UNIVERSITÉ LIBRE DE BRUXELLES
École polytechnique de Bruxelles

25 janvier 2013

INFO-H-100 – Informatique – Prof. Th. Massart
1ère année du grade de Bachelier en Sciences de l'Ingénieur
Interrogation de janvier

Remarques préliminaires

- On vous demande de répondre à **chaque question sur une feuille séparée**.
- N'oubliez pas d'inscrire votre nom, prénom et numéro de matricule sur chaque feuille.
- Vous disposez de 3 heures et vous ne pouvez pas utiliser de notes.
- La réponse à la question doit comprendre, si approprié, le code *Python* structuré et conforme aux règles de bonne pratique et conventions ainsi que des commentaires pertinents.
- Vous pouvez ajouter des fonctions si cela vous semble nécessaire.
- Sauf mention contraire, vous ne pouvez utiliser aucune fonction de librairies (pas d'`import`).

Question 1 - Théorie (5 points)

Que donnent les codes suivants ? Justifiez vos réponses et décrivez la situation et son évolution grâce à des diagrammes d'état (comme fait au cours).

1.
```
x=y=[0,2,4,6]
z=x[1:3]
print(x)
print(y)
print(z)
```

2.
```
x=[102,30]
while x[len(x)-1] != 0 :
  x.append(x[len(x)-2]%x[len(x)-1])
print(x)
```

3.
```
def foo(x,y):
    x[0]= y[1]
    x,y=y,x
    return x+y

z=[1,2,3]
z2=[4,5,6]
t=foo(z,z2)
print(z)
print(z2)
print(t)
```

4.
```
t=[0,0]
t[0]=t
print(t)
t[len(t)]=t
print(t)
```

5.
```
x=[[1,2,3],"Boum"]
y=x[:]
y[0][1]="Crac"
y[1]= 6
print(x)
print(y)
```

Question 2 - Approximation de $sin(x)$ (4 points)

La fonction $sin(x)$ peut être définie par la série suivante :

$$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + ... + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + ... = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

où x est exprimé en radians.

On vous demande d'écrire une fonction `sin_deg(x)` qui reçoit `x` en *degrés* et renvoie une approximation du sinus correspondant. Cette fonction doit être la combinaison de deux autres fonctions (que vous devez également écrire) : `sin(x)` qui reçoit un angle en *radians* et qui renvoie le sinus correspondant, et `deg_in_rad(x)` qui reçoit en argument un angle en degrés et qui renvoie le même angle en radians. Pour cette dernière fonction, vous devez employer la valeur `pi` de la libraire `math`.

Les calculs s'arrêteront quand la valeur absolue du dernier terme ajouté est inférieure à `EPS` ou lorsque le nombre de termes considérés atteint la borne `NMAX`. `EPS` et `NMAX` sont considérées comme des constantes définies globalement.

Pour rappel : $\theta_{rad} = \pi \cdot \theta_{deg}/180$

Question 3 - Les nombres de Bell (5 points)

Les nombres de Bell donnent le nombre de partitions d'un ensemble. Par exemple, le second nombre de Bell, $B_2 = 2$, indique qu'il est possible de partitioner un ensemble de deux éléments de deux manières distinctes (pour un ensemble $\{a, b\}$, les deux partitions sont $\{\{a, b\}\}$ et $\{\{a\}, \{b\}\}$. Le troisième nombre de Bell, $B_3 = 5$, correspond aux cinq partitionnements d'un ensemble à trois éléments (pour l'ensemble $\{a, b, c\}$, les cinq partitionnements possibles sont $\{\{a\}, \{b\}, \{c\}\}$, $\{\{a\}, \{b, c\}\}$, $\{\{b\}, \{a, c\}\}$, $\{\{c\}, \{a, b\}\}$, et $\{\{a, b, c\}\}$. La série démarre avec $B_0 = B_1 = 1$, et les quelques premiers termes sont (en reprenant les deux premiers) :

$$1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, \ldots$$

Les nombres de Bell peuvent se calculer grâce à un *schéma triangulaire*, c'est-à-dire en construisant le triangle de nombres suivant :

$$\begin{array}{ccccc} 1 & & & & \\ 1 & 2 & & & \\ 2 & 3 & 5 & & \\ 5 & 7 & 10 & 15 & \\ 15 & 20 & 27 & 37 & 52 \end{array}$$

Chaque ligne est construite en commençant par recopier le dernier nombre de la ligne précédente, puis en sommant le dernier nombre écrit avec le nombre juste au dessus. Ainsi, le 5 est calculé en faisant $3 + 2$. Il est ensuite copié à la ligne suivante, et le nombre suivant, 7, et calculé en faisant $5 + 2$. Le suivant est $7 + 3 = 10$, puis $10 + 5 = 15$. Comme il n'y a plus de nombre au-dessus de 15, on le recopie à la ligne et on recommence.

Les nombres de Bell sont les nombres en fin de ligne de ce triangle.

On vous demande d'écrire une fonction `bell(n)` qui renvoie le n^e nombre de Bell.

```
>>> bell(0), bell(1), bell(2), bell(3), bell(4)
(1, 1, 2, 5, 15)
```

Question 4 - Tri de voitures (6 points)

On représente une voiture par un dictionnaire dont les clefs sont "marque", "modèle", et "cartec" (pour caractéristiques techniques). Les trois clefs sont des chaînes de caractères. Les valeurs associées à `"marque"` et `"modele"` sont des chaînes de caractères. La valeur associée à `"cartec"` est un tuple de deux nombres qui correspondent respectivement à la puissance et au poids du véhicule.

On vous demande d'écrire une fonction qui trie, de façon *décroissante*, une liste de voitures en fonction du rapport `puissance/poids` de chaque voiture. Pour cela, vous pouvez utiliser soit le tri par insertion, soit le tri par sélection. **Veillez à préciser sur votre copie le tri utilisé.**

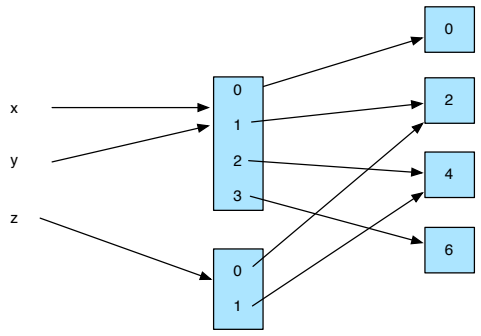
Par exemple :

```
>>> liste = [{"marque" : "Noble", "modele" : "M600", "cartec" : (650,1274)},
             {"marque" : "Spyker", "modele" : "C8", "cartec" : (406,1425)},
             {"marque" : "Wiesmann", "modele" : "MF3", "cartec" : (343,1180)},
             {"marque" : "Ariel", "modele" : "Atom 500", "cartec" : (500,550)}]
>>> print("Avant : ",liste)
Avant :  [{'marque': 'Noble', 'modele': 'M600', 'cartec': (650, 1274)},
          {'marque': 'Spyker', 'modele': 'C8', 'cartec': (406, 1425)},
          {'marque': 'Wiesmann', 'modele': 'MF3', 'cartec': (343, 1180)},
          {'marque': 'Ariel', 'modele': 'Atom 500', 'cartec': (500, 550)}]
>>> tri_selection(liste)
>>> print("Apres : ",liste)
Apres :  [{'marque': 'Ariel', 'modele': 'Atom 500', 'cartec': (500, 550)},
          {'marque': 'Noble', 'modele': 'M600', 'cartec': (650, 1274)},
          {'marque': 'Wiesmann', 'modele': 'MF3', 'cartec': (343, 1180)},
          {'marque': 'Spyker', 'modele': 'C8', 'cartec': (406, 1425)}]
```

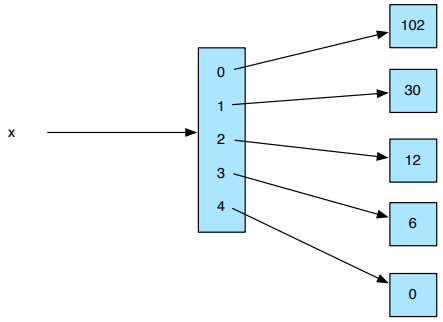
INFO-H-100 - Programmation
Interrogation de janvier
Corrections

Solution de la question 1

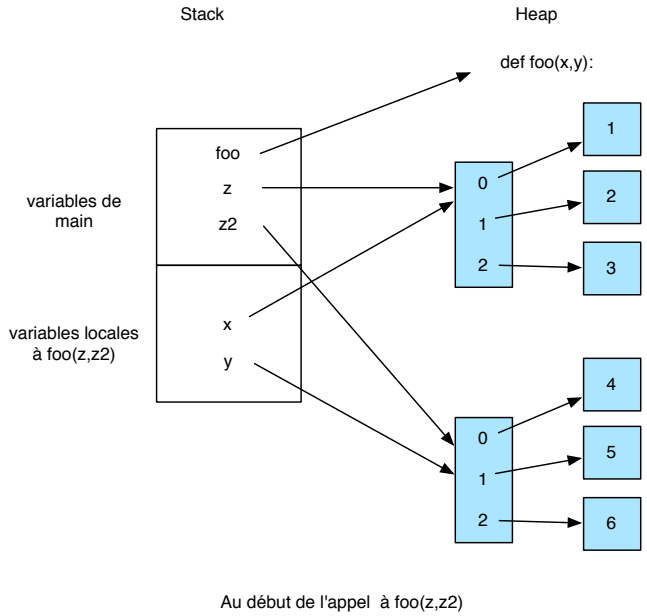
1. imprime
   [0, 2, 4, 6]
   [0, 2, 4, 6]
   [2, 4]



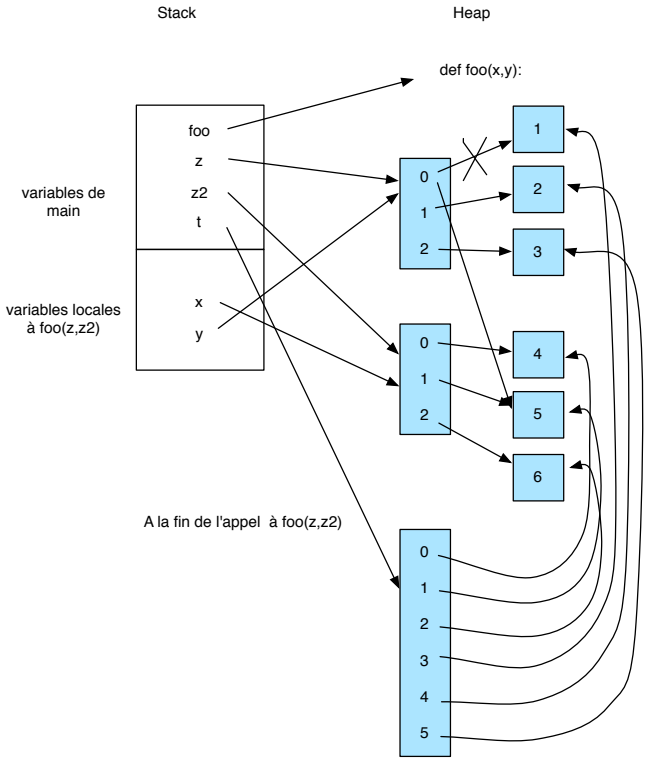
2. imprime
   [102, 30, 12, 6, 0]



3. imprime
   [5, 2, 3]
   [4, 5, 6]
   [4, 5, 6, 5, 2, 3]



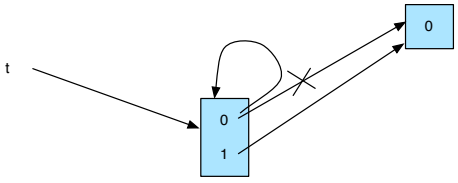
Au début de l'appel à foo(z,z2)



4. imprime

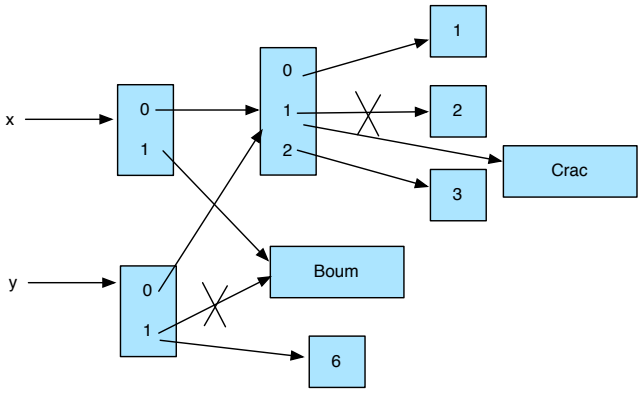
```
[[...], 0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list assignment index out of range
[[...], 0]
```

Les `...` est dû à la référence cyclique. L'erreur provient du fait que l'élément d'indice `len(t)` c'est-à-dire 2 n'existe pas dans `t`.



5. imprime

```
[[1, 'Crac', 3], 'Boum']
>>> print(y)
[[1, 'Crac', 3], 6]
```



Solution de la question 2

```
EPS = 0.0000001
NMAX = 10000

from math import pi

def deg_in_rad(x):
        return pi*x/180

def sin(x):
    somme = x
    x2 = x*x
    signe = 1
    terme = x
    den = 1
    nbTermes = 1
    facteurX = x
    n2 = 1
    while abs(terme) >= EPS and nbTermes < NMAX:
        nbTermes += 1
        signe = -signe
        n2 += 2
        den *= (n2)*(n2-1)
        facteurX *= x2
        terme = facteurX/den
        somme += signe*terme
    return somme

def sin_deg(x):
    return sin(deg_in_rad(x))

print(sin_deg(45))
print(sin_deg(60))
```

Solution de la question 3

```
def bell(n):
    return bell_line(n)[-1]

def bell_line(n):
    line = [1]
    for i in range(n-1):
        line = next_line(line)
    return line

def next_line(prev_line):
    new_line = [prev_line[-1]]
    for elem in prev_line:
        new_line.append(elem + new_line[-1])
    return new_line
```

Solution de la question 4

```
def puissance(car):
    return car["cartec"][0]

def poids(car):
    return car["cartec"][1]

def ratio_puissance_poids(car):
    return puissance(car)/poids(car)

def meilleur_ratio(car1,car2):
    return ratio_puissance_poids(car1) < ratio_puissance_poids(car2)

def tri_selection(ls):
    """ Procedure du tri par selection """
    for i in range(len(ls) - 1):
        pos = pos_min(ls, i)
        echange(ls, i, pos)


def echange(ls, i1, i2):
```

```
    """ Echange dans la liste 'ls', les valeurs des indices i1 et i2"""
    ls[i1], ls[i2] = ls[i2], ls[i1]


def pos_min(ls,i):
    mini = i
    for j in range(i,len(ls)):
        if meilleur_ratio(ls[mini],ls[j]):
            mini = j
    return mini

#############################################################################

def tri_insertion(ls):
    for i in range(1, len(ls)):
        j = pos_insert(ls[i], ls, i)
        deplacer(ls, i, j)


def deplacer(ls,i,j):
    """ Deplace dans la liste 'ls' la valeur en position i vers la position j"""
    val = ls[i]
    del ls[i]
    ls.insert(j,val)

def pos_insert(val, ls, n):
    """ Renvoi la position d'insertion de la valeur 'val' parmi les n 1er elements de la liste"""
    j = 0
    while j < n and meilleur_ratio(val,ls[j]):
        j += 1
    return j


################################

if __name__ == "__main__":

    liste = [{"marque" : "Noble", "modele" : "M600", "cartec" : (650,1274)},
             {"marque" : "Spyker", "modele" : "C8", "cartec" : (406,1425)},
             {"marque" : "Wiesmann", "modele" : "MF3", "cartec" : (343,1180)},
             {"marque" : "Ariel", "modele" : "Atom 500", "cartec" : (500,550)}]

    print("Avant : ",liste)
    tri_selection(liste)
    print("Apres : ",liste)

    liste = [{"marque" : "Noble", "modele" : "M600", "cartec" : (650,1274)},
             {"marque" : "Spyker", "modele" : "C8", "cartec" : (406,1425)},
             {"marque" : "Wiesmann", "modele" : "MF3", "cartec" : (343,1180)},
             {"marque" : "Ariel", "modele" : "Atom 500", "cartec" : (500,550)}]

    print("Avant : ",liste)
    tri_insertion(liste)
    print("Apres : ",liste)
```