

DS 3

Exercice 1 - Une fonction récursive facile.

Les puissances d'un nombre peuvent être définies récursivement par la formule :

$$\forall a \in \mathbb{R}, \forall n \in \mathbb{N}, \quad a^{n+1} = a \times a^n \quad \text{et} \quad a^0 = 1.$$

En se reposant sur la formule ci-dessus, proposer une fonction $puiss(a, n)$ qui calcule a^n en s'appelant elle-même,

Exercice 2 - Méthode de Newton. On se donne une fonction $f : \mathbb{R} \rightarrow \mathbb{R}$, dont on cherche à calculer un point d'annulation c'est-à-dire une solution de l'équation $f(x) = 0$. On va mettre en place la méthode de Newton, vue en cours de maths.

On suppose que la fonction f et sa dérivée f' ont été codées, et sont notées en python f et fp . Proposer une fonction $Newton(a, f, fp, E)$ qui calcule les valeurs de la suite définie par la méthode de Newton :

$$u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)} \quad \text{et} \quad u_0 = a,$$

jusqu'à ce que $|f(u_n)| < E$. La fonction renverra la dernière valeur calculée.

Bonus : On ajoutera un critère d'arrêt au cas où ce premier critère n'est pas atteint : on limitera l'algorithme à 100 itérations au maximum.

Exercice 3 - Programmer le produit matriciel.

On rappelle qu'étant données deux matrices carrées A et B , de tailles n , on définit leur produit matriciel par $C = AB$, avec

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}.$$

On souhaite coder ce produit en python, les matrices étant représentées au format *array* du module *numpy* (renommé *np*). Attention, dans ce format les indices démarrent à 0, et les tailles sont fixes, par exemple, $M = np.zeros((3, 3))$ crée une matrice de taille 3×3 remplies de 0, que l'on peut ensuite remplir, par exemple à l'aide de boucles sur les lignes et sur les colonnes.

Ce format fonctionne comme les autres itérables, avec quelques spécificités : la taille d'une matrice M (un couple d'entiers) est donné par la fonction $np.shape(M)$, et on accède à l'élément (i, j) avec la commande $M[i, j]$.

- Quelle ligne de code permet d'importer le module *numpy*, et de le renommer *np* ?
- Créer une fonction *identite(n)* qui crée la matrice identité de taille n .
- Etant données deux matrices carrées A et B de même taille, créer une fonction *produit(A, B)* qui renvoie la matrice produit $C = AB$. On recodera les coefficients du produit à la main, sans utiliser la commande *dot* de *numpy*.
- Etant données deux matrices A et B carrées de même taille, en utilisant les deux fonctions précédentes, écrire une fonction *test_inv(A, B)* qui renvoie *True* si B est l'inverse de A , et *False* sinon. Attention, la commande $==$ pour les matrices doit être complétée par la commande *.all* comme suit : $M.all() == N.all()$, qui renvoie *True* si les matrices M et N sont de même taille et égales, et *False* sinon.