Université de Caen **Année 2013/2014**
UFR de Sciences **session (2) de juin**

Licence L2,

Examen de Programmation Objet / Python

Durée: 2,0 H. Les documents ne sont pas autorisés.

Chaque candidat doit, au début de l'épreuve, porter son nom dans le coin de la copie qu'il cachera par collage après avoir été pointé. Il devra en outre porter son numéro de place sur chacune des copies, intercalaires, ou pièces annexées.

1. Écrire un générateur Python qui prend comme entrée un flot (un autre générateur, une séquence) numérique, sans longueur spécifiée. [Si vous ne savez pas ce qui est un générateur Python, tant pis, d'autres solutions ne seront pas acceptées]. La séquence engendrée montre la « tendance » des données source : si les valeurs des données augmentent, le résultat est +1, si diminuent : -1, si égaux : zéro. Il faut que le générateur construit mémorise le dernier élément lu depuis la source.
   Ainsi, si la séquence – source est : 3, 3, 6, -2, 7, 8, 4, 2, 7, 7, 2, 6, 1, 2, 2, 6, ..., le résultat engendré par le générateur doit être : 0, 1, -1, 1, 1, -1, -1, 1, 0, -1, 1, -1, 1, 0, 1, ...

2. Écrire un paquetage opérant sur les "nombres artificiels" étant des séquences de chiffres, par ex. `N(1,0,6,3)` représentant le nombre décimal 1063. Construire la classe **N** de tels nombres, ***par la surcharge des listes*** [si vous ne savez pas ce qui est la surcharge de la classe des listes, tant pis, d'autres solutions ne seront pas acceptées]. Le constructeur acceptera un nombre arbitraire de paramètres – chiffres comme ci-dessus. La représentation de ces objets doit avoir la forme un peu différente de la forme d'entrée : `N[1,0,6,3]`, avec les crochets. `N()` s'affiche comme `N[]`, et `N(9)` comme `N[9]`.

   - Si `a=N(1,0,6,3)`, une "tranche" (slice), par ex. `a[1:-1]` peut retourner une liste, ici `[0,6]`, si on surcharge les listes, mais pas leurs méthodes. Comment la convertir en objet de classe **N**?
   - Surcharger l'opérateur d'addition '+' et l'affichage.

   Une stratégie *possible* pour `(+)` est la suivante. Commencez par la procédure qui ajoute à une liste N(...) un seul chiffre. On ajoute ce chiffre au dernier élément, on vérifie si la somme dépasse 10, et on propage la retenue à gauche. On peut exploiter `divmod(n,m)` $\rightarrow$ (quotient, reste) de la division n/m (pour $m = 10$). Ensuite pour addition des listes utilisez la récursivité.

3. Les nombres complexes : $z = x + yi$, avec $i^2 = -1$ peuvent être implémentés comme des paires de réels : $C(x,y)$. Clifford a inventé un autre corps numérique, les ***nombres duals***, un peu plus simple ; un nombre dual p est une paire de réels a et b, disons, $D(a,b)$, écrit intuitivement comme $p = a + b\epsilon$ avec les règles arithmétiques intuitives, (par exemple on ajoute séparement les composantes réelle et duale). L'unité duale ϵ est irréductible, indépendante des réels (comme l'unité imaginaire dans le monde des nombres complexes), mais elle respecte l'identité : $\epsilon^2 = 0$. Donc, $(a + b\epsilon) \cdot (x + y\epsilon) = ax + (ay + bx)\epsilon$. **Écrire en Python une classe D qui représente de tels nombres, avec les méthodes standard : `__init__` et `__repr__`** raisonnables, et les quatres opérations arithmétiques : `+ - * /`, de manière la plus complète possible (et avec l'arithmétique mixte dual–réel). Pour la division, vérifier que $(a + b\epsilon) \cdot (a - b\epsilon)$ est un nombre réel (quelle est sa valeur?), et coder-la de manière similaire à la division des nombres complexes. La méthode `__init__` doit pouvoir prendre 1 ou deux arguments, dans le cas d'un seul, le coefficient de ϵ est considéré nul.

Écrivez de manière lisible, un texte difficile à déchiffrer risque d'être rejeté. Ne jamais copiez des morceaux de code inutile, tout code hors sujet peut être pénalisé. Évitez toute discussion qualitative « philosophique » inutile. Par contre, *commentez votre code*, sinon il risque d'être peu compréhensible.