Python-TP2-2014

October 4, 2014

TP 2 – Informatique CM3 – Septembre 2014

1 Python @ Polytech'Lille

Le texte de cette sessions de travaux pratiques est également disponible ici http://nbviewer.ipython.org/github/ecalzavarini/python-at-polytech-lille/blob/master/Python-TP2-2014.ipynb

Gestion des accents en Python Dans la première séance de TP, vous avez souvent rencontré un message d'erreure issue de l'utilisation des caractères accentués dans les scripts en Python. Voici une solution à ce problème. En première ligne d'un script, il faut insérer la ligne :

```
In [1]: # -*- coding: utf-8 -*-
```

ou bien

```
In [2]: # -*- coding: latin-1 -*-
```

Il s'agit d'un pseudo-commentaire indiquant à Python le système de codage utilisé, ici l'utf-8 (ou le latin-1) qui comprend les caractères spéciaux comme les caractères accentués, les apostrophes, les cédilles , etc .

Utiliser les fonctions en Python Vous savez déjà ce qui est une fonction mathématique d'une variable réelle Considérons par exemple la fonction f(x) suivante définie:
f : x —> 2 x + 1
pour la définir en Python :

```
In [3]: #definition d'une fonction
        def f(x):
            return 2 * x + 1

        #utilisation de la fonction

        print(f(4))
```

```
9
```

Avez-vous remarqué qu'à la deuxième ligne, on n'a pas commencé à écrire au début de la ligne? De la même façon que pour les structures de contrôle 'if' ou 'for', nous devons appliquer la règle d'indentation. Cette indentation est indispensable pour que l'interpréteur Python comprenne la fin d'un bloc de definition d'une fonction.

Le principe de définition de fonctions est intéressant pour deux raisons :

1) cela nous permet de ne pas répéter un calcul long à taper,

2) Python possède un type spécial dédié au fonctions, que l'on peut donc manipuler, mettre dans des listes pour les étudier les unes à la suite des autres.

Par exemple :

```
In [4]: print( f(1) , f(2) , f(3) , f(4) , f(5) , f(6))

(3, 5, 7, 9, 11, 13)

In [5]: print( type(f))

<type 'function'>

In [6]: # definition d'une seconde fonction
        def hi(name):
            print("hello " + name + " from Python!!!")

        #exemple d'utilisation
        hi("Mark")

hello Mark from Python!!!

In [7]: #definition  d'une trosieme fonction
        from random import choice
        def lettre():
            return choice('abcdefghijklmnopqrstuvwxyz')

        #exemple d'utilisation
        lettre()

Out[7]: 'i'

In [8]: #definition d'une liste de fonctions
        mes_fonctions = [f,hi,lettre]

        #utilisation
        mes_fonctions[1]("Dominique")

        mes_fonctions[2]()

hello Dominique from Python!!!

Out[8]: 'n'
```

1.0.1 Script 1 : calcul des forces sur un ballon de football

```
In [9]: from IPython.display import Image
        Image(filename='kick.jpg')

Out[9]:
```

Les forces sur un ballon de football en vol suite à un coup de pied d'un joueur sont deux: la force de la pesanteur (le poids F_P) et la force de traînée exercée par le frottement de l'air sur le ballon (F_T). Leurs expressions sont les suivantes :

$$F_P = M\ g$$

$$F_T = 0.5\ C_D\ \rho\ u^2\ S$$

Ou M est la masse du ballon, g l'accélération de gravité, C_D le coefficient de traînée, ρ la densité massique de l'air, u la vitesse du ballon et enfin S la section du ballon $S = \pi\ R^2$ (avec R le rayon).

Nous demandons d'écrire un script qui tout d'abord demande à l'utilisateur d'entrer les valeurs du rayon du ballon (en mètres), de la masse du ballon (en Kg) et de la vitesse du ballon ainsi que l'unité de mesure pour cette dernière (soit "m/s" ou "Km/h"). Ensuite le script devra calculer les forces F_P et F_T, afficher les deux valeurs ainsi que leur rapport F_T/F_P.

On demande de faire tout cela à l'aide de quatre fonctions :

1) une fonction qui convertit la vitesse de "km/s" en "m/s" si besoin

2) une fonction qui calcule la surface S du ballon à partir du rayon R

3) une fonction qui calcule F_T

4) une fonction qui calcule F_P

Les données du problème sont l'accélération de gravité $g = 9.81 m/s^2$, $\rho = 1.2 m\ kg/m^{-3}$, C D = 0.2.

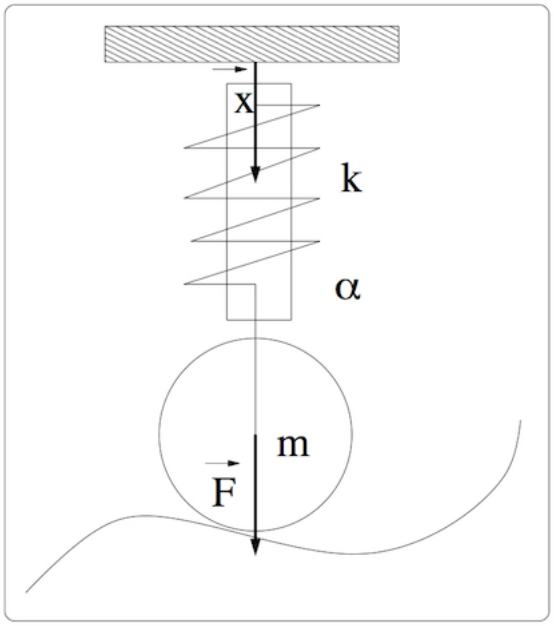
Tourner le script plusieurs fois et dire ce qui change dans le rapport F_T/F_P pour des valuers de vitesse croissante et à masse et rayon fixes.

Script 2 : etude de la fonction de transfert du système ressort - amortisseur d'un automobile
L'objet de ce script est d'illustrer à l'aide de Python les rôles respectifs joués par le ressort et l'amortisseur d'un système automobile. Pour ce faire, nous étudions un ressort couplé à un amortisseur en parallèle.

Comme dans la figure ci-dessous:

```
In [10]: from IPython.display import Image
         Image(filename='masse-ressort-amortisseur.jpg')
```

Out[10]:



L'équation vérifiée par le système est la suivante :

$$\ddot{x} + \frac{\omega_0}{Q}\dot{x} + \omega_0^2\, x = \frac{F}{m}\cos(\omega t)$$

Ici m est la masse de la roue, F et ω sont respectivement l'intensité de la force appliquée et sa pulsation, ω_0 la fréquence caractéristique du ressort et enfin le coefficient Q (dit coefficient de qualité).

Un système très amorti a un Q faible. À l'inverse, un Q élevé correspond à un système peu amorti. Pour fixer les idées, le Q d'une voiture avec des amortisseurs en bon état est légèrement supérieur à 1.

De la solution de l'equation du système ressort-amortisseur on trouve la fonction de transfert qui décrit la réponse du systeme en fonction de la pulsation ω.

En particulier la fonction de transfert s'écrit :

$$T(\omega) = \frac{F}{m\ \omega_0^2}\frac{1}{\sqrt{\left(1-\omega^2/\omega_0^2\right)^2 + Q^{-2}\ \omega^2/\omega_0^2}}$$

ou en utilisant la pulsation adimensionnée u = $\omega/\omega 0$:

$$T(u) = \frac{F}{m\ \omega_0^2}\frac{1}{\sqrt{\left(1-u^2\right)^2 + u^2/\ Q^2}}$$

Nous demandons d'écrire un script qui trace un graphique du module de la fonction de transfert en fonction de la pulsation adimensionnée u pour toutes les valeurs de Q dans l'intervalle [0 , 10] avec un incrementation de $\delta Q = 2.0$.

Prendre $F/(m\omega_0^2) = 2$

Ça pourrait être utile :

1.0.2 La boucle while

Le but de la boucle "while" est de répéter certaines instructions tant qu'une condition est respectée. On n'est pas donc obligé de savoir au départ le nombre de répétitions à faire.

```
In [11]: nb_repetitions = 3
         i = 1

         while i <= nb_repetitions :
             print "Et "+str(i)+"!"
             i = i+1
         print "Zéro!"
```

```
Et 1!
Et 2!
Et 3!
Zéro!
```

L'instruction "break" sert, non pas à interrompre le programme, mais à sortir de la boucle.

```
In [12]: nb_repetitions = 3
         i = 1

         while i <= nb_repetitions :
             print "Et "+str(i)+"!"
             i = i+1
             if i==3:
                 break
         print "Zéro!"
```

```
Et 1!
Et 2!
Zéro!
```

1.0.3 Script 3

Écrire un script qui calcule la valeur critique de Q, c'est-à-dire la valuer intermédiaire entre le régime des vibrations sur amorties et cel de résonance.

Pour faire cela, on doit calculer la valeur maximale de T pour des valeurs décroissantes de Q, et d'arrêter lorsque T_{max} est égal à F / (m ω20).

Utiliser encore : F / (m ω20)=2.

Calcul de la valeur maximale avec numpy Pour ce script, il peut être utile d'utiliser la fonction numpy pour calculer le maximum d'une liste :

```
In [13]:  import numpy as np

          val=np.linspace(0,2,100)

          val_max = np.amax(val)
```

```
print(val_max)
```

```
2.0
```

In [14]:
```
from IPython.core.display import HTML
def css_styling():
    styles = open('custom.css', 'r').read()
    return HTML(styles)
css_styling()
```

Out[14]:
```
<IPython.core.display.HTML at 0x10c0844d0>
```