

# PyLogo - discussion sur le projet

Excellente question théorique ! Oui, absolument, et c'est même très intéressant pédagogiquement.

## Approches possibles pour les objets en Logo

### 1. Approche "Dictionnaire/Table" (la plus simple)

Étendre la syntaxe actuelle avec des structures de données :

```
(* Nouveau type : table associative *)
table = "{" [ paire { "," paire } ] "}" ;
paire = identificateur ":" expression ;

(* Accès aux propriétés *)
acces_propriete = variable "." identificateur ;

(* Exemple d'utilisation *)
DONNE :vetement {nom: "Chemise", couleur: "Bleu", taille: 42}
ECRIS :vetement.nom
```

#### Avantages :

- Naturel pour les enfants (proche des objets du monde réel)
- Pas de concept de "classe" à enseigner
- Extension minimale de la grammaire

### 2. Approche "Prototype" (style JavaScript)

Pas de classes, seulement des objets qui se clonent :

```
(* Création d'un prototype *)
PROTOTYPE identificateur [ liste_proprietes ]
```

```

    { instruction }
FIN

(* Exemple *)
PROTOTYPE Vetement :nom :couleur
    DONNE :this.nom :nom
    DONNE :this.couleur :couleur
FIN

NOUVEAU :chemise Vetement "Chemise" "Bleu"

```

#### Avantages :

- Plus flexible
- Concept de "copie" plus intuitif qu'héritage
- Pas de hiérarchie complexe

## 3. Approche "Modèle" (ce que vous suggérez)

Une forme hybride avec des templates :

```

(* Définition d'un modèle *)
MODELE identificateur
    { declaration_propriete }
FIN

declaration_propriete = identificateur [ "=" expression ] ;

(* Instanciation *)
NOUVEAU :instance identificateur
DONNE :instance.propriete valeur

```

#### Avantages :

- Vocabulaire adapté aux enfants ("modèle" vs "classe")
- Séparation claire structure/données
- Proche de votre proposition

## 4. Approche "Liste nommée" (ultra-simple)

Réutiliser les listes existantes avec des conventions :

```
; Pas de nouvelle syntaxe !
DONNE :vetement [nom "Chemise" couleur "Bleu" taille 42]

; Fonction utilitaire
POUR OBTENIR :objet :cle
    ; Cherche la clé dans la liste
FIN

ECRIS OBTENIR :vetement "nom"
```

### Avantages :

- Zéro changement grammatical
- Enseigne la manipulation de listes
- Très flexible

# Ma recommandation pour un contexte éducatif

Je pencherais pour **l'Approche 4 (Listes nommées) au départ**, puis **Approche 1 (Dictionnaires)** si besoin :

### Pourquoi ?

1. **Progressivité** : Les enfants maîtrisent d'abord les listes, puis ajoutent la sémantique d'objet
2. **Pas de magie** : Tout est visible (c'est juste des listes !)
3. **Découverte** : "Tiens, si on mettait les noms et valeurs ensemble..."
4. **Transition naturelle** vers les vrais objets plus tard

# Extensions de la grammaire EBNF

Si on voulait ajouter des tables/objets :

```
(* Structures de données *)
facteur = nombre
        | variable
```

```
| chaine
| table      (* NOUVEAU *)
| liste
| ... ;
```

```
table = "{" [ entree_table { "," entree_table } ] "}" ;
entree_table = identificateur ":" expression ;
```

(\* Accès \*)

```
acces_membre = variable "." identificateur
               | variable "[" expression "]" ; (* accès par clé *)
```

(\* Modification \*)

```
affectation_membre = variable "." identificateur expression
                    | variable "[" expression "]" expression ;
```

# Question philosophique

## Faut-il ajouter les objets à Logo ?

### Pour :

- Modéliser le monde réel
- Préparer aux langages modernes
- Organiser le code complexe

### Contre :

- Logo enseigne la **pensée procédurale** d'abord
- Risque de complexifier trop tôt
- Les listes suffisent souvent

**Mon avis** : Commencer sans objets, les introduire comme "bibliothèques de fonctions" qui manipulent des listes structurées, puis éventuellement formaliser la syntaxe si le besoin émerge naturellement.

C'est cohérent avec la philosophie Logo : découverte progressive, pas abstraction imposée ! ☐

Revision #1

Created 2026-01-05 16:11:33 UTC by Nicolas Farrie

Updated 2026-01-05 16:12:47 UTC by Nicolas Farrie