
ITC MPSI - TP 3 : FONCTIONS

21 SEPTEMBRE 2026

Exercice 1 (Suite de Syracuse, avec des fonctions)

1. Écrivez une fonction `etape_syracuse` qui applique une étape de la fonction de Syracuse. Votre fonction prendra en entrée un entier naturel u et renverra

$$\begin{cases} \frac{u}{2} & \text{si } u \text{ est pair} \\ 3 * u + 1 & \text{si } u \text{ est impair} \end{cases}$$

2. Écrivez une fonction `delai_convergence` qui prend en entrée un entier naturel u , et qui renvoie le premier indice k tel que $s_k = 1$, où $(s_k)_{k \in \mathbb{N}}$ est la suite de Syracuse démarrant à u :

$$\begin{aligned} s_0 &= u \\ \forall k \in \mathbb{N}, s_{k+1} &= \begin{cases} \frac{s_k}{2} & \text{si } s_k \text{ est pair} \\ 3 * s_k + 1 & \text{si } s_k \text{ est impair} \end{cases} \end{aligned}$$

Votre fonction `delai_convergence` devra faire appel à `etape_syracuse`.

3. Écrivez une fonction `premiers_termes` qui prend en entrée deux entiers naturels u et n et qui renvoie la liste des n premiers termes de la suite de Syracuse démarrant à u . **Votre fonction `premiers_termes` devra utiliser la méthode `append`, et faire appel à `etape_syracuse`.**

Exercice 2 (Dérivation)

On considère une fonction réelle f dérivable en a et on se propose d'estimer sa dérivée en a par

$$f'(a) \approx \frac{f(a+h) - f(a-h)}{2h}$$

1. Écrire une fonction `derive` de paramètres `f`, `a` et `h` qui retourne une valeur approchée de $f'(a)$. Tester cette fonction dans des cas connus avec différentes valeurs de h (par exemple, la fonction carré, la fonction exponentielle, etc ...).
2. Écrire une fonction `derive_compose` de paramètres `f`, `g`, `a` et `h` qui retourne une valeur approchée de $(g \circ f)'(a)$. Cette fonction **doit** appeler la fonction `derive`.

n-uplets

Un n-uplet (ou *tuple* en anglais) est une collection de n valeurs. Un n-uplet se comporte presque comme une liste, à ceci près qu'il ne peut pas être modifié (on ne peut pas y ajouter des valeurs avec `append`, et on ne peut pas modifier une valeur contenue dedans).

La syntaxe pour définir un n-uplet constant est de mettre les différentes valeurs entre parenthèses, séparées par des virgules :

$(valeur1, valeur2, \dots)$

Par exemple, le code suivant

```
coordonnees = (2, 3)
abcisse = coordonnees[0]
ordonnee = coordonnees[1]
print("Le point", coordonnees, "a pour abcisse", abcisse,
      "et pour ordonnée", ordonnee)
```

a pour effet d'afficher

Le point (2, 3) a pour abcisse 2 et pour ordonnée 3

Exercice 3 (n-uplets)

1. Codez une fonction `produit_scalaire` qui prend en entrée deux n-uplets de même taille n (représentant les coordonnées de deux vecteurs de $\mathbb{R}^n$), et qui renvoie le produit scalaire des deux vecteurs en question.
2. Écrire une fonction `tangente` de paramètres `f`, `a` et `h`, qui retourne un tuple de valeurs `(m, p)` où l'équation approchée de la tangente au graphe de f en a est de la forme $y = mx + p$. Cette fonction **doit** appeler la fonction `derive` de l'exercice précédent.