DS d'informatique : corrigé

PTSI Lycée Eiffel

22 novembre 2013

Exercice 1

1. On peut faire extrêmement rapide en utilisant les divisions euclidiennes :

```
def ent(x) :
    return x//1
def par(n) :
    return n%2
```

2.
```
def mystere(i,j) :
    t=0
    while i>=1 :
        if par(i)==1 :
            t=t+j
        j=2*j
        i=ent(i/2)
    return t
```

Une remarque à propos de ce programme : il n'est en fait pas très judicieux d'utiliser les variables i et j à la fois comme paramètres de la fonction et comme variables locales (on modifie leurs valeurs en cours de calcul). Ce n'est pas très gênant ici (si vous faites tourner le programme sous Python, vous constaterez que les valeurs de i et j après exécution sont les valeurs initiales, comme si Pyhton créait de lui-même une variable locale distincte pour les calculs), mais il est plus soigneux de recopier les valeurs de i et j dans deux autres variables et de modifier ces dernières.

3. Détaillons les calculs pour mystere(8,9). Au départ, t reçoit la valeur 0.
   - premier passage dans la boucle while : i est pair, donc on ne midifie pas t, j est multiplié par 2 donc vaut maintenant 18, et i divisé par 2 et vaut maintenant 4.
   - deuxième passage dans la boucle while : i est pair, t reste nul, j devient égal à 36 et i à 2.
   - troisième passage dans la boucle while : i est pair, t reste nul, j devient égal à 72 et i à 1.
   - quatrième passage dans la boucle while : i est impair, on remplace t par 0+72, soit 72, j devient égal à 144 et i à 0.

   Le programme s'arrête ici, et sort la valeur 72.

   Décrivons le fonctionnement pour mystere(9,8), le point de départ étant le même, t reçoit la valeur 0.
   - i est impair, t devient égal à 8, j à 16 et i à 4.
   - i est pair, j devient égal à 32 et i à 2.
   - i est pair, j devient égal à 64 et i à 1.
   - i est impair, t devient égal à 8+64=72, j à 128 et i à 0.

   Le programme s'arrête et sort à nouveau la valeur 72.

4. Le programme calcule simplement le produit des deux entiers i et j. Une façon simple de le prouver est de constater qu'à la fin de chaque passage dans la boucle, la valeur de i*j+t reste inchangée. En effet, si i est pair, i est divisé par 2 et j multiplié par 2, donc i*j ne change pas, et t non plus ; et si i est impair, i devient égal à (i-1)/2, j à 2j, et t à t+j, et (i-1)/2*2j+t+j=ij-j+t+j=ij+t. Au début du programme, ij+t est égal au produit des deux nombres donnés puisque t=0. À la fin, au contraire ij+t=t puisque i est nul (c'est toujours un entier naturel et on continue tant qu'il est supérieur ou égal à 1), dona la valeur de t ressortie est bien égale au produit des deux nombres initiaux.

Exercice 2

1.
```
def ajoute(x,l) :
    l=[x]+l
def enlevetete(l) :
    l=l[1 :]
```
Bon, en fait, je dois être honnête avec vous : ces fonctions ne marchent pas car les passages par valeur dans les fonctions Python ne permettent pas de modifier l'argument quand il s'agit d'une liste comme ici (autre dit, l ne va pas être changée). Il faudrait remplacer la dernière ligne par un return [x]+l, par exemple, et du coup changer le ajoute(y,l) de la fonction définie ensuite dans l'énoncé par un l=ajoute(y,l).

En tout cas, la fonction ajoute prend deux paramètres, l'un de type entier (si on admet qu'on ne travaille qu'avec des listes de nombres entiers) et l'autre de type liste. La fonction enlevetete ne prend qu'un argument, de type liste.

2. (a) Puisque k est égal à 3, on va faire trois passages dans la boucle :
   - premier passage, y prend la valeur 0 (puisque 20 est pair), l devient égale à [0] et x à 10.
   - deuxième passage, y prend la valeur 0 (puisque 10 est pair), l devient égale à [0,0] et x à 5.
   - premier passage, y prend la valeur 1 (puisque 5 est impair), l devient égale à [1,0,0] et x à 2.

   Le résultat sorti est donc la liste [1,0,0].

   (b) La liste contiendra k éléments, qui sont les k derniers chiffres de l'écriture binaire de l'entier n (dans l'exemple ci-dessus, 20 s'écrit en binaire sous la forme 10100).

   (c)
```
def binaire(n) :
    l=[ ]
    x=n
    while x!=0 :
        y=x %2
        ajoute(y,l)
        x=(x-y)/2
    return l
```

3. (a) La première ligne amoce la définition de la fonction, dont le nom est fonctionb, et qui prend deux arguments : une liste l et un entier k. La deuxième ligne copie la liste l dans une variable locale nommée t. La troisième ligne initialise une autre variable locale u, de type liste, à la valeur liste vide. La quatrième ligne amorce une boucle while qui s'arrêtera lorsque la longueur de la liste t deviendra nulle, c'est-à-dire lorsque t sera devenue vide.

   (b) La fonction calcule simplement les k derniers chiffres de l'écriture binaire de chaque élément de la liste l, et ressort une liste de listes. Ici, on aura simplement le résultat [[1,1,0],[0,0,0],[0,1,1],[1,0,1]].