TP 4
Arithmétique

SABABADY Kamala et SELVARAJAH Dinusan

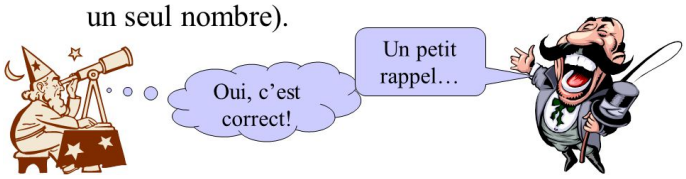
December 14, 2017

1 Partie 1 : *Un nombre est-il premier ?*

Soit a un nombre entier et b un nombre entier non nul. La division euclidienne de a par b est l'opération qui associe à a et à b deux nombres entiers q et r définie par : a = b * q + r avec r < b. En Python, le quotient et le reste de la division euclidienne d'un entier par un entier s'obtient de la façon suivante: pour le quotient on utilise a/b et pour le reste on cherche son modulo ($a\%b$).

Définition des nombres premiers

- Un nombre premier est un nombre qui ne peut donner un nombre entier comme résultat, quand on le divise, que s'il est divisé par 1 ou par lui-même. 1 n'est pas un nombre premier (il n'est divisible que par un seul nombre).



Prenons les exemples données dans la consigne et vérifions s'ils sont premiers ou pas :
1001 = 11 * 91 donc il n'est pas premier.
2017 est un nombre premier.
3001 est un nombre premier.
49999 est un nombre premier.
89999 = 7 * 12857 donc il n'est pas premier.

Nous avons défini une fonction python *is_prime* qui prend un argument un entier n et qui renvoi *true* si n est premier, *false* sinon.

Les nombre de Fermat F_0, F_1, F_2, F_3, F_4 sont premier mais F_5 n'est pas premier.

2 Partie 2 : *Crible d'Erasthotène, distribution des nombres premiers.*

"ÉRATOSTHÈNE de Cyrène est un astronome, géographe et mathématicien, nommé à la tête de la bibliothèque d'Alexandrie, il est resté célèbre pour son crible et pour avoir le premier mesuré le méridien terrestre."

On cherche des nombres premiers à l'aide du Crible d'Erathostène.
Voici les entiers naturels de 1 à 200:

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	40
41	42	**43**	44	45	46	**47**	48	49	50
51	52	**53**	54	55	56	57	58	**59**	60
61	62	63	64	65	66	**67**	68	69	70
71	72	**73**	74	75	76	77	78	**79**	80
81	82	**83**	84	85	86	87	88	**89**	90
91	92	93	94	95	96	**97**	98	99	100
101	102	**103**	104	105	106	**107**	108	**109**	110
111	112	**113**	114	115	116	117	118	119	120
121	122	123	124	125	126	**127**	128	129	130
131	132	133	134	135	136	**137**	138	**139**	140
141	142	143	144	145	146	147	148	**149**	150
151	152	153	154	155	156	**157**	158	159	160
161	162	**163**	164	165	166	**167**	168	169	170
171	172	**173**	174	175	176	177	178	**179**	180
181	182	183	184	185	186	187	188	189	190
191	192	**193**	194	195	196	**197**	198	**199**	200

Nous avons obtenu ce tableau en éliminant :

- 1 qui n'est pas un nombre premier donc **la case** est coloriée en rouge.
- Tous les multiples de 2 excepté **2**
- Tous les multiples de 3 excepté **3**
- Tous les **multiples de 5** excepté **5**
- Tous les multiples de 7
- Tous les multiples de 11
- Tous les multiples de 13

Pour les nombres premiers inférieurs à 200 on a tous les nombres restants en gras.

La fonction `primes` qui prend en argument un entier n et qui renvoie la liste de tous les premiers inférieurs à n est créée en faisant le test de chaque entier inférieur à cet entier n avec la fonction *is_prime* qu'on a vu précédemment et voir si l'entier est inférieur , si c'est le cas on l'enregistre dans une liste .

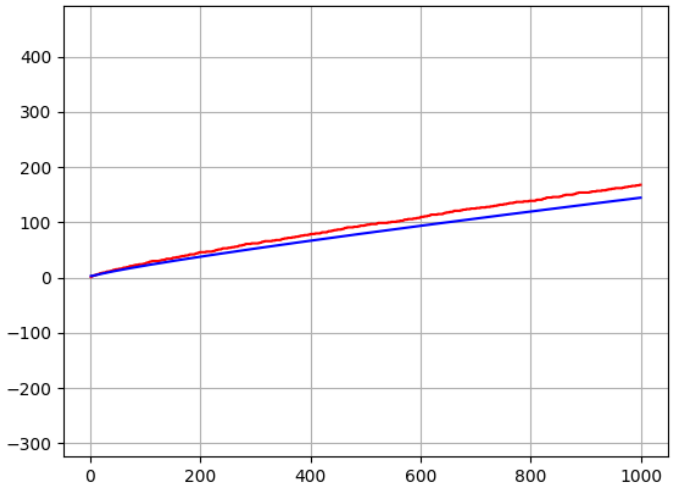
On a ensuite créée cette liste pour tous les premiers inférieurs à 1000 avec la fonction `primes` et nous l'avons enregistré dans un fichier texte primes.txt.

Tous les entiers premiers inferieurs a 1000:

[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]

On a défini une fonction $\pi(n)$ le nombre d'entiers premiers inférieurs à n , pour la définir on a appliqué la fonction `primes` à n et puis on calcule la longueur de la liste que l'on retourne.

Voici le graphique avec $\pi(n)$ en fonction de n pour n variant de 2 à 1000 (courbe rouge), dans ce même graphique nous avons rajouté la fonction $g(n) = \frac{n}{\log n}$ (courbe bleu):



On observe que la fonction $g(n)$ est proche de la courbe $\pi(n)$ donc des nombres premiers inférieur à N c'est-à-dire ici 1000.

Voici la table suivante des deux fonctions $\pi(n)$ et $g(n)$ pour n allant de 10 à 1000000 :

n	$\pi(n)$	$\frac{n}{\log n}$
10^1	4	4.34294
10^2	25	21.7147
10^3	168	144.765
10^4	1229	1085.74
10^5	9592	8685.89
10^6	78498	72382.4

On remarque que plus n est grand plus l'approximation est précise comme l'on observe sur le graphique.

3 Partie 3 : *Factorisation d'un entier en premiers.*

Le théorème fondamentale de l'arithmétique dit que tout entier positif peut être écrit sous forme d'un produit de nombres premiers, et cette décomposition est unique, à l'ordre près des facteurs.

La décomposition en facteurs premiers de 924 est : $924 = 2 * 2 * 3 * 7 * 11$

On a défini deux fonctions de deux manières différentes sur Python `factors` et `factors1`, les deux fonctions donnent le même résultat. Nous avons testé avec les nombres 924 et 60. Par exemple pour la fonction `factors1` qui prend en argument un entier n et qui renvoi la liste dans l'ordre croissant des facteurs premier de n , on a réalisé des divisions successives de n par les nombres premiers qui lui sont inférieurs . A chaque fois que n est divisible par un nombre premier on l'enregistre dans la liste puis on passe au suivant. Nous avons testé cette fonction pour 60 et on a obtenu $[2, 2, 3, 5]$

4 Partie 4 : *PGCD de deux entiers, identité de Bézout, algorithme d'Euclide.*

Le PGCD est le plus grand commun diviseur de 2 entiers. Soit a et b deux entiers naturels , le PGCD de a et b est le plus grand des diviseurs communs de a et b.

En utilisant la décomposition en facteurs premiers on a pour le pgcd de a = 4864 et b = 3458 :

$a = 4864 = 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 19 = 28 * 19$
$b = 3458 = 2 * 7 * 13 * 19$
$pgcd(4864; 3458) = 2 * 19 = 38$

L'énoncé de L'identité de Bézout est la suivante :
"Soient a et b deux entiers relatifs et d leur PGCD alors il existe deux entiers u et v tels que $a \cdot u + b \cdot v = d$."
A l'aide de l'algorithme d'Euclide étendu , on calcule le pgcd de 4864 et 3458 :

$4867 = 1 * 3458 + 1406$
$3458 = 2 * 1406 + 646$
$1406 = 2 * 646 + 114$
$646 = 5 * 114 + 76$
$114 = 1 * 76 + 38$
$76 = 2 * 38 + 0$

Donc le $pgcd(4864, 3458) = 38$.

Pour obtenir les coefficient de Bézout, il suffit de remonter l'algorithme d'Euclide à l'envers :

$1406 = 4864 - 3458$
$646 = 3458 - 2 * 1406$

$114 = 1406 - 2 * 646$
$76 = 646 - 5 * 114$
$38 = 114 - 76$

Puis on remplace successivement :
$38 = 114$–76
$38 = 114$–$(646$–$5 * 114)$
$38 = -646 + 6 * 114$
$38 = -646 + 6 * (1406$–$2 * 646)$
$38 = 6 * 1406$–$13 * 646$
$38 = 6 * 1406$–$13 * (3458$–$2 * 1406)$
$38 = 32 * 1406$–$13 * 3458$
$38 = 32 * (4864$–$3458)$–$13 * 3458$
$38 = 32 * 4864$–$45 * 3458$
On trouve ainsi les coefficients de Bézout x et y avec $x = 32$ et $y = -45$.

Nous avons défini une fonction `euclide` qui prend en arguments deux entiers a, b et qui renvoie x, y, d, où d est le pgcd de a, b et x, y les coefficients de Bézout. On le teste pour 4864 et 3458 et on obtient bien $d = 38$, $u = 32$ et $v = -45$