M1 Physique Appliquée — Université Paris-Sud XI
Année 2016-2017

Programmation et données numériques

TDs : programmation en Python

Un memento sur l'utilisation de Python est donné à l'adresse :
`http://lptms.u-psud.fr/wiki-cours/index.php/Memento_Python`
Il y a un lien vers le site du cours depuis le memento.

Préliminaires

Prise en main de Linux

Nous allons utiliser Linux pour programmer en Python. Nous n'avons besoin que de deux choses : (i) un `terminal` dans lequel vous aller taper des commandes `Linux` (en langage `bash`) ou utiliser Python en ligne de commande, et (ii) un éditeur de texte, par exemple `gedit` pour pouvoir écrire des programmes. Le résumé des commandes Linux utiles est donné dans le memento. Après avoir lu la page, effectuer les actions suivantes :

- ☞ Ouvrir un terminal, créer un répertoire `TD/`, s'y rendre.
- ☞ Lancer `Python3` en ligne de commande et taper

```
print("Hello world!")
```

quitter le programme en faisant

```
quit()
```

- ☞ L'autre manière d'utiliser `Python` est d'écrire le programme dans un fichier d'extension `.py` et de l'exécuter avec `Python`. Éditer avec `gedit` un fichier `script.py` dans lequel vous écrirez l'instruction précédente. Vous pouvez alors exécuter le programme en tapant dans la ligne de commande :

```
python3 script.py
```

Vous pouvez évidemment choisir n'importe quel autre nom que `script` pour votre fichier mais il est utile de conserver la bonne extension.

- ☞ Interrompre une exécution : si vous voulez arrêter une exécution qui prend un temps anormalement long, vous pouvez le faire avec la commande `Ctrl-C` (appuyer sur les touches `Ctrl` et `C` en même temps). Remplir le fichier `script.py` avec les lignes suivantes :

```
i=0
while i<10**9:
    i += 1
```

Lancer l'exécution puis l'arrêter avant qu'il ne s'arrête de lui-même.

Bien entendu, vous pouvez également manipuler les fichiers et vous déplacez dans l'arborescence via l'interface graphique. Il faut garder à l'esprit que lorsque vous utilisez le terminal, il faut être dans le dossier où se trouve le script pour pouvoir l'exécuter via la commande `python3 script.py`. Pensez à sauvegarder vos fichiers et vous les envoyer par mail ou les récupérer par clé usb.

Utilisation de Idle fourni avec Python

L'Idle est un logiciel fourni avec la distribution standard de Python et qui permet d'avoir à la fois la ligne de commande de `Python` ainsi qu'un éditeur de fichier depuis lequel on peut lancer l'exécution en tapant sur la touche `f5`. Sous Linux, l'Idle peut être lancé en tapant une des deux commandes (à vérifier) :

```
idle
idle3
```

☞ Refaire les quelques essais ci-dessus mais en utilisant l'Idle.

À vous de choisir la manière dont vous voulez travailler ! Sous Windows, le terminal n'étant pas fonctionnel, il est recommandé de travailler avec l'Idle fourni avec votre distribution de `Python`. Voir le site du cours pour plus de précision sur l'installation si vous souhaitez travailler à la maison.

S'inscrire sur le site france-ioi

On vous demande, en plus de résoudre les exercices de cette feuille, de vous inscrire sur le site `http://www.france-ioi.org/` et de rejoindre le groupe `M1PA-2016` avec le mot de passe `jaimepython`. Vous n'avez besoin que d'un navigateur web pour cela et vous pouvez donc travailler d'où vous voulez et sans avoir `Python` installé sur votre machine, il suffit d'un accès internet [1]. Le site offre un mélange progressif de cours et de petits exercices. Il permet de s'entraîner avec de nombreux exercices mais notez que vous n'êtes pas obligé(e) de faire tous les exercices pour valider un chapitre (il y a des exercices de validation).

Organisation des TDs

Le niveau en informatique est en général inhomogène, suivant votre parcours et votre expérience. Le but de ces premiers TDs est d'amener tout le monde à maîtriser les bases du langage `Python`.

- ☞ Lors des séances de TD ou chez vous, vous êtes libres d'apprendre à partir du memento et des exercices de TD, du site `france-ioi`, ainsi que des sites en lien depuis le site du cours. Cependant (voir ci-dessous), les TDs et le site `france-ioi` correspondent au programme de ce cours.
- ☞ Essayez d'être autonome dans votre apprentissage, d'avancer à votre rythme. Les enseignants en TD sont là pour répondre à vos questions et vous débloquez. N'hésitez pas à discuter avec vos camarades, à travailler en groupe (notamment pour les problèmes de la fin de l'énoncé), mais gardez en tête que vous devez comprendre et savoir faire les choses vous-même.
- ☞ **Il vous est demandé de valider les niveaux 1 et 2 de `france-ioi` pour le 20 octobre, date du contrôle continu qui aura lieu à la fin de la séance de TD. Ce contrôle évaluera votre maîtrise du langage Python.**
- ☞ Les corrigés de ce TD (hors problèmes) seront mis à disposition sur le site du cours après la séance du 6 octobre.
- ☞ Si vous êtes très à l'aise ou connaissez déjà bien la syntaxe, vous pouvez aider vos camarades, cela vous fera progresser. Vous pouvez faire les exercices Challenge qui sont facultatifs. Vous pouvez valider les niveaux supérieurs de `france-ioi`, comme les niveaux 3 et 4, Google finira bien par vous recruter.

1. N'hésitez cependant pas à installer `Python` sur votre ordinateur !

Apprentissage de la syntaxe de Python

On propose dans cette section de commencer par lire certaines pages du memento en essayant la syntaxe `Python` sur les petits exemples qui y sont donnés, puis de résoudre les exercices suivants. N'hésitez pas à prendre un papier et un crayon pour réfléchir à la solution.

Affichage, types, variables, affectation

Lire les quatre premiers chapitres du memento. Prenez le temps d'essayer la syntaxe avec l'interpréteur, de faire des exemples tirés du memento ou de vos envies, et d'apprendre à écrire plusieurs lignes sur un script et à le lancer. Une fois que vous êtes un peu à l'aise, résoudre les deux exercices suivant :

☞ Affichage, opérateurs – On donne les ingrédients (en gramme sauf pour les œufs) pour une recette de gâteau pour 4 personnes

```
sucre, beurre, farine, oeufs = 175, 150, 250, 4
```

Écrire un programme qui prend en entrée un nombre de personnes N (avec la fonction `input`) et qui affiche les bonnes quantités, au gramme près (nombres entiers), par exemple selon :

```
"Pour 6 personnes, il faut 262g de sucre, 225g de beurre, 375g de farine et 6 oeufs"
```

☞ Division entière, tests conditionnels – Un groupe de N étudiants dispose de voitures pouvant transporter M personnes. Écrire un programme commençant par

```
N = ...
M = ...
```

et affichant le nombre de voitures nécessaires au transport selon les exemples ci-dessous :
si $N = 21$ et $M = 7$

```
"Il faut 3 voitures transportant 7 personnes."
```

si $N = 19$ et $M = 7$

```
"Il faut 2 voitures transportant 7 personnes et une transportant 5 personnes."
```

Tester votre programme pour différentes valeurs de N et de M.

Utilisation de la bibliothèque de math

On présente l'utilisation de modules ou de bibliothèques à travers l'exemple de la bibliothèque `math` qui est fournie avec le langage et qui contient la plupart des fonctions mathématiques utiles. À une bibliothèque est associé un espace de nom, ici `math`, dans lequel sont contenues les fonctions et variables de la bibliothèques. Pour les utiliser, il faut importer la bibliothèque et il existe plusieurs mécanismes pour cela. Enfin, rappelons que l'appel d'une fonction se fait par l'utilisation des parenthèses selon la syntaxe `fonction(arguments)`.

Le mot-clé import permet d'importer l'espace de nom de la bibliothèque. On peut faire afficher l'aide sur la bibliothèque (et son contenu), à l'aide de la fonction `help()` :

```
import math
help(math)
```

Avec cette syntaxe, les fonctions ou constantes doivent être appelées avec l'espace de nom en préfacteur, séparé par un point du nom de la fonction, comme ceci :

```
print(math.pi)      # affiche la constante pi
print(math.sin(2))  # calcule sin(2), appel de la fonction sin()
help(math.sin)      # affiche l'aide sur la fonction sin de la bibliothèque
```

C'est la manière la plus sûre d'utiliser la bibliothèque. En effet, l'espace de nom assure qu'il n'y a pas de confusion entre vos définitions et celles de la bibliothèque.

Cependant, il est possible d'importer une fonction directement, en dehors de l'espace de nom par la syntaxe suivante

```
from math import sin # importe uniquement la fonction sin de la bibliothèque math
print(sin(2))        # pas besoin de préfixer par math., on peut directement utiliser sin()
from math import sin, cos, tan # importation de plusieurs fonctions à la fois
```

Il est enfin possible d'importer tout le contenu de la bibliothèque avec la syntaxe

```
pi = 1.0          # variable définie par l'utilisateur
from math import *
print(sin(pi))    # sin et pi ont été automatiquement importer
                  # la variable pi définie par l'utilisateur a été remplacée par celle de la bibliothèque
```

Une fois les bibliothèques importées, il n'y a plus besoin de les importer de nouveau. Une dernière chose : il est possible de renommer l'espace de nom d'une bibliothèque si on le veut, pour améliorer la lisibilité du code. Ainsi, on utilisera souvent par la suite la librairie `numpy` que l'on renomme `np` à l'aide de la syntaxe

```
import numpy as np
print(np.pi)    # numpy contient aussi la constante pi
```

- ☞ Utiliser la bibliothèque `cmath` adaptée aux nombres complexes pour calculer $\sqrt{-1}$ et $e^{i\pi}$.
- ☞ Les lignes suivantes renvoient une erreur :

  ```
  from math import sin
  print(sin(pi/2))
  ```

  Quel est le problème ? Comment le corriger ?

Répétitions, boucles

Lire le chapitre 5 du memento. En préliminaires, vous pouvez aussi faire les exercices de france-ioi du niveau I correspondant.

- ☞ Table de multiplication - Écrire un code qui affiche la table de multiplication des entiers de 1 à 10 sous la forme d'une succession de lignes de la forme :

  ```
  ...
  7 * 8 = 56
  ...
  ```

- ☞ Racine d'un nombre - Vous savez poser des additions, des multiplications et des divisions. Voyons un algorithme simple qui permet de calculer la racine carrée $\sqrt{x}$ en n'utilisant que les opérations $+, *, /$ et que vous pouvez donc faire à la main. Il repose sur la suite récurrente [2]

  $$a_{k+1} = (a_k + x/a_k)/2 \quad \text{avec } a_0 = x$$

  Compléter le code suivant pour mettre en œuvre l'algorithme :

  ```
  from math import sqrt
  x = 3
  a = ???
  for i in range(5):
      print(a)
      ap = ???
      ????????
  print(a, sqrt(x))
  ```

2. En considérant $x^2 = (a + \varepsilon)^2$ avec ε petit, pouvez-vous justifier la formule de l'algorithme puis son principe ?

Liste et dictionnaire

Lire le chapitre 6 du memento. En préliminaires, vous pouvez aussi faire les exercices de france-ioi correspondant.

- ☞ Générer par une commande simple une liste notée `L` contenant les entiers de 0 à 29. Extraire et faire afficher la liste des multiples de 7 contenus dans `L` à l'aide de la fonction `filter`. Trouver une commande qui inverse l'ordre des éléments de `L` ; faire afficher le résultat.
- ☞ Que va afficher le code ci-dessous ? Expliquer ses différentes étapes.

```
L = 103428
print( sum( [ int(c) for c in str(L) ] ) )
```

- ☞ Remplir dans un tableau à deux entrées `Table[i][j]` les résultats de la table de multiplication des entiers `i` et `j` de 1 à 10. Faire afficher le résultat comme une matrice 10×10.
- ☞ Que va afficher le programme suivant ? Expliquez son principe.

```
L =  [1,2,1,3,2,1,3,2,1,1,2,1]
d = {}
for i in L:
    if i in d : d[i] += 1
    else : d[i] = 1
print(d)
```

- ☞ On écrit les lignes suivantes :

```
a = [ 1, 2 ]
b = [ 3, 4 ]
a[0] = b
b[0] = a
```

  Que renvoie ensuite la commande : `print(a[0][0][0][0][0][0][1])` ? Expliquez.
  Que se passe-t-il pour un nombre impair de `[0]` avant le `[1]` ? Expliquez.
- ☞ Challenge - Après lecture de l'article `https://fr.wikipedia.org/wiki/Crible_d'Érastotène`, écrire un code qui affiche la liste des nombres premiers plus petits que N.

Fonctions et graphiques

Lire les chapitre 7 et 8 du memento ainsi que la page Graphe 2D.

- ☞ Tracer le graphe 2D de la fonction exponentielle avec **pylab** pour $x \in [-3, 3]$ avec 101 points et en donnant des titres pour les axes et le graphiques.
- ☞ Écrire une fonction `derive(f,x,eps)` qui renvoie une estimation de la dérivée selon la formule

$$f'_\varepsilon(x) \simeq \frac{f(x+\varepsilon) - f(x-\varepsilon)}{2\varepsilon}$$

  avec comme arguments : une fonction f, un réel x et une précision $\varepsilon = \text{eps}$.
  Appliquer votre résultat en estimant la dérivée de la fonction racine carrée $f(x) = \sqrt{x}$ au point $x = 3$ pour $\varepsilon = 10^{-6}$. Comparer au résultat analytique attendu.
- ☞ Le code ci-dessous retourne l'erreur suivante : `name 'x' is not defined`

```
def f():
    x = 0
f()
print(x)
```

  Pourquoi ? Quelle ligne rajouter pour résoudre ce problème ?

- ☞ Créer une fonction `Factorielle` qui renvoie $n!$, d'abord de façon non-récursive avec une boucle `for`, puis de façon récursive, c'est-à-dire en faisant en sorte que la fonction s'appelle elle-même. *On pensera à utiliser un test conditionnel dans le second cas.*
  Vérifier votre résultat avec celui de la fonction `factorial` de la bibliothèque `math`.
- ☞ Challenge - Après lecture de la page `https://fr.wikipedia.org/wiki/Algorithme_récursif`, proposer une implémentation en `Python` de l'algorithme récursif calculant le nombre de partitions d'un entier naturel en au plus q parties.

Problème : un générateur de nombres aléatoires

Un manière de générer des nombres aléatoires, très utiles pour les simulations en physique, est d'utiliser des suites congruentes de la forme

$$u_{n+1} = (a \times u_n + c)\%m$$

où a, c et m sont des paramètres, u_0 est appelée la graine.

- ☞ Écrire une fonction `aleatoire()` qui renvoie les nombres générés par la suite. On pourra faire une première version dans laquelle la fonction prend comme argument u_n pour renvoyer u_{n+1}. Essayer les paramètres suivants pour générer les 10 premiers nombres de la suite :

```
a, c, m = 25, 16, 256
graine = 10, 50 puis 125
a, c, m = 8121, 28411, 134456 # quickran choice
a, c, m = 16807,    0, 2**31-1 # standard minimum
```

  Quelles sont les valeurs minimale et maximale possibles pour les nombres générés ?
- ☞ Écrire une seconde version qui n'a pas d'argument mais qui génère les nombres à chaque appel et qui sera plus pratique pour créer des listes aléatoires. Il faudra dans ce cas utiliser le mot-clé `global`

```
u = ????
def aleatoire ():
    global u
     ...
    return ???
```

- ☞ Adapter le générateur précédent pour créer une fonction `Uniform(`$x_{\texttt{min}}, x_{\texttt{max}}$`)` qui génère des nombres réels distribués uniformément sur l'intervalle $[x_{\min}, x_{\max}]$.
- ☞ Utiliser le mini-générateur de nombres aléatoires pour générer une liste de $N = 1000$ réels aléatoires compris entre -1 et 1. *Si vous n'avez pas réussi à créer le générateur, utiliser la fonction `uniform()` de la bibliothèque `random` après avoir lu son aide.*
- ☞ Écrire *en une ligne* une expression qui renvoie la moyenne $\langle x \rangle$ des éléments à l'aide de fonctions Python adaptées aux listes.
- ☞ Écrire en une ligne une expression qui renvoie l'écart-type σ des éléments (la moyenne étant connue).

Problème : simulation du tir ballistique

L'objectif de cette partie est d'obtenir la répartition sur le sol de la quantité d'eau envoyée par un pommeau d'arrosage. On effectue une modélisation sommaire en supposant que le pommeau émet des gouttes de masse m que l'on traitera comme des points matériels et qui sont soumises au champ de pesanteur avec $g = 9.81\,\mathrm{ms}^{-1}$. On notera $\vec{F}$ la résultante des forces pour le principe bien qu'il n'y ait qu'une force en jeu.

Algorithme de Verlet : Le temps est discrétisé en intervalles élémentaires de durée dt et on note $t_i = i \times dt$ les temps intermédiaires. On note enfin les positions dans le plan xOz $\vec{r}_i = (x_i(t_i), z_i(t_i)) = \vec{r}(t_i)$, $\vec{F}_i = \vec{F}(\vec{r}_i)$ et les accélérations $\vec{a}_i = \vec{a}(t_i) = \vec{F}_i/m$, en supposant que la force ne dépend que de la position. Le schéma d'intégration des équations du mouvement est donné par

$$\vec{r}_{i+1} = 2\vec{r}_i - \vec{r}_{i-1} + \vec{a}_i\, dt^2$$

qui provient simplement de la discrétisation de la dérivée seconde. L'initialisation se fait à l'aide d'un développement de Taylor :

$$\vec{r}_1 = \vec{r}_0 + \vec{v}_0\, dt + \frac{1}{2}\vec{a}_0\, dt^2$$

Ce type d'algorithmes converge vers la solution exacte pour peu que $dt \to 0$.

On considère une seule goutte. Le mouvement se fait dans le plan xOz avec z la verticale. Les gouttes sont émises au temps $t = 0$ depuis une hauteur h en $x = 0$ et avec une vitesse initiale $\vec{v}_0 = v_0(\sin\theta\vec{e}_x + \cos\theta\vec{e}_z)$ et l'on utilisera l'angle θ comme paramètre.

- ☞ Écrire l'algorithme de Verlet sous forme d'une fonction `Verlet(r0,v0,tf,F,N=100)` qui prend en arguments les conditions initiales `r0` et `v0` sous forme de tuple, le temps final `tf`, la résultante des forces `F` sous forme d'un tuple `(Fx,Fz)` avec `Fx` et `Fz` deux fonctions, et qui renvoie un tuple `(x,z)` correspondant à la trajectoire $z(x)$ contenant `N` points.
- ☞ Représenter graphiquement les trajectoires obtenues pour $v_0 = 0.2$, $t_f = 0.5$, $N = 100$ et $\theta \in [\pi/5, \pi/4, \pi/3]$.
- ☞ Comparer à la solution exacte

$$z = h + \frac{1}{\tan(\theta)}x - \frac{g}{2v_0^2\sin^2\theta}x^2$$

- ☞ Tracer l'erreur commise en fonction du temps. Elle croît comme une puissance du temps : déterminer cette puissance.
- ☞ Tracer alors l'histogramme des résultats des positions obtenues au sol lorsque l'angle θ est parcouru uniformément dans $[\pi/5, \pi/3]$. *On pourra réfléchir au critère de contact avec le sol et l'approximer par une interpolation.*