TP1 — Rappels de Python

Exercice 1 – *Suite et dynamique de population*

On considère une suite u_n représentant le nombre d'habitants dans un pays au jour n. Soit $\alpha \in [0,1]$ le taux de naissance journalier et $\beta \in [0,1]$ le taux de décès. On suppose que $\beta > \alpha$. L'évolution de la population est donnée par la suite :

$$u_{n+1} = u_n + \alpha u_n - \beta u_n.$$

Question 1.1. Écrire une fonction qui prend u_n, α et β en entrée, et qui renvoie u_{n+1}.
Cette fonction ne construit pas la suite $(u_n)_n$, mais calcule simplement un terme u_{n+1} à partir du terme précédent u_n (notamment, une boucle n'est pas nécessaire pour cette fonction).
Tester cette fonction avec $u_n = 1$, $\alpha = 0.5$ et $\beta = 1$.

Question 1.2. Écrire une boucle `while` qui tourne tant qu'une erreur est plus grande que $\varepsilon = 0.001$. Dans cette boucle, on :

- calcule u_{n+1} à partir de u_n et de la fonction précédente,
- calcule l'erreur avec $E = |u_{n+1} - u_n|$,
- stocke u_n dans une liste L_u,
- affecte u_{n+1} dans u_n.

Avant la boucle, on initialisera u_n et on le stockera en tant que premier élément de la liste. Après la boucle, on affichera la liste L_u.

Question 1.3. Créer une liste L_n contenant les entiers de 0 à $K-1$, avec $K-1$ la taille de la liste L_u. Tracer le comportement de la suite pour différentes valeurs de α et β avec `matplotlib`. On pourra adapter le code ci-dessous.

```
1  # import de matplotlib
2  import matplotlib.pyplot as plt
3
4  # construit une liste x avec les nombres de 0 à 7
5  x = [0, 1, 2, 3, 4, 5, 6, 7]
6  # construit une liste y de même taille que la liste x
7  y = [-1, 4, 5, 2, 1, -2, 0, 3]
8
9  # place chaque couple de points de (x, y) dans le plan et les relie
10 # ici, le premier point est (0, -1), le deuxième est (1, 4), etc.
11 plt.plot(x, y, color="red", marker="+")
12 plt.show()
```

Question 1.4. On va manipuler la liste L_n. Dans un premier temps, supprimer les 4 derniers éléments de cette liste. Puis prendre une tranche des cette liste entre deux indices `i0` et `i1`, et copier ces tranches dans une nouvelle liste. Modifier le dernier élément de cette nouvelle liste. Enfin, afficher l'ancienne et la nouvelle liste, et commenter.

Exercice 2 – *Dictionnaire : base de données de notes*

On va construire et manipuler une petite base de données d'étudiants et leurs notes. Les données du semestre pour chaque étudiant seront stockées dans des listes. L'ensemble sera contenu dans un dictionnaire, dont :

— les **clés** seront les **prénoms** des étudiants,
— les **valeurs** seront les listes des notes.

Question 2.1. Créer un dictionnaire avec `"Marie"` de notes $[12, 15, 11]$ et `"Jean"` de notes $[14, 9.5, 15]$.

Question 2.2. Ajouter au dictionnaire `"Eva"` de notes $[10, 9, 11]$. Afficher le nombre d'éléments du dictionnaire.

Question 2.3. Avec une boucle, afficher les noms et les notes des étudiants ayant des notes en-dessous de la moyenne. Pour ceux qui ont une note strictement supérieure à 10, modifiez la note et passez-la à 10.

Exercice 3 – *Polynômes*

On va construire des fonctions permettant de calculer et d'afficher les valeurs de polynômes.
On considère le polynôme suivant, de degré k :

$$P(x) = \sum_{n=0}^{k} a_n x^n.$$

Question 3.1. Écrire une fonction qui prend en entrée la liste des coefficients a_n ainsi que le point x où la valeur de $P(x)$ sera calculée, et qui renvoie cette valeur. On pourra utiliser une boucle `for`.
Tester la fonction. Pour cela, on pourra prendre $a_n = 1$ pour tout $n \in [\![0, 6]\!]$.

Question 3.2. Tracer la valeur de $P(x)$ pour $x \in [-1, 1]$. On pourra s'inspirer de la **Question 1.3** pour l'utilisation de `plt.plot`. On pourra aussi utiliser la fonction `np.linspace` pour créer un vecteur `x` de taille `n` avec `n` éléments équirépartis de `a` (inclus) à `b` (inclus) : `x = np.linspace(a, b, n)`.

Question 3.3. En s'inspirant de la fonction de la **Question 3.1**, écrire des fonctions qui renvoient à la fois la valeur de $P(x)$ et le degré de P, de plusieurs façons :
— en modifiant simplement la fonction de la **Question 3.1** pour renvoyer aussi le degré,
— en utilisant une boucle `while` au lieu de la boucle `for`,
— en utilisant la fonction `enumerate`.
Vérifier que ces fonctions donnent bien les mêmes résultats que celle de la question **Question 3.1**.

Exercice 4 – *Partie entière*

Dans cet exercice, on propose d'implémenter une fonction partie entière. On rappelle que cette fonction est définie, pour $x \in \mathbb{R}$, comme l'unique entier $n \in \mathbb{Z}$ tel que

$$n \leq x < n + 1.$$

Question 4.1. Créer une fonction `partie_entière` qui calcule la partie entière d'un nombre x.
Pour cela, on pourra distinguer plusieurs cas suivant le signe de x, puis utiliser des boucles `while` pour trouver $E(x)$.
Tester ensuite cette fonction sur plusieurs nombres (par exemple, 0.3, 4, 13.7, -0.5, -3.5, ...).

Question 4.2. Tracer cette fonction pour $x \in [-5, 5]$.
Pour cela, utiliser une boucle (par exemple, `for`) pour créer une liste contenant les résultats de la fonction. Le tracé de la fonction pourra être fait avec la fonction `scatter` de la bibliothèque `matplotlib.pyplot`. On pourra adapter le code ci-dessous.

```
1  # import de matplotlib
2  import matplotlib.pyplot as plt
3
4  # construit une liste x avec les nombres de 0 à 7
5  x = [0, 1, 2, 3, 4, 5, 6, 7]
6  # construit une liste y de même taille que la liste x
7  y = [-1, 4, 5, 2, 1, -2, 0, 3]
8
9  # place chaque couple de points de (x, y) dans le plan, sans les relier
10 # ici, le premier point est (0, -1), le deuxième est (1, 4), etc.
11 plt.scatter(x, y, color="red", marker="+")
12 plt.show()
```

Question 4.3. Essayer d'appliquer directement la fonction `partie_entière` à `x = np.linspace(-5, 5, 100)`. Que se passe-t-il ? Comment expliquer ce comportement ?

Question 4.4. Comparer les résultats de la fonction `partie_entière` avec ceux de la fonction `floor` de la bibliothèque `numpy` (`np.floor`). Pour cela, on pourra tracer les résultats des deux fonctions. Laquelle est la plus facile d'utilisation ?