UNIVERSITÉ LIBRE DE BRUXELLES janvier 2014
École polytechnique de Bruxelles

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 (7 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). Si des print sont effectués donner aussi leur résultat. De plus, pour le dernier code de cette question, la fonction `kesse` prend comme paramètre une liste de valeurs entières entre 0 et $MAX - 1$. Expliquer en termes mathématiques, ce qu'elle renvoie.

1.
```
d= {"A":1,"B":2,"C":3}
print(dict([[d[i],i] for i in d]))
```

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

3.
```
empty = []
x=[i for i in range(len(empty))]
if x is empty :
   print("ok")
print(x)
print(empty)
```

4.
```
def foo(x):
    x=1
t=[0,0]
print(foo(t))
print(t)
```

5.
```
x=[1,[2,3,4],[5,6,7]]
y=x[:]
x[2]=8
y[1][1]=[6]
print(x)
print(y)
```

6.
```
def foo(x,y):
    x[0]= y[1]
    x,y=y,x
    return x+y

z=[1,2]
z2=[z,z]
t=foo(z,z2)
print(z)
print(z2)
print(t)
```

7.
```
MAX = 5

def kesse(s):
   n = len(s)
   c = [0]*MAX
   for j in s:
      c[j] +=1
   i=0
   while c[i] < n//2:
     c[i+1] += c[i]
     i = i+1
   return i

liste= [4,3,2,2,3,1,1,1,2,0,0,4]

print(kesse(liste))
```

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

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

$$arcsin(x) = x + \frac{1}{2}\frac{x^3}{3} + \frac{1.3}{2.4}\frac{x^5}{5} + \frac{1.3.5}{2.4.6}\frac{x^7}{7} + ... = \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1}$$

On vous demande d'écrire une fonction `arcsin(x)` qui reçoit un `x` et renvoie une apprixmation de l'arcsinus correspondant.

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.

Question 3 - Triangle trinomial (4 points)

Une façon possible, quoique bien étrange, de calculer 3^n est de réaliser que le résultat peut être obtenu en sommant tous les nombres de la $n^{\text{ème}}$ ligne du triangle suivant (les lignes sont comptées en commençant à 0) :

```
              1
           1  1  1
        1  2  3  2  1
     1  3  6  7  6  3  1
  1  4  10 16 19 16 10 4  1
           . . .
```

Par exemple, la somme des nombres 1, 4, 10, 16, 19, 16, 10, 4, 1 de la ligne numéro 4 donne 81, soit 3^4.

Chaque nombre de ce triangle est obtenu en sommant les 3 nombres situés juste au dessus, au dessus à gauche, et au dessus à droite, et en considérant que l'extérieur du triangle est formé de zéros. Bien sûr, la première ligne est formée d'un simple 1.

Vous devez écrire (au minimum) deux fonctions :

- Une fonction `trinomial(n)` qui reçoit une entier positif, `n`, et renvoie, sous forme d'une liste d'entiers, la ligne correspondante du triangle.
- Une fonction `threePower(n)` qui reçoit un entier positif, `n`, et renvoie en utilisant `trinomial(n)`, la valeur 3^n **calculée de la façon décrite ci-dessus**.

Question 4 - Tri de contacts (5 points)

On représente un contact par un dictionnaire dont les clefs sont le nom (`name`), le prénom (`first_name`) et l'adresse (`address`). Les valeurs associées à `name` et `first_name` sont des chaînes de caractères. La valeur associée à `address` est un tuple de quatre valeurs qui correspondent respectivement au nom de la rue `(str)`, au numéro de la maison `(int)`, au code postal `(int)` et enfin au nom de la commune `(str)`.

On vous demande d'écrire une fonction qui trie, de façon *croissante*, une liste de contacts en fonction du code postal de sa résidence. 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 :

```
>>> abook = [{"name" : "Dupont", "first_name" : "Jean", "address" : ("Rue du Centre",10,4500,"Tihange")},
             {"name" : "Leloup", "first_name" : "Pierre", "address" : ("Grand Place",1,1000,"Bruxelles")},
             {"name" : "du Four", "first_name" : "Jacques", "address" : ("Rue Haute",21,1050,"Ixelles")},
             {"name" : "Hainaut", "first_name" : "Kim", "address" : ("Rue du Monastere",4,4000,"Liege")}]

>>> print("Avant : ",abook)
>>> Avant :  [{'name': 'Dupont', 'address': ('Rue du Centre', 10, 4500, 'Tihange'), 'first_name': 'Jean'},
{'name': 'Leloup', 'address': ('Grand Place', 1, 1000, 'Bruxelles'), 'first_name': 'Pierre'},
{'name': 'du Four', 'address': ('Rue Haute', 21, 1050, 'Ixelles'), 'first_name': 'Jacques'},
{'name': 'Hainaut', 'address': ('Rue du Monastere', 4, 4000, 'Liege'), 'first_name': 'Kim'}]

>>> tri(abook)
>>> print("Apres : ",abook)
>>> Apres : [{'name': 'Leloup', 'address': ('Grand Place', 1, 1000, 'Bruxelles'), 'first_name': 'Pierre'},
{'name': 'du Four', 'address': ('Rue Haute', 21, 1050, 'Ixelles'), 'first_name': 'Jacques'},
{'name': 'Hainaut', 'address': ('Rue du Monastere', 4, 4000, 'Liege'), 'first_name': 'Kim'},
{'name': 'Dupont', 'address': ('Rue du Centre', 10, 4500, 'Tihange'), 'first_name': 'Jean'}]
```

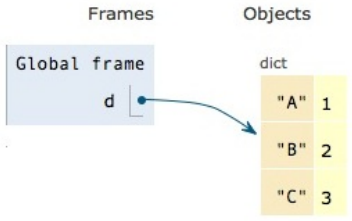
INFO-H-100 - Programmation
Interrogation de janvier
Corrections

Solution de la question 1 - Théorie (7 points)

Sachant que les diagrammes ci-dessous représentent les listes de façon raccourcie

1.
```
d= {"A":1,"B":2,"C":3}
print(dict([[d[i],i] for i in d]))
```

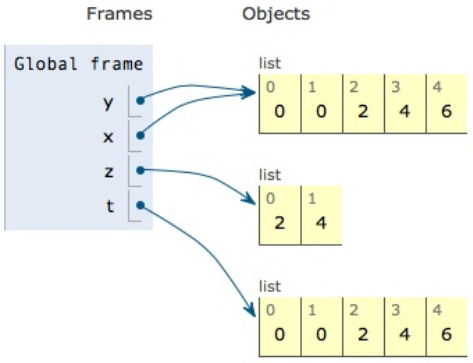
Imprime : `{1: 'A', 2: 'B', 3: 'C'}`



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

Imprime :

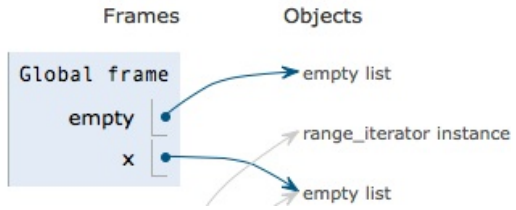
```
[0, 0, 2, 4, 6]
[0, 0, 2, 4, 6]
[2, 4]
[0, 0, 2, 4, 6]
```



3.
```
empty = []
x=[i for i in range(len(empty))]
if x is empty :
   print("ok")
print(x)
print(empty)
```

Imprime :

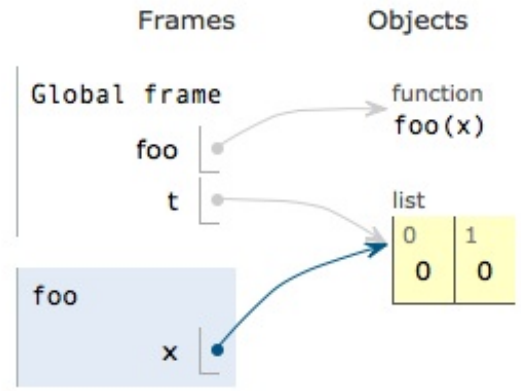
```
[]
[]
```

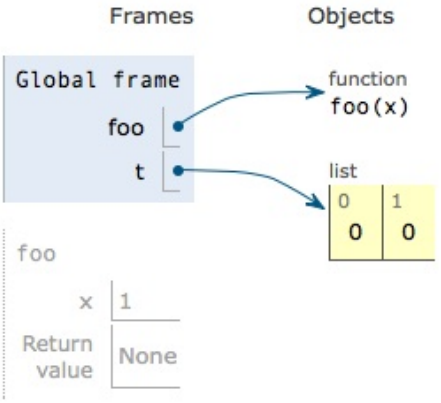


4.
```
def foo(x):
    x=1
t=[0,0]
print(foo(t))
print(t)
```

Imprime :

```
None
[0,0]
```

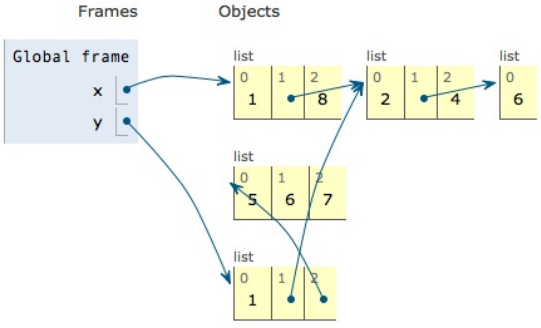




5.
```
x=[1,[2,3,4],[5,6,7]]
y=x[:]
x[2]=8
y[1][1]=[6]
print(x)
print(y)
```

Imprime :

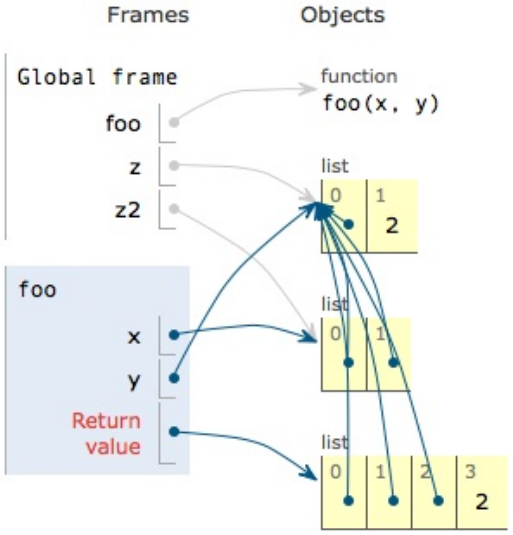
```
[1, [2, [6], 4], 8]
[1, [2, [6], 4], [5, 6, 7]]
```



6.
```
def foo(x,y):
    x[0]= y[1]
    x,y=y,x
    return x+y

z=[1,2]
z2=[z,z]
t=foo(z,z2)
print(z)
print(z2)
print(t)
```

Imprime :

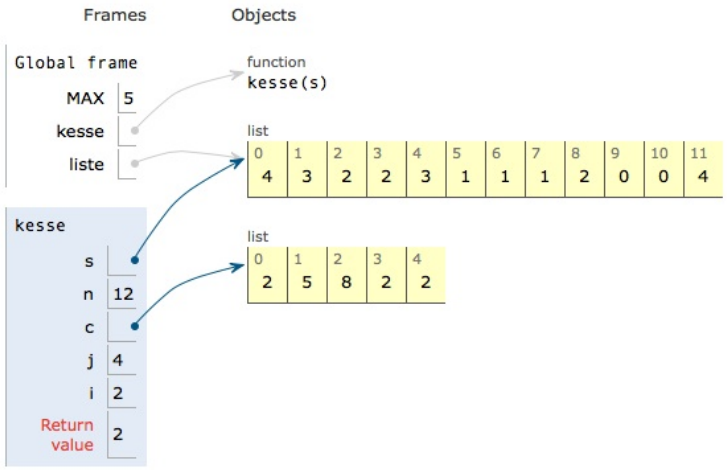
```
[[...], 2]
[[[...], 2], [[...], 2]]
[[[...], 2], [[...], 2], [[...], 2], 2]
```



7.
```
MAX = 5

def kesse(s):
    n = len(s)
    c = [0]*MAX
    for j in s:
        c[j] +=1
    i=0
    while c[i] < n//2:
      c[i+1] += c[i]
      i = i+1
    return i

liste= [4,3,2,2,3,1,1,1,2,0,0,4]

print(kesse(liste))
```

Imprime : 2

Cela correspond à la médiane le l'ensemble des valeurs dans `liste` (il y a moins de la moitié des nombres dans la liste de valeurs inférieures à la médiane m et de même pour le nombre de valeurs supérieures à m)



Solution de la question 2

```
EPS = 0.00000001
NMAX = 10000000
```

```
def arcsin(x):
    """  Descr: Calcule une approximation de l'arc-sinus
                pour le reel donne en parametre.
          Input: (float) x
         Output: (float) l'arc sinus de x
    """
    term = x
    result = term
    nb_terms = 1
    x2 = x**2
    den = 1
    factor1 = 1
    factorX = x
    while abs(term) >= EPS and nb_terms < NMAX:
        factor1 *= den/(den + 1)
        den += 2
        factorX *= x2
        term = factor1 * factorX / den
        result += term
        nb_terms += 1
    return result


print(arcsin(1)) # Affiche          1.5689429743270682
```

Solution de la question 3

```
# Triangle Trinomial (coefficients de (1 + x + x^2)^n)

def extend_line(ls):
    """  Descr: Renvoie une copie de ls étendue en ajoutant deux zeros
                avant et deux zeros apres.

          Input: (list) liste d'elements
         Output: (list) liste etendue
    """
    return [0, 0] + ls + [0, 0]

def next_line(current_line):
    """  Descr: Calcule la prochaine ligne du triangle a
                partir de la precedente.

          Input: (list) une ligne du triangle trinomial
         Output: (list) la ligne suivante
    """
    lc = extend_line(current_line)
    res = []
    for i in range(1, len(lc) - 1):
        res.append(lc[i - 1] +lc [i] + lc[i + 1])
    return res

def trinomial(n):
    """  Descr: Calcule la n-ieme ligne du triangle trinomial.

          Input: (int) numéro de ligne (commence en 0)
         Output: (list) n-ieme ligne du triangle trinomial
    """
    ls = [1]
    for i in range(n):
        ls = next_line(ls)
    return ls

def threePower(n):
    """  Descr: Calcule la n-ieme puissance de 3.

          Input: (int) exposant n
         Output: (int) n-ieme puissance de 3
    """
    return sum(trinomial(n))
```

Solution de la question 4

```
#########################################################
#  Fonctions communes au tri
#########################################################
```

```
def zip_code(user):
    """  Descr: Renvoie le code postal associe a
                l'utilisateur
         Input: (dict) utilisateur
        Output: (int) code postal
    """
    return user["address"][2]

def is_before(user1,user2):
    """  Descr: Compare l'ordre de deux utilisateurs
                sur base de leur code postal.
         Input: (dict,dict) deux utilisateurs
        Output: (bool) True si le premier utilisateur et
                       a ranger avant le deuxieme; False
                       sinon
    """
    return zip_code(user1) < zip_code(user2)

#########################################################
#  Tri par selection
#########################################################

def selection_sort(ls):
    """  Descr: Procedure du tri par selection.
         Input: (list) liste d'elements a trier
        Output: (vide) la liste en parametre est
                       transformee
    """
    for i in range(len(ls) - 1):
        pos = pos_selected(ls, i)
        swap(ls, i, pos)

def swap(ls, i1, i2):
    """  Descr: Echange dans la liste 'ls', les
                valeurs des indices i1 et i2
         Input: (list) liste dont deux elements sont
                       a echanger
                (int,int) indices des elements a echanger
        Output: (vide) la liste en parametre est
                       transformee
    """
    ls[i1], ls[i2] = ls[i2], ls[i1]

def pos_selected(ls,i):
    """  Descr: Renvoie la position du prochain
                element a ajouter a la liste
         Input: (list) liste partiellement triee
                (int) position a partir de ou chercher
        Output: (vide) la liste en parametre est
                       transformee
    """
    sel_pos = i
    for j in range(i+1,len(ls)):
        if is_before(ls[j],ls[sel_pos]):
            sel_pos = j
    return sel_pos

#########################################################
#  Tri par insertion
#########################################################

def insertion_sort(ls):
    """  Descr: Procedure du tri par insertion
         Input: (list) liste a trier
        Output: (vide) la liste en parametre est
                       transformee
    """
    for i in range(1, len(ls)):
        j = pos_insert(ls[i], ls, i)
        move(ls, i, j)


def move(ls,i,j):
    """  Descr: Deplace l'element en position i1 d'une
                liste ls vers la position i2
```

```
          Input: (list) liste traitee
                 (int,int) deux indices i1 et i2
          Output: (vide) la liste en parametre est
                         transformee
    """
    val = ls[i]
    del ls[i]
    ls.insert(j,val)

def pos_insert(val, ls, n):
    """  Descr: Renvoie la position d'insertion de la
                valeur 'val' parmi les n premiers
                elements de la liste
          Input: (int) valeur cherchee
                 (list) liste dans laquelle on cherche
                 (int) position a partir de laquelle on
                       cherche
          Output: (int) position dans la liste
    """
    j = 0
    while j < n and is_before(ls[j],val):
        j += 1
    return j

########################################################
#  Programme principal
########################################################

if __name__ == "__main__":

    abook = [{"name" : "Dupont", "first_name" : "Jean", "address" : ("Rue du Centre",10,4500,"Tihange")},
             {"name" : "Leloup", "first_name" : "Pierre", "address" : ("Grand Place",1,1000,"Bruxelles")},
             {"name" : "du Four", "first_name" : "Jacques", "address" : ("Rue Haute",21,1050,"Ixelles")},
             {"name" : "Hainaut", "first_name" : "Kim", "address" : ("Rue du Monastere",4,4000,"Liege")}]

    print("Avant : ",abook)
    selection_sort(abook)
    print("Apres : ",abook)

    abook = [{"name" : "Dupont", "first_name" : "Jean", "address" : ("Rue du Centre",10,4500,"Tihange")},
             {"name" : "Leloup", "first_name" : "Pierre", "address" : ("Grand Place",1,1000,"Bruxelles")},
             {"name" : "du Four", "first_name" : "Jacques", "address" : ("Rue Haute",21,1050,"Ixelles")},
             {"name" : "Hainaut", "first_name" : "Kim", "address" : ("Rue du Monastere",4,4000,"Liege")}]

    print("Avant : ",abook)
    insertion_sort(abook)
    print("Apres : ",abook)
```