. *classe LaTeX* : un format de base (*e.g.*, `article`, `report`, `book`, `slides`)
2. Un contenu
   - commence par `\begin{document}`
   - contient le contenu du document (texte, *etc.* . .)
   - termine par `\end{document}`

Structure d'un document LaTeX

1. Un préambule
   - commence au début du fichier
   - indique `\documentclass[options]{nom-de-classe}`, généralement au début [a]
   - contient des imports (`\usepackage{nom-de-librairie}`)
   - des définitions (de variable, de commande, de style, . . .)
   - des méta-données
   - se termine juste avant `\begin{document}`

   a. *classe LaTeX* : un format de base (*e.g.*, `article`, `report`, `book`, `slides`)
2. Un contenu
   - commence par `\begin{document}`
   - contient le contenu du document (texte, *etc.* . .)
   - termine par `\end{document}`
3. (Éventuellement du texte ignoré – *e.g.*, commentaires)
   - après `\end{document}`

Structure d'un document LaTeX

1. Un préambule
   - commence au début du fichier
   - indique `\documentclass[options]{nom-de-classe}`, généralement au début [a]
   - contient des imports (`\usepackage{nom-de-librairie}`)
   - des définitions (de variable, de commande, de style, . . .)
   - des méta-données
   - se termine juste avant `\begin{document}`

   a. *classe LaTeX* : un format de base (*e.g.*, `article`, `report`, `book`, `slides`)
2. Un contenu
   - commence par `\begin{document}`
   - contient le contenu du document (texte, *etc.* . .)
   - termine par `\end{document}`
3. (Éventuellement du texte ignoré – *e.g.*, commentaires)
   - après `\end{document}`

(oublions ceci)

Exemple

```
\documentclass{article}
\usepackage[french]{babel}
%le package babel gère le support linguistique.

\title{Je déclare le titre dans le préambule}
\author{Bruno GUILLON}
\date{Septembre 2022}

\begin{document}
\maketitle

J'ai utilisé la macro \verb|\maketitle|
pour générer l'entête de titre de mon article automatiquement,
à partir des méta-données préalablement indiquées en préambule
(titre, auteur et date).

\section{La classe article}
Cette classe définit des styles et certaines commandes
(notamment comment est rendu le titre).
\begin{center}
  Ici c'est centré et ce qui suit est en mode math:
  $\frac{3+4}{7}\times 8=8$.
\end{center}

\end{document}
```

Exemple

Je déclare le titre dans le préambule

Bruno GUILLON

Septembre 2022

J'ai utilisé la macro `\maketitle` pour générer l'entête de titre de mon article automatiquement, à partir des méta-données préalablement indiquées en préambule (titre, auteur et date).

1 La classe article

Cette classe définit des styles et certaines commandes (notamment comment est rendu le titre).

Ici c'est centré et ce qui suit est en mode math : $\frac{3+4}{7} \times 8 = 8$.

Semaine prochaine

- TP : manipulation du langage
  - quelques commandes les plus courantes
  - mode texte/mode math
  - gestion des retours à la ligne, paragraphes

Semaine prochaine

- TP : manipulation du langage
  - quelques commandes les plus courantes
  - mode texte/mode math
  - gestion des retours à la ligne, paragraphes

Défi

- Dans l'exemple précédent, à quoi servait la commande `\verb` ?