Partie n°3 : Le petit théorème de Fermat

Ce théorème a été souvent utilisé pour montrer qu'un nombre n'est pas premier, voici comment. Ce théorème s'énonce en disant que, si p est un nombre premier et $a \geq 2$ un nombre entier non divisible par p, alors $a^{p-1}-1$ est un multiple de p.

a) Le nombre 7 est premier donc quelque soit l'entier $a \geq 2$ non divisible par 7, on doit avoir a^6-1 divisible par 7. Vérifier cela, en testant la divisibilité de a^6-1 par 7 pour toutes les valeurs inférieures à 14 qui ne soient pas dans la table de 7. Recommencer le même travail pour d'autres nombres premiers inférieurs à 100.

Commençons par 7. Nous allons générer la liste des valeurs de la forme a^6-1 pour tout $a \geq 2$ non divisible par 7. Le programme que nous venons d'écrire pour p=7 reste valable pour p=11 ou pour toute autre valeur de p premier.

```
p=7
L=list(range(2,2*p) )
L0=list(a for a in L if a%p!=0)
print(L0)                    #  liste des entiers a non multiples de p
L1=[a**(p-1)-1 for a in L0]
print(L1)                    #  liste des entiers b =2**(p-1)-1
L2=[b%p for b in L1]
print(L2)                    #  que des restes nuls d'après le théorème
L3=[b//p for b in L1]
print(L3)                    #  les quotients de b par p
print('a \t a**{}-1 \t division par {}'.format(p-1,p))
for rang,a in enumerate(L0) :
    puissance=str(L1[rang])  #  pour un affichage plus propre
    while len(puissance)<10 :# (toutes les colonnes alignées)
        puissance+=' '
    print(a,'\t',puissance,'\t ',str(L3[rang])+'\u00D7'+str(p)+"+0")
```

```
[2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13]
[63, 728, 4095, 15624, 46655, 262143, 531440, 999999, 1771560, 2985983, 4826808
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[9, 104, 585, 2232, 6665, 37449, 75920, 142857, 253080, 426569, 689544]
a        a**6-1          division par 7
2        63               9×7+0
3        728              104×7+0
4        4095             585×7+0
5        15624            2232×7+0
6        46655            6665×7+0
8        262143           37449×7+0
9        531440           75920×7+0
10       999999           142857×7+0
11       1771560          253080×7+0
12       2985983          426569×7+0
13       4826808          689544×7+0
```

Nous donnons ci-dessous les sorties « propres » uniquement pour les premières valeurs de p : 2, 3, 5, 7 et 11. On constate que les valeurs de $b=a^{p-1}-1$ sont très rapidement très grandes, mais néanmoins, la propriété énoncée est toujours vraie (cela ne constitue en aucun cas une démonstration. Ce théorème, énoncé en 1637 par Pierre Fermat, a été véritablement démontré par Leibniz et ensuite Euler, un siècle seulement plus tard).

```
a        a**1-1          division par 2
3        2                1×2+0

a        a**2-1          division par 3
2        3                1×3+0
4        15               5×3+0
5        24               8×3+0

a        a**4-1          division par 5
2        15               3×5+0
3        80               16×5+0
4        255              51×5+0
6        1295             259×5+0
7        2400             480×5+0
8        4095             819×5+0
9        6560             1312×5+0
```

```
a        a**10-1         division par 11
2        1023             93×11+0
3        59048            5368×11+0
4        1048575          95325×11+0
5        9765624          887784×11+0
6        60466175         5496925×11+0
7        282475248        25679568×11+0
8        1073741823       97612893×11+0
9        3486784400       316980400×11+0
10       9999999999       909090909×11+0
12       61917364223      5628851293×11+0
13       137858491848     12532590168×11+0
14       289254654975     26295877725×11+0
15       576650390624     52422762784×11+0
16       1099511627775    99955602525×11+0
17       2015993900448    183272172768×11+0
18       3570467226623    324587929693×11+0
19       6131066257800    557369659800×11+0
20       10239999999999   930909090909×11+0
21       16679880978200   1516352816200×11+0
```

b) Pour prouver qu'un nombre n n'est pas premier, il suffit de montrer que $a^{n-1}-1$ n'est pas divisible par n (c'est la forme contraposée du théorème) pour au moins une valeur de $a \geq 2$ non divisible par n. Par exemple, 91 n'est pas premier car $2^{90}-1=1237940039285380274899124223$ n'est pas divisible par 91 (le reste est 63). Et, en effet, $91=7\times13$, 91 est un nombre composé (on aurait pu le savoir bien plus facilement, sans utiliser ce théorème).

Mettre au point un algorithme qui teste, avec cette méthode la non-primalité d'un entier n. On pourra par exemple tenter de prendre a=2 et montrer que $b=2^{p-1}-1$ n'est pas divisible par n (utiliser la fonction % : reste de la division euclidienne), si ce n'est pas le cas, on tente la division par le nombre premier suivant, etc. Tester, par exemple, la non-primalité de $10^3+1=1001$ puis de $10^7+1=10000001$.

Nous donnons, pour simplifier, une liste des premiers nombres premiers. Nous examinons ensuite la divisibilité de b par p (en fait on teste si le reste de la division de a^{p-1} par p donne 1, ce qui revient au même). Dans le cas d'une divisibilité, le nombre est pseudo-premier en base a (voir plus loin) ; il faut alors continuer l'investigation en prenant le nombre premier suivant. C'est dans le cas contraire, que la contraposée est efficace : on sait tout de suite que le nombre n'est pas premier. Dans ce cas, rien ne sert de

continuer les calculs (qui sont très longs pour des nombres *p* très grands) mais on peut chercher le premier nombre premier qui divise *p* (c'est ce que nous avons fait pour les nombres premiers de notre liste, qui va jusqu'à 101). Pour cela notre liste risque de ne pas être assez longue... Cela pourrait être intéressant de coupler cet algorithme avec celui de la partie 2 qui prolonge la liste des nombres premiers.

```
p=int(input('Entrer un nombre entier impair : '))#tester avec 1001, 10403 puis 10000001
Lprem=[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]#quelques nombres premiers
for a in Lprem :
    if (a**(p-1))%p==1 : #  p est-il premier?
        print('{} est pseudo-premier en base {}'.format(p,a))
    elif p%a==0 :
        print('{} est composé de {} et de {}'.format(p,a,p//a))
        break
    else :
        print('{} est composé, mais pas de {}'.format(p,a))
```

```
Entrer un nombre entier impair : 341
341 est pseudo-premier en base 2
341 est composé, mais pas de 3
341 est composé, mais pas de 5
341 est composé, mais pas de 7
341 est composé de 11 et de 31
```

```
Entrer un nombre entier impair : 1001
1001 est composé, mais pas de 2
1001 est composé, mais pas de 3
1001 est composé, mais pas de 5
1001 est composé de 7 et de 143
```

```
Entrer un nombre entier impair : 10000001
10000001 est composé, mais pas de 2
10000001 est composé, mais pas de 3
10000001 est composé, mais pas de 5
10000001 est composé, mais pas de 7
10000001 est composé de 11 et de 909091
```

Les calculs sont longs pour *p*=10000001 car les nombres calculés sont très grands. Nous n'écrirons pas en entier ici le nombre $2^{1000000}$ (il est constitué de 301 030 chiffres, à raison de 60 lignes par pages et de 100 chiffres par ligne, il faudrait 50 pages pour l'écrire) mais, pour en avoir une idée, donnons juste 2^{1000} qui est calculé pour tester la non-primalité de 1001 (l'algorithme calcule aussi 3^{1000} et 5^{1000}) ; ce nombre ne s'écrit qu'avec 302 chiffres :
107150860718626732094842504906000181056140481170553360744375038837035105112493612249319837881
569585812759467291755314682518714528569231404359845775746985748039345677748242309854210746050
623711418779541821530464749835819412673987675591655439460770629145711964776865421676604298316
52624386837205668069376.

c) Par contre, on ne peut utiliser directement la réciproque qui est fausse généralement : ce n'est pas parce qu'il existe un entier $a \geq 2$ premier avec l'entier impair *n* tel que $a^{n-1}-1$ soit un multiple de *n* (on écrit cela aussi $a^{n-1} \equiv 1 \,(mod\ n)$, que *n* est un nombre premier... De tels nombres impairs *n* sont appelés *nombres pseudo-premiers de base a* (on précise parfois que ce sont des nombres pseudo-premiers de Fermat car il existe d'autres sortes de nombres pseudo-premiers). Si *a* est égal à 2, et si *a* est quelconque (pas nécessairement premier avec *p*) de tels nombres pseudo-premiers sont appelés *nombres de Poulet*. Si la propriété est vérifiée pour tout entier *a* compris entre 2 et *n*, le nombre est appelé *nombre de Carmichael*.
Retrouver, à l'aide d'un algorithme, les dix premiers nombres de Poulet (la liste commence par 341, 561, 645, 1105 et 1387) et les dix premiers nombres de Carmichael (la liste commence par 561, 1105 et 1729).
Quels sont les premiers nombres pseudo-premiers de base 2, 3, 4, 5, etc. qui soient supérieurs à leur base ? (pour 2, 3 et 4 la réponse est 341, 91 et 15).
Pour les nombres de Poulet, nous avons besoin d'une liste de nombres premiers qui permette d'identifier si un pseudo-premier de base 2 (il passe positivement le test de Fermat pour la base *a*=2, il pourrait donc être premier) est un premier véritable. Nous employons donc la fonction *isPrem*() qui a été définie dans la partie 1. L'algorithme résultant est très simple et ne nécessite pas davantage de commentaires. Les premiers nombres de Poulet sont donc : 341, 561, 645, 1105, 1387, 1729, 1905, 2047, 2465, 2701, 2821, 3277, 4033, 4369, 4371, 4681, 5461, 6601, etc.

```
#recherche des nombres pseudo-premiers de Poulet
Lprem=[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]#quelques nombres premiers
nbrTrouve=0
n_teste=3
isPrem=False
while nbrTrouve<10:
    isPrem=isPremier(n_teste)
    if (2**(n_teste-1))%n_teste==1 and isPrem is False :
        nbrTrouve+=1
        print('{} est le nombre de Poulet n°{}'.format(n_teste,nbrTrouve))
    n_teste+=2
```

```
341 est le nombre de Poulet n°1
561 est le nombre de Poulet n°2
645 est le nombre de Poulet n°3
1105 est le nombre de Poulet n°4
1387 est le nombre de Poulet n°5
1729 est le nombre de Poulet n°6
1905 est le nombre de Poulet n°7
2047 est le nombre de Poulet n°8
2465 est le nombre de Poulet n°9
2701 est le nombre de Poulet n°10
```

On examine, dans cette recherche, tous les nombres impairs à partir de 3 mais il était précisé que la base *a*=2 et le nombre testé ne devait pas nécessairement être premier entre eux. Si l'on devient moins difficile pour les nombres testés et qu'on essaie tous les entiers à partir de 3 (on remplace *n-teste*+=2 par *n-teste*+=1), le résultat est le même : les premiers nombres entiers qui ne sont pas premiers mais qui paraisse

l'être avec la base a=2 selon le test de Fermat, sont les nombres précités.
Pour les nombres de Carmichael, nous procédons presque comme pour les nombres de Poulet. Il y a une boucle supplémentaire dans l'algorithme puisqu'on doit examiner tous les nombres impairs $a \geq 3$ comme des bases potentielles pour le test de Fermat. Nous supprimons de ces bases potentielles les nombres qui ne sont pas premiers avec le nombre testé (d'où l'utilisation de notre fonction *pgcd*() dans cet algorithme). Le mot-clef *continue* permet ici de poursuivre la boucle *for* lorsqu'on tombe sur une base potentielle qui n'est pas première avec le nombre testé. Les premiers nombres de Carmichael sont donc : 561, 1105, 1729, 2465, 2821, 6601, 8911, 10585, 15841, 29341, etc.

```
#recherche des nombres pseudo-premiers de Carmichael
Lprem=[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]#quelques nombres premiers
nbrTrouve=0
n_teste=3
isPrem=False
while nbrTrouve<10:
    isPrem=isPremier(n_teste)
    if isPrem is False :
        isCarmichael=True
        for i in range(2,n_teste) :
            if pgcd(i,n_teste)!=1 : continue
            if (i**(n_teste-1))%n_teste!=1 :
                isCarmichael=False
                break
        if isCarmichael is True :
            nbrTrouve+=1
            print('{} est le nombre de Carmichael n°{}'.format(n_teste,nbrTrouve))
    n_teste+=2
```

```
561 est le nombre de Carmichael n°1
1105 est le nombre de Carmichael n°2
1729 est le nombre de Carmichael n°3
2465 est le nombre de Carmichael n°4
2821 est le nombre de Carmichael n°5
6601 est le nombre de Carmichael n°6
8911 est le nombre de Carmichael n°7
10585 est le nombre de Carmichael n°8
15841 est le nombre de Carmichael n°9
29341 est le nombre de Carmichael n°10
```

On remarque que ces nombres sont tous des nombres de Poulet (c'est une évidence car la condition pour être un nombre de Poulet (passer le test pour la base 2) est contenue dans celle, plus sévère, pour être un nombre de Carmichael). Ainsi, le 10ème nombre de Carmichael est le 40ème nombre de Poulet (voir le tableau ci-dessous).
On a la preuve depuis 1994 qu'il y a une infinité de nombres de Carmichael, et qu'ils sont tous composés d'au moins trois nombres premiers (561=3×11×17, 1105=5×13×17, 1729=7×13×19, etc.). On sait aussi que les produits $(6k+1)(12k+1)(18k+1)$ quand ces trois facteurs sont premiers sont les nombres de Chernick, et

rang Poulet	nombre	rang Carmichael	rang Chernick	Décomposition	rang Poulet	nombre	rang Carmichael	rang Chernick	Décomposition
1	341			11×31	26	12 801			3×17×251
2	561	1		3×11×17	27	13 741			7×13×151
3	645			3×5×43	28	13 747			59×233
4	1 105	2		5×13×17	29	13 981			11×31×41
5	1 387			19×73	30	14 491			43×337
6	1 729	3	1	7×13×19	31	15 709			23×683
7	1 905			3×5×127	32	15 841	9		7×31×73
8	2 047			23×89	33	16 705			5×13×257
9	2 465	4		5×17×29	34	18 705			3×5×29×43
10	2 701			37×73	35	18 721			91×193
11	2 821	5		7×13×31	36	19 951			71×281
12	3 277			29×113	37	23 001			3×11×17×41
13	4 033			37×109	38	23 377			97×241
14	4 369			17×257	39	25 761			3×31×277
15	4 371			3×31×47	40	29 341	10		17×37×61
16	4 681			31×151	41	30 121			7×13×331
17	5 461			43×127	42	30 889			17×23×79
18	6 601	6		7×23×41	43	31 417			89×353
19	7 957			73×109	44	31 609			73×433
20	8 321			53×157	45	31 621			103×307
21	8 481			3×11×257	46	33 153			3×43×257
22	8 911	7		7×19×67	47	34 945			5×29×241
23	10 261			31×331	48	35 333			89×397
24	10 585	8		5×29×73	49	39 865			5×7×17×67
25	11 305			5×7×17×19	50	41 041	11		7×11×13×41

que ce sont tous des nombres de Carmichael. Par exemple, 1729=(6+1)(12+1)(18+1) est le 1er nombre de Chernick (il suffit de prendre k=1). Les nombres de Chernick sont beaucoup plus rares que les nombres de Carmichael, mais plus simples à obtenir ; le suivant est obtenu pour k=6 (il s'agit de 294 409) et ensuite il faut attendre k=35 pour trouver le 3ème (il s'agit de 56 052 361). Certains nombres de la forme $(6k+1)(12k+1)(18k+1)$ sont des nombres de Carmichael sans être des nombres de Chernick, le premier étant 172 081 qui s'écrit 31×61×91 avec 31 et 61 premiers, mais 91 n'est pas premier (91=7×13). On peut remarquer sur notre tableau que toutes les décompositions de nombres de Poulet ne font intervenir que des facteurs premiers différents (pas d'exposant supérieur à 1