Université de Caen **Année 2011/2012**
UFR de Sciences **session de décembre**

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. Tous connaissent les paires de réels, qui sont des nombres complexes : $z = x + yi$, avec $i^2 = -1$. 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, écrit comme $p = a + b\epsilon$ avec les règles arithmétiques typiques (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 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 (combien?), 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. Un bonus intéressant attend ceux qui seront capables d'implémenter la fonction exponentielle dans le domaine des duals.

2. **Le code de César**. Construire une classe de « pseudo-dictionnaires », des objets qui se comportent comme des dictionnaires indexés par des chaînes de caractères, par exemple **`d['yeux']`** donne **`'bhxa'`**. Cependant, **`d`** *n'est pas un vrai dictionnaire*, il n'accepte que des chaînes. Le pseudo-dictionnaire prend la chaîne-argument, caractère par caractère, ajoute 3 à son code, et reconstruit le caractère résultant, et les compose dans la chaîne résultante. Si le code du caractère dépasse le code de **`'z'`**, on boucle : **`'z'`** est suivi par **`'a'`**, **`'b'`**, etc. Pour simplicité, ignorons les lettres majuscules et les lettres accentuées (on considère uniquement les 26 lettres latines minuscules). Prévoir une paramétrisation globale par une variable de classe – le décalage, pas obligatoirement égal à 3, mais quelconque. Utiliser les fonctions : **`ord`** qui convertit un caractère en un entier, et **`chr`** – la conversion inverse. Vous pourrez avoir besoin de l'opérateur modulo : **`%`**. Construire conceptuellement d'abord la fonction qui effectue le transcodage, qui servira comme une *méthode d'indexation* dans la classe des objets en question. Est-ce qu'il est utile d'hériter du type des dictionnaires normaux?

3. Écrire une fonction ***générateur*** parametré par un autre générateur – source, et une valeur initiale, connue avant la consommation du flot-source :

```
def zcross(source,iniv=1):
  ...
  v = source.next()     # ou une boucle for, peut-être?...
  ...
  yield (...)
  ...
```

qui réalise un filtrage « zero-crossing » d'un flot de données (sans longueur spécifiée), en créant un nouveau flot de résultats. Quand la valeur dans le flot d'entrée passe du négatif vers positif, le générateur retourne (par **`yield`**, bien sûr) $+1$; de positif vers négatif : -1. Si les valeurs ne changent pas de signe, le filtre émet zéro.

Par exemple, si la séquence d'entrée est : **`1 -2 3 2 1 -2 -3 9 -1 1 -2 -1 3 5 -1 2 -1 ...`**, le flot de sortie sera **`-1 1 0 0 -1 0 1 -1 1 -1 0 1 0 -1 1 -1 ...`**.

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 sera *pénalisé.* Évitez toute discussion qualitative (« philosophique ») inutile. Par contre, *commentez votre code*, sinon il risque d'être peu compréhensible. Il est préférable que vous essayez de faire tous les exercices, même de manière imparfaite, que de vous concentrer sur un seul, sans toucher les autres.