

PILE NON BORNÉE

OBJECTIF : L'objectif de ce tp est de rendre l'élève capable :

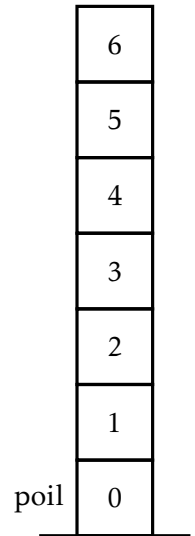
- d'utiliser une structure de pile
- établir des fonctions sur la classe `Pile` semblable aux méthodes de la classe `List`

1 Présentation de la classe `Pile`

Dans ce Tp, on n'a au départ qu'une pile `pile` construite avec une liste chaînée. Pour cela, on utilise le module `pile_non_bornée` (à télécharger) importé avec l'allias `p` pour obtenir une nouvelle classe `Pile`. Les seuls actions possibles sont :

- créer une pile `pile = Pile()`
- empiler `pile.empiler(x)` pour ajouter `x` en haut de la pile.
- dépiler `x = pile.depiler()` pour enlever le dernier élément de la pile et affecter sa valeur à `x`.
- tester si la pile est vide `pile.est_vide()`
- obtenir la valeur du sommet `x = pile.sommet()`

C'est un peu rudimentaire mais nous allons créer des fonctions permettant d'élaborer les outils (fonctions et méthodes) associées aux listes **Python**. Dans la partie suivante, nous utiliserons à nouveau les listes **Python** comme d'ordinaire.



On a bien sûr le droit d'utiliser les structures algorithmiques avec les tests et boucles conditionnelles `while` et inconditionnelles `for`.

Première étape : fabrication de la pile `poil` contenant les valeurs 0, 1, 2, 3, 4, 5, 6.

Q - 1 : Créer la pile `poil` en utilisant uniquement la méthode `empiler`.

2 Création des outils de base

Q - 2 : Écrire une fonction `inverse(pile)` qui dépile `pile` pour renvoyer la pile inverse de `pile`. Que vaut `pile` après avoir appliqué `inverse(pile)` ?

Q - 3 : Créer une fonction `copie(pile)` :

- qui vide `pile` pour construire la pile inverse de `pile` qu'on note `inv_pile`.
- qui reconstruit `pile` et construit `copie_pile` en vidant `inv_pile`
- qui renvoie `copie_pile`

Q - 4 : Écrire une fonction `longueur(pile)` qui renvoie le nombre d'éléments de `pile`. Prendre bien soin de travailler sur la copie de `pile`, grâce à la fonction précédemment créée.

On dispose donc ici d'une pile et des fonctions permettant de copier la pile et d'en connaître la longueur.

Q - 5 : Écrire une fonction `valeur(pile, i)` qui renvoie la $i^{\text{ème}}$ valeur de `pile` en partant du bas de la pile.

REMARQUE : si la pile ne contient pas l'indice `i`, la fonction `valeur(pile, i)` renvoie un message d'erreur.

Q - 6 : Écrire une fonction `concatenation(pile_p, pile_q)` qui permet d'ajouter à la pile `pile_p`, la pile `pile_q`.

Q - 7 : Écrire une fonction `supprime(pile, i)` qui permet de supprimer de la pile `pile`, l'élément `i`.

Q - 8 : Écrire une fonction `insertion(pile, x, i)` qui permet d'ajouter un élément `x` en position `i` dans la pile `pile`.

Q - 9 : Écrire une fonction `substitue(pile, x, i)` qui permet de remplacer l'élément `i` de la pile `pile` par `x`.

Q - 10 : Donner la complexité de toutes les fonctions créées.

Si vous avez d'autres idées et d'autres besoins, sentez vous libres des les programmer.