Listes, chaînes de caractères, dictionnaires

TP3

Exercice 1 (liste et compréhension)**.**

Dans chacun des cas, créer la liste demandée deux deux façons différentes. D'abord à l'aide d'une fonction, ensuite à l'aide d'une compréhension de liste.

1. les 16 premiers carrés entiers : 0, 1, 4, ..., 225.
2. les $f(x)$ pour $x \in [\![-10, 10]\!]$ pour $f(x) = 3x^2 - 5x - 20$
3. la liste des entiers dont le carré est inférieurs à 1532.
4. les diviseurs de 3876 (*validation : il y en a 24*).

Exercice 2 (Des bases)**.**

1. Écrire une fonction `maxi` qui détermine le maximum d'une liste de nombres.
2. Écrire une fonction `contient` qui prend en argument une liste et un élément et qui retourne `True` si l'élément est dans la liste et `False` sinon.
   Pour cette question, on ne doit pas utiliser l'instruction `in`.
3. Écrire une fonction `maxi_indice` qui détermine le plus petit indice du maximum d'une liste de nombres.
   Par exemple `maxi_indice( [1,3,4,2,4,0] ) == 2`
   Comment modifier cette fonction pour obtenir le plus grand indice du maximum ?
4. Écrire une fonction `nb_occurrence` qui prend en argument une liste et un élément et qui renvoie le nombre d'occurrence de l'élément dans la liste.

Exercice 3 (Des graphiques)**.**

Tracer un courbe avec Python

Pour tracer un graphe, on peut utiliser le module `matplotlib`. Voici un exemple de script qui permet de tracer quelques segments.

```
1 import matplotlib.pyplot as plt
2
3 x_list = [-1,  1, -1, 4]  # Création de la liste des abscisses
4 y_list = [-1, -1,  1, 0]  # Création de la liste des ordonnées
5
6 plt.axes(aspect='equal')  # le repère devra être orthonormé
7 plt.plot(x_list, y_list)  # définition du tracé
8 plt.show()                 # affichage
```

Notez que les axes ne sont pas centrés sur l'origine mais sur les bords de la figure. La ligne 6 n'est obligatoire que si on veut un repère orthonormé.

1.(a) Écrire une fonction `regulier` qui prend en argument trois nombres, `mini`, `maxi` et `nb` et qui retourne une liste de `nb+1` valeurs régulièrement espacées entre `mini` et `maxi`.
Par exemple `regulier(2, 4.5, 5) == [2, 2.5, 3, 3.5, 4, 4.5]`

(b) Tracer la fonction $f(x) = 0,3x^4 + 1,6x^3 + 1,2x^2 - x$ entre -4,5 et 1,5 à l'aide de 60 points régulièrement espacés.

2. Soit $\alpha \in [0;4]$ un paramètre. On définie la suite logistique de paramètre α par :

$$\begin{cases} u_0 = 0,1 \\ u_{n+1} = \alpha u_n(1-u_n) \quad \text{pour } n \in \mathbb{N} \end{cases}$$

Tracer le graphe des 20 premiers termes de la suite pour $\alpha = 1,5$. Que semble être le comportement de (u_n) à l'infini ?

Même question pour les 100 premiers termes de la suite pour $\alpha = 3,2$.

Même question pour les 350 premiers termes de la suite pour $\alpha = 4$.

Exercice 4 (Plus techniques)**.**

1. Écrire une fonction `moyenne` qui prend une liste de nombres en argument et qui calcule la moyenne des valeurs de la liste.

2. Écrire une fonction `variance` qui prend une liste de nombres en argument et qui calcule la variance des valeurs de la liste.
Pour rappel, la variance se calcule à l'aide de la formule :

$$Var(X) = \frac{1}{n}\sum_{k=1}^{n} {x_k}^2 - E(X)^2 \qquad \text{où } X = (x_1, \cdots, x_n) \text{ et } E(X) \text{ est la moyenne de } X$$

Exercice 5 (Chaines)**.**

1. Écrire une fonction `pyramide` qui prend un entier `n` en paramètre et affiche une pyramide de caractères `*`. Ce caractère peut être modifié par un second paramètre facultatif (valant `"*"` par défaut).

Note : Le caractère `\` s'écrit avec deux anti-slashs : `"\\"`.

```
 >>> pyramide(4)
*\
**\
***\
****>
***/
**/
*/
```

2. Écrire une fonction `envers` qui prend une chaîne de caractères et retourne la chaîne inverse.
Par exemple, `envers("maths") == "shtam"`.

3. Écrire une fonction `nb_voyelles` qui prend une chaîne de caractère en argument et retourne le nombre de voyelles.
Par exemple, `nb_voyelles("variation") == 5`.

Exercice 6 (Dictionnaire)**.**

1. Écrire une fonction `composition` qui prend une chaîne de caractères en argument et retourne un dictionnaire dont les clés sont des lettres et les valeurs sont les nombres d'occurrence de la lettre dans la chaîne. Par exemple
`composition('bouillotte') == {'b':1, 'o':2, 'i':1, 'l':2, 'u':1, 'e':1, 't':2}`

2. Écrire une fonction `maxi_lettres` qui prend en argument une chaîne de caractères et donne la liste des caractères les plus fréquents dans la chaîne (l'ordre des lettres n'a pas d'importance).
Par exemple, `maxi_lettre('bouillotte') == ['o', 'l', 't']`.