UNIVERSITÉ LIBRE DE BRUXELLES juin 2013
École polytechnique de Bruxelles

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

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 2 heures 45 minutes 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 - Paradigmes de Python (1 points)

Chaque langage de programmation reflète un ou plusieurs paradigmes de programmation. Python permet de programmer en utilisant plusieurs de ces paradigmes : lesquels et pour chacun illustrez vos propos avec un petit morceau de code Python.

Question 2 - Exceptions (1 points)

Que fait le code suivant sachant que l'exception "KeyboardInterrupt" est déclenchée par la frappe d'une touche d'arrêt d'exécution au clavier.

```
def foo(x):
    for i in range(10000000):
        x=x+i
    return x

z=0
flag=False
while not flag:
  try:
    res=foo(z)
    flag=True
  except KeyboardInterrupt:
    z=z+1
print(res)
```

Question 3 - Runtime et complexité (3 points)

- Donnez la complexité de la fonction multiply (en justifiant bien votre réponse)

```
def add(p1,p2):
    """Renvoie la somme de 2 polynomes"""
    if len(p1) < len(p2):
        p1,p2 = p2,p1
    new = p1[:]
    for i in range(len(p2)):
        new[i] += p2[i]
    return new

def mult_one(p,c,i):
    """Renvoie le produit du polynome p avec le terme c*x^i"""
    new = [0]*i #termes 0 jusque i-1 valent 0
    for pi in p:
        new.append(pi*c)
    return new

def multiply(p1,p2):
    """ Renvoie le produit de 2 polynomes"""
    if len(p1) > len(p2):
        p1,p2 = p2,p1
    new = []
    for i in range(len(p1)):
```

```
        new = add(new,mult_one(p2,p1[i],i))
    return new
```

– Avec le code supplémentaire suivant expliquez l'état du programme au moment de chaque "return" en décrivant avec des diagrammes d'état la gestion du stack et du heap runtime.

```
p= [1.0, 2.0]
q= [1.5, 0.0, 1.0]
print(multiply(p,q))
```

Question 4 - Récursivité (5 points)

Ecrivez une fonction qui reçoit en argument un arbre d'entiers et qui renvoie comme valeur un entier qui est la somme de tous les entiers présents dans l'arbre. Un arbre d'entiers est une liste dont chaque élément est soit un arbre d'entiers, soit un entier.

Rappel : Pour tester le type de la valeur associée à une variable `a`, vous pouvez utiliser l'opérateur `type(a)`. Par exemple, `type(a) == int` renverra true si la valeur associée à a est un entier. Les types vus au cours sont `int`, `str`, `float`, `list`, `tuple`, `dict`, `bool` et `set`.

Exemple :

```
>>> print(tree_sum([[[[1,2],3,4],4],5]))
19
```

Question 5 - Opérations sur fichiers (5 points)

Ecrire un programme qui supprime les commentaires d'un fichier Python (.py).

Exemple :

Pour le fichier suivant :

```
def read_file(filename):
    f = open(filename,"r+")      # Ouvre un fichier en lecture
    string = f.read()            # Lit le fichier et stocke le résultat dans la variable "string"
    f.close()                    # Ferme le fichier
    return string                # Retourne le contenu du fichier sous la forme d'une string
```

L'exécution du programme donnera le fichier résultant suivant :

```
def read_file(filename):
    f = open(filename,"r+")
    string = f.read()
    f.close()
    return string
```

Question 6 - Map/Reduce/Filter (5 points)

ATTENTION : cette question doit être résolue à l'aide de `map`, `reduce` et `filter`. Vous n'avez pas le droit d'utiliser de boucle (`for`, `while`).

Ecrivez une fonction young_employees qui reçoit une liste d'employés et une année et qui renvoie la liste des noms des employés qui sont plus jeunes que leur département. Un employé est représenté sous la forme d'un dictionnaire qui a au moins les clefs 'name', 'age' et 'department'. La valeur associée à la clef "department" est un tuple de trois éléments, dont le premier élément est le nom du département (str), le second élément est un identifiant unique du département (int), et le troisième élément est l'année de fondation du département (int).

Exemple :

```
>>> facsa = ("Faculté des Sciences Appliquées", "facsa", 1970)
>>> solvay = ("Solvay Business School", "solvay", 1990)
>>> employees = [
    {"name":"Pierre Dupont","age":20,"department":facsa},
    {"name":"Michael Dujardin","age":52,"department":facsa},
    {"name":"Brunhilde Atso","age":21,"department":solvay},
    {"name":"Jean D'Avignon","age":30,"department":solvay}]
>>> print(list(young_employees(employees,2013)))
['Pierre Dupont', 'Brunhilde Atso']
```

INFO-H-100 - Programmation
Interrogation de Juin
Corrections

Solution de la question 1

```
def echange(ls, i1, i2):                                    # O(1)
    ls[i1], ls[i2] = ls[i2], ls[i1]                         # O(1)

def triPolys(ls, x):                                        # O(len(ls) * len(ls) * max(O(n),O(m))
    def posMinFrom(ls, i):                                  # O(len(ls) * max(O(n),O(m))
        res = i                                             # O(1)
        while i < len(ls):                                  # O(len(ls) * max(O(n),O(m))
            if p_eval(ls[i], x) < p_eval(ls[res], x):       # O(max(n,m)) avec n=len(poly) ou m=len(poly)
                res = i                                     # O(1)
            i += 1                                          # O(1)
        return res                                          # O(1)

    for i in range(len(ls) - 1):                            # O(len(ls) * len(ls) * max(O(n),O(m))
        pos = posMinFrom(ls, i)                             # O(len(ls) * max(O(n),O(m))
        echange(ls, i, pos)                                 # O(1)

def p_eval(plist,x):                                        # O(n) avec n=len(poly)
    val = 0                                                 # O(1)
    facteur = 1                                             # O(1)
    for coeff in plist:                                     # O(n)
        val += coeff * facteur                              # O(1)
        facteur *= x                                        # O(1)
    return val                                              # O(1)
```

Solution de la question 2

Solution de la question 3

Solution de la question 4

Solution utilisant la fonction `sum`.

```
def tree_sum(tree):
    if (type(tree) == list):
        return sum(map(tree_sum, tree))
    else:
        return tree
```

```
def tree_sum(tree):
    if(type(tree) == list):
        return reduce(lambda a, b: a + b, map(tree_sum, tree))
    else:
        return tree
```

```
def tree_sum(tree):
    res = 0
    for el in tree:
        if (type(el) == list):
            res = res + tree_sum(el)
        else:
            res = res + el
    return res
```

Solution de la question 5

```
def write_to_file(filename,string):
    f = open(filename,"w+")
    f.write(string)
    f.close()

def remove_comments(nomfich):
    f = open(nomfich,"r+")
    string = ""
    for ligne in f.readlines():
        index = ligne.find("#")
        substring = ligne[:index]
        string += substring+'\n'
```

```
    f.close()
    write_to_file(nomfich,string)

remove_comments("fichier.py")
```

Solution de la question 6

```
print([f['name'] for f in employees if f['department'][2] < year-f['age']])
```

```
def young_employees(emps, year):
    return map(lambda e: e["name"], filter(lambda e: (year - e['department'][2]) > e['age'], emps))
```