Intro. au Calcul Scientifique

Warm up

Nom :
Prénom :
Section :

Question 1. Dans un fichier nommé `question1.py`, écrivez une fonction `plus_petits(x, k)` qui retourne les `k` plus petits éléments de la liste `x` et une fonction `plus_grands(x, k)` qui retourne les `k` plus grands éléments de `x`. (On suppose que les éléments de `x` sont comparables avec l'ordre usuel.)

Au travers de cette question, vous veillerez à

- apprendre le typage de générique de votre code (la cohérence de vos types peut être montrée dans votre éditeur grâce, par exemple, à `pylint`),
- documenter *adéquatement* votre code,
- comprendre la différence entre `.sort()` et `sorted()` et leur complexité algorithmique,
- installer `pytest` (via `pip` ou `uv`) et ajouter des *tests unitaires*.

Considérez également les questions suivantes.

- Que doit retourner `plus_petit([[2], [33, 1], [1, 7]], 2)` ?
- Qu'est-ce que le code suivant imprime (en expliquant pourquoi, en fonction de votre implémentation de `plus_petits`) ?

```
x = [[2], [3, 2], [3, 1]]
y = plus_petits(x, 2)
for a in y:
    a.sort()

print(x)
```

Question 2. Écrivez une fonction `positifs_croissants(x)` qui retourne les éléments positifs (les valeurs a telles que $a \geqslant 0$) de `x` triées par ordre croissant.

La question suivante nécessite d'avoir installé `matplotlib`.

Question 3. Soit $a \in]0, +\infty[$. On considère la fonction

$$f_a : \mathbb{R} \to \mathbb{R} : x \mapsto a \ln\left(\frac{2x}{x+1}\right) - \frac{x}{2x+1}.$$

Dans un fichier `q3.py`, écrivez une fonction `nombre_racines` qui prend a comme seul argument et retourne le nombre de racines de f_a. Prouvez[1] que votre programme est correct (il doit l'être comme si les ordinateurs pouvaient calculer avec tous les nombres réels même si, comme nous le verrons, certains réels ne sont pas représentables avec les nombres flottants).

[1] Au risque d'enfoncer des portes ouvertes, faire une preuve demande que tous les arguments invoqués soient rigoureux.