TD: piles et files:

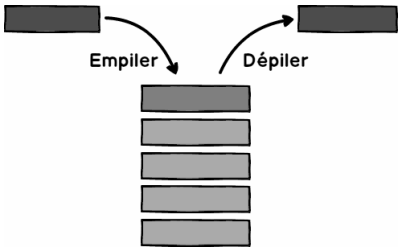

Figure 1: Schématisation d'une pile

Exercice 1: Pile à capacité non bornée:

On s'interesse au programme suivant:

```
0  def creer_pile():
       return []
2
   def empiler(p,x):
4      p.append(x)

6  def depiler(p):
       if len(p) > 0:
8          return p.pop()

10 def pile_vide(p):
       return p == []
12
   def taille(p):
14     return len(p)

16 def sommet(p):
       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'interesse au programme suivant:

```
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

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

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 diminue de 1
        return s


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'interesse au programme suivant:

```python
class Element:
    def __init__(self,x):
        self.val = x
        self.precedent = None
        self.suivant = None

    def __str__(self):
        return str(self.val) + " - " + str(self.suivant)



class File:
    def __init__(self):
        self.premier = None
        self.dernier = None

    def ajoute(self,x):
        e = Element(x)
        if self.premier == None:
            self.premier = e
        else:
            e.precedent = self.dernier
            self.dernier.suivant = e
        self.dernier = e

    def file_vide(self):
        return self.premier is None

    def retire(self):
        if not self.file_vide():
            e = self.premier
            if e.suivant is None:
                self.premier = None
                self.dernier = None
            else:
                self.premier = e.suivant
                self.premier.precedent = None
            return e.val

    def __str__(self):
        return str(self.premier)
```

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.
4. Ajouter un attribut taille et une méthode longueur permettant d'afficher le nombre d'éléments présents dans la 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 Creer une fonction affichage() qui permet l'affichage sous forme de liste.

2. Mêmes questions pour les files en utilisant Queue(-1).

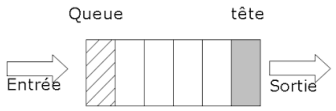

Figure 2: Schématisation d'une file