Math ECG 2. 2023-2024

Mathématiques Appliquées - F. Gaunard
http://frederic.gaunard.com
ENC Bessières, Paris 17e.

Informatique.

Semaine du 16 Octobre

.

On propose cette semaine des activités Python pour encourager un investissement réel dans cette partie du programme de math. Chaque activité illustre et utilise des notions du cours. Y répondre nécessite de réfléchir, d'écrire des ébauches de programmes, des les corriger car ils ne compilent pas. Bref, on retrousse ses manches et on y va. Et on ne soupire pas.

Activité 1 - Le paradoxe des anniversaires

L'objectif de cette première activité est de visualiser l'évolution de la probabilité qu'au moins deux personnes d'un groupe de n personnes fêtent leur anniversaire le même jour, en fonction du nombre d'individus dans le groupe.
On suppose, pour simplifier, qu'une année n'est constituée que de 365 jours et que les naissances sont réparties uniformément tout au long de l'année. On s'intéresse à la date mais pas à l'année (exemple: 3 Janvier).
Déterminer graphiquement avec Python à partir de quelle valeur n, la probabilité qu'au moins deux personnes soient nées le même jour dépasse 1/2. Même question avec 99%.

Activité 2 - Le code de César

Jules César, général et stratège romain, a été (à ce qu'il semble) le premier militaire officiel à chiffrer ses messages. Sa méthode était assez simple : il décalait les lettres de 3 rangs dans l'alphabet.
On propose alors l'écriture d'une fonction de codage. Pour que ce soit plus chouette, la fonction devra prendre en argument le décalage (vers la droite ou vers la gauche - on aura en tête qu'en décalant `'z'` de 1 vers la droite, on retombe sur `'a'` - et la valeur de celui-ci) mais qui ne décale qu'une lettre sur deux (en commençant toujours par la deuxième de chaque mot) et le mot que l'on veut coder.

☞ Par exemple, la commande `codage(2, 'amicalement')` devra renvoyer `'aoieaneoept'`.

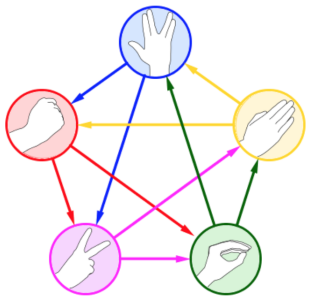
Activité 3 - Plus grande somme d'une sous-liste

On considère une liste `L` d'entiers. On veut écrire un programme la plus grande valeur possible de la somme des termes consécutifs d'une sous-liste de L.

☞ Par exemple, la commande `recherche_s_liste([2, -2, -3, 2, 3, -4, 2, 3, -1])` renverra 6.

Activité 4 - Graphes *Shifumi*

Inutile d'en rappeler les de *Shifumi* (ou Pierre-Feuille-Ciseaux). Ce jeu bien connu connait aussi des variantes, comme celle (présentée ci-après) popularisée par la série télévisée *The Big Bang Theory*.



Un *graphe de Shifumi* est un graphe censé représenté les règles d'une confrontation. Les sommets sont les *figures* possible. Il y a un arc sortant d'un sommet vers un autre si la figure du sommet dont on sort l'emporte sur celle du somment où on entre.

(1) Représenter graphiquement puis sous Python le graphe du *shifumi* classique.
(2) Représenter le graphe d'un *shifumi* équilibré à 5 sommets (Pierre, Feuille, Ciseaux, Lézard, Spock). Le représenter en Python.
(3) Écrire une fonction Python d'en tête `def shifumi(G, nb_manches) :` qui permet un duel entre l'utilisateur et l'ordinateur (dont les figures sont choisies aléatoirement de manière équiprobable en) en `nb_manches` confrontations successives (selon les règles représentées par le grpahe `G`) et qui affiche le score final.
(4) Soit $n \geq 3$ un entier impair. Écrire une fonction d'en-tête `graphe_shifumi(n):` qui renvoie un graphe de Shifumi équilibré à n sommets.

Activité 5 - Le truel

Trois gentlemen, Mr. White, Mr. Grey et Mr. Black se retrouvent opposés dans un *truel*, où chaque adversaire tire à son tour jusqu'à ce qu'il n'en reste qu'un.

Les trois hommes n'ont pas les mêmes niveaux de tir; Mr. Black fait mouche à chaque tir alors que Mr. Grey ne touche que deux fois sur trois. Quant à Mr. White, il ne réussit son tir qu'une fois sur trois. Les hommes sont des gentlemen, ils laissent donc Mr. White tirer en premier, puis Mr. Grey et enfin Mr. Black.

Mr. Black visera toujours, tant que celui-ci est vivant, Mr. Grey, et Mr. Grey essaiera également toujours de toucher Mr. Black. Mr. White en revanche se demande s'il doit, commencer par faire tomber Mr. Black ou Mr. Grey voire tirer en l'air et les laisser s'entretuer, donnant ainsi lieu à trois stratégies, numérotées 1, 2 et 3.

Duel between Onegin and Lenski, Ilya Repin, 1899. Pushkin Museum, Moscou. Domaine public.

Comparer les trois stratégies avec Python, *via* simulation d'un grand nombre de duels pour chacune des trois stratégie et émettre une conjecture sur celle la plus intéressante à suivre pour Mr. White.

On pourra commencer à écrire des fonctions qui simulent des duels entre Mr. White et Mr. Grey avec dans chaque cas un premier tireur différent.

Activité 6 - La ruine du joueur

José se rend au casino *Les requins de la côte* avec s euros en poche ($s \in \mathbb{N}^\star$) et l'envie de faire fortune. Après s'être vêtu de ses habits de lumière, il s'installe à une table où, à chaque partie, il gagne avec probabilité $p \in]0;1[$ un euro et perd avec probabilité $q = 1 - p$ un euro.

On note N la somme dont dispose le casino (on peut raisonnablement supposer que $s < N$). José décide qu'il arrêtera de jouer s'il devient ruiné (c'est à dire lorsque sa fortune tombe à 0) ou lorsque ce sera le cas pour le casino. On admet (même si on pourrait le montrer) que le jeu s'arrête à un moment presque sûrement.

(1) Écrire une première fonction d'en-tête `def casino(s, N, p):` qui à chaque appel renvoie le graphe de l'évolution de la fortune de José jusqu'à l'arrête du processus.
(2) On introduit la variable aléatoire T qui prend la valeur 1 si José est ruiné et 0 si le casino est ruiné. Établir une conjecture quant à la loi de T.

Activité 7 - Théorème de Zeckendorf

On rappelle que la suite de Fibonacci (F_n) est la suite de nombre réels définis par $\begin{cases} F_0 &= 0 \\ F_1 &= 1 \\ F_{n+2} &= F_{n+1} + F_n \end{cases}$

Les premiers termes de la suite sont $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55...$

Théorème

Le **Théorème de Zeckendorf** dit la chose suivante.
Pour tout entier non nul n, il existe un unique entier k et un unique $k-$uplet d'entiers $(c_1, .., c_k)$ vérifiant

- $c_1 \geq 2$
- $c_i + 1 < c_{i+1}$

tels que

$$n = \sum_{i=1}^{k} F_{c_i}$$

Cette décomposition de n en somme de nombres de la suite de Fibonacci est appelée *décomposition de Zeckendorf de n.*

Par exemple $17 = 13 + 3 + 1 = F_7 + F_4 + F_2$ donc $k = 3$ et $(c_1, c_2, c_3) = (2, 4, 7)$.
Le but de cet exercice est d'écrire un programme qui renvoie cette décomposition pour un nombre n pris en argument.

(1) Écrire une fonction `def fibo(k):` qui renvoie le terme F_k de la suite (F_n).

(2) Écrire une fonction `def tri(L):` qui renvoie la liste des termes de `L` triés par ordre croissant.

(3) Écrire une fonction `def recherche(x, L):` qui prend en argument un réel x et une liste `L` (déjà triée dans l'ordre croissant, de première terme inférieur ou égal à x et de dernier terme strictement supérieur à x) et qui renvoie le plus grand élément de la liste inférieur ou égal à x.

(4) Écrire une fonction `def Zeckendorf(n):` qui renvoie la décomposition de Zeckendorf de n. *On pourra commencer par écrire la liste des nombres de Fibonacci inférieurs à n (ainsi que le premier strictement supérieur) et faire une boucle descendante sur cette liste.*