DS 2 - Informatique - CPBX 1

21 novembre 2012 - Durée : 1h30 minutes

Les documents, les ordinateurs et les téléphones ne sont pas autorisés.
Les modules Python ne sont pas autorisés pour écrire les programmes.

Exercice 1

Que font les programmes suivants ? Vous représenterez toute la mémoire allouée par l'ordinateur et toutes les valeurs prises par les espaces mémoires en barrant les espaces mémoires et les valeurs temporaires.

Programme 1 :

```
1 def fonction1( u ):
2     return u + 1
3 def fonction2( u ):
4     return fonction1( fonction1( u ) )
5 u = 3
6 u = fonction2( fonction2( u ) )
7 print( u )
```

Programme 2 :

```
1 s = [0 , 0]
2 t = s
3 t[0] = 5
4 s[1] = 9
5 print( s )
6 print( t )
```

Exercice 2

1. Que font les fonctions suivantes ?

   Fonction 1 : (t est un tableau, i et j sont des entiers)

```
1 def mystere1( t, i, j ):
2     t[i] = t[i] + t[j]
3     t[j] = t[i] - t[j]
4     t[i] = t[i] - t[j]
```

   Fonction 2 : (a et b sont des réels)

```
1 def mystere2( a, b ):
2     return ( (a<b) and b ) + ( (b<a) and a )
```

   En python, `False and a` vaut `False`, `True and b` vaut `b`, `False + b` est égal à `b + False` et vaut `b`, enfin `False + False` vaut 0.

2. Proposez une implémentation plus simple et plus lisible des fonctions `mysteres`.

Exercice 3: Quel jour est-ce aujourd'hui ?

On souhaite pourvoir déterminer le jour de la semaine (lundi, mardi, mercredi, etc ...) d'une date donnée. Pour cela, on peut utiliser la congruence de Zeller, qui est une formule qui

donne la date d'un jour donné. Cette formule est définie par :

$$h = \left(q + \left\lfloor \frac{13(m+1)}{5} \right\rfloor + K + \left\lfloor \frac{K}{4} \right\rfloor + \left\lfloor \frac{J}{4} \right\rfloor + 5.J \right) \bmod 7$$

où

- h est le jour de la semaine codé avec la convention : 0=samedi, 1=dimanche, 2=lundi, ... ;
- q est le jour du mois ;
- m est le mois de l'année codé avec la convention : 3=mars, 4=avril, 5=mai, ..., 12=décembre, 13=janvier, 14=février) ;
- K est l'année du siècle (année mod 100) ;
- J est le siècle ($\lfloor$année/100$\rfloor$).

1. Écrivez une fonction `jour_de_la_semaine(l)` qui prends en paramètre une liste l de trois entiers, où :
   - $l[0]$ est le jour du mois,
   - $l[1]$ le mois de l'année en utilisant le codage usuel (janvier=1, février=2, ... décembre=12)
   - $l[2]$ est l'année,

   et qui renvoie le jour de la semaine codée en utilisant le codage usuel (1=lundi, 2=mardi, ..., 7=dimanche).
2. Écrivez un programme `afficher_jour(l)` qui affiche le jour de la date passée en paramètre en utilisant les même paramètres et les même conventions que la fonction `jour_de_la_semaine(l)`.

Exercice 4

On souhaite calculer les coéfficients qui apparaissent lors du développement de $(x+1)^n$. Il est bien connu que

$$(x+1)^n = \sum_{k=0}^{n} C_n^k x^k \qquad \text{avec} \qquad C_n^k = \frac{n!}{k!(n-k)!}.$$

1. Proposez une implémentation de la fonction `binomial_1(n,i)` qui prends en paramètres deux entiers n et i et qui renvoie C_n^i.
2. Écrivez un programme qui calcule la somme $\sum_{k=0}^{n} C_n^k (-1)^k$. Y-a-t-il un moyen simple pour vérifier si votre programme est juste ?
3. Démontrez que, pour tout $i \in [1, n]$, $C_{n+1}^i = C_n^{i-1} + C_n^i$ et $C_n^0 = C_n^n = 1$.
4. En utilisant la formule de récurence précédente, écrivez une fonction `calcul_binomiaux(T)` qui prends en paramètre une liste T à $n+1$ éléments vérifiant $T[i] = C_n^i$ et qui renvoie une nouvelle liste U à $n+2$ éléments où $U[i] = C_{n+1}^i$. Par exemple, `calcul_binomiaux([1])` renvera `[1,1]`, `calcul_binomiaux([1,1])` renvera `[1,2,1]`, `calcul_binomiaux([1,2,1])` renvera `[1,3,3,1]`, `calcul_binomiaux([1,3,3,1])` renvera `[1,4,6,4,1]`, etc ...
5. Écrivez une fonction `binomiaux(n)` qui renvoie la liste T contenant $n+1$ éléments définis par $T[i] = C_n^i$.
6. Écrivez une fonction `binomial_2(n,i)` qui calcule C_n^i sans utiliser de factorielles.

Solutions des exercices

Exercice 1

Voir solutions du DS 1 et du TP 3.

Exercice 2

1. La fonction `mystere1` échange les valeurs contenues dans `t[i]` et `t[j]` ;
   La fonction `mystere2` renvoie la valeure maximale entre a et b.
2.

```
def echange( t, i, j ):
    tmp = t[i]
    t[i] = t[j]
    t[j] = tmp
```

```
def max( a, b ):
    if a > b:
        return a
    return b
```

Exercice 3

```
def fonction_de_zeller( q, m, k, j ):
    return ( q + (13*(m+1))/5 + k + k/4 + j/4 + 5*j ) % 7

def jour_de_la_semaine( l ):
    q = l[0]
    annee = l[2]
    m = l[1]
    if( m <= 2 ):
        m = m + 12
    k = annee % 100
    j = annee / 100
    jour = fonction_de_zeller( q, m, k, j )
    if jour <= 1 :
        jour = jour + 6
    else:
        jour = jour - 1
    return jour

def afficher_jour( l ):
    jours = [
        "", "Lundi", "Mardi", "Mercredi", "Jeudi",
        "Vendredi", "Samedi", "Dimanche"
    ]
    print( jours[ jour_de_la_semaine(l) ] )
```

Exercice 4

1.
```
def factorial( n ):
    res = 1
    for i in range(1,n+1):
        res *= i
    return res

def binomial_1(n,i):
    return factorial(n)/factorial(i)/factorial(n-i)
```

2. Le programme qui calul $\sum_{k=0}^{n} C_n^k(-1)^k$ est :

```
def calcul( n ):
    somme = 0
    signe = 1
    for k in range(n+1):
        somme = somme + binomial_1(n,k) * signe
        signe = - signe
    return somme
```

On sait que $0^n = ((-1)+1)^n = \sum_{k=0}^{n} C_n^k(-1)^k$. Pour vérfier si le programme est juste il suffit de vérifier qu'il renvoie 0 quelque soit la valeure de n donnée en paramètre.

3. $C_n^0 = C_n^n = \frac{n!}{0!n!} = 1$.
$C_n^i + C_n^{i-1} = \frac{(n)!}{i!(n-i)!} + \frac{n}{(i-1)!(n-i+1)!} = \frac{n!(n-i+1)+n!i}{i!(n-i+1)!} = \frac{(n+1)!}{i!(n+1-i)!} = C_{n+1}^i$.

4. Voici le programme `calcul_binomiaux` :

```
def calcul_binomiaux( T ):
    n = len( T )
    res = [1]
    for i in range(1,n):
        res.append( T[i]+T[i-1] )
    res.append( 1 )
    return res
```

5. Voici le programme `binomiaux` :

```
def binomiaux( n ):
    res = [1]
    for i in range(n):
        res = calcul_binomiaux( res )
    return res
```

6. Voici le programme `binomial_2` :

```
def binomial_2( n, i ):
    return binomiaux( n )[i]
```