TP n°4 : fonctions

PTSI Lycée Eiffel

17 octobre 2013

1 Lecture et analyse de programmes

Pour chacun des programmes suivants, **AVANT** de les exécuter sous Python, lisez le programme, cherchez à comprendre ce qu'il fait, et prévoyez ce que renverra $f(10)$ ainsi que $f\left(\frac{1}{2}\right)$, puis vérifiez vos hypothèses en recopiant les programmes et en les exécutant.

Première fonction

```
> def f(x) :
>         y=3*x
>         z=y*x
>         return(z-y)
```

Deuxième fonction

```
> def f(x) :
>         if x < x*x :
>                 return x
>         else :
>                 return x*x
```

Troisième fonction

```
> def f(x) :
>         a=0
>         s=0
>         while s<x :
>                 a=a+1
>                 s=s+a
>         return a
```

2 Quelques petits exemples pour commencer à programmer vous-même

1. Programmer une fonction en Python prenant comme paramètres un nombre x et un entier n et calculant x^n (on pourra évidemment programmer une boucle à l'intérieur de la définition de la fonction).
2. Programmer une fonction en Python prenant comme paramètre un entier n et calculant le n-ème terme de la suite de Fibonacci (définie, rappelons-le, par les conditions initiales $F_0 = 0$ et $F_1 = 1$ et par la relation de récurrence $F_{n+2} = F_{n+1} + F_n$).

3. Programmer une fonction en Python prenant comme paramètre un entier n et calculant la somme des chiffres de n (en base 10, bien entendu).

3 Pour s'entraîner avec la récursivité

Reprendre chacun des trois exercices précédents, mais en écrivant à chaque fois une fonction récursive. Dans le cas du calcul de puissance, réfléchir à une version vraiment intelligente, ne calculant pas x^n sous la forme $x \times x^{n-1}$ mais plutôt sous la forme $x^{\frac{n}{2}} \times x^{\frac{n}{2}}$ (du moins si n est pair). On essaiera même d'estimer le nombre de multiplications nécessaires pour calculer x^n. Dans le cas de la suite de Fibonacci, on testera le programme récursif sur des valeurs de n de l'ordre de quelques dizaines, et on se demandera pourquoi c'est beaucoup plus lent que la version non récursive.

4 Amusons-nous un peu avec les nombres entiers

Toute cette partie du TP est consacrée à l'écriture d'algorithmes faisant intervenir les nombres entiers, et notamment la notion de nombre premier. Il existe de nombreuses façons d'écrire les différents programmes demandés, notamment celui permettant de savoir si un entier donné est premier ou non. N'hésitez pas, une fois que vous avez écrit des programmes qui tournent, à comparer avec ce qu'ont pu faire vos camarades, et à réfléchir à des améliorations pour optimiser la rapidité d'exécution de vos programmes. Mais prenez quand même d'abord le temps de bien tout faire par vous-même !

1. Programmer une fonction prenant comme paramètre un entier n et déterminant s'il est premier ou non (la fonction renverra donc un booléen, autrement dit une variable ne pouvant prendre comme valeur que **True** ou **False**).
2. Programmer une fonction prenant comme paramètre un entier n et affichant la liste de tous les nombres premiers inférieurs ou égaux à n.
3. Programmer une fonction prenant comme paramètres deux entiers n et p et déterminant leur pgcd à l'aide de l'algorithme d'Euclide (si vous ne savez pas comment fonctionne cet algorithme, on prendra le temps de l'expliquer !). On pourra écrire deux versions, l'une récursive, l'autre non.
4. Programmer une fonction prenant comme paramètre un entier n et déterminant son plus petit facteur premier.
5. Programmer une fonction prenant comme paramètre un entier n et calculant sa décomposition en facteurs premiers (sous la forme d'une liste de facteurs premiers, éventuellement répétés k fois si le facteur apparait élevé à la puissance k dans la décomposition de n).

5 Pour ceux qui ont tout trouvé trop facile

La monnaie actuellement en cours en France est l'euro, et on dispose de pièces d'1 et 2 euros, et de billets de 5, 10, 20, 50, 100, 200 et 500 euros (on oublie les centimes pour simplifier un peu). Programmer une fonction en Python prenant comme paramètre un entier n et déterminant de combien de façons différentes on peut payer une somme de n euros à l'aide de toutes les pièces et billets disponibles (on pourra commencer en utilisant uniquement les pièces).