TP Informatique 3 : Nombres premiers, travail sur les listes

1 Nombres premiers

On souhaite construire une liste contenant l'ensemble des nombres premiers strictements inférieurs à n. Dans un premier temps, on prendra $n = 100$.

1.1 Première méthode

Un nombre k est premier si et seulement si aucun entier p vérifiant $2 \leq p < k$ n'est un diviseur de k.

1. Utiliser cette propriété pour écrire un programme construisant une liste `P` des nombres premiers inférieurs à n.

2. Combien y a-t-il de nombres premiers inférieurs à 100 ?

1.2 Amélioration

En réalité, il est inutile de tester la divisibilité par tous les entiers inférieurs à k. En effet, il est aisé de montrer qu'un nombre k est premier si et seulement si aucun nombre *premier* p avec $p < k$ n'est un diviseur de k.

3. Modifier le programme précédent pour construire plus rapidement la liste `P`.

1.3 Crible d'Érathostène

En fait, on peut encore gagner un peu de temps. Plutôt que de tester la divisibilité des entiers k par les premiers p, on peut, dès que l'on trouve un premier p, affirmer que ses multiples ne sont *pas* des nombres premiers.

Pour ce faire, nous allons donc avoir besoin d'une grande liste, que l'on désignera par le nom `L`, qui contiendra des booléens, `True` ou `False`, indiquant à tout instant de l'algorithme si chacun des entiers entre 0 et $n-1$ sont possiblement premiers (si `L[k]` est égal à `False`, alors on sait avec certitude que k n'est pas premier).

Dans un premier temps, on suppose que tous les entiers sont candidats à être des nombres premiers, autrement dit la liste `L` ne contient que des `True`.

4. Écrire un programme qui construit une liste `L` contenant n booléens, tous initialisés à `True`, et une liste vide `P` destinée à recevoir les nombres premiers.

On peut dès à présent affirmer que 0 et 1 ne sont pas des nombres premiers. Il faut refléter ça dans notre liste `L` !

5. Ajouter une ou plusieurs lignes au programme pour que les deux premiers éléments de `L` contiennent la valeur `False`, indiquant que 0 et 1 ne peuvent pas être premiers.

Ensuite, on applique la méthode dite du *crible d'Ératosthène* :

- on considère tous les éléments de la liste `L` un par un, dans l'ordre ;
- si l'élément est égal à `False`, on ne fait rien ;
- si l'élément est égal à `True`, alors l'entier correspondant à l'index de l'élément considéré est un nombre premier ; on l'ajoute alors à la liste `P`, puis on place `False` dans la liste `L` pour tous les index multiples de l'index de l'élément

Illustrons cet algorithme. Initialement, la liste (pour $n = 40$) peut être vue comme ceci (les chiffres dans les cases correspondent aux index, pour simplifier la lecture, les cases claires correspondent à la valeur `True`, les cases sombres à la valeur `False`) :

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

0 n'est donc pas premier, 1 non plus, mais 2 est premier. On « élimine » donc ses multiples en plaçant un `False` dans toutes les cases dont les index sont des multiples de 2 :

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

3 est premier, puisque la valeur d'index 3 est `True`, on élimine ses multiples :

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

4 n'est pas premier (`L[4]=False`), mais 5 l'est, même chose :

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

Et ainsi de suite jusqu'à ce que l'on ait parcouru toute la liste.

6. Écrire un programme implémentant le crible d'Ératosthène, puis l'utiliser pour construire la liste P des entiers strictement inférieurs à n.

7. Justifier que, lorsque l'on élimine les multiples de k, il suffit en fait de débuter à k^2, et que si $k > 2$, seuls les multiples impairs de k doivent être éliminés. Et donc qu'il est inutile de chercher à supprimer les multiples de k si $k > \sqrt{n}$.

8. Comment peut-on obtenir l'ensemble des entiers de la forme $(2i+1)k$ (en supposant k impair) où i est un entier et vérifiant $k^2 \leq (2i+1)k < n$?

9. En déduire une amélioration de l'algorithme (on traitera le cas $k = 2$ à part, puis les entiers impairs k inférieurs ou égaux à $\sqrt{n}$, et enfin les entiers impairs restants).

On souhaite visualiser la liste L afin de voir comment sont répartis les nombres premiers inférieurs à n. Pour ce faire, on peut utiliser la fonction `matshow` du module `matplotlib.pyplot`.

Il faut cependant, préalablement, transformer la liste L en une liste L2 contenant 10 listes de 10 éléments, pour obtenir une structure en deux dimensions comme le tableau qui a servi d'illustration précédemment, qui sera plus simple à afficher et à lire.

10. Construire la liste L2 selon les spécifications proposées.

11. Afficher le résultat avec les lignes suivantes :

```
from matplotlib.pyplot import matshow, show
matshow(L2, cmap='gray')
show()
```

Les différents entiers sont ordonnés en ligne, comme sur le tableau de la page précédente. Les nombres premiers apparaissent en blanc, les autres en noir. Dans le cas présent, l'index des lignes indique la dizaine, celui de la colonne les unités.

1.4 Aller plus loin

12. Obtenir une image similaire à la précédente pour dix mille entiers (on construira cent lignes de cent entiers)

13. Déterminer le nombre de nombres premiers inférieurs à un million.

14. Quel est le cent-millième nombre premier ?

15. Tracer, à l'aide de la fonction `plot` du module `matplotlib`, la courbe du nombre de nombres premiers inférieurs à k en fonction de k pour $k \leq 10^6$.

2 Quelques listes itérées particulières

Liste de Cantor

Les premiers termes ($n = 0$ à $n = 3$) de la liste de Cantor sont les suivants :

```
[ 1 ]
[ 1, 0, 1 ]
[ 1, 0, 1, 0, 0, 0, 1, 0, 1 ]
[ 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
  0, 1, 0, 1 ]
```

1. Étudier comment on passe d'un terme au suivant, et écrire un programme qui construit la liste de Cantor à un ordre n choisin puis tracer l'histogramme correspondant. On rappelle (voir le TP précédent) qu'il est possible de tracer un histogramme en utilisant la commande `bar` du module `matplotlib.pyplot`, par exemple ainsi :

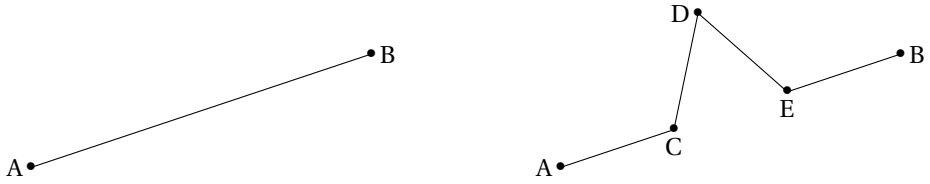
```
bar(range(len(L)), L, width=1.0)
```

Liste de Koch

Dans ce second cas, on part de deux listes, notées X et Y, qui contiennent respectivement les abscisses et les ordonnées de points du plan. On prendra comme valeurs initiales :

```
X = [ 0, 1 ]
Y = [ 0, 0 ]
```

On souhaite créer la fractale dite de Koch en créant, itérativement, des listes de coordonnées de plus en plus longues. L'itération se fait en ajoutant, entre chaque paire de points consécutifs dans les listes (que l'on note par exemple A et B), trois points supplémentaires, notés C, D, E, en respectant les règles géométriques suivantes :



Les quatres segments de la figure de droite sont de même longueur, et A, C, E et B sont alignés.

2. Construire les listes X et Y à l'ordre n (qui doivent contenir chacune $4^n + 1$ valeurs), et afficher le résultat obtenu avec la commande `plot(X, Y)` qui permet de tracer une ligne brisée reliant chacun des points.

3 Nombres chanceux

Les nombres chanceux sont des entiers particuliers, dont la construction est similaire aux nombres premiers avec le crible d'Ératosthène.

On commence par construire la liste des entiers des nombres impairs inférieurs à n.

1	3	5	7	9	11	13	15	17	19
21	23	25	27	29	31	33	35	37	39
41	43	45	47	49	51	53	55	57	59
61	63	65	67	69	71	73	75	77	79

Le deuxième nombre de cette liste est 3. On conserve donc les deux premiers nombres, et on retire le troisième (soit 5). On conserve deux nombres supplémentaires, et on retire le suivant (11). Et ainsi de suite.

1	3	5	7	9	11	13	15	17	19
21	23	25	27	29	31	33	35	37	39
41	43	45	47	49	51	53	55	57	59
61	63	65	67	69	71	73	75	77	79

Le troisième nombre restant est 7, on va donc conserver les six premiers nombres, éliminer le septième, et ainsi de suite.

1	3	5	7	9	11	13	15	17	19
21	23	25	27	29	31	33	35	37	39
41	43	45	47	49	51	53	55	57	59
61	63	65	67	69	71	73	75	77	79

On continue avec le quatrième nombre restant, et on poursuit jusqu'à épuisement des nombres dans la liste. Les nombres restants sont appelés nombres *chanceux*

1. Déterminer la somme des cent premiers nombres chanceux.

2. Combien y a-t-il de nombres premiers chanceux inférieurs à dix mille ?