Recursivité

Pour faire ce TP, on utilisera Future Coder et Python Tutor et prendra soin du bien simuler le code.

I Approche descendante

Implémentez les fonctions suivantes

Exercice 1. *Fonction d'Ackermann*

La fonction d'Ackermann est une fonction à deux paramètres définie par récurrence comme ceci :

$$\mathrm{A}(m,n) = \begin{cases} n+1 & \text{si } m = 0 \\ \mathrm{A}(m-1,1) & \text{si } m > 0 \text{ et } n = 0 \\ \mathrm{A}(m-1,\mathrm{A}(m,n-1)) & \text{si } m > 0 \text{ et } n > 0. \end{cases}$$

Exercice 2. *Fonction 91 de McCarthy*

La fonction de McCarthy est une fonction définie par récurrence comme ceci :

$$f(n) = \begin{cases} n-10 & \text{si } n > 100 \\ f(f(n+11)) & \text{sinon.} \end{cases}$$

II Fonctions récursives

Exercice 3. *Suite récurrente simple*

On définit $u_0 = 12$ et $u_{n+1} = \sqrt{u_n + 1}$.

Écrire une fonction récursive qui prend en entrée un entier $n \geq 0$ et calcule la valeur de u_n.

Exercice 4. *Suite de Syracuse*

Une suite de Syracuse est définie par récurrence :

$$\mathrm{S}_{n+1} = \begin{cases} \dfrac{\mathrm{S}_n}{2} & \text{si } \mathrm{S}_n \text{ est pair,} \\ 3\mathrm{S}_n + 1 & \text{si } \mathrm{S}_n \text{ est impair.} \end{cases}$$

Le premier terme n'est pas fixé.

Écrire une fonction récursive `R` qui prend en entrée un entier $s \geq 1$ et calcule le premier entier $n \geq 0$ tel que $\mathrm{S}_n = 1$ si $\mathrm{S}_0 = s$.

On commencera par établir une relation de récurrence pour `R` qui determine `R(s)` en fonction de `R(s_suiv)` avec `s_suiv` le terme suivant `s` dans la suite de Syracuse qui débute en `s`.

Exercice 5. *Somme*

Écrire une fonction récursive `somme_rec(l,i)` qui prend en entrée une liste `l` d'entiers, un entier `i` tel que `0 ⩽ i ⩽ len(l)` et calcule la somme des `i` premiers éléments de `l`.

Finaliser l'exercice en implémentant une fonction `somme(l)` qui renvoie la somme des éléments de `l` avec une approche descendante du calcul.

Exercice 6. *Maximum*

Écrire une fonction récursive `maximum_rec(l,i)` qui prend en entrée une liste `l` d'entiers, un entier `i` tel que `0 ⩽ i ⩽ len(l)` et calcule le maximum des `i` premiers éléments de `l`.

On pourra utiliser la fonction `max(a,b)` de Python qui renvoie le plus grand élément parmis `a` et `b`.

On choisira de renvoyer $-\infty$ (`-float("inf")`) pour le maximum d'une liste vide.

Finaliser l'exercice en implémentant une fonction `maximum(l)` qui renvoie le maximum des éléments de `l` avec une approche descendante du calcul.

Exercice 7. *Recherche linéaire*

Écrire une fonction récursive `recherche_rec(l,x,i)` qui prend en entrée une liste `l` d'entiers, un entier `i` tel que `0 ⩽ i ⩽ len(l)`, un objet `x` et renvoi l'indice de `x` dans `l[:i]` et `-1` si `x` n'est pas dans `l[:i]`.

Exercice 8. *Fibonnacci*

On définit $F_0 = 0, F_1 = 1$ et $F_{n+2} = F_{n+1} + F_n$.

Écrire une fonction récursive qui prend en entrée un entier $n \geq 0$ et calcule la valeur de F_n.

Testez la fonction sur de petites valeurs, puis augmentez petit à petit. Améliorer la fonction pour pouvoir calculer F_{100}.

Exercice 9. *Coefficient Binomial*

On utilisera **exclusivement** la définition du binôme par la formule de récurrence du triangle de Pascal :

$$\binom{n}{k} = \begin{cases} 0 & \text{si } k > n \\ 1 & \text{si } k = 0 \text{ ou } k = n \\ \binom{n-1}{k} + \binom{n-1}{k-1} & \text{sinon.} \end{cases}$$

Écrire une fonction récursive qui prend en entrée deux entiers `n` et `k`, et qui renvoie $\binom{n}{k}$.