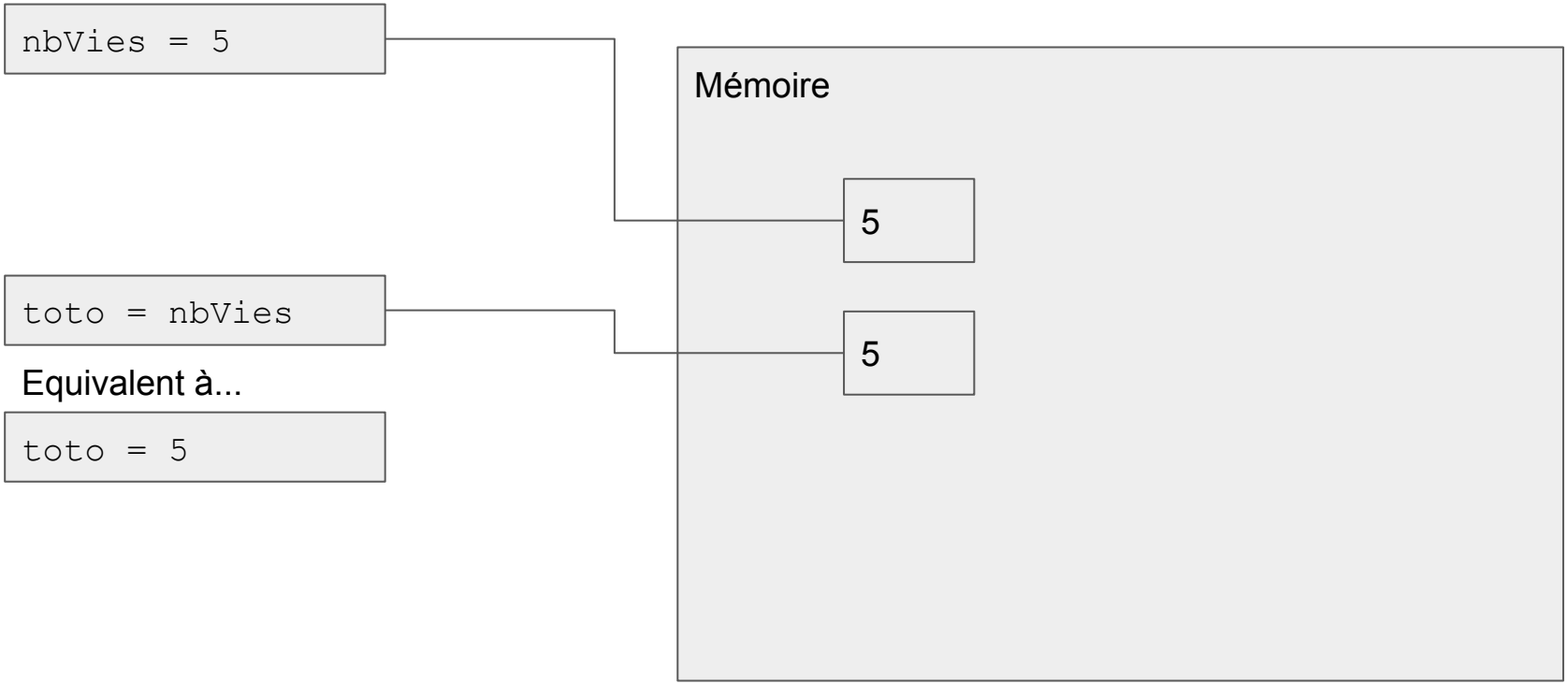
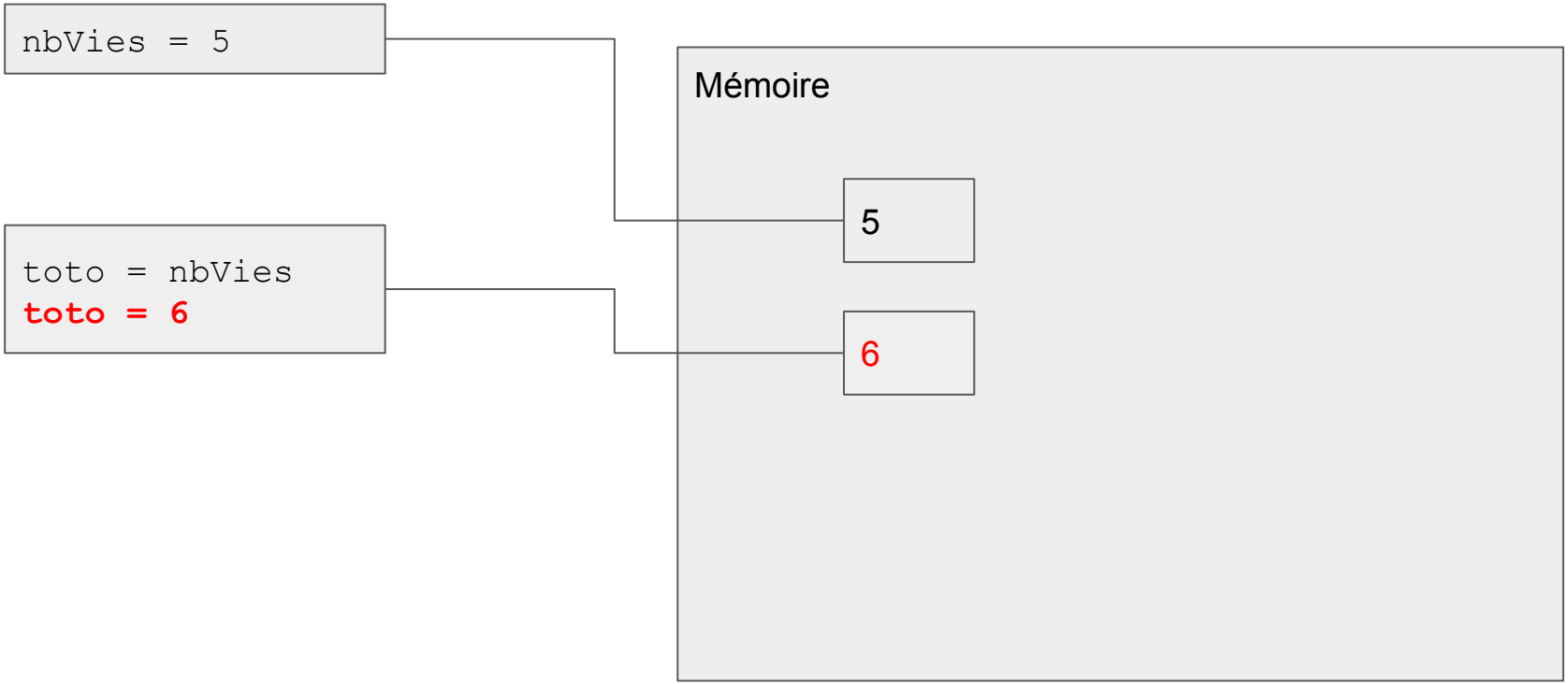

CODER UN INVENTAIRE
DE RPG AVEC UNE LISTE
Lua
LÖVE
?

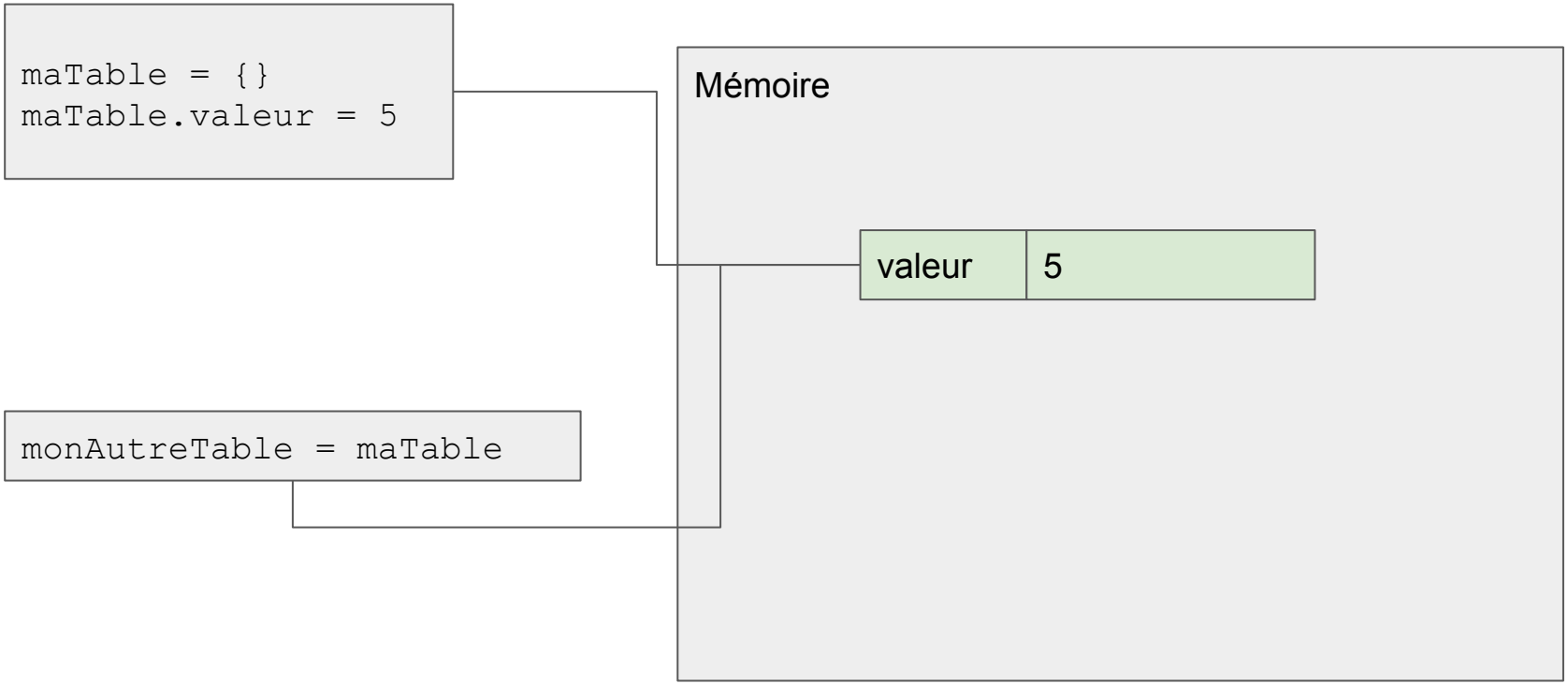
Les variables sont des valeurs



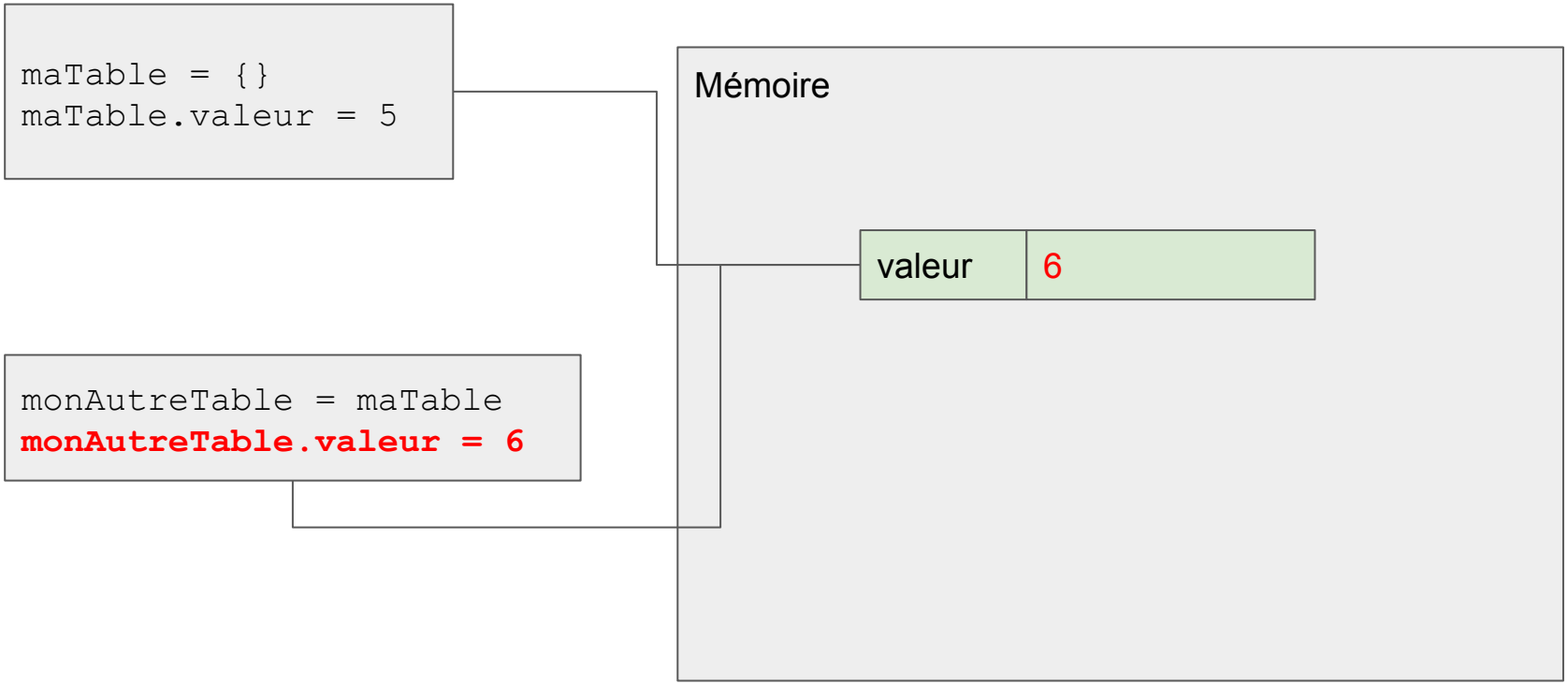
Les variables sont des valeurs



Les tables sont des références



Les tables sont des références

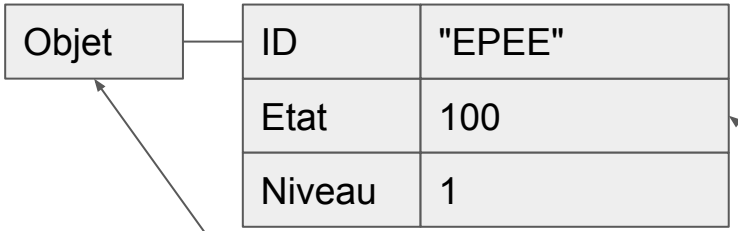


Créons une table...

```lua
local objet = {}
objet.ID = "EPEE"
objet.Etat = 100
objet.Niveau = 1
```

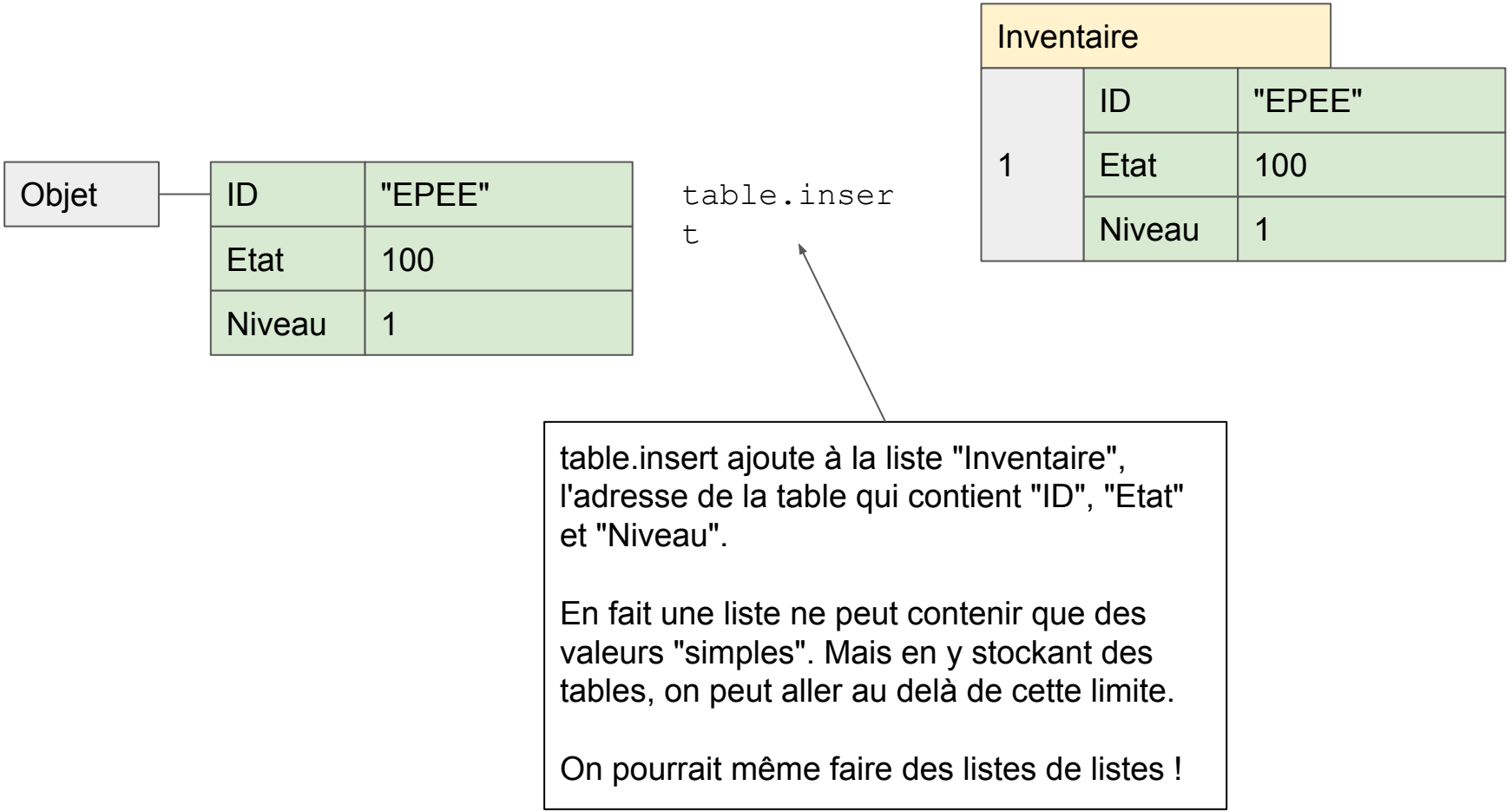
```lua
Inventaire = {}
```

Et on va utiliser une liste pour stocker les objets d'inventaire. En réalité on va donc stocker des adresses !

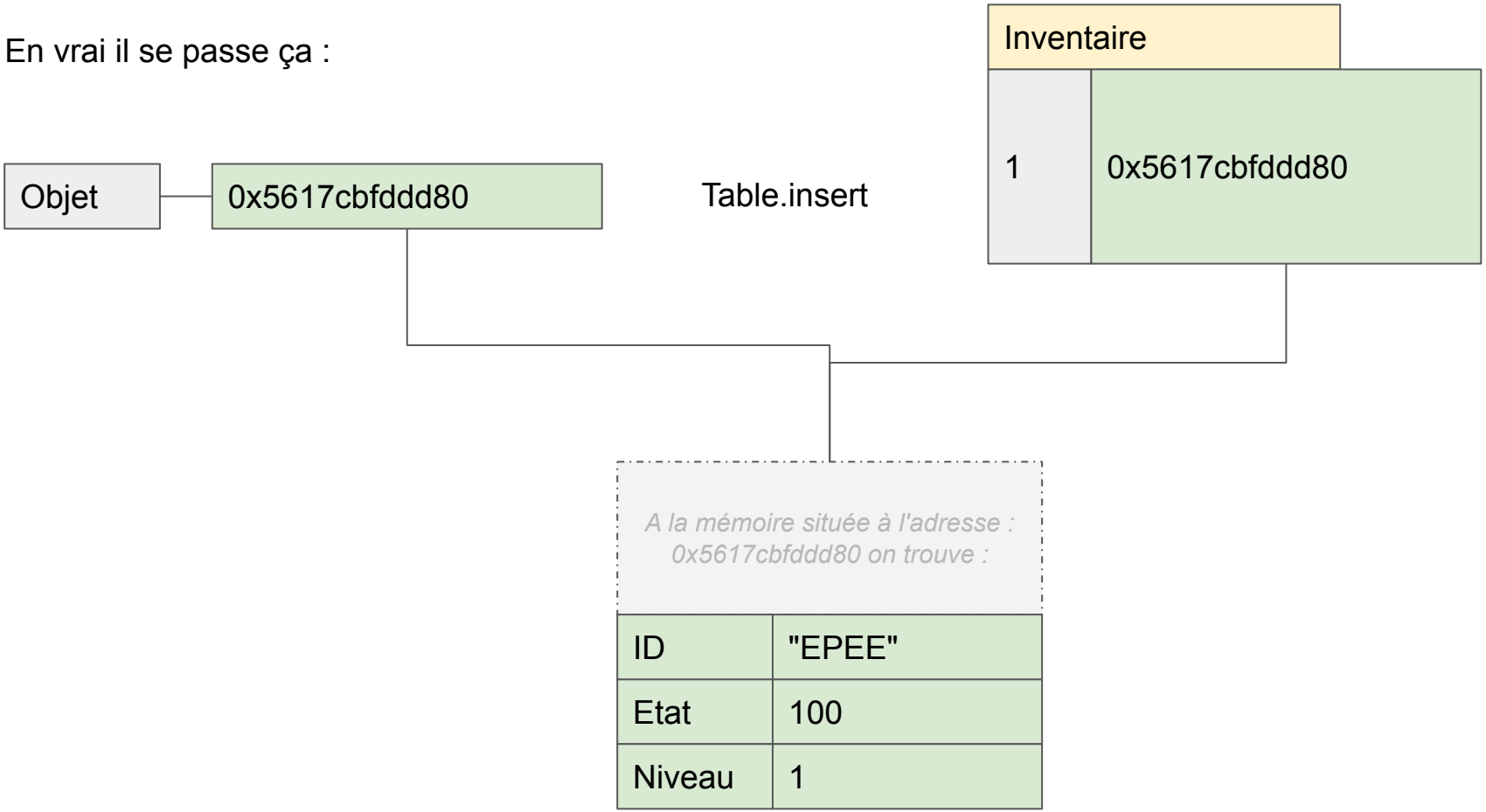


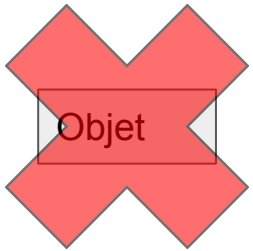
Objet est une variable contenant une adresse, exemple : 0x5617cbfddd80

A l'adresse 0x5617cbfddd80 en mémoire, se trouvent les données de la table


Inventaire
1
ID
"EPEE"
Etat
100
Niveau
1
Objet
ID
"EPEE"
Etat
100
Niveau
1
table.insert
table.insert ajoute à la liste "Inventaire", l'adresse de la table qui contient "ID", "Etat" et "Niveau".
En fait une liste ne peut contenir que des valeurs "simples". Mais en y stockant des tables, on peut aller au delà de cette limite.
On pourrait même faire des listes de listes !

En vrai il se passe ça :





Quand on supprime la variable contenant l'adresse, on ne supprime pas les données de la table.

Les données restent en mémoire tant qu'au moins une variable contient leur adresse.

Inventaire		
1	ID	"EPEE"
	Etat	100
	Niveau	1

On va ajouter 3 tables à la liste "Inventaire".

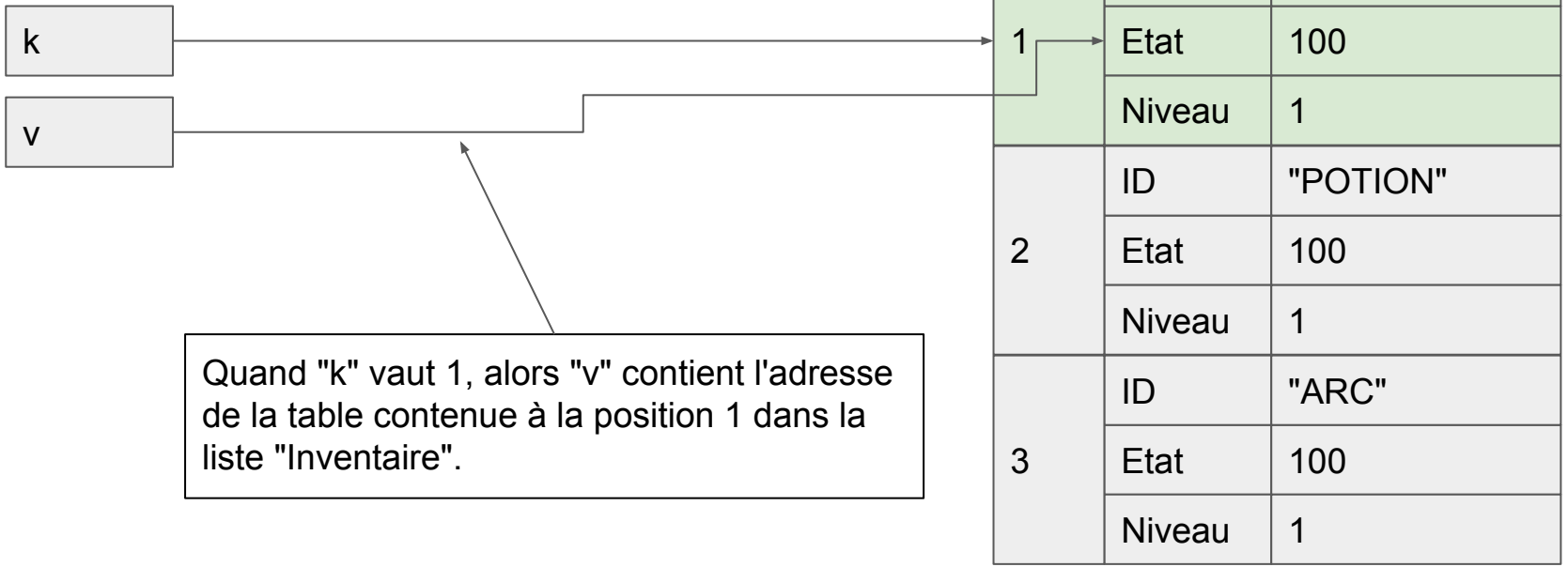
Inventaire

1	ID	"EPEE"
	Etat	100
	Niveau	1
2	ID	"POTION"
	Etat	100
	Niveau	1
3	ID	"ARC"
	Etat	100
	Niveau	1

```
for k,v in ipairs(Inventaire)
```

k

v




```
end
```

Quand "k" vaut 1, alors "v" contient l'adresse de la table contenue à la position 1 dans la liste "Inventaire".

```
for k,v in ipairs(Inventaire)
```

k

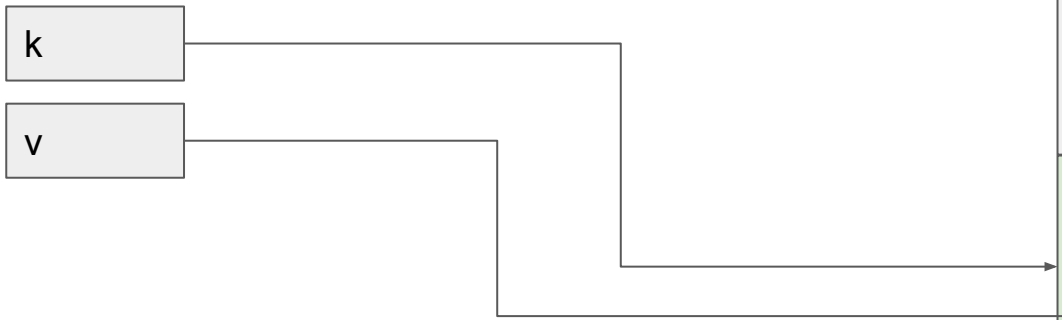
v	
v.ID	"EPEE"
v.Etat	100
v.Niveau	1

```
end
```

Inventaire		
1	ID	"EPEE"
	Etat	100
	Niveau	1
2	ID	"POTION"
	Etat	100
	Niveau	1
3	ID	"ARC"
	Etat	100
	Niveau	1

v "pointe" donc vers les données de l'épée.
On peut donc accéder aux membres de la table en y ajoutant un point puis le nom du membre.

```
for k,v in ipairs(Inventaire)
```



Inventaire		
1	ID	"EPEE"
	Etat	100
	Niveau	1
2	ID	"POTION"
	Etat	100
	Niveau	1
3	ID	"ARC"
	Etat	100
	Niveau	1

```
end
```

Au tour de boucle suivant, v pointe vers les données de l'élément suivant de la liste et k contient l'index suivant (dans notre cas : 2).

```
for n=1,#Inventaire do
    local objet = Inventaire[n]



end
```

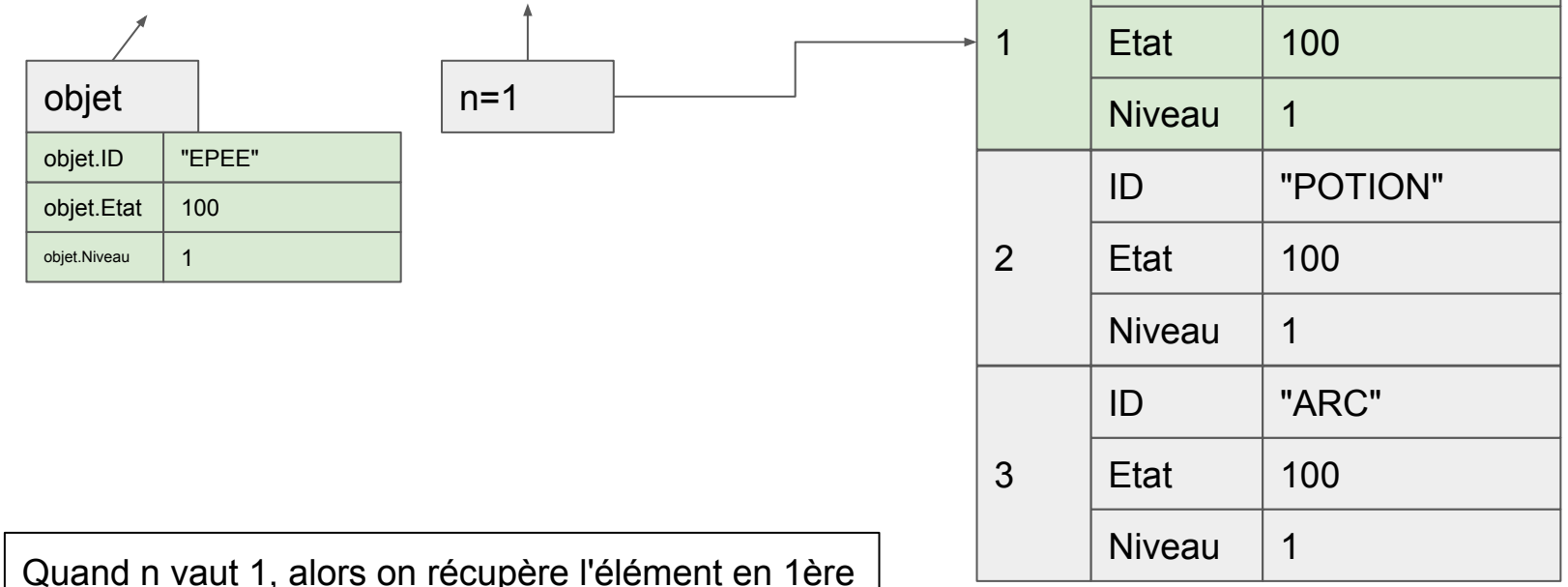
L'autre méthode : faire une boucle sur l'index (ici de 1 à 3) et aller "chercher" l'élément de la liste via cet index et en stockant sa référence dans une variable locale temporaire (ici "objet").

Attention : le fait de nommer ma variable "objet" est arbitraire, je peux aussi l'appeler "concombre".

Inventaire

1	ID	"EPEE"
	Etat	100
	Niveau	1
2	ID	"POTION"
	Etat	100
	Niveau	1
3	ID	"ARC"
	Etat	100
	Niveau	1

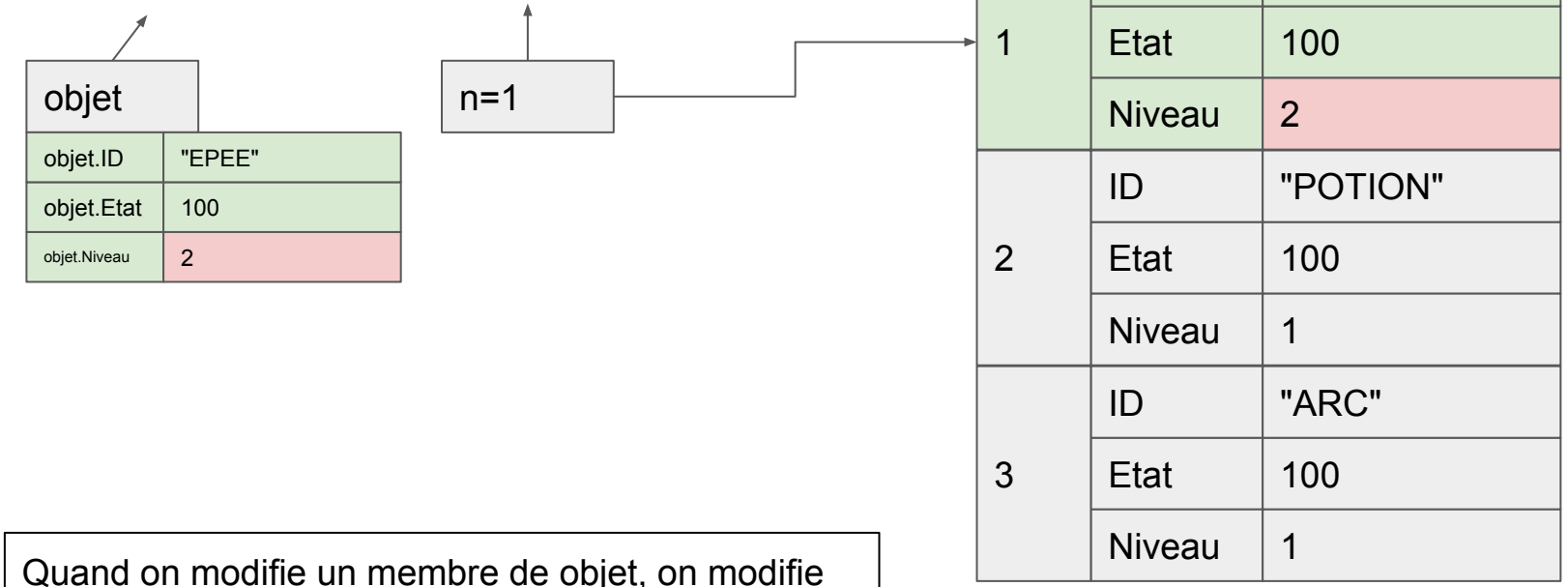
```
for n=1,#Inventaire do
    local objet = Inventaire[n]
```




```
end
```

Quand n vaut 1, alors on récupère l'élément en 1ère position dans la liste.

```
for n=1,#Inventaire do
    local objet = Inventaire[n]
```




```
end
```

Quand on modifie un membre de objet, on modifie directement la table en mémoire.

```
for n=1,#Inventaire do
    local objet = Inventaire[n]
```

Inventaire		
1	ID	"EPEE"
	Etat	100
	Niveau	2
2	ID	"POTION"
	Etat	100
	Niveau	1
3	ID	"ARC"
	Etat	100
	Niveau	1

```
end
```

Quand n vaut 2, alors on récupère l'élément en 2ème position dans la liste.

Quelle méthode choisir ?

ipairs	**pairs**	**for + index**
Quand on a utilisé uniquement "table.insert" et qu'on n'a pas besoin de supprimer d'élément de la liste pendant qu'on la parcoure.	Quand on s'est amusé avec les index à la main : maTable["EPEE"] = ...	Quand on a utilisé uniquement table.insert et qu'on doit supprimer des éléments (seul moyen de parcourir à l'envers). Ou quand on n'aime pas pairs/ipairs ...

Je suis désolé mais...

Les structures de données telles que les tables/objets, les tableaux et les listes, c'est indispensable pour coder des jeux vidéo.

Alors bossez ce concept à DONF !