

Quelques notions d'informatique à destination des professeurs de sciences

Ceci est une proposition de plan. Nous en traiterons une partie selon le temps et l'envie pendant les séances dont nous disposons... Vous pouvez proposer d'autres thèmes bien entendu.

1 Culture informatique

Nous pouvons prendre une dizaine de minutes pour parler de la naissance de l'informatique et répondre aux questions suivantes :

1. Qu'est-ce qu'un ordinateur ?
2. Qu'est-ce qu'un algorithme ?
3. Un ordinateur peut-il tout calculer ?

2 Exemples

Je peux montrer quelques exemples de sujets traités en TP avec mes élèves. Le but n'est pas de comprendre le code utilisé mais de voir ce qui est attendu dans le supérieur, ce dont Python est capable, et de se rendre compte que le principe n'est souvent pas compliqué. Par ailleurs, on peut penser que la mise en continuité avec le supérieur est une des raisons de l'introduction de l'informatique dans les programmes de lycée.

Il s'agit de sujets informatiquement simples, traités avec des élèves en ayant fait deux heures par semaine pendant un an.

- Méthode d'Euler : le pendule à ressort mouillé ;
- Équation de la chaleur ;
- Équations proies prédateurs (SVT) ;
- Proportion des formes d'un tri-acide en fonction du pH dans une solution ;
- ...

3 Le B-A BA

Dans les exemples ci-dessous, on va découvrir :

- Comment créer et utiliser un programme ;
 - La notion de variable, globale et locale ;
 - Les tests et les booléens.
1. Écrire une fonction `durée_parcourt` prenant en entrée une vitesse `v` et une longueur `l` et renvoyant le temps nécessaire pour parcourir cette distance à cette vitesse.
 2. Écrire une fonction qui calcule le volume d'une boule en fonction de son rayon.
On introduit l'utilisation d'une bibliothèque (ici pour obtenir π).
 3. Écrire une fonction qui calcule la masse d'une boule en fonction de son rayon et sa masse volumique.
 4. Écrire une fonction qui calcule le poids d'une boule en fonction de son rayon et sa masse volumique.
On introduit ici la notion de variable globale.

Dans la suite, pour que les formules ne soient pas trop évidentes, on prend une sphère creuse. Les fonctions ci-dessous prendront comme argument le rayon `R`, l'épaisseur `e` de la paroi, et la masse volumique `rho` du matériaux de la paroi.
 5. Écrire une fonction qui calcule l'accélération d'une boule plongée dans l'eau.
La formule devenant compliquée, il est judicieux d'enregistrer les résultats intermédiaires dans des variables locales.
 6. Écrire une fonction `flotte` qui indique si la boule flotte (on mettra toujours les mêmes arguments en entrée). Ainsi, `flotte(R, rho, e)` renverra `True` si la boule flotte et `False` sinon.
On utilise ici un test `if`, et les booléens `True` et `False`.

7. « Tâche complexe » : plongeons un cube de glace de longueur de côté a à la température T_g dans un verre contenant un volume V de whisky à la température T_w . Calculer la température finale du whisky.

Si nous décidons de traiter ce problème (en fonction du temps restant...) le but est que vous choisissiez vous-même les étapes (et donc les fonctions) intermédiaires.

J'ai trouvé sur Wikipédia les données suivantes : chaleur latente de fusion de la glace : 333 Jg^{-1} , capacité thermique massique de la glace : $2.06 \text{ Jg}^{-1}\text{K}^{-1}$, capacité thermique massique de l'éthanol : $2.44 \text{ kJkg}^{-1}\text{K}^{-1}$, capacité thermique massique de l'eau : $4.18 \text{ Jg}^{-1}\text{K}^{-1}$, pourcentage alcoolique volumique du Loch Lomond (le whisky du capitaine Haddock) 46%.

4 Les tableaux

Pour manipuler plusieurs données, on utilise des tableaux (List en Python).

4.1 Exemple de manipulation de tableaux à deux cases

Un vecteur ou un point peut être modélisé par un tableau à deux ou trois cases. C'est un bon moyen pour commencer à manipuler les tableaux.

Ci-dessous, on fixe une base orthonormée (\vec{e}_x, \vec{e}_y) avec \vec{e}_x horizontal et \vec{e}_y « vers le haut ».

1. Écrire une fonction pour calculer la norme d'un vecteur, puis la distance entre deux points.
2. Écrire une fonction prenant un vecteur \vec{u} et un réel λ et renvoyant le vecteur $\lambda\vec{u}$.
3. Un objet est fixé à un ressort dont l'autre extrémité est fixée à l'origine du repère. Écrire une fonction prenant en entrée la masse de l'objet, sa position (tableau de deux éléments), la constante de raideur du ressort, sa longueur au repos, et renvoyant la force qu'il subit.
4. Écrire une fonction pour additionner deux vecteurs.
5. Écrire une fonction prenant les mêmes arguments que dans la pénultième question, et renvoyant l'accélération de l'objet.

En poursuivant sur cette lancée, on peut obtenir la trajectoire de l'objet par la méthode d'Euler.

4.2 Manipulation de tableaux de données

En pratique, on voudra souvent traiter un tableau contenant les résultats de nombreuses mesures. Si \mathbf{t} est un tableau, on peut parcourir tous ses éléments par un `for x in t` :

C'est l'analogue du « $\forall x \in t$ » des mathéux. On peut naturellement remplacer x par tout autre nom.

Exemples :

1. Calculer une moyenne.
2. Calculer l'écart-type.
3. Appliquer un traitement à chaque élément. Par exemple écrire une fonction prenant un tableau \mathbf{t} et renvoyant le tableau des carrés des éléments de \mathbf{t} .

Possibilité de pythonneries ici.

4. Ne conserver que certains éléments. Par exemple écrire une fonction pour garder uniquement les éléments non nuls d'un tableau.

Possibilité de pythonneries ici aussi.

5 Utilisation de données expérimentales

5.1 La bibliothèque matplotlib

La bibliothèque matplotlib permet d'effectuer des tracés. Vous avez vu la syntaxe en stage académique.

Dans les exemples suivants, nous nous plaçons dans la peau d'Isaac Newton regardant tomber les pommes.



Le fichier `pommes.py` crée deux tableaux résultant d'une campagne d'observation de pommiers : la première contient la hauteur d'où sont tombées les pommes, et l'autre les temps de chute.

1. Tracer le nuage de points correspondant au temps de chute en fonction de la hauteur.
2. Écrire une fonction prenant en entrée deux nombres a et b et traçant par dessus la courbe précédente la droite d'équation $y = ax + b$.
3. Tracer la courbe d'équation $y = \sqrt{\frac{2x}{g}}$, au cas où.

5.2 Régression linéaire

On trouve une fonction de régression linéaire dans la bibliothèque `scipy.stats`. Elle s'appelle `linregress` et prend en entrée deux tableaux : celui des abscisses et celui des ordonnées des points. Elle renvoie 5 valeurs dont les deux premières nous intéressent : la pente et l'ordonnée à l'origine.

Une fonction plus simple et plus générale est disponible dans `numpy` : c'est `polyfit`. Elle prend en entrée la liste des abscisses, la liste des ordonnées, et le degré de polynôme de régression à calculer (donc pour une régression linéaire, prendre 1), et elle renvoie les coefficients du polynôme de régression.

Concrètement, pour une régression linéaire on utilisera `np.polyfit(abscisse, ordonnées, 1)` et on récupérera le tableau à deux cases [ordonnée à l'origine, pente].

1. Tracer la droite de régression linéaire correspondant aux données précédentes.

À présent soyons sérieux :

2. Tracer le nuage de points de la hauteur en fonction du temps de chute.
3. Calculer la régression d'ordre 2 de ce nuage de points.

Il faudra calculer un polynôme et tracer à partir de ses coefficients... Soit on le fait ensemble, soit je donne un programme directement, selon temps et motivation.

4. Tracer icelle.

5.3 Lecture de fichiers

On enregistre fréquemment des données dans des fichiers csv. Les données de notre pommier sont dans `pommes.csv`. Le programme `lit_csv` permet d'extraire le contenu d'un tel fichier.

Adapter la feuille de calcul précédente à la situation où les données sont dans un fichier csv.

5.4 Plus dur : un exemple sans pommes mais avec une boucle for

Le fichier `résistances.py` contient des mesures d'intensité et de tension aux bornes d'une résistance.

On propose pour déterminer la résistance de cette résistance¹ de tracer la courbe tension en fonction de l'intensité, puis de tracer un tas de droites pour différents coefficients directeurs (utiliser une boucle « pour ») afin de regarder laquelle passe au plus près des points.

On pourra vérifier à l'aide d'une régression linéaire comme précédemment.

1. C'est la « résistance de ce conducteur ohmique » qu'il faut dire ?

6 Avancement d'une réaction chimique

Suivant une suggestion du programme de première, étudions l'avancement d'une réaction chimique totale.

Attention : nous passons à un niveau supérieur de programmation !

1. Choisir une réaction chimique totale et pas trop triviale.
2. Pour commencer, écrivons un programme prenant en entrée la quantité des différents réactifs, et traçant les courbes donnant la quantité de matière des différents composés en fonction de l'avancement de la réaction. Voici les grandes lignes de ce programme :
 - On crée un tableau, initialement vide, pour chaque composant ;
 - Pour faire varier l'avancement de réaction, on peut utiliser `for xi in np.linspace(début, fin, nombre de valeurs)`.
 - Pour chaque valeur de ξ , on calcule la quantité de chaque composant, et on la met dans le tableau correspondant, via un `tableau.append(valeur)`
 - Quand on a fini la boucle, les tableaux sont pleins. On plotte alors chacun et on affiche le graphe.
3. Le graphe obtenu, on peut regarder à l'œil le moment où un réactif est épuisé, et lire la composition finale du système.

Remarque : Rajouter un `plt.grid()` avant le `plt.show()` affiche une grille, bien pratique ici.

4. Maintenant une version savante : nous allons faire avancer la réaction tant qu'il reste des réactifs. (Dans la version précédente, on calculait la composition pour tous les avancements, même lorsqu'une quantité devenait négative ...)
 - (a) Écrire une fonction `il_reste_des_réactifs` prenant en entrée les quantités des différents réactifs et renvoyant `True` si et seulement si ces nombres sont tous positifs.
 - (b) Maintenant, écrivons un programme prenant en entrée les quantités initiales de réactif, ainsi qu'un nombre `dxi` et faisant avancer la réaction par pas de `dxi` tant qu'il reste des réactifs. À ce moment, renvoyer les quantités des composants.

On utilise une boucle « tant que », `while` en Python.

5. Pour ceux qui s'ennuient, on optimise (de beaucoup) la fonction précédente par la méthode de la dichotomie (au programme de math de terminale)...