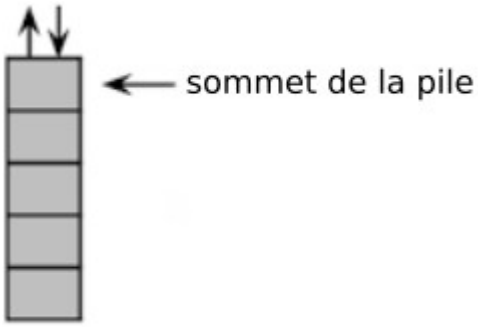
Les piles

On retrouve dans les piles une partie des propriétés vues sur les listes.

Dans les piles, il est uniquement possible de manipuler le dernier élément introduit dans la pile.

On prend souvent l'analogie avec une pile d'assiettes:

Dans une pile d'assiettes la seule assiette directement accessible et la dernière assiette qui a été déposée sur la pile.



Les piles (*stack*) sont basées sur le principe LIFO (Last In First Out: le dernier rentré sera le premier à sortir).

Quelques exemples d'usage courant d'une pile:

– Dans un navigateur web, la pile sert à mémoriser les pages web visitées, ainsi, lors de l'appui sur la fléche de retour en arrière, les pages sont envoyées dans l'ordre de visite

– Quand on fait un "annuler" (Ctrl+z)

Une pile est une collection d'objets sur lesquels il est possible d'utiliser diverses opérations.

(On retrouve souvent ce principe LIFO en informatique).

Voici les opérations que l'on peut réaliser sur une pile:

•Savoir si la pile est vide

- (pile_vide?, is_empty)

•Empiler un nouvel élément au sommet de la pile

- (empiler, push)

•Récupérer l'élément au sommet de la pile tout en le supprimant

- (dépiler, pop)

•Accéder à l'élément situé au sommet de la pile sans le supprimer de la pile

- (sommet)

•Connaitre le nombre d'éléments présents dans la pile

- (taille)

Exemples:

Soit une pile P composée des éléments suivants:

12, 14, 8, 7, 19 et 22 (le sommet de la pile est 22)

Pour chaque exemple ci-dessous on repart de la pile d'origine:

•pop(P) renvoie 22 et la pile P est maintenant composée des éléments suivants:

- 12, 14, 8, 7 et 19 (le sommet de la pile est 19)

•push(P,42) la pile P est maintenant composée des éléments suivants:

- 12, 14, 8, 7, 19, 22 et 42

•sommet(P) renvoie 22, la pile P n'est pas modifiée

•si on applique pop(P) 6 fois de suite, pile_vide?(P) renvoie vrai

•Après avoir appliqué pop(P) une fois, taille(P) renvoie 5

Le type *list* de Python contient tout ce qu'il faut pour implémenter une pile. Le code est suffisamment simple pour ne pas nécessiter l'écriture d'une nouvelle classe (le *push* de la pile correspond au *append* de la liste).

```
pile = []
pile.append("3")
pile.append("2")
pile.append("1")
while pile:
        print(pile.pop())
```

-> 1 2 3

Une 1° implémentation de la structure:

```
def pile():
    """
    Retourne une liste vide
    """
    return []

def vide(p):
    """
    Retourne True si la pile est vide, False sinon
    """
    return p == []

def empiler(p, x):
    """
    Ajoute l'élément x à la pile p
    """
    p.append(x)

def depiler(p):
    """
    Dépile et renvoie l'élément en haut de la pile
    """
    assert not vide(p), 'Pile vide'
    return p.pop()
```

– Testez les instructions suivantes:

```
p = pile()
for i in range(5):
        empiler(p, 2*i)
a = depiler(p)
print(a)
print(vide(p))
```

– Ecrire les fonctions:
  – taille(p) -> retourne la taille de la pile
  – sommet(p) -> retourne le sommet de la pile, sans le supprimer

Une 2° implémentation de la structure:

```
class Pile:
    """
    Classe Pile
    Création d'une instance de la pile avec une liste
    """

    def __init__(self):
        """
        Initialisation de la pile
        """
        self.L = []

    def vide(self):
        """
        Regarde si la pile est vide
        """
        return self.L == []

    def depiler(self):
        """
        Dépile
        """
        assert not self.vide(), 'Pile vide'
        return self.L.pop()

    def empiler(self, x):
        """
        Empile
        """
        self.L.append(x)
```

– Testez les instructions suivantes:

```
p = Pile()
for i in range(5):
    p.empiler(2*i)
print(p.L)
a = p.depiler()
print(a)
print(p.L)
print(p.vide())
```

- Réalisez les méthodes suivantes:
  - taille(self) -> renvoie la taille de la pile
  - sommet(self) -> renvoie le sommet de la pile, sans le supprimer

Contrôle du parenthésage d'une expression:

Ecrire une fonction ***verification(expr)*** qui vérifie si une expression mathématique passée en chaine de caractère à bien toutes ses parenthèses ouvrantes et fermantes bien placées.

- Créer une pile
- Parcourir l'expression de gauche à droite
- Chaque fois que l'on rencontre une "(", on l'empile
- Si l'on rencontre une ")" et que la pile n'est pas vide, on dépile, sinon retourne False
- A la fin, la pile doit être vide

Pour aller plus loins:

- Faire en sorte de dire combien de parenthèses il manques

Sources:

- https://pixees.fr/informatiquelycee/n_site/nsi_term_structDo_liste.html
- Prépabac Tle générale spécialité NSI, édition Hatier
- https://isn-icn-ljm.pagesperso-orange.fr/NSI-TLE/co/section_chapitre2.html