

TD: piles et files:

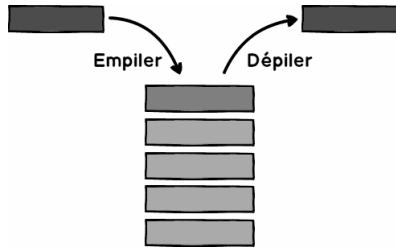


Figure 1: Schématisation d'une pile

Exercice 1: Pile à capacité non bornée:

On s'intéresse au programme suivant:

```
0 def creer_pile():
1     return []
2
3 def empiler(p, x):
4     p.append(x)
5
6 def depiler(p):
7     if len(p) > 0:
8         return p.pop()
9
10 def pile_vide(p):
11     return p == []
12
13 def taille(p):
14     return len(p)
15
16 def sommet(p):
17     if len(p) > 0:
18         return p[-1]
```

piles_fonction.py

1. Expliquer avec précision le rôle de chacune de ces fonctions.
2. Les tester en créant une pile.
3. Reprogrammer ces fonctions dans une classe Pile et les tester.

Exercice 2: Pile à capacité bornée:

On s'intéresse au programme suivant:

```
0 def creer_pile(c):           #pile de capacite c
    p = (c+1) * [None]       #une pile vide
    p[0] = 0                 #la taille de la pile
    return p
4
6 def empiler(p,x):
    if p[0] < len(p) - 1:    #si la taille de la pile n'est pas pleine
        p[0] = p[0] + 1     #la taille augmente d'une unite
        p[p[0]] = x         #l'element est place dans la pile
8
10 def depiler(p):
    if p[0] > 0:             #si la pile est non vide
        s = p[p[0]]         #s est le sommet de la pile
        p[p[0]] = None      #le sommet est retire de la pile
        p[0] = p[0] - 1     #la taille est diminuee de 1
12
14
16
18 p1 = creer_pile(3)
```

pile_capacite_bornee1.py

1. Expliquer avec précision le rôle de chacune de ces fonctions.
2. Les tester en créant une pile à capacité bornée.
3. Ajouter une fonction `pile_vide()` permettant de tester si la pile est vide.
4. De même avec une fonction `taille()` permettant de donner la taille de la pile et une fonction `sommet()` permettant de renvoyer le sommet de la pile.
5. Reprogrammer ces fonctions dans une classe `Pile` et les tester.

Exercice 3: Pile à capacité bornée, fonctions supplémentaires:

1. Reprendre les fonctions `creer_pile`, `empiler`, `depiler`, `pile_vide`, `taille` et `sommet` de l'exercice 2 et les écrire dans un fichier `mes_piles.py`. Dans la suite, ces fonctions constituent l'interface et sont les seules fonctions utilisables.
2. Créer une pile de capacité 5. Empiler les caractères A, B, C, D. Vérifier que la liste représentant la pile est `[4, A, B, C, D, None]`.
3. Écrire une fonction `vider_pile` qui prend en argument une pile `p` et renvoie la pile vidée.
4. Écrire une fonction `inverse_pile` qui prend en argument une pile `p` et renvoie une autre pile avec les éléments empilés dans l'ordre inverse. La pile `p` peut être vidée.
5. Reprendre la question précédente avec la condition que la pile `p` ne soit pas modifiée.

Exercice 4: File:

On s'intéresse au programme suivant:

```
0 class Element:
1     def __init__(self, x):
2         self.val = x
3         self.precedent = None
4         self.suivant = None
5
6     def __str__(self):
7         return str(self.val) + " - " + str(self.suivant)
8
9
10 class File:
11     def __init__(self):
12         self.premier = None
13         self.dernier = None
14
15     def ajoute(self, x):
16         e = Element(x)
17         if self.premier == None:
18             self.premier = e
19         else:
20             e.precedent = self.dernier
21             self.dernier.suivant = e
22             self.dernier = e
23
24     def file_vide(self):
25         return self.premier is None
26
27     def retire(self):
28         if not self.file_vide():
29             e = self.premier
30             if e.suivant is None:
31                 self.premier = None
32                 self.dernier = None
33             else:
34                 self.premier = e.suivant
35                 self.premier.precedent = None
36             return e.val
37
38     def __str__(self):
39         return str(self.premier)
40
```

file.py

1. Expliquer avec précision le rôle des méthodes de la classe Element.
2. Expliquer avec précision le rôle des méthodes de la classe File.
3. Les tester en créant une file.

Exercice 5: Pile et file dans la librairie standard de Python:

Dans la librairie standard de Python, une implémentation existe pour les piles et les files dans le module `queue` (documentation à <https://docs.python.org/fr/3/library/queue.html>)

1. Pour les piles:
 - 1.1 Créer une pile en utilisant `LifoQueue(-1)` du module `queue`.
 - 1.2 Tester les méthodes `put`, `qsize`, `empty` et `get`.
 - 1.3 Créer une fonction `affichage()` qui permet l'affichage sous forme de liste.
2. Mêmes questions pour les files en utilisant `Queue(-1)`.



Figure 2: Schématisation d'une file