Fiche d'exercices - **Chapitre A.2 - La programmation orientée objets**

Exercice 1 - Sur ordinateur

Prenons l'implémentation des piles sous la forme de liste python.
Écrire un fonction `retourne_pile(p)` qui prend une Pile en paramètre et qui renvoie une nouvelle Pile contenant les même éléments de la pile `p` mais inversée.
Par exemple, si les éléments de la pile `p` sont a, b, c alors les éléments de notre pile renversé seront c, b, a.

> Proposer une solution permettant de ne pas modifier la pile `p` initial. (La pile `p` à la fin de l'exécution de la fonction doit être identique à la pile `p` au début de la fonction.)

Exercice 2 - Sur ordinateur

Prenons l'implémentation des piles en Programmation Orientés Objets. Écrire les fonction suivantes :

1. Écrire la fonction `affiche` qui prend en paramètre une pile et qui retourne la chaîne de caractère permettant un affichage de la pile.

> Votre fonction ne doit pas modifier la pile

```
>>> p = Pile()
>>> p.empile(5)
>>> p.empile(8)
>>> p.empile(7)
>>> print(affiche(p))
|7|
|8|
|5|
```

2. Écrire la fonction `creation_pile` qui prend en paramètre une liste python et renvoie la pile contenant les valeurs de la liste dans l'ordre de lecture.

```
>>> p = creation_pile([5,4,5,6,2,9,0])
>>> print(affiche(p))
|0|
|9|
|2|
|6|
|5|
|4|
|5|
```

3. Écrire la fonction separation qui prend en argument une pile d'entier et qui sépare dans deux nouvelles piles les nombres pairs et impairs. Votre fonction renverra un tuple contenant les deux piles

```
>>> p = creation_pile([8,4,6,7,9,4,7])
>>> p1, p2 = separation(p)
>>> print(affiche(p1))
|4|
|6|
|4|
|8|
>>> print(affiche(p2))
|7|
|9|
|7|
```

> Écrire une fonction `taille` qui prend une pile en paramètre et qui renvoie la taille de la pile. (Sans la modifier)
> Écrire une fonction `diviser` qui prend une pile en paramètre et qui renvoie de pile contenant respectivement la première et la deuxième moitié de la pile.

Exercice 3 - Sur ordinateur

Prenons l'implémentation des files en Programmation Orientés Objets. Ecrire les fonctions suivantes :

1. Écrire la fonction `affiche` qui prend en paramètre une file et qui retourne la chaîne de caractères permettant un affichage de la file.

> Votre fonction ne doit pas modifier la file !

```
>>> f = File()
>>> f.enfile(5)
>>> f.enfile(8)
>>> f.enfile(7)
>>> print(affiche(f))
|7|8|5|
```

2. Écrire la fonction creation_file qui prend en paramètre une liste python et renvoie la file contenant les valeurs de la liste stockées dans l'ordre de lecture.

```
>>> f = creation_file([5,4,5,6,2,9,0])
>>> print(affiche(f))
|0|9|2|6|5|4|5|
```

3. Écrire la fonction copie qui prend en paramètre une liste python représentant une file et renvoie une copie de cette même file.

```
>>> f = creation_file([5,4,5,6,2,9,0])
>>> f2 = copie(f)
>>> print(f)
|0|9|2|6|5|4|5|
>>> print(f2)
|0|9|2|6|5|4|5|
```

Exercice 4 - Sur ordinateur

Nous allons étudier la vérification des délimiteurs dans une expression.
On distingue trois type de délimiteurs, les parenthèses « (» et «) », les accolades « { » et « } » et les crochets « [» et «] ». L'objectif est de déterminer si une chaîne de caractère (qui pourrait être un programme) est correctement délimitée, c'est-à-dire si chacun des délimiteurs ouvrant possède un délimiteurs fermant qui lui est associés.

1. Parmi les expressions suivantes, lesquels sont correctement délimitée ?
   (a) `a(b)c`
   (b) `a[b(c))`
   (c) `a()()cc(f())`
2. Est-il possible de vérifier une chaîne de caractères uniquement en comptant les délimiteurs, si non, donner un contre exemple.
3. Écrire une fonction `delim_pile` qui renvoie un booléen qui vaut `True` si la chaîne est correctement délimitée et `False` sinon.

> L'objectif est de modifier `delim_pile` pour qu'il puisse vérifier qu'un fichier HTML est correctement écrit
> - Comment lire un fichier texte caractère par caractère ?
> - Comment repérer les balises ouvrantes et les balises fermantes ?
> - Comment récupérer le nom d'une balise ouvrante ?
> - Comment récupérer le nom d'un balise fermante ?
> - Écrire une fonction get_balise qui renvoie la liste des balises ouvrantes et fermantes d'un fichier html passé en paramètre.
> - Modifier votre fonction `delim_pile` pour qu'il vérifie la délimitation d'un fichier html.

Exercice 5 - Sur feuille

Les interfaces des structures de données abstraites Pile et File sont proposées ci-dessous. On utilisera uniquement les fonctions ci-dessous :
Structure de données abstraite : Pile

- Opérations :
  - $creer\_pile\_vide : \emptyset \rightarrow Pile$
    creer_pile_vide() renvoie une pile vide
  - $est\_vide : Pile \rightarrow booléen$
    est_vide(pile) renvoie True si pile est vide, False sinon
  - $empiler : Pile, Element \rightarrow \emptyset$
    empiler(pile, element) ajoute element à la pile pile.
  - $depiler : Pile \rightarrow Element$
    depiler(pile) renvoie l'élément au sommet de la pile en le retirant de la pile

Structure données abstraite : File

- Opérations :
  - $creer\_file\_vide : \emptyset \rightarrow File$
    creer_file_vide() renvoie une file vide
  - $est\_vide : File \rightarrow booléen$
    est_vide(file) renvoie True si file est vide, False sinon

- $enfiler : File, Element \rightarrow \emptyset$
  enfiler(file, element) ajoute element à la file file.
- $defiler : Pile \rightarrow Element$
  defiler(file) renvoie l'élément au sommet de la file en le retirant de la file

1. On considère la file F suivante :

Enfilement > |"rouge", "vert", "jaune", "rouge", "jaune"| > Défilement
(a) Quel sera le contenu de la pile P et de la file F après l'exécution du programme suivant :

```
P = creer_pile_vide()
while not est_vide(F):
    empiler(P, defiler(F))
```

(b) Créer une fonction taille_file qui prend en paramètre une file F et qui renvoie le nombre d'éléments qu'elle contient . Après appel de cette fonction, la file F doit avoir retrouvé son état d'origine.

2. Écrire une fonction former_pile qui prend en paramètre une file F et qui renvoie une pile P contenant les éléments que la file.

Le premier élément sorti de la file devra se trouver au sommet de la pile ; le deuxième élément sorti de la file devra se trouver juste en dessous du sommet, etc...
**Exemple : Si F =

"Rouge"	"Vert"	"Jaune"	"Rouge"	"Jaune"

P=

"Jaune"
"Rouge"
"Jaune"
"Vert"
"Rouge"

3. Écrire une fonction nb_elements qui prend en paramètre une file F et un élément elt et qui renvoie le nombre de fois où elt est présent dans la file F.

   Après l'appel de cette fonction, la file F doit avoir retrouvée son état d'origine

4. Écrire une fonction verifier_contenu qui prend en paramètre une file F et trois entiers : nb_rouge, nb_vert et nb_jaune.

   Cette fonction renvoie le booléen True si "rouge" apparaît au plus nb_rouge fois dans la file F, "vert" apparaît au plus nb_vert fois dans la file F et "jaune" apparaît au plus nb_jaune fois dans la file F. Elle renvoie False sinon. On pourra utiliser les fonctions précédentes.