Prise en main de Python - 2ème partie

Travaux pratiques

Voici une liste d'exercices. Pour chacun d'entre eux, un programme doit être réalisé en Python. Le niveau des exercices ne dépasse pas celui du lycée, mais cela ne veut pas dire qu'ils sont tous directement adaptés aux élèves.

Exercice 1

Écrire un programme qui affiche toutes les triplets d'entiers strictement positifs (x, y, z) tels que :

$$xyz + xy + yz + zx + x + y + z + 1 = 244$$

Exercice 2

Écrire un programme qui affiche les valeurs de la fonction f définie par $f(x) = x^2 - 3x + 7$ pour x variant de -1 à 1 avec un pas de $0,2$.

Exercice 3

Écrire un programme qui demande un entier n et qui affiche le terme correspondant de la suite définie par $u_{n+1} = 2u_n - 5n + 1$.

Exercice 4

Écrire un programme qui demande un entier n et qui affiche le terme correspondant de la suite définie par $u_n = \sum_{k=0}^{n} \frac{1}{k^2}$.

Exercice 5

Écrire un programme qui demande un entier A et qui affiche le plus petit entier n tel que $n^2 + 3n >$ A.

Exercice 6

Écrire un programme qui demande un réel ε et qui affiche le plus petit entier n tel que $4 + \frac{1}{n^2+1} \in]4 - \varepsilon; 4 + \varepsilon[$.

Exercice 7

Soit f la fonction définie sur I $= [2; 3]$ par $f(x) = 2\sin(x) - x + 1$.
Cette fonction est continue et strictement décroissante sur I. De plus $g(2) > 0$ et $g(3) < 0$. L'équation $g(x) = 0$ admet donc une unique solution α dans I.
Écrire un programme qui demande un réel ε et qui met en œuvre l'algorithme de dichotomie pour déterminer un encadrement de α d'amplitude inférieure à ε.

Exercice 8

Écrire un programme qui demande les coordonnées de 3 points et qui teste leur alignement.

Exercice 9

Écrire un programme qui demande un entier n, qui simule n lancer d'une pièce de monnaie et qui affiche la fréquence d'apparition des événements "Pile" et "Face".
Indication : pour obtenir un entier aléatoire compris au sens large entre deux entiers a et b, on utilise `randint(a,b)`.

Exercice 10

Un expérience aléatoire consiste à lancer deux dés cubiques bien équilibrés dont les faces sont numérotées de 1 à 6.
Écrire un programme qui calcule une valeur approchée de la probabilité d'obtenir 7 pour la somme des deux dés.

Exercice 11

Marche aléatoire : sur un axe gradué par les entiers relatifs, un pion se trouve initialement à l'origine.
On lance une pièce : si on obtient Pile, l'abscisse du pion est augmentée de 1 ; si on obtient Face, elle est diminuée de 1.
On note D_n l'événement : "le pion est revenu à l'origine après n déplacements".

1. Écrire un programme qui simule n déplacements et qui affiche l'abscisse finale du pion.
2. Dans cette question, le pion effectue toujours un trajet de 6 déplacements. Écrire un programme qui simule n trajets de 6 déplacements et qui affiche la fréquence de réalisation de l'événement D_6 (valeur approchée de $P(D_6)$).

Exercice 12

L'objectif de cet exercice est de calculer une valeur approchée de l'intégrale $I = \int_0^1 \frac{4}{1+x^2}\,dx$ par la méthode des rectangles.

Écrire un programme qui demande un entier n et qui calcule cette valeur approchée en utilisant n rectangles.

Exercice 13

L'objectif de cet exercice est de calculer une valeur approchée de l'intégrale $I = \int_0^1 \frac{4}{1+x^2}\,dx$ par la méthode des trapèzes.

Écrire un programme qui demande un entier n et qui calcule cette valeur approchée en utilisant n trapèzes.

Exercice 14

L'objectif de cet exercice est de calculer une valeur approchée de l'intégrale $I = \int_0^1 \frac{4}{1+x^2}\,dx$ par la méthode de Simpson.

Écrire un programme qui demande un entier n et qui calcule cette valeur approchée en utilisant n paraboles.

Exercice 15

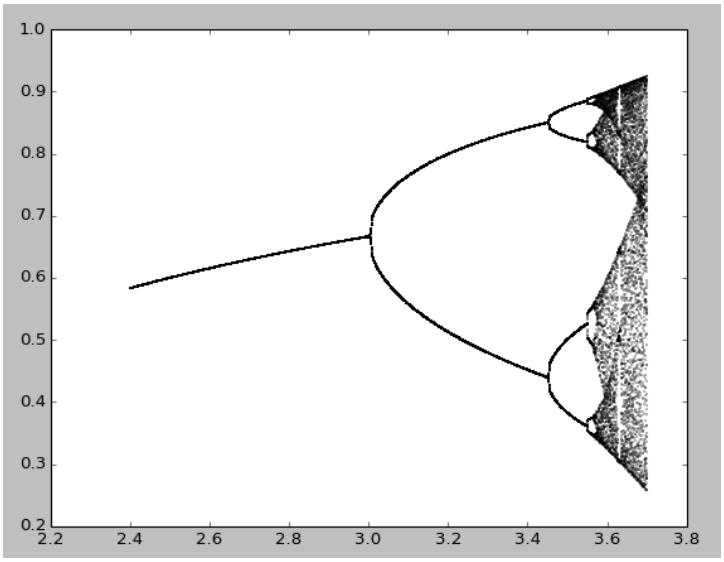
L'objectif de cet exercice est de calculer une valeur approchée de l'intégrale $I = \int_0^1 \frac{4}{1+x^2}\,dx$ par la méthode de Monte-Carlo.

Écrire un programme qui demande un entier n et qui calcule cette valeur approchée en utilisant n points.
*Indication : pour obtenir un **réel** aléatoire dans un intervale $[a;b]$ (loi uniforme), on utilise* `uniform(a,b)`.

Exercice 16

On appelle suite logistique la suite (x_n) définie par $x_0 = 0,5$ et, pour tout entier naturel n, $x_{n+1} = rx_n(1-x_n)$, où r est un réel.
Cette suite est liée à la modélisation de l'évolution d'une population réalisée par le mathématicien belge Pierre-François Verhulst (1804-1849).
Le comportement asymptotique de cette suite dépend fortement de la valeur du réel r : elle peut être convergente, avoir deux points d'attraction, quatre, etc., ou bien avoir un comportement chaotique.
La figure ci-dessous est réalisée comme suit :

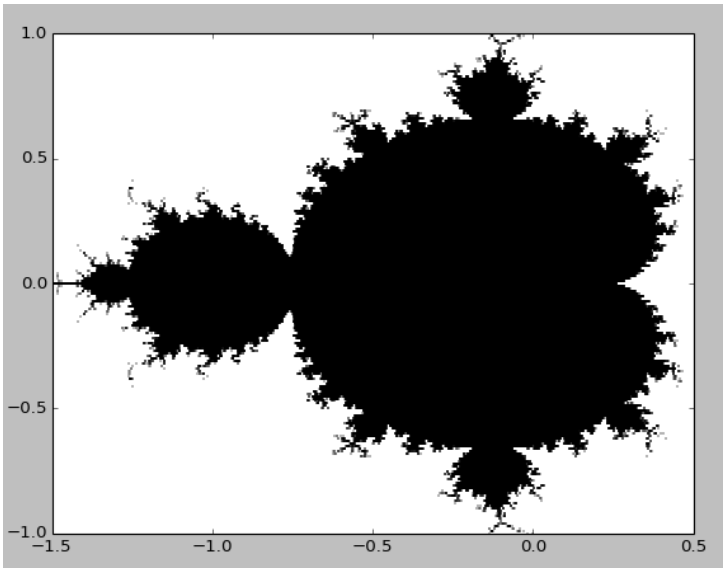
- Pour tous les entiers r compris entre 2,4 et 3,7, avec un pas de 0,01, on a placé les points de coordonnées (r, x_n) pour n variant de 50 à 100.



On visualise ainsi, pour chaque valeur de r, le comportement asymptotique de la suite (x_n).
Écrire un programme qui produit cette figure.
Indication : on ne peut pas ajouter les points au graphique un par un car, d'une part, l'affichage ne se fera qu'à la fin du programme et, d'autre part, cela induirait une utilisation déraisonnable de la mémoire vive de l'ordinateur et un temps d'exécution tout aussi déraisonnable. Pour afficher les points, on construit donc deux listes : une liste `liste_x` *contenant les abscisses, et une liste* `liste_y` *contenant les ordonnées, puis on utilise l'instruction* `repere.plot(liste_x, liste_y, 'k.', markersize=1)`*.Les deux derniers paramètres sont optionnels et ne sont là que pour l'esthétique.* `'k.'` *indique de représenter les points en noir et sous forme de point.* `'markersize=1'` *indique la taille des points. Pour ajouter un nombre* `a` *à la fin d'une liste, on tape par exemple* `liste_x = liste_x + [a]` *ou au choix* `liste_x.append(a)`*. Le programme doit se terminer par l'instruction* `repere.show()` *qui affichera l'image produite.*

Exercice 17

Pour tout nombre complexe c, on s'intéresse à la suite de nombre complexes (z_n) définie par $z_0 = 0$ et, pour tout entier naturel n, $z_{n+1} = z_n^2 + c$.
Le célèbre ensemble de Mandelbrot (Benoît Mandelbrot, mathématicien franco-américain, 1924-2010) est l'ensemble des points C, d'affixe c, du plan complexe tels que la suite (z_n) ne diverge pas vers l'infini (en module).



Écrire un programme qui dessine l'ensemble de Mandelbrot. R$e(c)$ doit varier entre $-1,5$ et $0,5$ avec un pas de $0,01$ et I$m(c)$ doit varier entre -1 et 1 avec un pas de $0,01$.
Pour déterminer si la suite diverge vers l'infini ou non, on calculera z_{30} et on considéra que (z_n) diverge vers l'infini si $z_{30} > 100$ (dans ce cas, le point C n'est pas tracé).
Indication : on peut, mais ce n'est pas une obligation, calculer directement avec les nombres complexes en Python. L'instruction `'z=complex(a,b)'` *définit le nombre complexe* $a + ib$.

Pour continuer ...

Se référer au manuel de Edupython : il contient beaucoup d'exemples très intéressants.