Examen de TP, fin du second semestre

Ouvrez dans pyzo le fichier `TP_note_sem2.py` situé dans le répertoire `Devoirs/fleck/TP_note_sem2/`. N'oubliez pas que l'exécution doit se faire via `Ctrl-SHIFT-E` pour que tout marche bien.

Partie I

Codage de Cesar

On cherche programmer une méthode simple pour crypter et décrypter un texte. La méthode proposée est vraiment élémentaire. Ce codage a été utilisé notamment par Jules César pour certaines de ses correspondances.

Nous voulons crypter un texte `texte` composé de caractères en minuscules soit 26 lettres différentes (on supposera que le texte est écrit sans la ponctuation, sans les majuscules et sans les accents). Nous allons commencer par représenter chaque lettre par un entier. Pour cela, on vous fournit la liste appelée `Alphabet` définie comme suit :

$$\texttt{Alphabet} = [\texttt{'a'}, \texttt{'b'}, \texttt{'c'}, \texttt{'d'}, \texttt{'e'}, \ldots, \texttt{'x'}, \texttt{'y'}, \texttt{'z'}]$$

Cela permet de représenter une lettre par un entier entre 0 et 25.

1. Construire une procédure `chiffrage` prenant en entrée une lettre sous forme de chaîne de caractère et renvoyant l'entier qui représente cet entier. Par exemple `chiffrage('e')` doit renvoyer 4 puisque `'e'` est la 5e lettre de l'alphabet. Si l'entrée n'est pas une lettre minuscule non accentuée, votre fonction doit renvoyer `None`.
2. Construire une procédure `lettrage` prenant en entrée un entier de $[\![0, 25]\!]$ et renvoyant la lettre correspondante. Par exemple `lettrage`(20) doit renvoyer `'u'` puisque `'u'` est la 21e lettre de l'alphabet.
3. Construire une procédure `Mise_en_entier(texte)` qui transforme la chaîne de caractère `texte` en une liste d'entiers de $[\![0, 25]\!]$. Par exemple `Mise_en_entier('hello world!')` doit donner la liste

   ```
   [7, 4, 11, 11, 14, 22, 14, 17, 11, 3]
   ```

   On notera que l'on élimine les espaces et autres signes de ponctuation : on ne code que les lettres.

'h'	'e'	'l'	'l'	'o'	'w'	'o'	'r'	'l'	'd'
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
[7,	4,	11,	11,	14,	22,	14,	17,	11,	3,]

Le principe du codage. — C'est de décaler les lettres de l'alphabet vers la gauche de d positions. Par exemple, en décalant les lettres de 1 position, le caractère `'a'` se transforme en `'z'`, le `'b'` se transforme en `'a'`, ..., `'z'` se transforme en `'y'`. Le texte `'avecesar'` devient `'zudbdrzq'`.

4. Que donne le codage de `'maitrecorbeau'` en utilisant un décalage de 5 ? On stockera le résultat [1] dans la variable `maitrecorbeau`.
5. Écrire une fonction `codage_cesar(texte,d)` qui prend en argument une chaîne de caractère `texte` et un entier d et qui renvoie la chaîne de caractère codée avec un décalage de d vers la gauche.

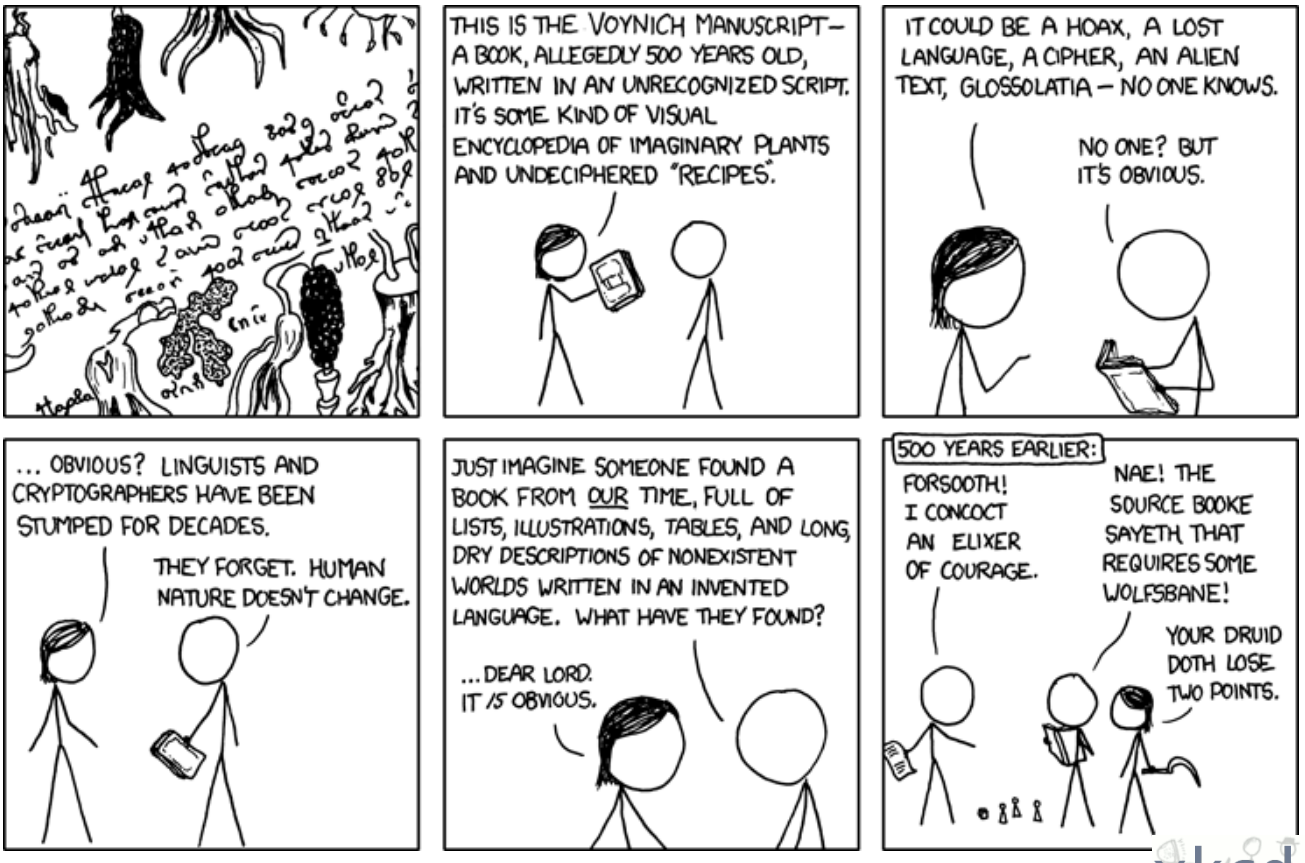
   $$\texttt{'cesar'} \xrightarrow[\texttt{codage\_cesar(\_,8)}]{} \texttt{'uwksj'}$$

6. Écrire une fonction `decodage_cesar(texte,d)` qui prend en argument une chaîne de caractère `texte` et un entier d mais qui renvoie le décalage dans l'autre sens.

1. Calcul à la main accepté.

Pour pouvoir décoder un texte, il faut connaître la valeur de décalage que l'on appelle *la clef*. Une manière de la déterminer automatiquement est d'essayer de deviner cette valeur. L'approche la plus couramment employée est de regarder la fréquence d'apparition de chaque lettre dans le texte crypté. En effet, la lettre la plus fréquente dans un texte suffisament long [2] est le lettre `'e'`.

7. Écrire une fonction `frequence(texte)` qui prend en argument un texte sous forme de chaîne de caractère et qui retourne la liste de taille 26 dont le i^{e} élément contient le nombre d'apparitions de la lettre numéro i dans `texte`.
8. Écrire une fonction `decodageauto_en_e_majeur(texte)` qui prend en argument le texte crypté sous forme de chaîne de caractère et qui retourne le texte d'origine en calculant la clef pour que la lettre `'e'` soit la plus fréquente dans le texte décrypté [3].
9. Écrire une fonction `top_secret()` qui décode le message du fichier `secret.txt`.



`Wait, is that the ORIGINAL voynich manuscript? Where did you GET that?`
`Wanna try playing a round of Druids and Dicotyledons?`

2. Hormis bien sûr dans un texte tiré du roman « La disparition » de George Perec.
3. Il y aura une unique lettre la plus fréquente dans les chaînes testées.

Partie II

Fabrication d'un simulateur à N corps

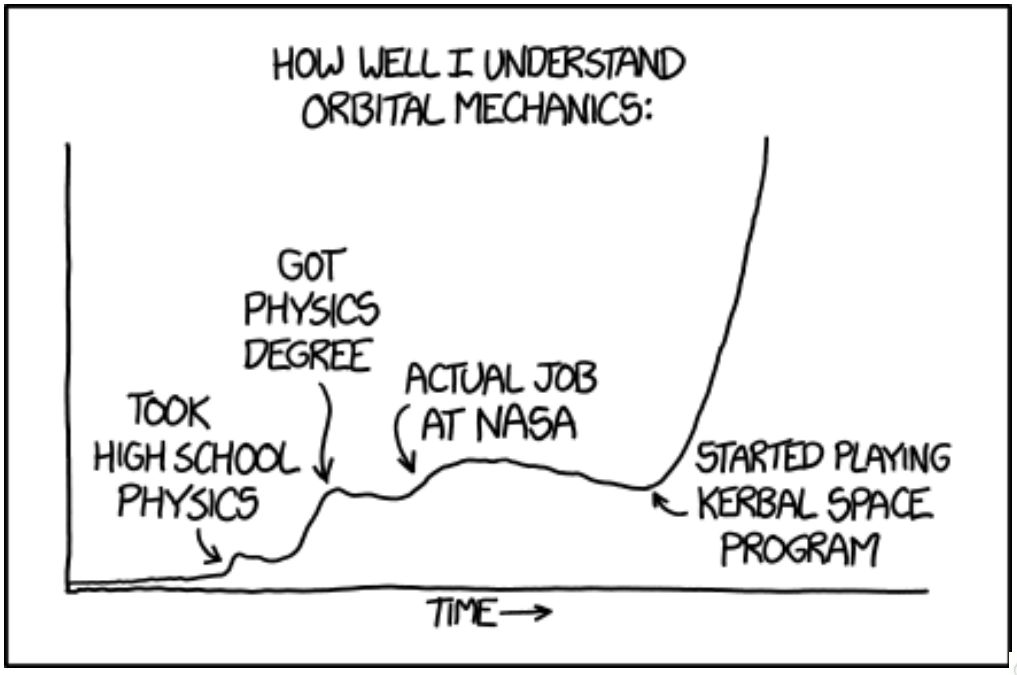
Dans cette partie, on va s'attacher à produire un simulateur complet à N corps qui va intégrer les équations du mouvement de chacun des N corps à l'aide de la méthode de Verlet.

On s'intéresse donc à la dynamique d'un système de N corps en interaction gravitationnelle. Dans la suite, les corps considérés sont assimilés à des points matériels P_j de masses m_j où $j \in [\![0\,;\, N-1]\!]$ avec $N \geq 2$ un entier positif donné. Le mouvement de ces points est étudié dans un référentiel galiléen muni d'une base cartésienne orthonormée $(\overrightarrow{e^{(0)}}, \overrightarrow{e^{(1)}}, \overrightarrow{e^{(2)}})$. L'interaction entre deux corps j et k est modélisée par la force gravitationnelle. L'action exercée par le corps k sur le corps j est décrite par la force

$$\overrightarrow{F_{k/j}} = G\,\frac{m_j m_k}{r_{jk}^3}\,\overrightarrow{P_jP_k} = G\,\frac{m_j m_k}{\left(\sum_{\ell=0}^{2}\left(x_k^{(\ell)} - x_j^{(\ell)}\right)^2\right)^{3/2}}\sum_{\ell=0}^{2}\left(x_k^{(\ell)} - x_j^{(\ell)}\right)\overrightarrow{e^{(\ell)}}$$

où $x_k^{(\ell)}$ est la ℓ^{e} coordonnée spatiale associée à la particule k. Pour simplifier, on prendra $G = 1$ dans toute la suite.

1. Écrire une fonction `force2(m1,p1,m2,p2)` qui prend en arguments deux flottants `m1` et `m2` représentant respectivement les masses des deux particules en interaction ainsi que deux listes `p1` et `p2` de trois éléments représentant les positions de chaque particule par rapport à l'origine du référentiel. La fonction doit renvoyer une liste de trois flottants représentative de la force que la particule 2 exerce sur la particule 1 dans la base cartésienne (ne pas oublier qu'on prend $G = 1$).
2. Écrire une fonction `forceN(j,m,pos)` qui prend en paramètres l'indice `j` d'un corps, la liste `m` des masses des N corps du système étudié ainsi que la liste `pos` de leurs positions. La fonction doit renvoyer $\overrightarrow{F_j}$, la force exercée par tous les autres corps sur le corps `j` sous la forme d'une liste de ses trois composantes cartésiennes.



xkcd.com

```
To be fair, my job at NASA was working on robots
and didn't actually involve any orbital mechanics.
The small positive slope over that period is
because it turns out that if you hang around at NASA,
you get in a lot of conversations about space.
```

Le schéma de Verlet proposé par le physicien français Loup Verlet en 1967 est un schéma d'intégration d'une équation de la forme $\ddot{y} = f(y)$ qui donne un résultat bien meilleur que la méthode d'Euler. Posons $J_n = [\![0\,;\, n]\!]$ avec n le nombre de pas d'intégration $\mathrm{d}t$ que l'on veut effectuer après avoir pris connaissance des conditions initiales. En notant $z = \dot{y}$ la dérivée temporelle de la fonction y recherchée, le schéma d'intégration va permettre, comme dans la méthode d'Euler, de définir deux suites $(y_i)_{i\in J_n}$ et $(z_i)_{i\in J_n}$ définies par les relations de récurrence suivantes

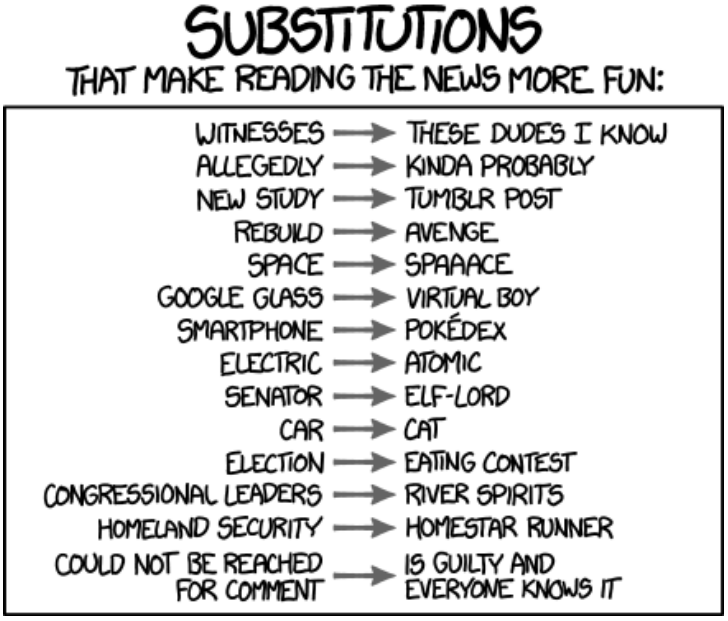
$$y_{i+1} = y_i + z_i\,\mathrm{d}t + f_i\,\frac{\mathrm{d}t^2}{2} \qquad \text{et} \qquad z_{i+1} = z_i + \left(\frac{f_i + f_{i+1}}{2}\right)\mathrm{d}t$$

en ayant pris le soin de poser $f_i = f(y_i)$ et $f_{i+1} = f(y_{i+1})$. En se souvenant que l'application de la relation fondamentale de la dynamique à la particule j s'écrit, en utilisant les notation introduites précédemment

$$\frac{\mathrm{d}^2\overrightarrow{OP_j}}{\mathrm{d}t^2} = \frac{\overrightarrow{F_j}}{m_j}$$

on va pouvoir utiliser le schéma de Verlet pour intégrer les équations du mouvement.

3. Écrire une fonction `positions_suivantes(m,pos,vit,dt)` qui prend en paramètres la liste `m` des masses des N corps du système étudié, la liste de leurs positions à l'instant t_i, la liste des vitesses au même instant et le pas d'intégration `dt`. La fonction doit renvoyer la liste des positions des N corps à l'instant $t_{i+1} = t_i + \mathrm{d}t$ calculées à l'aide du schéma de Verlet [4].
4. Écrire une fonction `etats_suivants(m,pos,vit,dt)` qui prend les mêmes paramètres que la fonction précédente mais doit renvoyer la liste des positions et la liste des vitesses des N corps à l'instant t_{i+1} en utilisant le schéma de Verlet. Attention, certains tests vont contrôler la vitesse d'exécution pour s'assurer que l'algorithme global est au plus quadratique en N.
5. À l'aide des listes `masses`, `positions` et `vitesses` prédéfinies dans le fichier à remplir, exécuter votre algorithme de $t = 0$ jusqu'à $t = 10$ par pas de temps `dt=1e-4` pour produire un graphique permettant de visualiser l'évolution des différentes particules dans le plan (Oxy) sur l'intervalle de temps considéré. Ce graphique sera enregistré dans le répertoire courant sous le nom `integration_verlet.png`. Votre nom devra figurer dans le titre du graphique.



`INSIDE ELON MUSK'S NEW ATOMIC CAT`

4. N'oubliez pas de diviser la force par la masse correspondante !