1 DS 3 Groupe 1

Devoir Surveillé n°3 : Parcours séquentielle d'un tableau	Thème 4 : Langage et programmation
	DEVOIR

Exercice n°1 :

Écrire une fonction recherche qui prend en paramètres un tableau tab de nombres entiers triés par ordre croissant et un nombre entier n, et qui effectue une recherche dichotomique du nombre entier n dans le tableau non vide tab. Cette fonction doit renvoyer un indice correspondant au nombre cherché s'il est dans le tableau, -1 sinon.

Exemples :

```
>>> recherche([2, 3, 4, 5, 6], 5)
3
>>> recherche([2, 3, 4, 6, 7], 5)
-1
```

```
assert recherche([2, 3, 4, 5, 6], 5)==3
assert recherche([2, 3, 4, 6, 7], 5)==-1
```

Exercice n°2 :

Écrire une fonction qui prend en paramètre un tableau d'entiers non vide et qui renvoie la moyenne de ces entiers. La fonction est spécifiée ci-après et doit passer les assertions fournies.

```
def moyenne (tab):
    '''
    moyenne(list) -> float
    Entrée : un tableau non vide d'entiers
    Sortie : nombre de type float
    Correspondant à la moyenne des valeurs présentes dans le
    tableau
    '''
```

```

```

```
assert moyenne([1]) == 1
assert moyenne([1,2,3,4,5,6,7] == 4
assert moyenne([1,2]) == 1.5
```

Exercice 3 : Moyenne coefficientée

On dispose ici de deux tableaux de même longueur : un tableau de notes (sur 20) et un tableau de coefficients. Il s'agit de calculer la moyenne coefficientée correspondante. Par exemple avec :

```
notes  = [12, 15, 14, 18]
coeffs = [1,  2,  1,  4 ]
```

la moyenne sera calculée ainsi :

```
(12*1 + 15*2 + 14*1 + 18*4)/(1 + 2 + 1 + 4)
```

Dans la fonction ci-dessous on appelle `num` (pour *num*érateur) ce qui correspond à `(12*1 + 15*2 + 14*1 + 18*4)` et `denom` (pour *dénom*inateur) ce qui correspond à `(1 + 2 + 1 + 4)`.

Compléter la fonction `moyenne_coefficientee` ci-dessous qui prend en paramètre deux tableaux non vides `notes` et `coeffs` et renvoie la moyenne coefficientée correspondante.

```
def moyenne_coefficientee(notes, coeffs):
    pass
```

Tester votre fonction grâce au jeu de tests ci-dessous.

```
notes = [10, 10, 20]
coeffs = [1, 3, 1]
assert moyenne_coefficientee(notes, coeffs) == 12.0

notes = [15, 17, 13, 19, 15, 11]
coeffs = [2, 3, 5, 5, 3, 8]
assert moyenne_coefficientee(notes, coeffs) == 14.384615384615385

notes = [8, 8, 8, 8, 12]
coeffs = [1, 1, 1, 1, 4]
assert moyenne_coefficientee(notes, coeffs) == 10.0
```

Exercice n°4 : Tableaux et extremums : plus grand dénivelé

En randonnée cycliste, pédestre ou à skis, on différencie les dénivelés positif et négatif. Ici nous ne ferons pas la différence.

Par exemple avec ce tableau :

```
[7, 4, 3, 6, 7, 4, 3, 1, 8]
```

on rencontre huit dénivelés lors de son parcours -3, -1, 3, 1, -3, -1, -2, 7.

Plus formellement, dans un tableau `tab` de taille `n` le dénivelé à l'indice `i` est égal à `tab[i+1] - tab[i]` (à condition que l'indice `i+1` existe).

Voici le schéma d'un tableau `tab` de taille `n` :

```
-----------------------------------------------
| indices | 0 | 1 | 2 | 3 | ... | n-2 | n-1 |
-----------------------------------------------
```

```
| valeurs | . | . | . | . | ... |  .  |  .  |
-----------------------------------------------
```

Le plus grand indice possible est donc `n-1`.

Compléter la fonction `plus_grand_denivele` ci-dessous qui prend en paramètre un tableau `tab` **de plus de deux nombres** et renvoie le plus grand dénivelé qui existe dans ce tableau.

```
def plus_grand_denivele(tab):
    pass
```

Tester votre fonction en utilisant le jeu de tests ci-dessous.

```
assert plus_grand_denivele([2, 5, 8, 3, -2, 13, 17, 18, 16, 13, 11, 5]) ==
assert plus_grand_denivele([17, 18, 16, 13, 11, 5, 2, -1, -14, -15]) == 1
assert plus_grand_denivele([22, 16, 13, 11, 5, 2, -1, -14, -19]) == -2
assert plus_grand_denivele([22, 28]) == 6
```