Notes et références

Tu trouveras ici des commentaires et des indications de lectures sur chacune des activités.

Ressources générales

- *Python au lycée – tome 1.* Indispensable !
  — À télécharger ici : exo7.emath.fr
  — Version papier en vente à prix coûtant : amazon.fr/dp/1986820033
  — Tous les codes `Python` ainsi que les fichiers sources : github.com/exo7math/python1-exo7
  — Les vidéos d'introduction à `Python` : youtube.com/PythonAuLycée
- *Apprendre à programmer avec Python 3* de Gérard Swinnen, éditions Eyrolles. Le livre est disponible gratuitement en téléchargement, selon la licence *Creative Commons BY-NC-SA* :
  inforef.be/swi/python.htm
- La documentation officielle de `Python` :
  docs.python.org/fr/3/

1. Suites arithmétiques – Suites géométriques

Ces deux types de suites sont idéales pour réviser les différentes formules permettant de calculer les termes soit directement, soit par récurrence. On peut regretter que `Python` n'effectue pas des calculs exacts même pour les opérations élémentaires. Par exemple les fractions sont transformés en nombres flottants :

- `1/3` renvoie `0.33333...`
- `1/10 + 1/5` renvoie `0.30000000000000004` et n'est pas égal à `3/10`.

2. Nombres complexes I

C'est une activité assez simple pour se remettre à `Python` en terminale. C'est très agréable que `Python` calcule directement avec les nombres complexes avec encore le bémol que les calculs s'effectuent en nombres flottants. L'activité incontournable est de résoudre les équations du second degré. On insistera sur la programmation sous forme de fonctions. Attention une fonction n'est pas là pour afficher des résultats, mais une fonction renvoie des valeurs. C'est aussi l'occasion d'insister sur l'aspect géométrique des nombres complexes.

3. Nombres complexes II

Le module `cmath` permet d'aller plus loin avec les nombres complexes et la forme module/argument. On peut se contenter d'utiliser les fonctions prédéfinies mais ici on propose en plus de reprogrammer ces fonctions. Les coordonnées polaires sont importantes au delà des nombres complexes que ce soit en mathématique ou en physique. Encore une fois on insiste sur la visualisation.
Notez l'astuce de Gauss (1777-1855) pour réduire le produit de deux nombres complexes avec seulement trois multiplications réelles. C'est une astuce similaire à la base de la multiplication rapide de Karatsuba (en 1960). La notion de complexité est abordée dans un autre chapitre.

4. Dérivée – Zéros de fonctions

Les notions de dérivée et de tangente sont essentielles pour l'étude de fonction. À l'aide de `Python` on peut trouver des valeurs approchées de limites et donc des dérivées. Savoir programmer le tracé du graphe d'une fonction permet de mieux appréhender la calcul infinitésimal et la notion de « epsilon ».
Deux activités sont fondamentales : la dichotomie et la méthode de Newton. La dichotomie est à la base du concept « diviser pour régner » et on retrouvera ce concept lors de la recherche dans une liste (chapitre « Le mot le plus long »). La méthode de Newton, qui utilise la notion de tangente, permet de comprendre qu'un bon algorithme est plus important qu'un ordinateur puissant !

5. Exponentielle

Il est important d'appréhender la croissance de l'exponentielle (c'est toujours plus que l'on « croît »). L'exponentielle est essentielle pour la modélisation de nombreux phénomènes physiques, par contre c'est la bête noire des informaticiens car tout phénomène exponentiel déborde rapidement les capacités d'un ordinateur (voir le chapitre « Tri - Complexité »).

6. Logarithme

Le logarithme a autant d'importance que l'exponentielle. La difficulté supplémentaire c'est qu'il y a plusieurs bases et plusieurs notations pour le logarithme. Le mieux est quand même de commencer par le logarithme décimal, vu comme l'exposant d'une puissance de 10. Mais bien sûr le logarithme népérien est préféré en mathématiques et le logarithme en base 2 est préféré en informatique. L'histoire du calcul des logarithmes à travers les âges est illustrée par quelques algorithmes.

7. Intégrale

Il s'agit d'une activité très facile du point vue informatique. Le calcul approché d'intégrales est indispensable lorsque qu'on ne peut pas calculer de primitives pour une fonction, ce qui est la cas dans la « vraie vie ». Le calcul approché est de nouveau l'occasion de comparer l'efficacité des différents algorithmes.

8. Programmation objet

La programmation objet est une façon différente de concevoir ses programmes. Si on veut mieux comprendre `Python` il faut comprendre ce concept. Mais à part le chapitre suivant, toutes les activités de ce livre peuvent se passer de programmation objet. L'avantage de la programmation objet, c'est qu'un l'objet contient à la fois ses variables et ses fonctions. L'exemple de plusieurs tortues devrait être éclairant : chaque tortue a sa position, sa couleur, chaque tortue peut avancer indépendamment des autres tortues. Par contre les premiers pas en programmation objet sont délicats et il faut beaucoup de lignes de code même pour un petit programme.

9. Mouvement de particules

C'est l'activité où la programmation objet prend tout son sens : chaque particule est un objet indépendant mais qui cependant interagit avec les autres particules. Pour ceux qui cherchent des idées en lien avec la programmation objet et la modélisation du monde qui nous entoure, le site :

natureofcode.com

est plein de projets, mais codés avec un autre langage. Le livre et sa version en ligne sont gratuits.

10. Algorithmes récursifs

Il s'agit d'un concept avancé de programmation qui demande pas mal de remue-méninges. La similarité entre le principe de récurrence et la récursivité, illustré par l'inusable exemple de la factorielle, permet de bien démarrer. Mais la récursivité permet d'écrire de façon simple des successions compliquées et imbriquées de solutions. Voir le déroulé complet d'appels récursifs reste fascinant ! C'est ce que l'on fait dans les activités graphiques à la fin. C'est sûrement un des chapitres qu'il vaut mieux étudier par petites touches en plusieurs fois. On retrouvera la récursivité dans d'autres chapitres.

11. Tri – Complexité

Deux notions très importantes en informatique. Tout d'abord la notion de tri, bien que le terme juste soit celui d'ordre. L'intérêt est immédiat : on n'imagine pas chercher un mot dans un dictionnaire dont les noms ne seraient pas classés par ordre alphabétique. Il existe de nombreux algorithmes de tri chacun apportant de petites ou de grandes améliorations. On présente ici seulement trois algorithmes simples, le meilleur algorithme nécessitant la récursivité.
La notion de complexité est très théorique même si on comprend bien la différence entre un algorithme lent ou rapide. On n'aborde pas ici la notion de complexité pour la mémoire. La notion de complexité est liée à la notion de limite : un algorithme en $O(n^2)$ peut être plus rapide qu'un algorithme en $O(n)$ pour de petites valeurs de n, mais pour n « assez grand » c'est l'algorithme en $O(n)$ le plus rapide. C'est l'occasion de comparer les suites $(\ln n)$, (n) et (e^n).

12. Calculs en parallèle

Les processeurs sont de plus en plus puissants (voir la loi de Moore qui donne un bon exemple de phénomène exponentiel) mais pour aller encore plus vite il faut distribuer le travail : c'est le calcul parallèle. L'idée est simple mais la mise en œuvre est complexe, d'ailleurs avec `Python` il n'est pas facile d'effectuer des tâches en parallèle. Les activités proposées sont seulement des simulations de calcul parallèle.

13. Automates

Les automates sont une version simple du « jeu de la vie ». On peut étudier toutes les règles possibles et c'est l'occasion de revoir l'écriture binaire.

14. Cryptographie

Union parfaite des mathématiques et de l'informatique : la cryptographie ! Il faut tout d'abord comprendre qu'un message secret se transforme en une suite d'entiers et ensuite que le chiffrement et le déchiffrement sont des opérations mathématiques. L'aspect le plus amusant c'est l'attaque d'un message secret : les ordinateurs sont capables de rechercher des milliers de combinaisons en quelques secondes. Par contre c'est une grave erreur de croire que c'est le nombre de combinaisons qui fait la solidité d'un système de chiffrement. C'est l'erreur faite par les Allemands lors de la seconde guerre mondiale avec la machine *Enigma*. Grâce à Turing les Alliés ont pu décrypter les messages des Allemands. L'histoire de la cryptographie est racontée dans le livre *Histoire des codes secrets* de Simon Singh (Le livre de Poche).

15. Le compte est bon

« Le compte est bon » est exactement le genre de problème difficile à résoudre pour un humain mais simple pour un ordinateur. Le nombre de combinaisons à étudier n'est pas très élevé et une recherche exhaustive bien menée suffit à trouver une solution. Ce qui complique la mise en œuvre c'est qu'il ne faut pas se contenter des solutions séquentielles (voir l'activité) ce qui conduit à un algorithme récursif.

16. Le mot le plus long

Encore un problème qui est de petite taille pour un ordinateur mais qui fait réfléchir les humains. Par contre pour le rendre simple à un ordinateur il faut prendre le problème par le bon bout. Un humain prend les lettres et à partir des lettres cherche des mots français. Ce n'est pas la bonne approche pour un ordinateur car il y a beaucoup de tirages possibles et beaucoup de combinaisons pour chaque tirage. Par contre des mots français il y en a environ 100 000, ce qui est peu pour un ordinateur. Donc avec la méthode d'indexation et un bon algorithme de recherche dans une liste triée c'est très facile.

17. Images et matrices

Cette activité n'est pas très difficile : pour le début une matrice est juste utilisée comme un tableau de nombres. L'application de la convolution sur les images est assez impressionnante et utilisée telle quelle dans les logiciels de retouche d'images. Pour déformer les images on considère cette fois les matrices comme des applications du plan dans lui-même.

18. Ensemble de Mandelbrot

La fractale de Mandelbrot demande un peu d'effort de la part du programmeur et beaucoup de calculs pour l'ordinateur. Par contre tout ce travail vaut le coup et vous plonge dans le monde infini des fractales. On peut programmer cette activité sans connaître les nombres complexes. Si on veut vraiment dessiner beaucoup de fractales avec `Python` il faut chercher des méthodes pour accélérer les calculs d'un facteur 100 (voir cython, numba, numpy, pycuda).

19. Images 3D

Pour visualiser des objets en dimension 3, `matplolib` est appréciable car on peut faire tourner les objets dans l'espace. Pour bien comprendre que l'on ne dessine que des images du plan il faut étudier les différentes projections possibles. Une autre écriture possible des perspectives serait d'utiliser les matrices. Le calcul vectoriel est très utile pour la visualisation en dimension 3 car les objets compliqués sont représentés par une multitude de triangles de l'espace où chaque triangle est traité indépendamment. Par exemple pour connaître l'éclairage d'un triangle on effectue le produit scalaire entre un rayon lumineux et un vecteur normal au triangle.

20. Sudoku

Encore un problème qui fait transpirer les humains plus que les ordinateurs. Mais ici il faut en plus se creuser la tête pour programmer la résolution. Le « retour en arrière » (*backtracking* en anglais) est une méthode générale qui permet de résoudre de nombreux problèmes (ici le problème des 8 reines et le sudoku).

21. Fractale de Lyapunov

Une nouvelle série de fractales dans la continuité de Mandelbrot. Cependant il faut connaître un peu plus de mathématiques (les logarithmes) et les calculs sont encore plus longs.

22. Big data I

La statistique de base (moyenne, écart-type, régression linéaire, distribution de Gauss…) fournit des outils puissants pour le traitement des données même de grande taille. Notez que dans l'activité sur la classification bayésienne naïve les formules sont présentées sous une forme simplifiée.
Les identités fictives utilisées dans les activités ont été créées à l'aide de l'outil rigolo :

fr.fakenamegenerator.com

23. Big data II

Lorsque l'on parle de « big data » on parle de données dépassant le téraoctet. Il alors trop difficile ou trop long de traiter les données une par une. Les activités présentées sont des méthodes récentes adaptées à ces volumes d'informations. Le test probabiliste d'un parenthésage correct est basé sur le cours « Algorithmes de flux de données » de Claire Mathieu au Collège de France. Le perceptron est avant tout lié à l'intelligence artificielle et plus spécifiquement aux réseaux de neurones.