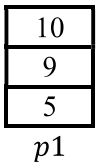
TP : MANIPULER DES PILES

Exercice 1 :

1) Créer une pile $p1$ qui contiendra les valeurs suivantes :



$p1$

Remarque : pour vérifier votre travail, une fois le programme compilé, vous pouvez taper $p1$ dans la console et visualiser le contenu de la pile.

```
>>> p1
Bas | 5    9    10   | Haut
```

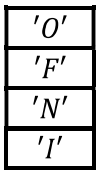
2) Il y a une erreur de saisie : il fallait mettre au milieu de la pile la valeur 8 au lieu de la valeur 9. Modifier le contenu de la pile $p1$ (sans créer de nouvelle pile).

Exercice 2 :

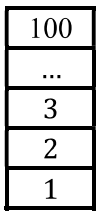
Créer les quatre piles suivantes :


$p2$ (pile vide)

$p3$



$p4$



$p5$

Exercice 3 :

Ecrire une fonction qui prend comme paramètre d'entrée une pile et retourne l'élément situé au sommet de la pile. Le contenu de la pile ne doit pas être modifié. Si la pile est vide, None est retourné.
Le nom de cette fonction est $sommet$ et son entête est :

```
def sommet(pile):
```

Tester votre fonction avec chacune des piles $p1, p2, \ldots p5$.

Exercice 4 :

Ecrire une fonction qui prend comme paramètre d'entrée une pile et permute les deux éléments situés au sommet de la pile. On supposera que la pile en entrée a une taille au moins égale à 2 et la fonction ne retournera rien.
On l'appellera $permute$ et son entête est :

```
def permute(pile):
```

Tester cette fonction avec la pile $p5$
Indication : taper dans la console
>>> permute(p5)
>>> p5

Exercice 5 :

a) Ecrire une fonction $troisieme\_elem(pile)$ qui prend comme paramètre d'entrée une pile et qui retourne l'antépénultième élément de cette pile. Le contenu de la pile ne sera pas modifié. On supposera que la pile en entrée a une taille au moins égale à 3. La tester avec la pile $p4$

b) Ecrire une fonction $dixieme\_elem(pile)$ qui prend comme paramètre d'entrée une pile et qui retourne le dixième élément de cette pile (dixième en comptant à partir du sommet de la pile). Le contenu de la pile ne sera pas modifié. On supposera que la pile en entrée a une taille au moins égale à 10. La tester avec la pile $p5$.

c) Ecrire une fonction $base\_pile(pile)$ qui prend comme paramètre d'entrée une pile et qui retourne l'élément situé à la base de la pile. Le contenu de la pile ne sera pas modifié. On supposera que la pile en entrée n'est pas vide. La tester avec les différentes piles .

Exercice 6 :

Il s'agit d'écrire une fonction $taille(pile)$ qui prend comme paramètre d'entrée une pile et retourne la taille de cette pile (le nombre d'éléments contenus dans cette pile). Le contenu de la pile ne doit pas être modifié suite à l'exécution de cette fonction.

Voici la solution proposée par un élève :

```
def taille(pile):
    res = 0
    while not pile.estVide():
        elem = pile.depiler()
        res = res + 1
    return res
```

Qu'en pensez-vous ? Justifiez votre réponse et proposez éventuellement une correction.

Exercice 7

1) Ecrire une fonction $inverse(pile)$ qui prend une pile en argument et renvoie une autre pile constituée des mêmes éléments placés dans l'ordre inverse. La pile peut être modifiée suite à l'exécution de la fonction

2) Ecrire une fonction $inversebis(p)$ qui prend une pile p en argument et renvoie une autre pile constituée des mêmes éléments placés dans l'ordre inverse. Le contenu de la pile p ne doit pas être modifié.