TD14. Premières fonctions récursives

Bases de la Programmation Impérative – Chapitre 4 : recursivite
version du 7 juin 2024

L'objectif de ce TD est d'écrire nos premières fonctions récursives et de **bien comprendre le flot de contrôle** lors que ces fonctions récursives sont exécutées.

Préambule :

Dans ce TD nous allons faire de nouveaux dessins :) À titre d'exemple, voici un programme Python exécutant une fonction récursive ainsi que le dessin associé à l'exécution de cette fonction. Assurez-vous d'avoir bien compris ce qui se passe lors de l'exécution et demandez-vous ce qu'il se passe si on remplace `3` par `-1` lors de l'appel initial à la fonction `affiche_n_fois_hello`.

```python
def affiche_n_fois_hello(n):
    # Condition d'arrêt
    if n == 0:
        return

    # Corps de la fonction
    print("hello")

    # Appel(s) recursif(s)
    affiche_n_fois_hello(n - 1)

    # Inutile mais aide à bien comprendre
    return None

affiche_n_fois_hello(3)
print("C'est la fin !")
```

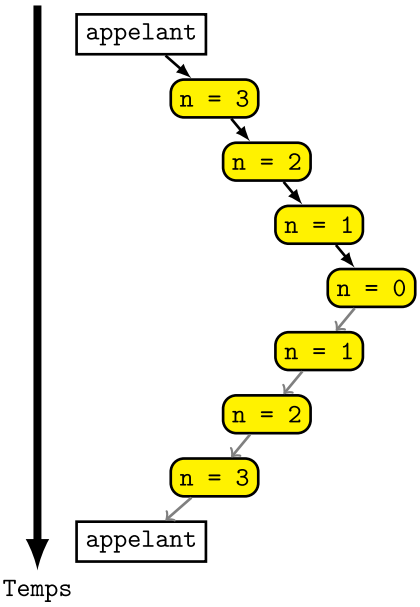

Fig. 1 : exécution affiche n fois

Exercice 1 : conjecture de Syracuse

On se propose de ré-implémenter le calcul du nombre d'étapes pour que la suite de Syracuse atteigne 1. Pour rappel la suite de Syracuse est définie à partir d'un entier u_0 quelconque par :

- si u_i est pair alors $u_{i+1} = u_i/2$
- sinon $u_{i+1} = 3 \times u_i + 1$

Question 1

Implémenter la fonction `calcule_nb_etapes_avant_1` en utilisant la récursivité.

Question 2

Exécuter pas-à-pas l'appel de fonction `calcule_nb_etapes_avant_1(10)`. Pour cela, dessiner schématiquement l'exécution sur le modèle du dessin de l'exécution de la fonction `affiche_n_fois_hello` ci dessus.

Exercice 2 : exponentiation rapide

On souhaite calculer x^y où `x` est un `float` ou un `int` et `y` un `int` strictement plus grand que zéro.

Question 1

En remarquant que pour `y` pair on a $x^y = (x^{y/2})^2$, proposer une fonction récursive calculant très rapidement x^y.

Question 2

Exécuter pas à pas l'appel de fonction `exponentiation(2, 10)`. Pour cela, dessiner schématiquement l'exécution sur le modèle du dessin de l'exécution de la fonction `affiche_n_fois_hello` ci dessus.

Exercice 3 : suite de Fibonacci

On considère le programme suivant :

```
1  def fibonacci(rang):
2      """Renvoie le terme de rang donné de la suite de Fibonacci."""
3      print("rang", rang)
4      if rang <= 1:
5          return rang
6      return fibonacci(rang - 1) + fibonacci(rang - 2)
7
8  if __name__ == "__main__":
9      print(fibonacci(4))
```

Question 1

Qu'affiche ce programme sur la sortie standard ?

Question 2

Intuitivement, quelle est la complexité de la fonction `fibonacci` ?

Exercice 4 : dichotomie récursive

Question 1

Implémenter une recherche dichotomique récursive dans un tableau trié. La fonction implémentée renvoie `True` si l'élément cherché est présent dans le tableau et `False` sinon.

Exercice 5 : Fibonacci qui va vite (pour aller plus loin)

Question 1

Implémenter une autre version de la fonction `fibonacci` bien plus efficace que celle de l'exercice 3.