

## Environnements de code basés sur le package listings

**Exemple d’environnement de code** Utilisez les environnements script, method, classdef et example. Chacun a un nom et un label optionnel. Avec les labels, nous pouvons avoir des références croisées de With labels we can have cross references to `\egref{history}` exemple 1, `\scrref{helloworld}` script 2, `\clsref{myclass}` classe 3, et `\mthref{doit}` méthode 4.

```
\begin{example}[history]{La première chose que Smalltalk peut faire}{}
3 + 4 --> 7
\end{example}
```

Exemple 1 – *La première chose que Smalltalk peut faire*

```
3 + 4 → 7
```

```
\begin{script}[helloworld]{La première chose que vous pouvez tester}
Transcript show: 'hello world'
\end{script}
```

Script 2 – *La première chose que vous pouvez tester*

```
Transcript show: 'hello world'
```

```
\begin{classdef}[myclass]{définition de MyClass}
Object subclass: #MyClass
instancevariables: '...'
...
\end{classdef}
```

Classe 3 – *définition de MyClass*

```
Object subclass: #MyClass
instancevariables: '...'
...
```

```
\begin{method}[doit]{Une méthode doit}
MyClass>>>doit
<tab>^ super doit
\end{method}
```

Méthode 4 – *Une méthode doit*

```
MyClass>>doit
↑ super doit
```

L’environnement de code ordinaire n’a ni titre, ni label, ni numérotation :

```
\begin{code}{}
Du code tout court
\end{code}
```

Du code tout court

## Les environnements Listings et ses macros L'environnement de code

```
\begin{code}{}
...
\end{code}
```

accepte du code ordinaire ou *verbatim* et traduit certains caractères spéciaux comme  $\wedge$  en  $\uparrow$ . Même les tabulations sont maintenus (ce qui n'est pas vrai pour le *verbatim*).

```
"Tout les caractères correctes: ↑ $ ' % \<< >> ← { }"
"Si vous voulez vraiment un point d'exclamation !, vous devez écrire explicitement
BANG"
"Si vous voulez vraiment un chapeau de renvoi ainsi: ^ au lieu de ↑, vous devez écrire
CARET."
| y |
true & false not & (nil isNil) ifFalse: [self halt].
y ← self size + super size.
#($a #a 'a' 1 1.0)
do: [:each | Transcript
  show: (each class name);
  show: ' ';
  show: (each printString).
{ 1 + 2 . 3 \ 4 . 1 << 3. 2 >> 5 . 1 % 2 }.
↑ x < y
```

La table QWERTY :

```
!@#$%↑&*( )_+
1234567890-=
QWERTYUIOP{}
qwertyuiop[]
ASDFGHJKL: ""| (deux fois " pour annuler l'italique")
asdfghjkl;' \
ZXCVBNM<>?
zxcvbnm,./
```

Échappement LaTeX dans le code :

```
\begin{code}{}
code ordinaire et !\textbf{gras}!
\end{code}
```

code ordinaire et **gras**

Dans le code *inline* avec `\ct` comme, *par ex.*, `\ct{1 + 2 --> 3}` imprimé ainsi : `1 + 2 --> 3`, le texte peut suivre immédiatement. Les “caractères-bloc” autour de `\ct` peuvent être n’importe quelle paire de caractères ; utile si vous voulez `{ et }` dans le code.

## Les caractères spéciaux avec `\ct`

`↑ ~ # $ ' % \ < > < > { } ! -- --> =`  
`\ct=^ ~ # $ ' % \ \ < > _ { } ! -- --> =`

## Conventions spéciales

`\ct{Class>>>method}` s’imprime `Class»method`.  
`\ct{3 + 4 - 5 --> 2}` s’imprime `3 + 4 - 5 --> 2`.

Utiliser `@TEST` pour inclure le code dans des tests automatiques et utiliser `-->` pour représenter leur résultat respectif.

```
\begin{code}{@TEST}
true          --> true
3@4           --> 3@4
$a            --> $a
#(1 2 3)      --> #(1 2 3)
\end{code}
```

## Autres macros

**url** Ne pas oublier le `http : //` : `http://SqueakByExample.org`, `http://pharo-project.org` ou `http://pharo-project.org/PharoParLExemple`

**noms** `SUnit` *xUnit* `Smalltalk` `Pharo` `Squeak` `SqueakSource`

**annotations originales** À CORRIGER! ~~please rephrase this~~ please insert this text  
~~delete this~~ ou encore ~~change this~~ → to this Andrew ►...◄

### annotations de l’édition française

À CORRIGER! MARTIAL: *LE RESTE N’EST PAS SYNCHRONE*


- `\arevoir` ~~Texte à revoir soit parce que c’est incertain, soit parce que c’est une formulation “pas très compréhensible”;~~
- `\arelire` ~~Changement par rapport à la version de Squeak Par l’Exemple ou raffinement par rapport à la version originale Pharo By Example;~~
- `\arevoir` ~~Totalement faux mais à garder car existe toujours dans la version originale.~~

**abréviations** *c-à-d.* *c-à-d.* *par ex.*, *par ex.*, etc

**Macros Smalltalk** sep : »

**scat et prot** Catégorie *Kernel-Objects* et protocoles *accessing*. Les paquetages (ou *packages*) s'écrivent comme les catégories avec `\scat`.

**menu** Le menu 

**button** Le bouton 

**do this** `\dothis` →  *Télécharger immédiatement Pharo.*

**Raccourcis-clavier** `\short{d}` → CMD-d

**Les clics de souris** Pour l'édition Pharo, les clics sont écrits :

- `\click` pour cliquer
- `\clickant` pour cliquant
- `\clickz` pour cliquez

Le *c* de `\click` peut être majuscule. Les alternatives de `\click` sont `\actclick` et `\metaclick` pour “cliquer avec le bouton d'action” et “meta-cliquer” respectivement. La règle de la capitalisation (première lettre en majuscule) existe aussi.

## Citations de bas de page

Il y a un grand livre sur Squeak par Ducasse<sup>1</sup>.

## Les points importants

L'heure est grave

## Remarques de l'édition française

**les termes et phrases toute faites** `\mantra` pour “tout est objet”

---

1. ?,.

mot anglais	commande	imprimé
callback	\callback	<i>callback</i>
changeset	\changeset	<i>change set</i>
sender	\sender	<i>sender</i>
sender	\sender	<i>sender</i>
senders	\senders	<i>senders</i>
Senders	\Senders	<i>Senders</i>
implementors	\implementors	<i>implementors</i>
implementor	\implementor	<i>implementor</i>
truetype	\truetype	<b>TrueType</b>
\bam <sup>2</sup>	\bamfr	atomes rebondissants

## accents dans les environnements de code

**cas des commentaires** Il suffit de baliser chaque mot<sup>3</sup> accentué avec des points d'exclamation !.

```
\begin{code}{}
"Des !caractères! !accentués!"
\end{code}
```

*"Des caractères accentués"*

**autres éléments** \normcode et emcode permettent d'écrire dans le code entre ! un mot accentué sans qu'il soit en italique comme dans tous commentaires : il sera ainsi respectivement de typographie normal et grasse.

```
BladeRunner>>run
self mayRun ifFalse: [self error: 'Reste caché alors !'].
self runningFree
```

## convention

- Nous parlons à la première personne du pluriel et vous lisez à la seconde personne du pluriel — pas de "on" ;
- Les notes de bas de pages commencent par une majuscule et finissent par un point ;
- Les légendes (*caption*) des figures commencent normalement par un verbe à l'impératif et finissent par un point ;
- les articles de `itemize` finissent par ; sauf le dernier qui se termine par un point.

3. Nous ne pouvons pas baliser toute une phrase en raison d'une implémentation *cra-cra* du package `listings`.

**méthodologie** Utilisez les `hyphenation` dans `common.tex` pour forcer les césures. En cas de doutes sur les `index` (`\index` et `\seeindex`) durant la traduction, laissez-les tels quels : il sera plus facile pour le responsable de l'indexation de faire des corrections d'une version originale cohérente que d'une version anglaise incohérente.

Les relecteurs devraient explorer les commentaires suivants dans le code  $\LaTeX$  :

- `\% ATTENDRE` signifie "en attente de correctionsconfirmations dans la version originale";
- `\% CHANGE` signifie "changement par rapport à Squeak Par l'Exemple". Souvent précédé d'une balise `\arelire` dans les parties I, II, III et IV du document;
- `\% REVOIR` signifie "phrase lourde, terme francisé incorrect, terme à franciser, inexactitude par rapport à la version actuelle de Pharo".

Ce document est une base technique ; pour les conventions sur le vocabulaire ou les mises à jours des outils (*par ex.*, les paquetages `MorphicExtras` et consorts), référez-vous aux fichiers `README.txt`, `CHANGES.txt`, `mailing-lists` et [http://community.ofset.org/index.php/Squeak\\_par\\_l%27exemple](http://community.ofset.org/index.php/Squeak_par_l%27exemple).

# Bibliographie

**Stéphane Ducasse:** Squeak : Learn Programming with Robots. APress, 2005,  
ISBN : 1-59059-491-6