PYTHON : LISTES ET CHAÎNES DE CARACTÈRES

EXERCICE 1

Corriger l'instruction suivante :

```
'J'aime Python'
```

EXERCICE 2

1) En utilisant le *slicing*, programmer une fonction **miroir** qui renvoie la chaîne de caractères écrite en sens inverse.
Par exemple **miroir**(`'agro'`) renvoie `'orga'`.

2) Un palindrome est un texte qui se lit de la même façon dans les deux sens. Par exemple « kayak » est un palindrome.
Programmer la fonction **palindrome** qui teste si une chaîne de caractères est un palindrome.

EXERCICE 3 (Des étoiles plein les yeux)

- Écrire une fonction **etoilesHoriz** qui prend un nombre entier positif `n` en argument, et affiche autant d'étoiles horizontalement.
  Par exemple pour $n = 3$, elle affiche :
  ***
- Écrire une fonction **etoilesVert** qui prend un nombre entier positif `n` en argument, et affiche autant d'étoiles verticalement.
  Par exemple pour $n = 3$, elle affiche :
  *
  *
  *
- Écrire une fonction **etoilesTri** qui prend un nombre entier positif `n` en argument, et affiche un triangle rectangle dont les côtés ont n étoiles.
  Par exemple pour $n = 3$, elle affiche :
  *
  **
  ***

EXERCICE 4

Sans l'exécuter sur l'ordinateur, essayer de prévoir ce que fait l'algorithme ci-contre, puis vérifier sa conjecture en exécutant ce programme.

Programmer ensuite le même algorithme en utilisant une boucle **for**.

```
def mystere(liste):
    res = 0; i = 0
    while i < len(liste):
        res += liste[i]
        i += 1
    return res
```

EXERCICE 5 (Opérations statistiques sur une liste)

1) Programmer une fonction **produit** qui renvoie le produit de tous les éléments d'une liste de nombres.

2) Programmer une fonction **maximum** qui renvoie le maximum d'une liste de nombres.

3) Programmer une fonction **moyenne** qui renvoie la moyenne d'une liste de nombres.

EXERCICE 6 (Opérations entre ensembles)

On a deux ensembles, dont on place les éléments dans des listes. On suppose donc que les éléments de chaque liste sont distincts deux à deux. L'ordre des éléments dans la liste est indifférent.

1) Programmer une fonction**appartenir**(`a,liste`) qui teste si l'élément `a` appartient à `liste`. La fonction renvoie **True** ou **False**.

2) Programmer une fonction **estInclus**(`liste1,liste2`) qui teste si tous les éléments de `liste1` sont dans `liste2`. La fonction renvoie **True** ou **False**.

3) Programmer une fonction **intersection**(`liste1,liste2`) qui renvoie la liste de l'intersection.

4) Programmer une fonction **egalite**(`liste1,liste2`) qui renvoie si les listes sont égales (d'un point de vue ensembliste).

EXERCICE 7 (Coefficients binomiaux)

1) Triangle de Pascal

(a) Écrire une fonction **trianglePascal**(`n`) qui *affiche* le triangle de Pascal jusqu'à la ligne $n \geq 0$.
On construira le triangle en utilisant la formule du triangle de Pascal.
Essayer de programmer avec le moins de calculs possible et sans occuper trop de place dans la mémoire.

(b) En modifiant légèrement l'algorithme précédent, programmer une fonction **binome**(`n,k`) qui renvoie le calcul de $\binom{n}{k}$ pour $(n,k) \in \mathbb{N}^2$.

2) Méthodes directes

(a) Programmer une fonction **factorielle**(`n`), qui calcule $n!$ pour $n \in \mathbb{N}$.

(b) En déduire une fonction **binome2**(`n,k`) qui calcule directement à partir de la définition la valeur de $\binom{n}{k}$.

(c) Écrire une fonction **binome3**(`n,k`) qui calcule la valeur de $\binom{n}{k}$ à partir de la définition : $\binom{n}{k} = \prod_{i=1}^{k} \frac{n-k+i}{i}$.

3) Comparaison

(a) Comparer **binome**(`80,20`), **binome2**(`80,20`), **binome3(80,20)**.
Qu'en pensez-vous ?

(b) En utilisant la fonction `time` de la bibliothèque du même nom, comparer les temps d'exécution des trois algorithmes précédents.