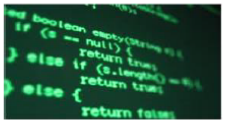
NSI (TERMINALE) :

Programmation par récursivité



Implémentation d'une fonction puissance

Exercice 1 : calcul de 2^n avec n entier naturel

1ère définition : **Implémentation de la fonction puissance par la méthode itérative**

si $n = 0$ alors $2^n = 1$, sinon $2^n = \underbrace{2 \times 2 \times 2 \times \ldots\ldots\ldots\ldots \times 2}_{n \text{ fois}}$

1°) Implémenter une fonction `calc(n)` prenant en paramètre un entier naturel n et retournant en sortie la valeur de 2^n avec une méthode itérative (boucle !!!)

2ième définition : **Implémentation de la fonction puissance par récursivité.**

si $n = 0$ alors $2^n = 1$, sinon $2^n = 2 \times 2^{n-1}$

Notons puissance(n) la fonction donnant la valeur de 2^n.
Avec cette notation, la définition devient : **si $n = 0$ alors puissance (0) = 1 sinon puissance (n) = $2 \times$ puissance($n - 1$)**

ALGORITHME :

Donnée(s) : n un entier naturel
fonction puissance(n)
Si n=0 **alors**
  renvoyer 1
Sinon
  renvoyer 2*puissance(n-1)

On en déduit que si $n > 0$ alors la fonction puissance s'appellera elle-même. On parle de définition par **récursivité**.

2°) Implémenter cette fonction `puissance(n)`

Exercice 2 : comparaison des vitesses d'éxécution

Comparer les vitesses d'exécutions des programmes version itérative et version récursive de l'exemple précédent. Pour tester la vitesse d'exécution d'une fonction on utilise le module **`timeit`**, comme le montre le code ci-dessous pour un entier naturel aléatoirement choisi entre 400 et
une liste de 1000 entiers choisis aléatoirement entre 500 et 1000 avec le module **`random`**.

```
from timeit import default_timer as timer
from random import randint

# Les deux fonctions ici

N=randint(500,1000)

debut=timer()
print(calc(N))
fin=timer()
print(f"temps version itérative {fin-debut}")
debut=timer()
print(puissance(N))
fin=timer()
print(f"temps version récursive {fin-debut}")
```

Que pouvez-vous en conclure ici ?