Programmation Impérative en Python

Examen de Rattrapage L1-Info

UNS, Juin 2016

*LE FORMULAIRE EN DERNIERE PAGE EST LE SEUL DOCUMENT AUTORISE. Tâchez de soigner l'**indentation**, la limpidité et l'exactitude du code. Répondez aux questions sur la copie d'examen qui vous est fournie, ne joignez aucune autre feuille ni intercalaire. N'hésitez pas à utiliser le résultat d'une question précédente même si vous ne l'avez pas faite. Et ne perdez pas de temps...*

A1 **Algorithmes sur les NOMBRES**. Programmez une fonction `est_premier(n)` prenant un entier `n` et retournant `True` si `n` est premier et `False` sinon. On rappelle que `n` est premier si $n \geq 2$ et si les seuls diviseurs de `n` sont `1` et `n`. Exemples : `est_premier(7)` $\rightsquigarrow$ `True`, `est_premier(9)` $\rightsquigarrow$ `False`.

A2 Faites calculer au toplevel, en une ligne, la liste des nombres premiers jusqu'à `30` :

```
>>> .........
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

A3 Montrez, sans définir aucune fonction, quelques lignes de code Python permettant d'afficher le plus petit nombre premier contenant quatre fois le chiffre 7.

B1 **Algorithmes sur LISTES, DICTIONNAIRES et ENSEMBLES**. Programmez une fonction `subst(x,y,L)` retournant une nouvelle liste obtenue en remplaçant chaque apparition de `x` par `y` dans la liste `L`. Aucune mutation sur `L`. Exemple :

```
>>> LT = [1,2,3,4,1,2,3,4]
>>> subst(2,3,LT)
[1, 3, 3, 4, 1, 3, 3, 4]
```

```
>>> LT
[1, 2, 3, 4, 1, 2, 3, 4]
```

B2 Même question que précédemment mais on fait *muter* la liste `L`! Aucun résultat pour cette nouvelle fonction `msubst(x,y,L)`.

```
>>> LT = [1, 2, 3, 4, 1, 2, 3, 4]
>>> msubst(2,3,LT)
>>> LT
[1, 3, 3, 4, 1, 3, 3, 4]
```

B3 Complétez la troisième ligne ci-dessous.

```
>>> LM = [1,2,3,4,1,2,3,4]
>>> msubst(2,3,msubst(4,2,LM))
........
```

B4 Programmez une fonction `ins_avant(x,y,L)` retournant une nouvelle liste obtenue en insérant l'élément `x` avant la *première* apparition de l'élément `y` dans `L`. Aucune mutation sur `L`. Exemples :

```
>>> ins_avant(10,6,[1,4,6,9,6,12])
[1, 4, 10, 6, 9, 6, 12]
```

```
>>> ins_avant(10,6,[8,4,7,9,12])
[8, 4, 7, 9, 12]
```

B5 Programmez une fonction `compter(L)` prenant une liste de nombres `L` et retournant un dictionnaire `{x:k,...}` dont les clés `x,...` seront les éléments distincts de `L` et la valeur associée à une clé sera le nombre de fois que la clé apparaît dans `L`. Exemple :

```
>>> compter([2,4,2,5,3,4,2,5,5,6])    # 2 apparaît 3 fois, 3 apparaît 1 fois, etc
{2: 3, 3: 1, 4: 2, 5: 3, 6: 1}
```

B6 En utilisant la fonction précédente, programmez la fonction `les_plus_frequents(L)` retournant l'*ensemble* des éléments de `L` apparaissant le plus grand nombre de fois. Exemple :

```
>>> les_plus_frequents([2,4,2,5,3,4,2,5,5,6])
{2, 5}
```

C **Algorithmes sur les CHAÎNES**. Programmez une fonction `majmin(s)` prenant une chaîne `s` représentant une phrase en français et retournant une nouvelle chaîne obtenue en mettant en majuscules les lettres minuscules de `s` et inversement.

```
>>> majmin('Il fait très BEAU !!!') == 'iL FAIT TRÈS beau !!!'
True
```

D **Algorithmes sur les FICHIERS**. Programmez une fonction `defs(f,n)` prenant une chaîne de caractères `f` représentant un nom de fichier sur le disque avec une extension. La fonction commencera par vérifier si `f` se termine bien par l'extension `.py` et déclenchera une `ValueError('Mauvais fichier')` si ce n'est pas le cas. S'il s'agit bien d'un fichier Python, elle va afficher à l'écran les `n` premières lignes qui débutent une définition de fonction (donc uniquement la ligne du `def`). L'exemple ci-dessous utilise le fichier `exam2.py` des solutions de cet examen.

```
>>> defs('exam2.py',3)
def est_premier(n) :
def subst(x,y,L) :                  # sans mutation de L
def msubst(x,y,L) :                 # avec mutation de L
```

E1 **Algorithmes sur les POLYNÔMES CREUX**. En utilisant le module `polycreux` (cf Annexe), programmez *par récurrence* sur $\texttt{n} \geq 1$ la fonction `q(n)` retournant le polynôme creux noté mathématiquement $(x-1)(x-2)\dots(x-n)$. Exemple :

```
>>> (q(2),q(3))    # (x-1)(x-2) et (x-1)(x-2)(x-3)
([[1, 2], [-3, 1], [2, 0]], [[1, 3], [-6, 2], [11, 1], [-6, 0]])
```

E2 Comment feriez-vous afficher, en quelques lignes de code Python, le coefficient du terme en x^5 du polynôme `q(10)` ? Vous afficherez bien entendu `0` si ce terme n'existe pas.

ANNEXE : Vocabulaire Python de base en L1 Sciences - 2016

Types étudiés : `int, float, bool, str, list, tuple, set, dict`
Test de type : `type` Ex: `if type([2,5,3]) == list : ...`
Modules, on indique entre crochets les objets les plus importés :
`fractions[Fraction,gcd]` `math[pi,sqrt,factorial,sin,log,...]` `random[randint]`
`time[time,sleep]` `turtle[*]` `polycreux[*]` `os[...]` `tkinter[*]`
Importations :
```
from ... import ...
import ...
from ... import ... as ...
```
Affichage (les crochets indiquent des arguments optionnels) :
`print(x,y,...,[sep=' '],[end='\n'])`
Opérateurs conditionnels : `and` `or` `not`
Egalité : `==` `!=`
Intervalles d'entiers (itérateurs) : `range(n)` `range(a,b)` `range(a,b,s)`
Nombres : `+` `-` `*` `%` `//` `/` `**` `abs`
Exceptions (simples) :
```
try :
    ...
except :
    ...
```
Chaînes de caractères (`str`*) :* `+` `i*s` `len(s)` `in` `s[i]` `==` `int(s)` `str(n)`
`bin(n)` `chr(i)` `ord(c)` `s.count(s)` `s.index(s)` `s.isupper()` `s.islower()` `s.isalpha()`
`s.isdigit()` `s.upper()` `s.lower()` `s.replace(old,new)` `s.split(sep=' ')` `L.join(s)`
Listes (`list`*) :*
`+` `i*L` `len(L)` `in` `L[i]` `==` `L.index(x)` `L.count(x)` `L.append(x)`
`L.pop(i)` `sorted(L)` `L.sort()` `L.reverse()` `L.insert(i,x)` `L.extend(`L_1`)`
Tuples (`tuple`*) :*
`+` `i*t` `len(t)` `in` `t[i]` `==` `t.index(x)` `t.count(x)`
Polynômes creux [[c_n*,*e_n*],...] (module* `polycreux`*) :*
`poly0()` `is_poly0()` `coef(p)` `degre(p)` `add(p,q)` `sub(p,q)` `mul(p,q)`
`mul_ext(k,p)` `valeur(p,x)` `fpoly(p)`
Ensembles (`set`*) :*
`in` `&` `|` `-` `<` `<=` `==` `len(E)` `E.add(x)` `E.pop()` `E.remove(x)`
Dictionnaires (`dict`*) :*
`len(d)` `in` `d[k]` `d.update(d)` `d.keys()` `d.values()`
Fichiers texte :
`f.read()` `f.readline()` `f.readlines()` `f.write(s)` `f.writelines(L)` `f.close()`
`open(s,'r',encoding=...)` `open(s,'w',encoding=...)` `open(s,'a',encoding=...)`
Graphisme tortue (module `turtle`*) :* `forward(d)` `back(d)` `left(a°)` `right(a°)`
`down()` `up()` `ht()` `st()` `goto(pt)` `goto(x,y)` `pos()` `heading()`
`setheading(a°)` `pencolor(s)` `dot(diam)` `tracer(b)` `reset()`
Programmation par objets :
```
class      __init__(self,...)      __repr__(self)
           __add__(self,obj)       __len__(self)
```