Python-TP4-2016

18 Novembre 2016

TP 4 – Informatique CM3 – 2016-2017

1 Python @ Polytech'Lille

1.0.1 Modalités pour accomplir le TP et compte rendu

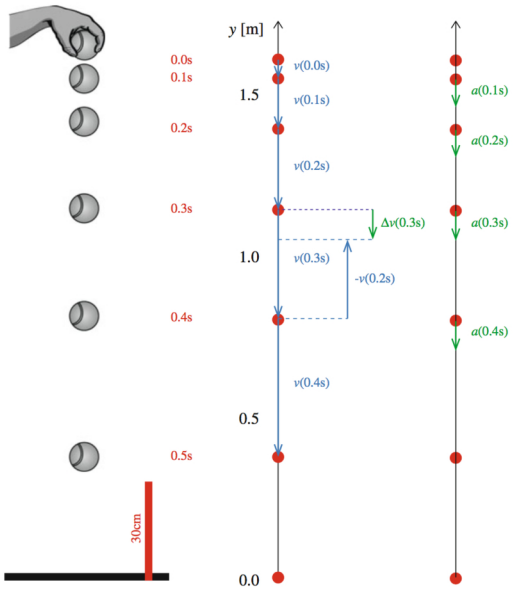
Vous aurez à écrire 2 scripts (script1.py , script2.py). Les scripts doivent être accompagnés par un document descriptif unique (README.txt). Dans ce fichier, vous devrez décrire le mode de fonctionnement des scripts et, si besoin, mettre vos commentaires et réponses aux questions. Merci d'y écrire aussi votre nom et votre prenom. Tous les fichiers doivent être mis dans un dossier appelé TP4-nom et ensuite être compressés dans un fichier archive TP4-nom.tgz .

Enfin vous allez envoyer ce fichier par email à l'enseignant: soit Enrico (enrico.calzavarini@polytech-lille.fr) soit Stefano (stefano.berti@polytech-lille.fr).

Vous avez 2 heures à disposition pour ce TP. Le compte rendu est à rendre avant la fin de la séance.

1.1 Script 1 : Mouvement de chute libre d'une balle de tennis

Le but de ce script est d'étudier la cinématique d'une balle de tennis qui tombe d'une hauteur initiale y_0, dans le champ de la pesanteur, et qui rebondit sur le sol ($y = 0$), à partir de données expérimentales, et de comparer ensuite les mesures de la position et de la vitesse de la balle avec les prédictions d'un modèle mathématique simple.



La figure précédente illustre le mouvement de la balle. Elle présente les résultats de mesures (de basse résolution temporelle), effectuées avec un appareil photographique digitale, de la position (colonne de gauche), de la vitesse (colonne centrale) et de l'accélération (colonne de droite) de la balle à des intervalles de temps de 0.1s.

Ces mesures fournissent une première indication du fait que l'accélération de la balle est approximativement constante pendant le mouvement, mais elles sont assez grossières. Au but d'étudier plus en détail la dynamique et de comparer les mesures avec un modèle mathématique, des mesures de plus haute résolution temporelle sont nécessaires. Pour cette raison, le mouvement de la balle a été aussi enregistré avec un capteur monté sur la balle même. Le capteur fournit la position $y(t)$ de la balle avec une résolution temporelle Δt bien plus importante.

Nous supposerons ici de ne pas connaître la résolution Δt à priori; elle devra donc etre déterminée à partir des données.

Les données sont contenues dans un fichier que vous recevrez de votre enseignant, où elles sont organisées en deux colonnes: la premiére correspond aux intants de temps t_i et la deuxième aux positions $y_i \equiv y(t_i)$.

Question 1. Lire les données du fichier; calculer Δt et tracer le graphique de la position en fonction du temps. Préciser les noms des variables, ainsi que leurs unités de mesure, dans les labels des axes.

Remarques informatiques. Une manière pratique pour lire un ensemble de données organisées dans deux colonnes dans un fichier appelé *nomdufichier.dat* est la suivante, basée sur la commande *loadtxt* de numpy. Après lecture, les variables (de type *array*) a et b contiendront, respectivement, les données de la première et de la deuxième colonne.

```
In [2]: import numpy as np # au debut du script

In [3]: a,b=np.loadtxt('nomdufichier.dat',usecols=[0,1],unpack=True) # lecture
```

Question 2. Tracer les graphiques de la vitesse $v(t_i) = \dfrac{y(t_i + \Delta t) - y(t_i)}{\Delta t}$ et de l'accélération $a(t_i) = \dfrac{v(t_i + \Delta t) - v(t_i)}{\Delta t}$ en fonction du temps. Préciser les noms des variables, ainsi que leurs unités de mesure, dans les labels des axes.

Question 3. Que peut-on dire du mouvement à partir de ces graphiques? En particulier, peut-on conclure que l'accélération est constante, comme il semblerait à partir de son allure pour $t \leq 0.5$s? Pour répondre à cette question il est nécessaire d'analyser plus en détail le comportement de l'accélération dans cet intervalle de temps.

Nous demandons, donc, de tracer le graphique de $a(t)$ pour $t \leq 0.5$s et de commenter la dépendance temporelle de l'accélération.

Remarques informatiques. Pour faire ce graphique nous devons connaître la valeur de l'indice i la plus grande pour laquelle $t \leq 0.5$s. Les commandes *where* et *amax* de numpy pourraient se révéler utiles à ce fin. Les exemples ci-dessous illustrent leur utilisation.

```
In [4]: a=np.array([-5,-3,0,1,2,3,4,5,7]) # definition array
        b=np.where(a>2) # elements de a plus grands que 2
        print(b)

(array([5, 6, 7, 8]),)


In [5]: np.amax(b) # indice max

Out[5]: 8
```

Question 4. Nous souhaitons maintenant comparer les données expérimentales avec le modèle mathématique le plus simple qu'on puisse imaginer pour cette situation, celui du mouvement uniformément accéléré de la balle:

$$y(t) = y_0 - \frac{1}{2}gt^2,$$

où $g = 9.8\text{m/s}^2$ est l'accélération de la pesanteur et $y_0 = 1.6\text{m}$ est la position initiale de la balle.

Tracer dans un même graphique les courbes correspondantes aux donnees expérimentales et à la prédiction théorique du modèle, pour la position et pour la vitesse (dans un autre graphique), pour les indices i tels que $t \leq 0.5\text{s}$. Utiliser une légende dans le graphique pour préciser quelle courbe correspond aux données et quelle autre correspond au modèle.

Question 5. En se basant sur les analyses effectuées, formuler une conclusion à propos de la possibilité de décrire les données avec le modèle.

Quelle est la grandeur physique qui est moins bien représentée dans le modèle?

Comment pourrait-on améliorer le modèle du point de vue de la mécanique?

Question 6. Nous voulons, finalement, faire un zoom de la trajectoire de la balle autour des rebonds.

Déterminer les indices i qui correspondent aux temps pour lesquels la balle rebondit, en imposant une condition sur la vitesse.

Tracer ensuite un graphique de la tracjectoire pour chaque impact avec le sol, en utilisant une fenêtre temporelle correspondant à 10 points avant l'impact et 10 points après. Utiliser le style point-ligne pour le graphique, afin de mieux mettre en évidence les différentes positions de la balle au cours du temps.

Question bonus. Calculer la valeur moyenne de l'accélération dans la phase de descente de la balle avant le premier rebond, entre le premier et le deuxième, entre le deuxième et le troisième. Pour ce but, définir et utiliser une *fonction* qui calcule la valeur moyenne pour chaque phase, en utilisant des conditions sur la vitesse et l'accélération pour identifier les indices i correspondant à la phase de descente.

On constate que le module de l'accélération augmente, en se rapprochant de g. Sauriez-vous expliquer pourquoi?

Remarques informatiques. Les commandes suivantes pourraient etre utiles pour implémenter la *fonction*.

```
In [6]: a=np.array([-5,-3,0,1,2,3,4,5,7]) # definition array
        b=np.where(a>2) # elements de a plus grands que 2
        print(b)
        np.amin(b) # indice min

(array([5, 6, 7, 8]),)


Out[6]: 5

In [7]: np.mean(b) # moyenne des elements de b

Out[7]: 6.5
```

2 Script 2 : Crypter et déchiffrer un message

Il s'agit de développer un programme permettant de crypter un mot en remplaçant chaque lettre de l'alphabet par les coordonnées (numéro de la ligne; numero de la colonne) de sa position dans une matrice carrée.

–	0	1	2	3	4
0	A	F	K	P	U
1	B	G	L	Q	V *et* W
2	C	H	M	R	X
3	D	I	N	S	Y
4	E	J	O	T	Z

Exemple : Le mot “LILLE” sera crypté par la liste de nombres à deux chiffres suivante : “12 31 12 12 40”.

Question 1. Écrire la fonction de cryptage et l’utiliser pour crypter les mots suivants : “MECANIQUE” , “POLYTECH”, “WALLONIE” et aussi votre nom de famille.

Votre script devra demander le mot à crypter à l’utilisateur et ensuite afficher le mot crypté sur l’écran.

Question 2. Écrire la fonction de decryptage.

Utilisez par la suite cette fonction pour vérifier que votre nom de famille est décrypté correctement.

Enfin décrypter les mots suivants:

“31 32 01 42 23 22 00 43 31 13 04 40” , “03 34 43 21 42 32” , “03 23 42 11 23 00 22 22 00 43 31 42 32”.

Remarques algorithmiques et informatiques L’algorithme pour le cryptage du texte sera le suivant:

1) On définit une tableau (par exemple en utilisant “array” de la bibliothèque numpy) de taille 5x5 comportant les lettres de l’alphabet dans la disposition donnée.
2) Pour chaque lettre du mot à crypter, on determine ses coordonnées dans le tableau. Conseil : utilisez la commande “while” pour la boucle sur les lettres du mot à crypter.
3) On affiche la liste de toutes les coordonnées séparées par des espaces. Cette sequence correspond au mot crypté qui sera affiché sur l’écran.

Remarque: Les lettres “V” et “W” sont toutes les deux cryptées par “14”. La lettre “W” des mots a crypter peut donc etre remplacé par la lettre “V” avant le cryptage. Vous pouvez utiliser la commande “list” pour separer les lettres dans une chaine de caracteres.

```
In [8]: nom = "ALBERT"
        print(nom)
```

```
ALBERT
```

```
In [9]: lettres_dans_nom = list(nom)
        print(lettres_dans_nom)
```

```
['A', 'L', 'B', 'E', 'R', 'T']
```

Pour créer une liste vide :

```
In [10]: une_liste = []
```

Pour ajouter des éléments à la liste :

```
In [11]: une_liste.append("A")
         une_liste.append("B")
         une_liste.append("C")
         print(une_liste)
```

```
['A', 'B', 'C']
```