Formation :
Python pour l'enseignant de Physique-Chimie au lycée

Table des matières

— *Le lien vers cet énoncé en ligne est ici ; téléchargez-le (lien* `Download` *en haut du document) et utilisez-le pour accéder aux liens cliquables de l'énoncé :*

œ

— Tous les liens soulignés du présent énoncé sont cliquables et permettent d'accéder aux tutoriels, à des fichiers aides ou aux fichiers corrections.

— *Vous aurez auprès de vous les documents mis à disposition sur le site académique et en premier lieu :*
  — `Tracés de graphes avec Python` signalé dans ce poly par : VADEMECUM
  — `Tutoriel pour Python` signalé dans ce poly par : TUTORIEL

— *Vous traitez les exercices dans l'ordre (difficulté croissante).*

— *Pour chaque grande partie, vous créerez un fichier (stipulé dans l'énoncé).*

— *Chaque partie peut se traiter selon 3 niveaux de difficulté :*
  — **niveau 1** : on suit les questions dans l'ordre, en s'aidant ponctuellement des tutoriels et sans copier les codes proposés.
  — **niveau 2** : dès le début de chaque exercice, des aides à la résolution sous forme de morceaux de code sont données ; on copie-colle ces codes dans les programmes et on les complète pour répondre aux questions posées
  — **niveau 3** : on utilise les corrections données en fin de chaque partie. Le travail alors et de **commenter** les différentes lignes du programme, c'est à dire d'expliquer à quoi sert chaque ligne.
    RAPPELS :
    — les commentaires sont des éléments du programme qui ne sont pas lus par Python, mais qui sont présents pour aider l'utilisateur du programme
    — les commentaires sur une ligne sont introduits par # (apparaissent en rouge sous IDLE)
    — les commentaires sur plusieurs lignes sont introduits et terminés par un triple guillemet : `"""` (apparaissent en vert sous IDLE)

— À chacun de choisir son niveau de difficulté ; conseil :
  — Par défaut, commencez par le niveau 2, c'est ainsi que l'énoncé est rédigé.
    Si vous vous sentez à l'aise, vous pouvez tenter de réaliser les exercices sans les aides (niveau 1).
    Si vous avez des difficultés, passez au niveau 3.

— *Enfin, et même si cela n'est pas précisé, pensez à compiler votre code régulièrement pour vérifier si vous répondez bien aux questions.*

1 Algorithmie et Chimie, exercice 1

Au programme de Première (Spécialité) : Déterminer la composition de l'état final d'un système siège d'une transformation chimique totale à l'aide d'un langage de programmation.

On considère la réaction totale entre l'hydrogène sulfureux (H_2S) et le dioxyde de soufre (SO_2) qui produit du soufre et de l'eau et modélisée par l'équation : $2\ H_2S + SO_2 \longrightarrow 3\ S + 2\ H_2$

L'exercice peut être rédigé comme un programme. Vous pourrez le nommer `exercice1.py`.

Allez chercher le code présent à ce lien. Copiez-coller le dans votre programme. Ce code est à compléter.

1. Créez 4 **variables** qui contiendront les valeurs des coefficients stoechiométriques des différentes espèces. On pourra nommer ces variables `coeff_xxx`.

   Pour des précisions sur la notion de variable, TUTORIEL PAGE 9.

2. Créez 4 variables, `n0_H2S`, `n0_SO2`, `n0_S`, `n0_H2O` qui contiendront les quantités de matière initiales de chacune des quatre espèces.

   Le programme devra demander à l'utilisateur les valeurs de ces 4 quantités.

   AIDES :
   — la fonction `input()` (TUTORIEL PAGE 11) permet d'interagir avec l'utilisateur du programme.
   — Attention : la fonction `input()` renvoie une chaîne de caractères ; il faudra penser à la transformer en nombre "flottant".

3. Écrire les formules permettant de calculer les avancements maximaux possibles pour les deux réactifs ; on les notera `xmax_H2S` et `xmax_SO2`. Vous utiliserez les variables définies précédemment.

4. En utilisant une condition (TUTORIEL PAGE 14) ou par une autre méthode (voir ci-dessous), trouvez l'expression de l'avancement maximal `xmax` de cette réaction en fonction des résultats de la question précédente.

   Faites afficher sur la console la valeur de cet avancement maximal.

   AIDES :
   — Si vous ne souhaitez pas traiter la question par la condition, Python dispose des fonctions `min()` et `max()` qui retournent le minimum et le maximum d'une liste passée en paramètre.
   — Pour afficher quelque chose en console, on utilise la fonction `print()` (TUTORIEL PAGE 11).

5. Calculez ensuite les valeurs finales de quantité de matière dans les 4 espèces, que vous appellerez `nF_H2S`, `nF_SO2`, `nF_S` et `nF_H2O` à partir des variables définies précédemment.

   Votre programme devra ensuite afficher sur la console les valeurs finales des quantités de matière des 4 espèces.

↬ `aide à la résolution : exercice1-aide.py`

► `lien vers la correction de cet exercice : exercice1-correction.py`◄

2 Représenter des points expérimentaux sur un graphe, exercice 2

Au programme de Seconde : représenter les positions successives d'un système modélisé par un point lors d'une évolution unidimensionnelle ou bidimensionnelle à l'aide d'un langage de programmation.

On souhaite représenter la trajectoire d'un point au cours de son mouvement. L'étude a été faite par Avimeca, le fichier de sortie travaillé pour arriver aux 3 listes `X`, `Y` et `T` présentes dans le fichier ici.

1. Copiez-coller l'intégralité de ce code dans votre programme que vous pourrez appeler `exercice2.py`.
2. Faites afficher les coordonnées spatiales et temporelles du 5ème point de la trajectoire.

   AIDES :
   — Python numérote à partir de "0" (TUTORIEL PAGE 13).
   — `L[2]` représente donc le 3ème élément de la liste L (TUTORIEL PAGE 13).

   Vous pourrez utiliser le VADEMECUM sur le tracé de graphes (sections 1 et 2) pour la suite.

3. Construisez la trajectoire à l'aide de la fonction `plt.plot()` ; vous incorporez un "label" à cet objet afin de faire afficher une légende.
4. Donnez un nom aux axes et un titre au graphe ("Lancer d'une balle de golf").
5. Limitez les axes d'abscisses et d'ordonnées par les valeurs minimales et maximales des deux listes.
6. Construisez la légende puis faites afficher le tout.

↪ `aide à la résolution : exercice2-aide.py`

▶ `lien vers la correction de cet exercice : exercice2-correction.py`◀

3 Représenter une fonction sur un graphe, exercice 3

3.1 Un problème

1. Placez-vous dans la **console** Python. Créez une liste `L`, `L = [1, 2, 3]` (pour les listes, voir TUTORIEL PAGES 12-13).
2. On souhaite multiplier tous les éléments de la liste par 3,5. On aimerait pouvoir écrire `3.5*L` et avoir le résultat. Testez cette formule. Quel est le retour ?
3. Python n'accepte pas le calcul d'une fonction sur une liste (ici $f(x) = 3,5 \times x$) ; c'est dommage mais c'est normal, les listes peuvent contenir tout type de données, des nombres comme des chaînes de caractères.

   Pour pouvoir réaliser l'opération souhaitée, c'est à dire faire agir une fonction sur tous les éléments d'une liste, on peut soit passer par une boucle et faire agir la fonction sur tous les éléments de la liste, soit transformer notre liste en un nouvel objet, un **tableau**. Celui-ci ne contient que des nombres, et du coup, les opérations naturelles sont autorisées (notamment celle qu'on souhaite réaliser).

   Pour y parvenir, nous avons besoin de la bibliothèque **numpy** qui crée et gère les tableaux et les opérations sur ces derniers.

   Puis nous tranformerons notre liste L en tableau :

code Python

```
>>> L = [1, 2, 3]
>>> import numpy as np # on importe la bibliothèque numpy sous l'alias np
>>> T = np.array(L) # on transforme la liste L en tableau par la fonction array de la bibli numpy
>>> print(T)
>>> 3.5*T
>>> dir(np) # pour voir toutes les fonctions de la bibliothèque : il y a une fonction cos (cosinus)
```

On constate qu'à présent on peut faire le calcul de façon "naturelle".

Conclusions :

— La bibliothèque numpy nous permettra d'obtenir l'image d'un tableau de données par une fonction. Dès que vous voulez tracer une courbe théorique, utilisez cette bibliothèque.

— Autre point important : Numpy permettra de plus de réaliser des opérations "naturelles" entre les éléments de tableaux.
Soient 2 tableaux `U` et `I` de dimension 1 et de même longueur. Alors `U/I` réalisera le calcul membre à membre attendu et retournera un tableau contenant l'opération de division terme à terme.

3.2 Ondes et signaux, exercice 3

Au programme de Première (Spécialité) : Représenter un signal périodique et illustrer l'influence de ses caractéristiques (période, amplitude) sur sa représentation.

Vous pouvez créer un fichier `exercice3.py` ; Copiez-coller à l'intérieur `le code présent à cette adresse`.
On souhaite représenter l'intensité d'un signal périodique qui s'écrit sous la forme : $I(x) = I_0 \times \cos(k \cdot x + \phi)$

— L'étude se fait dans la région de l'espace des x : [0,5]
— On prendra $I_0 = 3$
— On prendra $k = \pi$
— On prendra $\phi = \dfrac{\pi}{2}$

1. À quoi servent chacune des trois premières lignes ? (aide : TUTORIEL PAGE 10)

2. Entrez les données de l'énoncé, I_0, k et ϕ dans des variables aux noms appropriés.

3. Construisez votre tableau `X` de valeurs pour les antécédents (les x).

   AIDES :
   — Numpy propose entre autres (une fois importé sous l'alias `np`) :
     — la fonction `np.arange()` qui est équivalente à la fonction range (TUTORIEL PAGE 12) mais qui permet de prendre un pas décimal : `np.arange(1,10,0.01)` donne la série de valeurs entre 1 (inclus) et 10 (exclu) par pas de 0,01.
     — la fonction `np.linspace()` qui permet de donner un intervalle et le nombre de points à prendre dans l'intervalle : `np.linspace(1,10,2000)` renvoie un tableau de 2000 valeurs (régulièrement réparties) entre 1 (inclus) et 10 (inclus).

4. Construire alors le tableau `I` constitué de toutes les images $I(x) = I_0 \times \cos(k \cdot x + \phi)$.

   AIDES :
   — Numpy connaît beaucoup de fonctions mathématiques ; ainsi, `np.cos(X)` renvoie les images d'un tableau X par la fonction cosinus.
   — Rappel du TUTORIEL PAGE 10 : pour voir l'ensemble des fonctions d'une bibliothèque (par exemple numpy), pensez à la fonction `dir()` (voir plus haut).

   AIDE :
   — vous pourrez utiliser le VADEMECUM sur le tracé de graphes (sections 1 et 2) pour la suite.

5. Vous disposez à présent de deux tableaux, vous pouvez obtenir une représentation graphique. Commencez par construire la courbe à l'aide de la fonction `plt.plot()` ; vous incorporez un "label" à cet objet afin de faire afficher une légende.

6. Donnez un nom aux axes et un titre au graphe.

7. Limitez les axes d'abscisses et d'ordonnées par les valeurs minimales et maximales des deux tableaux.

8. Construisez la légende puis faites afficher le tout.

↬ `aide à la résolution : exercice3-aide.py`

▶ `lien vers la correction de cet exercice : exercice3-correction.py` ◀

4 Autour de la vitesse, exercice 4

Au programme de Seconde : Représenter des vecteurs vitesse d'un système modélisé par un point lors d'un mouvement à l'aide d'un langage de programmation.

Vous pouvez créer un fichier `exercice4.py` ; Copiez-coller à l'intérieur `le code présent à cette adresse`.

4.1 Un aperçu de la trajectoire

1. En vous servant de ce qui a déjà été fait, représentez la trajectoire du point au cours de son mouvement.

4.2 Étude pour un point particulier

2. En utilisant les éléments des listes X, Y et T, le tuto page 12 et suivantes, calculer les coordonnées du vecteur vitesse du 3ème point de la trajectoire (`vx` et `vy`), puis affichez-les.

   RAPPEL :
   - Python numérote les listes à partir du numéro "0" ;
   - Afin de se mettre en concordance avec la définition de dérivée en maths, on privilégiera en PC comme formule de calcul de vitesse :

   $$\vec{v}(n) = \frac{\vec{r}(n+1) - \vec{r}(n)}{t(n+1) - t(n)}$$

3. Déduisez-en alors la norme du vecteur vitesse pour le troisième point de la trajectoire (`v`). Calculez-la puis faites afficher cette norme.

   AIDES :
   - Mettre une expression x au carré sous Python s'écrit : `x**2`.
   - On peut utiliser la fonction racine carrée de la biblitohèque `math` : `sqrt()`
   - On peut aussi mettre l'expression à la puissance un demi : `(xxxx)**(1/2)`

4. On souhaite maintenant tracer le vecteur vitesse en ce point. Il existe pour cela en Python la fonction `arrow()` et la fonction `quiver()` du paquet `matplotlib.pyplot`.
   Nous nous intéressons uniquement ici à la fonction `arrow`.

   POINT COURS :

   Une fois importé le paquet `matplotlib.pyplot` sous l'alias `plt`, la fonction arrow est appelée par :

   ```
   plt.arrow(x,y, dx, dy, length_includes_head = "true", head_width = 0.02,
   color = "...")
   ```

   avec :
   - x : la coordonnée selon Ox du point de base du vecteur
   - y : la coordonnée selon Oy du point de base du vecteur
   - dx : la longueur selon Ox du vecteur
   - dy : la longueur selon Oy du vecteur
   - color (optionnel) : à choisir parmi red, blue, green, cyan, black, yellow, ...
   - le paramètre length_includes_head = "true" (optionnel) permet d'avoir la longueur de la pointe de la flèche dans la longueur totale de la flèche
   - le paramètre head_width (optionnel) permet de jouer sur l'épaisseur de la pointe de la flèche

   À l'usage, les vecteurs sont souvent trop grands ou trop petits ; il faudra donc appliquer un facteur multiplicatif aux longueurs dx et dy selon Ox et Oy.

   En vous servant de cela, tracez le vecteur vitesse au troisième point de la trajectoire.

4.3 Étude pour tous les points

5. En utilisant une boucle (TUTORIEL, page 17 et suivantes), tracez tous les vecteurs possibles pour les points où cela est possible.

AIDES :
— On pourra penser à l'utilisation d'une boucle `for` associée à la fonction `range` :
`for i in range(0,10) :` i va parcourir toutes les valeurs de 0 à 9 (TUTORIEL page 12)
— le nombre d'éléments d'une liste `L` peut être donné par `len(L)` (TUTORIEL page 13)

6. Comment comprenez-vous la ligne du programme :
```
plt.text(0,0, "vecteurs vitesse", color = "magenta")
```
À quoi sert-elle ? Comment le vérifieriez-vous ? Vous pouvez rédiger ces réponses en commentaires, directement dans le code.
7. Rajoutez un nom aux axes et un titre au graphique.

↬ `aide à la résolution : exercice4-aide.py`

► `lien vers la correction de cet exercice : exercice4-correction.py`◄

5 Modélisation à partir de points expérimentaux, exercice 5

Au programme de Seconde : Représenter un nuage de points associé à la caractéristique d'un dipôle et modéliser la caractéristique de ce dipôle à l'aide d'un langage de programmation.

On parle de modélisation dans la partie Ondes et Signaux du programme de Seconde. Nous allons le traiter en mécanique, ce qui permettra de voir un cas plus général de l'utilisation d'une fonction particulière.

La trajectoire précédente de l'exercice 4 semble être modélisable par une parabole (de type $a \cdot x^2 + b \cdot x + c$).

POINT COURS :
La bibliothèque `numpy` sait "fitter" une courbe par un polynôme de degré n grâce à la fonction `polyfit`.
Considérons 2 listes X (abscisses) et Y (ordonnées) dont on souhaite approcher la représentation graphique par un polynôme de degré 3 ($a \cdot x^3 + b \cdot x^2 + c \cdot x + d$), la fonction `polyfit` nous renverra une liste des coefficients $[a, b, c, d]$.

La commande est la suivante :
```
import numpy as np
liste_des_coeffs = np.polyfit(X,Y,3)
```
On peut alors accéder au coefficient a en écrivant `liste_des_coeffs[0]` pour l'afficher ou s'en servir pour tracer la courbe "fittée".

REMARQUE :
Pour réaliser un régression linéaire, il suffira de choisir un polynôme de degré 1.

Vous pouvez créer un fichier `exercice5.py` ; Copiez-coller à l'intérieur `le code présent à cette adresse`.

1. Commencez par représenter la trajectoire du point considéré en pointillés bleus. Légendez cette trace.
2. En utilisant la fonction polyfit, trouvez et faites afficher les coefficients polynômiaux qui permettent le mieux d'approximer notre courbe.
3. Pour tracer notre courbe modèle, il serait intéressant de pouvoir écrire :
```
Y_modele = a*X**2 + b*X + c
```
Mais pour réaliser ce calcul "naturel", X doit être un tableau et pas une liste.
— Commencez par transformer votre liste X en tableau que vous pourrez nommer X à nouveau. (Vous avez déjà réalisé cela en bas de page 4 du présent poly)
— Construisez alors votre tableau modèle `Y_modele`.
— Tracez alors la courbe représentative de votre modèle sur le même graphe en trait plein rouge. Légendez la. Conclusion.
4. Rajoutez des noms aux axes et un titre au graphique.

↬ `aide à la résolution : exercice5-aide.py`

► `lien vers la correction de cet exercice : exercice5-correction.py`◄