Vous avez vu les années antérieures que l'information pouvait être stockée par les ordinateurs avant d'être réutilisée. Vous avez vu et verrez de nombreux formats : les fichiers, les photos, les tableaux, les piles, les files et les bases de données.

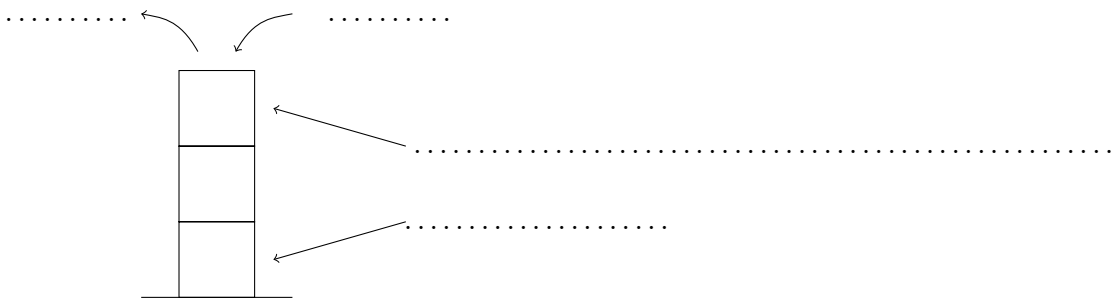
Nous allons présenter le fonctionnement des piles, étudier leur mise en pratique, avant de les implémenter avec le langage Python.

I Les piles

1 Structure

Une pile est comme son nom l'indique un ensemble d'éléments qui sont empilés les uns sur les autres afin de ne former qu'une seule pile.

expl : pile de carte, pile de pièces, pile de cartons, etc.



Lorsque l'on retire un élément à la pile, on dit qu'on ..... cet élément. Lorsque l'on ajoute un élément à la pile, on dit qu'on ..... cet élément.

Remarquons que ces opérations ne peuvent se faire qu'au sommet de la pile (sinon la pile tomberait et n'existerait plus).

2 Opérations

Informatiquement, il faut donc créer plusieurs fonctions qui nous permettent de manipuler une pile. Il faut tout d'abord pouvoir en créer une, empiler (ajouter) un élément et dépiler (retirer) un élément.

Ainsi nos premières fonctions sont :
- **creer_pile** : .....,
- **depiler(p)** : .....,
- **empiler(p,e)** : .....

On remarque que **creer-pile** peut prendre en argument un entier. Cette disposition est nécessaire lorsque la pile est de taille finie fixée (pour des problèmes d'allocation mémoire, on donne alors en argument la taille maximale de la pile).

Afin d'accroître la facilité d'utilisation de la pile, on peut y ajouter d'autres fonctions qui ne modifient pas la pile, telles que :
- **taille(p)** :.....,
- **est_vide(p)** :.....,
- **sommet(p)** :.....

Exo 1 : Écrire les instructions pour créer une pile, la remplir de sorte qu'il y ait AABCDB (B étant le sommet) et accéder au C (sans le dépiler). Dépiler C et vérifier que la pile ne soit alors pas vide.

3 Utilisations

? Quand rencontre-t-on les piles ?

À chaque fois qu'on souhaite que la dernière information arrivée soit la première utilisée on optera préférentiellement pour une pile. C'est le cas lorsque l'on modélise un entrepôt à une entrée, dans l'historique d'un navigateur (ou logiciel), dans une calculatrice utilisant la notation polonaise inverse, pour la fonction 'annuler' d'un traitement de texte, lors de l'appel de fonction réccursives, dans le jeu des piles d'Hanoi, ...

Ainsi on rencontre les piles lorsque seul le dernier élément nous intéresse dans l'immédiat et lorsqu'on ne connaît pas *a priori* le nombre d'éléments étudiés. En effet une pile est *a priori* infinie.

À l'inverse, on préfèrera un tableau lorsque l'accès à un élément quelconque est nécessaire ou lorsqu'on connaît *a priori* le nombre d'éléments étudiés.

Exo 2 : dans les exemples suivants, préciser quelle est la structure la plus appropriée entre pile et tableau. Pourquoi ?

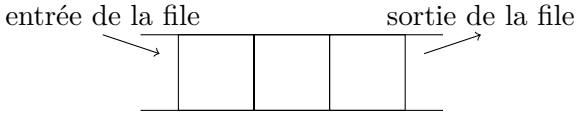
- se souvenir de l'historique des actions effectuées par un utilisateur (en vue d'une annulation),
- garder le listing des adhérents d'une association,
- garder une liste des devoirs à faire d'un élève de prépa
- comptabiliser le nombre de point de vie (max 100) d'un personnage de jeu vidéo,

Exo 3 : Souvent un logiciel propose une fonction **annuler** l'action (UNDO) et une fonction **annuler l'annulation** (REDO). En considérant que le logiciel possède une pile de toutes les actions faites et la lit pour donner l'état courant,

1. proposer une fonction UNDO avec la structune nécessaire,
2. ajouter alors une fonction REDO.

On remarque de plus que le premier arrivé sera le dernier sorti. En anglais, on utilise l'accronyme **LIFO** pour "Last In, First Out".

Nota : Il existe une structure qui permette l'inverse, *i.e.* le premier arrivé sera le premier à sortir (en anglais **FIFO** pour "First In, First Out"). Il s'agit du cas de la file d'attente où le premier à être entré dans la queue sera le premier servi. La structure informatique correspondante s'appelle une **file**.



Nota : une structure de pile ou de liste est assez limitée si les données peuvent être accessibles selon de multiples critères. Dans ce cas, il est fréquent d'utiliser une base de données.

II Utilisation et réalisation d'une pile

1 Utilisation papier d'une pile

Exo 4 : À l'aide des macros vues en cours, proposer une fonction qui

1. dépile k éléments d'une pile (si elle contient moins de k éléments, alors elle sera vidée)
2. inverse toute la pile,
3. inverse le sommet et le pénultième élément de la pile, 
4. inverse le sommet et le premier élément de la pile. 

Les langages informatiques possèdent une grammaire stricte qu'il faut respecter afin que l'ordinateur comprenne le code écrit. Sans cette première étape correcte, l'ordinateur n'effectuera pas le programme. Cette étape constitue le 'parsage' d'un code. Par exemple, lorsque dans un programme python il manque un '`:`' après le '`else`' ou après le '`def`', pyzo renvoie une erreur.

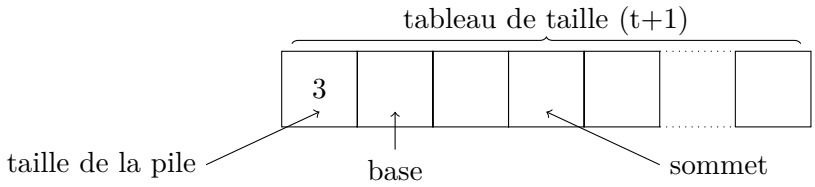
Il en va de même en mathématiques lorsqu'une expression est mal écrite : $(3 \times 5 + 3)$ est correcte, $3 \times 5 + 3)$ est incorrecte.

Exo 5 : À l'aide des macros vues, proposer une manière de vérifier le 'bon' parenthésage d'une expression mathématique.

→ Vérifier votre algorithme avec quelques exemples [*e.g.* $4\times(4+((3+2)-1\times(3-8)))$ ou $3+(1-(2)/(2+1)$ ou $(2+3))+(1$ etc.].

2 Réalisation d'une pile

Exo 6 : On cherche à simuler le fonctionnement d'une pile à l'aide d'un tableau en python. Afin de gagner en efficacité, on retient l'indice du tableau qui contient le sommet de la pile dans la première case. Ce qui donne schématiquement,



Écrire en `python` les six fonctions vues en cours.

```
def creer_pileF(t):
    p=(t+1)*[None] # la pile est un tableau de taille determinee
    p[0]=xxx       # la 1ere case possede l'indice du sommet de la pile
    xxx


...
```

→ Vérifier votre solution en créant une pile finie, en ajoutant des éléments, en la dépilant, etc.

3 Utilisation d'une pile

Sous pyzo, charger le fichier '`CPGE-Info-Piles-TP-eleve.py`' qui comporte les fonctions de bases permettant la gestion d'une pile infinie et comnpléter-le à partir de la ligne 50.

Exo 7 : Le fichier `CPGE-Info-Piles-TP.txt` contient la définition de quatre piles. Dans ce fichier, une pile débute par son nom et se termine par le caractère %.

Écrire un programme qui à partir du fichier, définit les 4 piles définies dedans.

On utilisera les commandes :

```
file = open("fichier.txt","r")    # ouvre le fichier en lecture seule
line = file.readline()            # renvoie 1 ligne du fichier comme 1 string
file.close()                      # ferme le fichier
```

Rq : `file.readline()` renvoie toute la ligne **avec** le caractère '`\n`' (retour chariot). Ainsi la fin d'un fichier sera atteinte lorsque `file.realine()` renvoie une chaîne vide.

Exo 8 : À partir des fonctions de bases, implémenter une fonction `'melange'` qui, étant donner deux piles, mélange leurs éléments dans une nouvelle pile de sorte que : tant qu'une pile au moins n'est pas vide, elle retire un élément du sommet d'une des piles choisie aléatoirement et l'empile sur la nouvelle pile. À la fin, les deux piles initiales sont vides.

rq : il est possible en chargeant le module `'numpy'` d'utiliser un générateur aléatoire ;
e.g. `numpy.random.randint(1,3)` renvoie aléatoirement un entier dans $[1;3[$.

Exo 9 : La notation polonaise inverse (du Polonais Jan Lukasiewicz 1878-1956) (ou notation post-fixée) consiste à écrire chaque opérande puis l'opérateur. Ainsi $3-5$ s'écrit `[3,5,-]`, $1-2\times 3$ s'écrit `[1,2,3,*,-]` et $(1-2)\times 3$ s'écrit sans besoin de parenthèses `[1,2,-,3,*]`.

Si on considère l'entrée de l'expression mathématiques comme un tableau, on peut alors évaluer cette expression à l'aide d'une pile. On lit le tableau en empilant les nombres lus puis, dès qu'on atteint un opérateur, on dépile les deux derniers éléments, on effectue l'opération et on empile le résultat. On arrête une fois le tableau vide. On ne se focalisera que sur les opérateurs $-$ et $\times$.

Proposer une fonction `eval_polo(p)` qui, à partir d'un tableau correspondant aux entrées d'une expression en notation polonaise inverse, évalue la-dite expression.