



EPITECH
INNOVATIVE
PROJECTS

La Vie est un Jeu !

Exemple de la classe LaTeX

Résumé: Ce document montre les différentes fonctionnalités de la documentclass EIP.



Le Groupe

Barbara Lepage db0company@gmail.com
Lepage Barbara lepage.barbara@gmail.com
Caradec Guillaume guillaume.caradec@gmail.com
Corsin Simon simoncorsin@gmail.com
Glorieux François fra.glorieux@gmail.com
Klarman Nicolas nickoas@gmail.com
Lassagne David david.lassagne@gmail.com
Louvigny Guillaume guillaume@louvigny.fr
El-Outmani Youssef youssef.eloutmani@gmail.com
Le-Cor Wilfried wilfried.lecor@gmail.com
Lenormand Frank lenormf@gmail.com



Informations du document

Type du document	Doc exemple
Nom du groupe	La Vie Est Un Jeu



Table des révisions

Version	Auteur	Commentaire	Date
0.0	Lepage Barbara	Exemple document class	08/01/2012
1.0	Lepage Barbara	Exemple mis à jour version cute pixel art back-ground	02/07/2012



Table des matières

I	Exemple de mise en forme	6
I.1	Lien	6
I.2	Mise en forme du texte	6
I.3	Insertion d'une image	7
II	D'autres exemples	8
II.1	De jolies boîtes	8
II.2	Une liste	10
III	Extrait d'un cours réel	11
III.1	Exercice 1	12
IV	Expressions et types	13
IV.1	Exercice 2	13
IV.2	Exercice 3	13
IV.3	Exercice 4	14
IV.4	Exercice 5	14



Chapitre I

Exemple de mise en forme

I.1 Lien

Ceci est un lien : <http://google.com>

Vous connaissez Google ?

I.2 Mise en forme du texte

texte en gras

texte en italique

texte souligné

texte normal



1.3 Insertion d'une image





Chapitre II

D'autres exemples

II.1 De jolies boîtes

Pour mettre en avant du texte :



Vous devez faire attention à ce message.



Ce message apporte une information.



Ceci est un conseil à suivre.



C'est terminé, bouclé, validé !



C'est en cours...



Ce n'est pas fait, c'est faux, c'est une négation.



II.2 Une liste

Et pour finir, une petite liste :

- élément
- autre élément
- Pourquoi pas une sous-liste ?
 - hello
 - world
- Et encore un élément !



Chapitre III

Extrait d'un cours réel

Vous devez utiliser l'interprete pour resoudre les exercices de ce TP. Je vous rappel que la commande shell pour lancer l'interprete OCaml est "ocaml".

Exemple :

```
1 >ocaml
2      Objective Caml version 3.10.2
3
4 #
```



Vous pouvez utiliser la commande `rlwrap` avec en parametre l'interprete `ocaml` pour beneficier d'une edition de ligne confortable.

La version d'ocaml que vous utilisez peut varier de celle de l'exemple, mais cela n'influra pas sur le contenu de ce module.

Pour quitter l'interprete, vous pouvez taper la directive "`#quit`" (avec le `#`) ou faire un `<ctrl> + d`.

Si vous voulez conserver le code que vous allez taper dans ce TP, vous pouvez l'ecrire dans un fichier et le charger dans l'interprete grace a la directive "`#use "fichier.ml"`" (avec le `#` et les doubles quotes autour du nom du fichier de sources). Cette directive va lire, compiler et evaluer votre fichier directement dans l'interprete.



Le symbole "`;;`" sert a marquer la fin d'une commande dans l'interprete `ocaml`. Il est donc inutile de le mettre a la fin de vos expressions dans votre fichier de sources.

Exemple :



```
1 >cat exemple.ml
2 let greetings = "Bonjour les tech2 !" (* Notez l'absence de ";" *)
3 >ocaml
4     Objective Caml version 3.11.1
5
6 # #use "exemple.ml";;
7 val greetings : string = "Bonjour les tech2 !"
8 # greetings;;
9 - : string = "Bonjour les tech2 !"
10 #
```

III.1 Exercice 1

- Creez un fichier nomme "exercice_1.ml"
- Ecrivez dans ce fichier "let exercice_1 = "Reussi !"" suivi d'un retour a la ligne
- Sauvegardez, et dans votre shell, lancez l'interprete ocaml
- A l'invite de l'interprete, utilisez la directive "#use" pour evaluer le fichier "exercice_1.ml"



Chapitre IV

Expressions et types

Pour vous aider, je vous encourage tres fortement a consulter la documentation officielle du langage a l'adresse <http://caml.inria.fr/pub/docs/manual-ocaml/index.html> et la documentation du module Pervasives. Disponible a l'adresse <http://caml.inria.fr/pub/docs/manual-ocaml/libref/Pervasives.html>, ce module de la bibliotheque standard est ouvert par default dans tous vos programmes.

IV.1 Exercice 2

Parcourez la documentation du module Pervasives a l'adresse <http://caml.inria.fr/pub/docs/manual-ocaml/libref/Pervasives.html>, principalement la section "comparaisons".

IV.2 Exercice 3

Tentez de predire le type et la valeur de chacune des expressions suivantes, puis verifiez si vous avez raison a l'aide de l'interprete. Il est tout a fait possible que certaines de ces expressions s'evaluent en erreurs...

- `let a = 42;;`
- `a;;`
- `let b = "suspens...";;`
- `let c = ();;`
- `let d = 42 + 0;;`
- `let e = 42.0 +. 0;;`
- `let f = 30 and g = 12;;`
- `let h = f + g;;`
- `let i = let j = 50 and k = 8 in j - k;;`
- `let l = 42 in let m = l - 42 in l + m;;`



- `let n = 42 and o = n - 42 in n + o;;`

IV.3 Exercice 4

Tentez de predire le type et la valeur de chacune des expressions suivantes, puis verifiez si vous avez raison a l'aide de l'interprete. Il est tout a fait possible que certaines de ces expressions s'evaluent en erreurs...

- `let fonction_p = fun a b -> a + b;;`
- `let fonction_q a b = a + b;;`
- `fonction_q 21 21;;`
- `fonction_q;;`
- `let fonction_r () = 42;;`
- `let fonction_s a = 42;;`
- `let fonction_t a = a;;`
- `let fonction_u a b = a b;;`
- `let fonction_v a b c = a b c;;`
- `let fonction_w a b c = a (b c);;`
- `let fonction_x () = let a = 42 in let b = 42 in a - b + 42;;`
- `let y = "a" in let fonction_z a b = b ^ y in fonction_z;;`
- `fonction_z;;`

IV.4 Exercice 5

Tentez de predire le type et la valeur de chacune des expressions suivantes, puis verifiez si vous avez raison a l'aide de l'interprete. Il est tout a fait possible que certaines de ces expressions s'evaluent en erreurs...

- `let a = 42 in if a > 0 then true else false;;`
- `let str = "ocaml" in if str <> "" then print_endline str;;`
- `if 42 = 24 then (*) else (+);;`
- `let _ = match 42 with 0 -> "zero" | n -> "42";;`
- `let rec f x = if x > 0 then (g (x - 1)) else 1 and g x = if x > 0 then (f (x - 1)) else 0;;`