# PILES

## Chapitre 3

jeudi 11 octobre 2018

Lycée Chaptal – PT*

http://www.mickaelprost.fr

1. Une pile est une structure de données reposant sur le principe du « dernier arrivé, premier sorti ». Structure LIFO (*Last in, First out*)
2. On peut représenter de façon imagée une pile par une pile de livres.

3. On ne peut accéder qu'au dernier élément ajouté, le sommet de la pile. On peut récupérer cet élément (le dépiler) et il disparaît.
   → Fonction pop
4. On peut rajouter un élément (l'empiler).
   → Fonction push

- Dans le cadre du programme, l'implémentation se fera à l'aide d'une liste.
- Plusieurs façons de procéder :

  piles à capacité finie **VS** piles non bornées

- Nécessité de définir trois fonctions de base :
  1. `depiler`
  2. `empiler`
  3. `liste_vide`

- On fixe la taille maximale $c$ des piles utilisées.
- Une pile contenant les éléments $e_1, \ldots, e_n$, pour $n \leqslant c$, sera représentée par une liste `L` telle que :
  1. `len(L) = c`
  2. `L[0] = n`
  3. Pour tout $i \in \{1, \ldots, n\}$, `L[i] = e_i`.

| L = | n | e_1 | e_2 | ··· | e_n | None | ··· | None |
|---|---|---|---|---|---|---|---|---|

- Exemple :

| |
|---|
| 'a' |
| 'b' |
| 'c' |

$\longrightarrow$ L =

| 3 | 'a' | 'b' | 'c' | None | None | None |
|---|---|---|---|---|---|---|

- On ne s'autorise à accéder qu'à :
  1. la taille de la pile : `L[0]`
  2. au dernier élément de la pile : `L[L[0]]`

- Fonction `creer_pile()`

```
>>> c = 7
>>> def creer_pile():
...     p = (c+1)*[None]
...     p[0] = 0
...     return p
...
>>> pile1 = creer_pile()
>>> print(pile1)
[0, None, None, None, None, None, None, None]
```

- Fonction `empiler(p, e)`

```
>>> def empiler(p, e):
...     n = p[0]
...     p[n+1] = e
...     p[0] = n + 1
...
>>> empiler(pile1, 'a')
>>> print(pile1)
[1, 'a', None, None, None, None, None, None]
```

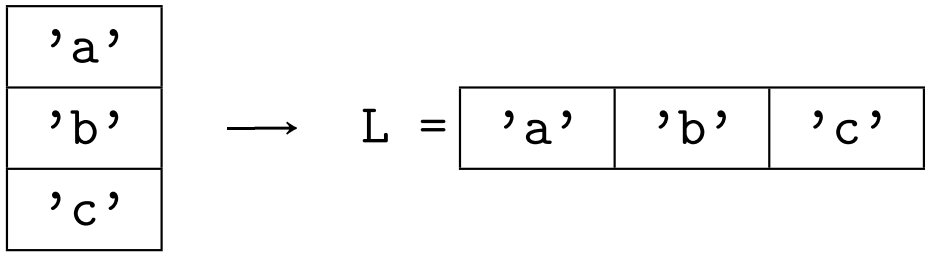
- Fonction `depiler(p)`

```
>>> def depiler(p):
...     n = p[0]
...     assert n > 0, 'La pile est vide !'
...     if n != 0:
...         p[0] = n-1
...         return p[n]
...
```

```
>>> empiler(pile1, 'b')
>>> print(pile1)
[2, 'a', 'b', None, None, None, None, None]
```

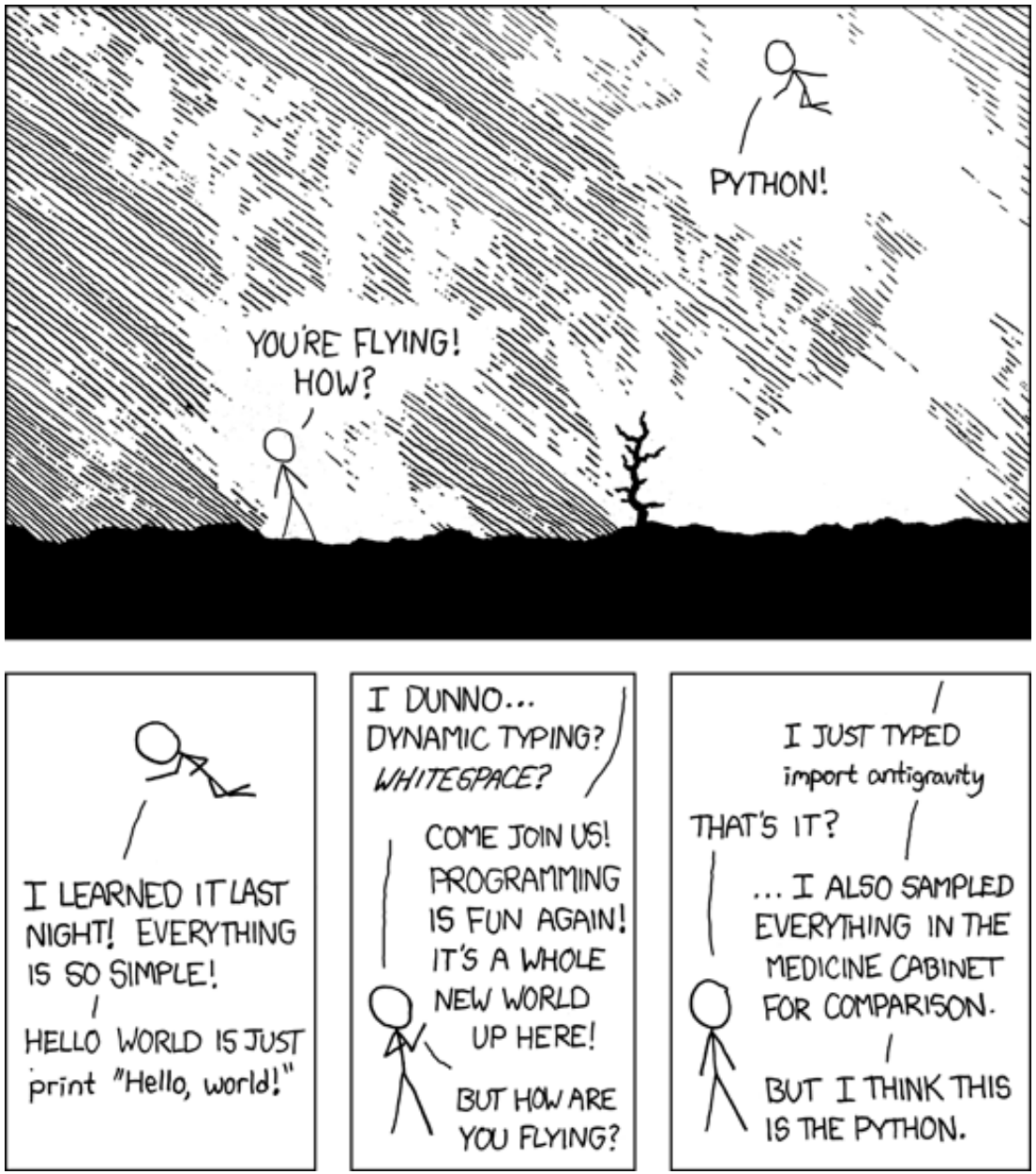
```
>>> x = depiler(pile1)
>>> print(x)
b
>>> print(pile1)
[1, 'a', 'b', None, None, None, None, None]
>>> empiler(pile1, 'c')
>>> print(pile1)
[2, 'a', 'c', None, None, None, None, None]
```

- Il suffit d'utiliser une liste Python dont les éléments sont exactement ceux de la piles.
- Exemple :




- On utilise les méthodes pop et append sur les listes pour empiler et dépiler.

```
>>> pile2 = ['a', 'b', 'c']
>>> pile2.append('z')
>>> print(pile2)
['a', 'b', 'c', 'z']
>>> pile2.pop()
'z'
>>> print(pile2)
['a', 'b', 'c']
```


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.