Arithmétique et Python 1 - divisibilité / congruences

Yannick Le Bastard

31 juillet 2024

Table des matières

Résumé : L'Arithmétique ! Le sujet est si vaste et si passionnant qu'il convient de se limiter dès le début tant les résultats et les approches sont foisonnantes. En ce qui nous concerne, nous travaillerons essentiellement avec des entiers naturels ou relatifs conformément aux programmes du secondaire, mais n'excluons aucun prolongement si le besoin s'en fait sentir. Nous reprendrons dans le corps de cet article des résultats élémentaires mais fondamentaux en nous limitant à une présentation de niveau secondaire. Nous irons parfois plus loin dans l'esprit des Olympiades de mathématiques, et nous rencontrerons quelques fonctions arithmétiques classiques généralement enseignées dans le supérieur.
Le présent document est cependant essentiellement à destination des enseignants en devenir ou en exercice, et suppose connus les thèmes abordés, ainsi qu'une certaine familiarité avec le langage Python, notamment l'utilisation raisonnée des listes.
Beaucoup de notions seront présentées de manière duale, en illustrant un cadre purement théorique par une approche informatique efficiente. L'évolution des programmes du secondaire nous incite en effet à ce cheminement afin de les exposer en classe à l'aide de l'outil TICE. Ce-dernier peut et doit servir de "testeur", d'approche inventive et rigoureuse, avant de donner, si le contexte le permet, une solution standard d'un exercice ou d'un problème intéressant.
Réactiver les souvenirs de collège des élèves de seconde sur les nombres premiers et l'algorithme d'Euclide, leur exposer une vraie démarche scientifique, les faire chercher bien évidemment sans quoi aucune appropriation n'est possible, est une activité essentielle avant la formalisation rigoureuse des concepts. Et puis, depuis fort longtemps maintenant, arithmétique et informatique font ménage commun dans les transactions bancaires ou en sécurité militaire. Si la reine des sciences est La Mathématique et si la reine des mathématiques est l'Arithmétique, sa petite cousine l'informatique ne peut que s'enorgueillir de la parer de ses plus beaux atours !

Je remercie chaleureusement ma collègue Hélène Carles pour sa relecture attentive, et l'exercice n'est pas simple, ainsi que pour ses suggestions pertinentes.

1 Généralités sur les entiers - Divisibilité

Nous supposons acquise la notion d'entier (naturel ou relatif), de valeur absolue, et nous admettrons les propriétés suivantes :

- **Toute partie A *non vide* de $\mathbb{N}$ admet un plus petit élément m,**
- **Toute partie A *non vide*, et *majorée* de $\mathbb{N}$ admet un plus grand élément M,**
- **Le corps $\mathbb{R}$ est *archimédien* : $\forall (x,y) \in \mathbb{R}_+^* \times \mathbb{R}$ il existe $n \in \mathbb{N}^*$ tel que $nx > y$.**

1.1 Divisibilité

Théorème 1-1-1 (de la division euclidienne) : Soient $(a,b) \in \mathbb{N} \times \mathbb{N}^*$. Alors il existe un *unique* couple $(q,r) \in \mathbb{N}^2$ tel que $a = bq + r$, et $0 \leq r < b$.

Démonstration :
Unicité : Soient $(a,b) \in \mathbb{N} \times \mathbb{N}^*$. Supposons qu'il existe $(q,r) \in \mathbb{N}^2$ tel que $a = bq + r$, et $0 \leq r < b$, ainsi que $(q',r') \in \mathbb{N}^2$ tel que $a = bq' + r'$, et $0 \leq r' < b$.
Mais alors, $bq + r = bq' + r'$ et donc $b|q - q'| = |r - r'|$. Or $|r - r'| < b$, d'où nécessairement $|q - q'| = 0$ *i.e* $q = q'$, puis $r = r'$.
Existence : Soit $A = \{k \in \mathbb{N}; a - bk \in \mathbb{N}\}$. $A \neq \emptyset$: $0 \in A$ et A est majoré par $a+1$: $a - b(a+1) = a(1-b) - b < 0$. Donc A admet un élément maximum que nous noterons q. Posons alors $r = a - bq$.
Supposons par l'absurde que $r \geq b$. Alors : $0 \leq r - b = a - b(q+1)$. D'où $q+1 \in A$ et $q+1 > q$. Contradiction avec q maximal. D'où $r < b$.

Remarque 1-1-2 : Pour mieux faire ressortir la définition de $\mathbb{R}$ archimédien, nous aurions aussi pu introduire l'ensemble $A = \{n \in \mathbb{N} \ ; nb \leq a\}$ et considérer son élément maximal.

Exercice résolu 1-1-3 : Nous pouvons traduire la preuve d'existence précédente informatiquement sans l'aide des opérateurs modulo % et // renvoyant respectivement le reste de la division euclidienne de a par b et le quotient de la division euclidienne de a par b. Moralement, si $a \geq b$, on construit une suite arithmétique de premier terme a et de raison $-b$. Notons $q+1$ le premier entier tel que $a - b(q+1) < 0$. Alors q est le quotient recherché et $r = a - bq$. Nous donnons ici un script sans test de vérification ($a \geq 0$ et $b > 0$). Le lecteur complètera.
Remarquons que $q = E(a/b)$, que l'on note aussi $q = [a/b]$.

```python
#script de H. Carles
#Entree : deux entiers naturels a et b ( b non nul )
#Sortie : le quotient dans la division euclidienne de a par b
def division(a,b) :
    c = a           #on part de a
    if c < b :
        return 0
    else :
        q = 0       #le compteur de pas
        while c >= 0:
            c -= b   #on enleve b a chaque pas
            q += 1   #increment du compteur
        return q-1

a = int(input("saisir un entier naturel a : "))
b = int(input("saisir un entier naturel b : "))
print(a,"= ",b," *", division(a,b),"+",a-b*division(a,b))
```

Corollaire et définition 1-1-4 : De ce qui précède, nous pouvons déduire que pour tout couple $(a, b) \in \mathbb{Z} \times \mathbb{N}^*$,il existe un unique couple $(q, r) \in \mathbb{Z} \times \mathbb{N}$ tel que $a = bq + r$, et $0 \leq r < b$. q s'appelle le *quotient* et r le *reste* dans la division euclidienne de a par b.

Corollaire et remarque 1-1-5 : Si $(a, b) \in \mathbb{Z}^2$, pour tout couple $(a, b) \in \mathbb{Z} \times \mathbb{Z}^*$, il existe un unique couple $(q, r) \in \mathbb{Z} \times \mathbb{N}$ tel que $a = bq + r$, et $0 \leq r < |b|$.
Si nous n'imposons pas au reste r d'être dans $\mathbb{N}$ en remplaçant la condition $0 \leq r < |b|$ par la condition $|r| < |b|$, alors l'unicité est perdue. Par exemple : $24 = 5 \times 4 + 4$, mais aussi $24 = 5 \times 5 - 1$.

■ Vous pouvez faire ici l'exercice 0.

Définition 1-1-6 : Soient a et b deux entiers ($b \neq 0$). On dit que a est **divisible** par b (ou que a est un **multiple** de b) s'il existe un entier k tel que $a = kb$.
Autrement dit, a est divisible par b si le reste r de la division euclidienne de a par b est nul.

Remarque 1-1-7 : il résulte de cette définition que 0 est divisible par n'importe quel entier ($k = 0$ convient toujours). Nous exclurons ce cas dans le script suivant mais le lecteur pointilleux le prendra en compte via un test de saisie.

Exercice corrigé 1-1-8 : Écriture d'un entier N non nul sous la forme $a \times b$ ($a \leq b$) : par exemple si l'on saisit $N = 20$, il sera renvoyé :
$N = 1 \times 20$
$N = 2 \times 10$
$N = 4 \times 5$

```python
N = int(input("Saisir un entier N>=2 : "))
D = [i for i in range(1,N+1) if N%i == 0] #diviseurs de N
for div in D :
    if div <= N//div :          #pour eviter les doublons
        print(N,"=",div,"*",N//div)
```

■ Vous pouvez faire ici les exercices 1 et 2.

Propriété 1-1-9 : Soient a, b et c trois entiers non nuls.

1. Si $a|b$ et si $b|c$, alors $a|c$ (la relation de divisibilité est transitive),
2. Si $a|b$ et si $a|c$, alors pour tout entiers $(k,l) \in \mathbb{Z}^2$, $a|kb+lc$,
3. Si $a|b$, alors $|a| \leq |b|$,
4. Si $a|b$ et $b|a$, alors $a = \pm b$

Démonstration :

1. $a|b$ donc il existe $u \in \mathbb{Z}$ tel que $b = ua$. De même, il existe $v \in \mathbb{Z}$ tel que $c = vb$. D'où $c = uva$ et donc $a|c$.
2. $a|b$ donc il existe $u \in \mathbb{Z}$ tel que $b = ua$. De même, il existe $v \in \mathbb{Z}$ tel que $c = va$. Mais alors pour tout $(k,l) \in \mathbb{Z}^2$, $kb + lc = (ku + lv)a$ d'où le résultat.
3. $a|b$ donc il existe $u \in \mathbb{Z}$ tel que $b = ua$. De plus, $a, b > 0$ donc nécessairement $u \neq 0$, donc $|b| = |u||a| \geq |a|$.
4. Immédiat de par ce qui précède.

Notation et exemples 1-1-10 : Notons $\mathcal{D}_k$ l'ensemble des diviseurs d'un entier k. Nous savons que $\mathcal{D}_k \neq \emptyset$ puisque de manière évidente $1 \in \mathcal{D}_k$.

1. Résoudre dans $\mathbb{Z}^2$ l'équation :
   (a) $n+1|n+7$
   (b) $10|n^2 + (n+1)^2 + (n+3)^2$
2. Un entier naturel N est dit *parfait* si la somme de ses diviseurs stricts est égal à N. Par exemple, 6 a pour diviseurs stricts 1, 2 et 3 et $1+2+3=6$. Donc 6 est un nombre parfait. Écrire un script Python qui détermine la liste de tous les nombres parfaits inférieurs ou égaux à 10000.
3. Trouver le plus petit entier n tel que $2|n-1,\ 3|n-2,\ \ldots,\ 9|n-8$.

Solutions : Théorique comme pratique !
Les équivalences des questions 1 et 2 sont évidentes, mais il peut être utile de raisonner par condition nécessaire et suffisante en classe.

1. La solution théorique n'appelle pas de solution pratique :
   (a)

$$\begin{aligned} n+1 \mid n+7 &\Leftrightarrow n+1 \mid (n+1)+6 \\ &\Leftrightarrow n+1 \mid 6 \\ &\Leftrightarrow n+1 \in \{-6;-3;-2;-1;0;1;2;3;6\} \\ &\Leftrightarrow n \in \{-7;-4;-3;-2;-1;0;1;2;5\} \end{aligned}$$

   (b)

$$\begin{aligned} 10 \mid n^2+(n+1)^2+(n+3)^2 &\Leftrightarrow 10 \mid 3n^2+8n+10 \\ &\Leftrightarrow 10 \mid 3n^2+8n \end{aligned}$$

   En attendant le paragraphe sur les congruences, examinons pour le moment à l'aide de Python les cas $n = a + 10k,\ a = 0, 1, \ldots, 9\ ;\ k \in \mathbb{Z}$ (les "a" sont les restes de la division euclidienne de n par 10) et calculons $3n^2+8n$. Si n vérifie $3n^2+8n \equiv 0[10]$ alors n convient. Il suffit en fait de tester les restes a.

```
def diviseurs() :
    L = []
    for a in range(10) :          #a varie de 0 a 9
        if (3*a**2+8*a)%10 == 0 :
            L.append(a)
    return L

print(diviseurs())
```

On trouve $[0;4]$. Autrement dit, les entiers n solutions sont de la forme $n = 10k$ ou $n = 4 + 10k$, $k \in \mathbb{Z}$.

2. Les listes en Python sont vraiment pratiques !

```
def ListeParfaits(N) :
    L = []
    for k in range(2,N) :
        D = [i for i in range(1,k) if k%i == 0] #les diviseurs stricts de k
        if k == sum(D) :
            L.append(k)
    return L

print(ListeParfaits(10000))
```

On trouve 6, 28, 496, 8128.

3. Faisons appel à Python.

```
def diviseurMin() :
    n = 1
    condition = [(n-j+1)%j == 0 for j in range(2,10)]
    while condition != [True for i in range(8)] :
        n += 1
        condition = [(n-j+1)%j == 0 for j in range(2,10)]
    return n

print(diviseurMin())
```

On trouve 2519.

1.2 Écriture dans une base quelconque

Nous avons tous l'habitude de raisonner en base 10 de par notre numérotation décimale classique. Mais historiquement, c'est le système sexagésimal (base 60) qui était utilisé depuis la plus haute antiquité (dès 3000 av J.C par les sumériens, puis vers 2000 av J.C par les babyloniens. Il en fut de même en Inde et en Chine, puis par la culture arabe qui l'emprunta à la culture indienne. Les grecs suivirent à leur tour). Nous le retrouvons à l'heure actuelle dans le décompte du temps : 1 heure = 60 minutes ; 1 minute = 60 secondes.

Écriture en base 10 Tout entier naturel a s'écrit sous la forme

$$a = \sum_{i \geq 0} a_i 10^i, \quad 0 \leq a_i \leq 9$$

où les coefficients a_i sont presque tous nuls *i.e* il existe un rang N à partir duquel tous les a_i sont nuls.

Dans la pratique, on posera alors $a = \sum_{i=0}^{N-1} a_i 10^i$.

Écriture en base b

Théorème 1-2-1 : Soit b un entier supérieur ou égal à 2. Tout entier $a > 0$ s'écrit de manière unique sous la forme :

$$a = \sum_{i \geq 0} a_i b^i,\ a_i \in \{0; 1; \ldots; b-1\},\ \text{où les } a_i \text{ sont presque tous nuls}$$

L'unicité signifie que si $a = \sum_{i=0}^{n} a_i b^i = \sum_{j=0}^{m} a'_j b^j \quad ; \quad a_i, a'_j \in \{0; 1; \ldots; b-1\}$, alors $m = n$ et $a_i = a'_i$ pour tout $i \in \{0, \ldots, n\}$.

Un exemple pour mieux comprendre : Considérons $a = 236$. La division euclidienne de a par 5 donne :
$236 = 47 \times 5 + 1$
Divisons le quotient précédent 47 par 5 :
$47 = 9 \times 5 + 2$
De même :
$9 = 1 \times 5 + 4$
$1 = 5 \times 0 + 1$ (quotient nul)
soit : $236 = (((5 \times 0 + 1) \times 5 + 4) \times 5 + 2) \times 5 + 1$.
En développant : $236 = 1 \times 5^3 + 4 \times 5^2 + 2 \times 5^1 + 1$.
On écrit : $136 = \overline{1421}^5$ ou si aucune confusion n'est à craindre : $136 = \overline{1421}$.
Le chiffre des unités est le reste de la division euclidienne par 5.
Il suffit donc de **remonter depuis le dernier reste obtenu jusqu'au premier** afin d'obtenir l'écriture de n en base 5.

Démonstration : Elle est à retenir absolument car son principe algorithmique (division en cascade) donne l'idée du script permettant de convertir un entier naturel écrit en base 10 dans sa forme écrite en base b. Tout repose sur le principe de division euclidienne vu auparavant !
$n \longleftarrow$ entier naturel
$b \longleftarrow$ entier naturel ≥ 2
$r \longleftarrow n\%b$
$L \longleftarrow [r]$ (L est une liste)
Tant que $n//b \neq 0$ faire :
  $n \longleftarrow n//b$
  $r \longleftarrow n\%b$
  Ajouter r à L
Fin Tant que
Inverser L
Afficher les éléments de L

Ceci permet de donner le script suivant (base 10 vers base b) :

```
def conversionBase10b(n,b) :      #de la base 10 vers la base b
    r = n%b
    L = [r]
    while (n//b) != 0 :
        n = (n//b)
        r = n%b
        L.append(r)
```

```
    L.reverse()
    return L

n = int(input("Saisir un entier naturel en base 10 : "))
b = int(input("saisir une base b (entier >=2) : "))
print(conversionBase10b(n,b))
```

Remarquons que la réciproque est facile à établir :

```
def conversionBaseb10(n,b) :      #de la base b vers la base 10
    n,nb = str(n),0
    for i in range(len(n)) :
        nb += int(n[len(n)-1-i])*b**i
    return nb

b = int(input("saisir une base b (entier >=2) : "))
n = int(input("Saisir un entier naturel en base b : "))
print(conversionBaseb10(n,b))
```

Exercice résolu 1-2-2 : Quelques exemples :

1. Convertir $\overline{234}^{(5)}$ et $\overline{11001000110}^{(2)}$ en base dix,
2. En base supérieure à dix, on utilisera les lettres A, B, C, ...comme chiffres au-delà de 9. Convertir $\overline{AAB}^{(12)}$ en base 10 puis en base 7. Même question avec $\overline{A8D}^{(12)}$ en base 10 puis en base 4.

Solution :

1. $\overline{234}^{(5)} = 4 + 3 \times 5 + 2 \times 5^2 = 69$.
   $\overline{11001000110}^{(2)} = 2 + 2^2 + 2^6 + 2^9 + 2^{10} = 1606$
2. $\overline{AAB}^{(12)} = 11 + 10 \times 12 + 10 \times 12^2 = 1571$ en base 10, puis (utilisation du script précédent) : 4403 en base 7.
   $\overline{A8D}^{(12)} = 13 + 8 \times 12 + 10 \times 12^2 = 1549$ en base 10, puis 120031 en base 4.

Opérations usuelles dans une base : Profs comme élèves : en retenue !

Le principe est le même qu'en base 10 : il ne faut pas oublier les retenues. Donnons par exemple la table d'addition en **base 6** :

+	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	10
2	2	3	4	5	10	11
3	3	4	5	10	11	12
4	4	5	10	11	12	13
5	5	10	11	12	13	14

De même, en base 6, $10 + 10 = 20$, $20 + 40 = 100$, $20 + 50 = 110$, etc.

Un exemple d'addition en base 6 : 345 + 23.
En base 6, 5+3 = 12. On pose 2 on retient 1.
4 +2 = 10 , donc 1+ (4+2) = 11. On pose 1 et on retient 1.
Enfin, 1+3 = 4. **Mettre sous forme posée**.
Au final, $\overline{345}^{(6)} + \overline{23}^{(6)} = \overline{412}^{(6)}$.
Faisons de même pour la table de multiplication en base 4 :

×	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	10	12
3	0	3	12	21

■ Vous pouvez faire ici les exercices 3 et 4.

Écriture en binaire, octodécimal et hexadécimal C'est le mode de représentation usuel des nombres par un ordinateur. Nous y reviendrons en détail dans un thème d'étude approfondissant les notions vues dans cet article.

D'après le théorème d'écriture d'un entier en base b, nous savons que tout entier $a > 0$ s'écrit de manière unique sous la forme :

$$a = \sum_{i \geq 0} a_i 2^i, \ a_i \in \{0; 1\}, \text{ où les } a_i \text{ sont presque tous nuls}$$

Il s'agit de la **représentation binaire** d'un entier.
En base 8, nous parlons de **représentation octodécimale**, et en base 16 de **représentation hexadécimale**.

1.3 Congruences

Voici une section extrêmement importante ! Nous allons moduler tout ça ...

Définition 1-3-1 : Soit $n \in \mathbb{N}^*$. On dit que deux entiers a et b sont *congrus modulo* n, et on note $a \equiv b[n]$ ou $a \equiv b \pmod{n}$ si $a - b$ est un multiple de n, ou de manière équivalente si n divise $a - b$.

Remarque 1-3-2 : La relation $a \equiv 0[n]$ signifie que n divise a.

Propriété 1-3-3 :

1. La relation de congruence modulo n est une *relation d'équivalence*.
2. Cette relation est *compatible avec l'addition* : si $a \equiv b[n]$ et si $c \equiv d[n]$, alors
$$a + c \equiv b + d[n]$$
3. Cette relation est *compatible avec la multiplication* : si $a \equiv b[n]$ et si $c \equiv d[n]$, alors
$$ac \equiv bd[n]$$
4. Soit d un diviseur positif de n. Si $a \equiv b[n]$, alors $a \equiv b[d]$,
5. Soit r le reste de la division euclidienne de $a \in \mathbb{Z}$ par $n \in \mathbb{N}^*$. Alors $a \equiv r[n]$.

La démonstration de ces propriétés est immédiate. Prouvons juste 3 :
$a \equiv b[n]$ donc il existe un entier k tel que $a = b + kn$. De même, il existe un entier l tel que $c = d + ln$. D'où $ac = (b + kn)(d + ln) = bd + (bl + kd + kln)n$. CQFD.

Remarque 1-3-4 : Nous verrons d'autres propriétés des congruences une fois connues les notions de PGCD et de PPCM.

Corollaire 1-3-5 : Soit $n \in \mathbb{N}^*$. Tout entier $a \in \mathbb{Z}$ est congru à un unique entier $r \in \{0, 1, \ldots, n-1\}$.

Corollaire 1-3-6 : Soit $n \in \mathbb{N}^*$. Si $a \equiv b[n]$, alors pour tout entier naturel k, $a^k \equiv b^k[n]$.

Cette notion de congruence permet de prouver les critères de divisibilité classiques par 3, par 9 et par 11.

Propriété 1-3-7 : Soit $n = a_n \times 10^n + a_{n-1} \times 10^{n-1} + \cdots + a_0$, $(0 \leq a_i \leq 9)$ l'écriture en base 10 d'un entier n. Alors modulo 3 (resp. modulo 9), l'entier n est congru à la somme $a_n + \cdots + a_0$ de ses chiffres modulo 10.
Le nombre n est ainsi multiple de 3 (resp. de 9) si la somme de ses chiffres est un multiple de 3 (resp. de 9).

Démonstration : $10 \equiv 1[3]$, donc pour tout entier naturel i, $10^i \equiv 1[3]$. Ainsi, $a_n \equiv \sum_{i=0}^{n} a_i[3]$.

Propriété 1-3-8 : Soit $n = a_n \times 10^n + a_{n-1} \times 10^{n-1} + \cdots + a_0$, $(0 \leq a_i \leq 9)$ l'écriture en base 10 d'un entier n. Alors modulo 11, l'entier n est congru modulo 10 à la somme alternée de ses chiffres en partant du chiffre des unités : $a_0 - a_1 + a_2 + \cdots + (-1)^n a_n$.

Remarque 1-3-9 : En Python, la notion de congruence entre deux entiers naturels existe via l'instruction %. Testez par exemple dans le shell :
>>>10%3
1 # on a bien $10 = 3 \times 3 + \mathbf{1}$
En revanche, il y a souci entre deux entiers relatifs (*cf* remarque 1-1-5). Testez :
>>>-10%-3
-1

Exercice résolu 1-3-10 :

1. Prouver que pour tout entier naturel n, $3^{2n} - 2^n$ est multiple de 7,
2. Pour quelles valeurs de n le nombre $4^n + 2^n + 1$ est-il divisible par 7 ?
3. Prouver que l'équation diophantienne $56x^2 + 87y^2 = 134$ n'a pas de solution dans $\mathbb{Z}$.

Solution :

1. $9 \equiv 2[7]$, donc (corollaire 1-3-6) $3^{2n} - 2^n \equiv 2^n - 2^n = 0[7]$, ce qui prouve le résultat annoncé.
2. Il suffit de regarder les restes de la division euclidienne de n par 7 en vertu du corollaire 1-3-5.

```
def Restes() :
    L = []
    for reste in range(7) :
        if (4**reste+2**reste+1)%7 == 0 :
            L.append(reste)
    return L

print(Restes())
#Le script renvoie [1, 2, 4, 5]
```

On en déduit que les entiers de la forme $1 + 7k$, $2 + 7k$, $4 + 7k$, $5 + 7k$ $(k \in \mathbb{Z})$ sont solutions.

3. Nous allons raisonner par l'absurde en supposant que l'équation admette des solutions entières. Le principe de la démonstration est que si $A = B$ dans $\mathbb{Z}$, alors pour tout entier naturel n on a $A \equiv B[n]$. En choisissant de "bonnes valeurs" de n, la contradiction apparaitra.
   La décomposition en facteurs premiers des coefficients donne $56 = 2^3 \times 7,\ 87 = 3 \times 29$ et $134 = 2 \times 67$. Raisonnons modulo 7, histoire de supprimer un terme.
   L'équation diophantienne se réécrit ($87 \equiv 3[7]$ et $134 \equiv 1[7]$) : $3y^2 \equiv 1[7]$.
   Le corollaire 1-3-5 nous permet de nous limiter au cas des restes, autrement dit il suffit de regarder si pour $y \in \{0, 1, \ldots, 6\}$, l'équation $3y^2 \equiv 1[7]$ a une solution. On vérifie aisément que ce n'est pas le cas. Ce qui achève la preuve.
   Remarque : pourquoi ne pas avoir raisonné modulo 2 afin de supprimer deux termes ?

■ Vous pouvez faire ici l'exercice 5.

1.4 PGCD-PPCM

Propriété et définition 1-4-1 : Considérons deux entiers a et b non tous deux nuls. L'ensemble des diviseurs communs $\mathcal{D}_a \cap \mathcal{D}_b$ à a et à b est **fini** et **non vide**. Il possède donc un plus grand élément, appelé **plus grand commun diviseur** de a et de b, et noté **PGCD(a,b)** ou $a \wedge b$.
De même, l'ensemble des multiples communs *positifs* à a et à b est non vide (mais infini). Il possède donc un plus petit élément, appelé **plus petit commun multiple** de a et de b, et noté **PPCM(a,b)** ou $a \vee b$.
On définit de même par récurrence le PGCD et le PPCM de n entiers.

Démonstration (pour le PGCD) : $\mathcal{D}_a \cap \mathcal{D}_b \subset \mathbb{Z}$, est non vide (1 est élément) et borné (par $\max |a|, |b|$), donc admet un plus grand élément positif.

Notation 1-4-2 : On note aussi $\mathcal{D}(a, b)$ l'ensemble des diviseurs communs à a et b.

Définition 1-4-3 :

1. Deux entiers naturels a et b sont dits **premiers entre eux** si leur plus grand diviseur commun (PGCD) est égal à 1 *i.e* $a \wedge b = 1$,
2. N entiers naturels $a_1, a_2, \ldots, a_N$ sont dits **premiers dans leur ensemble** si leur plus grand diviseur commun (PGCD) est égal à 1. On note $a_1 \wedge a_2 \wedge \cdots \wedge a_N = 1$,
3. N entiers naturels $a_1, a_2, \ldots, a_N$ sont dits **premiers entre eux deux à deux** si pour tout $(i, j) \in \{1; \ldots; N\}$ $a_i \wedge a_j = 1$.

Remarque 1-4-4 : N entiers premiers entre eux dans leur ensemble ne le sont pas nécessairement deux à deux. Considérer par exemple 2, 5 et 10. Ce résultat est à comparer avec la notion d'indépendance d'événements en probabilités : dans ce dernier contexte, des événements deux à deux indépendants ne le sont pas nécessairement mutuellement. Nous reviendrons sur ce point en exercice.

Propriété 1-4-5 : Revenons sur la notion de divisibilité.

1. $a \wedge 0 = |a|$
2. Si $d = a \wedge b$, alors n divise a et b si et seulement si n divise d *i.e* si $n \in \mathcal{D}(a, b)$, alors $n \mid a \wedge b$.
   Ceci signifie que d est maximal pour la relation de divisibilité entre entiers.

3. Si $m = a \vee b$, alors n est un multiple de a et de b si et seulement si n est un multiple de m.
4. Si $a,\ b,\ n$ sont des entiers non nuls (avec $n > 0$), alors $(na) \wedge (nb) = n(a \wedge b)$.
5. Si $n > 0$ divise a et b, alors $\frac{a}{n} \wedge \frac{b}{n} = \frac{1}{n} a \wedge b$
6. Si $d = a \wedge b$, alors il existe $(a', b') \in \mathbb{Z}^2$ tels que $a = da',\ b = db'$ et $a' \wedge b' = 1$.
   Très utile en pratique !
7. Soient $(a, b, k, l, k', l') \in \mathbb{Z}^6$. Alors $\mathcal{D}(a, b) \subset \mathcal{D}(ak' + bl', ak + bl)$, avec égalité si et seulement si $|k'l - kl'| = 1$. En particulier, on a alors $a \wedge b = (ak + bl) \wedge (ak' + bl')$.

Démonstration :

1. $0 = a \times 0$ et $a \neq 0$. De plus $a \wedge 0 \in \mathbb{N}^*$, d'où $a \wedge 0 = |a|$.
2. Évident par définition de $a \wedge b$
3. De même !
4. Posons $d = a \wedge b$ et $D = na \wedge nb$. d divise a et b, donc nd divise na et nb. Par 2. nd divise D. Réciproquement D divise na et nb. Donc l'*entier* $\frac{D}{n}$ divise a et b, et donc $\frac{D}{n}$ divise d. D'où D divise nd. Il résulte que $D = \pm nd$ et comme tous deux sont positifs, $D = nd$.
5. Même raisonnement
6. Soit $d = a \wedge b$. Comme d divise a et b, posons $a' = \frac{a}{d}$ et $b' = \frac{b}{d}$. Supposons que a' et b' ne soient pas premiers entre eux. Alors $\delta = a' \wedge b' > 1$. Mais alors $d\delta$ divise a et b, donc divise d. D'où $\delta = 1$. Contradiction.
7. L'inclusion est évidente. Ceci entraine que $d = a \wedge b$ divise $D = (ak + bl) \wedge (ak' + bl')$. Réciproquement, supposons que $|k'l - kl'| = 1$. Pour fixer les idées, disons que $k'l - kl' = 1$. Alors D divise $l(ak' + bl') - l'(ak + bl) = a$. De même, D divise b, d'où D divise d. On en déduit que $d = D$.

Remarque technique 1-4-6 : Pour prouver l'égalité de deux PGCD, on prouve fréquemment que chacun divise l'autre.

Algorithme d'Euclide 1-4-7 : Soient $(a, b) \in \mathbb{N} \times \mathbb{N}^*$.

1. $a \wedge b = b \wedge r$ où r désigne le reste de la division euclidienne de a par b.
2. $a = bq_0 + r_0$, où $0 \leq r_0 < b$. Si $r_0 = 0$ alors c'est fini : $a \wedge b = b$, sinon on recommence en remplaçant a par b et b par r_0 :
   $b = q_1 r_0 + r_1$, où $r_1 < r_0$. Si $r_1 = 0$, alors c'est fini : $a \wedge b = r_0$, sinon on recommence en remplaçant b par r_0 et r_0 par r_1, etc.
   La suite d'entiers positifs ou nuls $(r_i)_{i \geq 0}$ est strictement décroissante, et ne pouvant être infinie, s'annule à partir d'un certain rang. Le PGCD est le dernier reste non nul.
3. Ce qui précède nous permet d'établir l'algorithme suivant (dont le nombre de pas est fini) :
   $a \longleftarrow$ entier naturel
   $b \longleftarrow$ entier naturel non nul
   $r \longleftarrow a\%b$
   Tant que $r \neq 0$ faire :
   $a \longleftarrow b$
   $b \longleftarrow r$
   $r \longleftarrow a\%b$
   Fin Tant que
   Afficher $PGCD(a, b) = b$ (dernier reste non nul)

```
def Euclide(a,b) :
    r = a%b
    while r != 0 :
        a = b
        b = r
        r = a%b
    return b

a = int(input("saisir un entier naturel a "))
b = int(input("saisir un entier naturel b "))
print("Le PGCD de",a,"et",b,"est de :",Euclide(a,b))
```

Exemple 1-4-8 : Déterminons le PGCD de 144 et 38. Nous allons faire apparaitre en plus les quotients des divisions euclidiennes successives afin de "remonter" l'algorithme d'Euclide. Ceci nous permettra de déterminer deux entiers relatifs u et v tels que $au + bv = d$. Nous obtenons ainsi l'algorithme d'Euclide étendu.
Descente : Posons $a = 144$ et $b = 38$; $144 = 38 \times 3 + 30$.

a	b	r	q
144	38	30	3
38	30	8	1
30	8	6	3
8	6	2	1
6	**2**	0	3

Le dernier reste non nul est 2, donc $144 \wedge 38 = 2$.
La nature même de l'algorithme d'Euclide nous permet de définir une suite récurrente linéaire d'ordre 2 en posant conventionnellement (notations Python) :
$r_{-2} = a$, $r_{-1} = b$ et pour tout entier naturel n : $r_{n-2} = q_n r_{n-1} + r_n$, où $q_n = r_{n-2}//r_{n-1}$, sous réserve que $r_{n-1} \neq 0$ bien sûr !

Ainsi définie, la suite d'entiers naturels (r_n) est strictement décroissante, et comme elle ne peut être infinie, elle est donc nulle à partir d'un certain rang $N + 1$.
On a $r_{-2} \wedge r_{-1} = r_{-1} \wedge r_0 = \cdots = r_N \wedge r_{N+1} = r_N \wedge 0 = r_N$.

Remontée : Commençons "à la main". L'analyse du tableau nous apprend que :

$$\begin{aligned} 2 &= 8 - 6 \times 1 \\ &= (38 - 30) - (30 - 8 \times 3) \\ &= 38 - 30 \times 2 + 8 \times 3 \end{aligned}$$

Or $30 = 144 - 38 \times 3$, d'où $8 = 38 \times 4 - 144$. D'où :

$$2 = 38 - (144 - 38 \times 3) \times 2 + (38 \times 4 - 144) \times 3 = 144 \times (-5) + 38 \times 19$$

Le couple $(u, v) = (-5, 19)$ vérifie $144u + 38v = 2$.

Généralisons la démarche précédente. Nous poserons $d = a \wedge b$.
À chaque étape de l'algorithme d'Euclide, on a une égalité de la forme :

$$r_{n-2} = r_{n-1} q_n + r_n$$

À l'avant-dernière étape, on a $r_N = d$, et donc une égalité de la forme :

$$r_{N-2} = r_{N-1}q_N + d$$

soit :

$$d = r_{N-2} - q_N r_{N-1}$$

À l'étape précédente, on a :

$$r_{N-1} = r_{N-3} - q_{N-1}r_{N-2}$$

Ces deux égalités permettent d'exprimer d comme combinaison linéaire de r_{N-3} et de r_{N-2}. Continuant à remonter, on aboutit à une inégalité de la forme :

$$d = ur_{-2} + vr_{-1} = au + bv$$

Nous allons construire par récurrence deux suites $(\tilde{u}_n)_{n\geq -2}$ et $(\tilde{v}_n)_{n\geq -2}$ telles que pour tout entier $n \geq -2$: $\boxed{r_n = a\tilde{u}_n + b\tilde{v}_n \quad (1)}$. Les coefficients u_N et v_N conviendront alors.

Rappelons que l'on a posé $r_{-2} = a$ et $r_{-1} = b$.
De manière à travailler avec des indices positifs ou nuls, posons pour tout entier naturel n : $a_n := r_{n-2},\ u_n := \tilde{u}_{n-2}$ et $v_n := \tilde{v}_{n-2}$.
Posons $(u_0, v_0) = (1, 0)$ et $(u_1, v_1) = (0, 1)$ de sorte que la relation (1) écrite avec (a_n) soit valide : $a_n = au_n + bv_n \quad (1)$.
D'autre part, $a_n = q_{n+2}a_{n+1} + a_{n+2}$, soit :

$$\boxed{a_n - q_{n+2}a_{n+1} = a_{n+2} \quad (2)}$$

Se servant de (1), nous en déduisons que :

$$a_{n+2} = au_n + bv_n + q_{n+2}(au_{n+1} + bv_{n+1})$$

soit :

$$a_{n+2} = a(u_n - q_{n+2}u_{n+1}) + b(v_n - q_{n+2}v_{n+1})$$

Ceci nous amène à poser $u_{n+2} = u_n - q_{n+2}u_{n+1}$ ainsi que $v_{n+2} = v_n - q_{n+2}v_{n+1}$.
On rappelle que $q_{n+2} = a_n//a_{n+1}$.

La notion d'affectation parallèle en Python est très utile pour définir des suites récurrentes d'ordre $n \geq 2$. C'est cette dernière qui nous permet d'écrire un script plus compact, sans utiliser de "variable tampon".

```
def EuclideEtendu(a,b):
    if a%b == 0 :           #b divise a
        return [0,1]    #car alors PGCD(a,b)=b
    else :
        u,U = 1,0
        v,V = 0,1
        q,r = a//b,a%b
        while r != 0 :
            u,U = U,u-q*U
            v,V = V,v-q*V
            a,b = b,r
            r = a%b
            q = a//b

    return [U,V]

print(EuclideEtendu(a,b))
```

■ Vous pouvez faire ici les exercices 6 et 7.

Théorème 1-4-9 : Tous reposent sur la notion de division euclidienne. Ils sont très utilisés en pratique.

1. **Théorème de Bachet-Bezout** : Si $d = a \wedge b$, alors il existe $(u, v) \in \mathbb{Z}^2$ tel que $au + bv = d$.
2. **Théorème de Bezout** : a et b sont des entiers premiers entre eux, si et seulement si il existe des entiers u et v tels que $au + bv = 1$.
3. **Théorème de Gauss** : Si $a \mid bc$ et si $a \wedge b = 1$, alors $a \mid c$.
4. Si deux entiers a et b premiers entre eux divisent un entier n, alors ab divise n.

Démonstration :

1. Le théorème de Bachet-Bezout résulte de l'algorithme d'Euclide étendu.
2. Supposons que les entiers a et b soient premiers entre eux. Alors $d = 1$. On conclut avec le théorème de Bachet-Bezout.
3. Si $a \mid bc$, alors il existe un entier k tel que $bc = ak$. De plus, puisque $a \wedge b = 1$, le théorème de Bezout nous assure l'existence de deux entiers u et v tels que $au + bv = 1$. Multipliant par c, nous obtenons $auc + bcv = c$, *i.e* $auc + akv = c$ *i.e* $a(uc + kv) = c$. Comme $uc + kv \in \mathbb{Z}$, on en déduit que $a \mid c$.
4. Puisque a et b divisent n, il existe deux entiers k et l tels que $n = ak = bl$ $(*)$. En particulier $a \mid bl$ et comme $a \wedge b = 1$, le théorème de Gauss nous assure que $a \mid l$. Donc il existe un entier m tel que $l = am$. Reportant ceci dans (*), nous obtenons que $n = abm$, et donc $ab \mid n$.

Exercice résolu 1-4-10 : Le PGCD de deux nombres entiers a et b est $d = 4$. Les quotients successifs dans la recherche du PGCD par l'algorithme d'Euclide(poursuivi jusqu'au reste 0) sont 2, 3, 1, 2. Déterminez a et b.
Même question avec $d = 6$ et les quotients 10, 1, 2, 3, 1, 2.
Sauriez-vous programmer ceci ?

Nous pouvons présenter les divisions euclidiennes successives à l'aide d'un tableau :

a	b	r	q
$a_0 = a$	$a_1 = b$	a_2	$q_2 = 2$
a_1	a_2	a_3	$q_3 = 3$
a_2	a_3	a_4	$q_4 = 1$
a_3	$a_4 = 4$	$a_5 = 0$	$q_5 = 2$

Nous savons de plus que $a_n = a_{n+2} + q_{n+2}a_{n+1}$. L'idée est de se servir de cette relation pour remonter au fur et à mesure à a et b. La programmation est idéale pour ce travail !

```
def RemonteeEuclide(PGCD,Liste_q) :
    Liste_a = [0,PGCD]
    long=len(Liste_q)
    Liste_q.reverse()    #pour partir du dernier quotient au premier
    for i in range(2,long+2) :    #decalage car Liste_a de longueur 2
        Liste_a.append(Liste_q[i-2]*Liste_a[i-1]+Liste_a[i-2])

    print("a =",Liste_a[long-1 +2])
    print("b =",Liste_a[long-2 +2])
```

```
10
RemonteeEuclide(4,[2,3,1,2])
#il s'affiche a = 100 et b = 44
RemonteeEuclide(6,[10,1,2,3,1,2])
#il s'affiche a = 2310 et b = 216
```

Vérification pour la seconde question : Calculons par l'algorithme d'Euclide le PGCD des nombres 2310 et 216.
$2310 = 216 \times 10 + 150$
$216 = 150 \times 1 + 66$
$150 = 66 \times 2 + 18$
$66 = 18 \times 3 + 12$
$18 = 12 \times 1 + 6$
$12 = 6 \times 2 + 0$
Le PGCD des nombres 2310 et 216 est le dernier reste non nul du procédé, c'est-à-dire 6. La liste des quotients obtenus est par ailleurs celle voulue.

Application 1-4-11 : équations diophantiennes $ax + by = c$
Nous supposons ici que les coefficients a et b sont des entiers naturels.
Notons $\boxed{(E) \ : \ ax + by = c}$. Soit $d = a \wedge b$.
Cas 1 : $d \nmid c$.
Alors (E) n'a pas de solution.
Cas 2 : $d \mid c$.
Soient alors (a', b', c') tels que $a = da', \ b = db'$ et $c = dc'$.
Divisant chaque membre de (E) par d, on obtient l'**équation équivalente** :
$(E') \ : \ a'x + b'y = c'$.
Mais on sait que $a' \wedge b' = 1$. D'après le théorème de Bachet-Bezout (ou l'algorithme d'Euclide étendu), nous savons qu'il existe un couple $(u_0, v_0) \in \mathbb{Z}^2$ tel que $a'u_0 + b'v_0 = 1$. Multipliant ceci par c', nous obtenons :
$a'(c'u_0) + b'(c'v_0) = c'$. Le couple $(x_0, y_0) = (c'u_0, c'v_0)$ est donc une solution particulière de (E').

$$\begin{cases} a'x + b'y = c' & (1) \\ a'x_0 + b'y_0 = c' & (2) \end{cases}$$

Soustrayant (2) de (1), nous obtenons : $a'(x - x_0) = b'(y_0 - y)$ (3).
Ainsi, $b' \mid a'(x - x_0)$ et comme $a' \wedge b' = 1$, le théorème de Gauss nous assure que $b' \mid x - x_0$, donc il existe un entier k tel que $x - x_0 = kb'$. Reportant dans (3), on obtient $y = y_0 - ka'$.
L'ensemble des solutions de (E') est donc inclus dans $\{(x_0 + kb', y_0 - ka'), \quad k \in \mathbb{Z}\}$. La réciproque est évidente.

Conclusion : $\boxed{\mathcal{S}_{(E)} = \{(x_0 + kb', y_0 - ka'), \quad k \in \mathbb{Z}\}}$.

Propriété 1-4-12 (retour sur les congruences) : Soient $(a, b) \in \mathbb{Z}^2$ et $n \in \mathbb{N}^*$. Alors :

1. Si $ac \equiv bc[n]$ et si $c \wedge n = 1$, alors : $a \equiv b[n]$,
2. $ac \equiv bc[n] \Leftrightarrow a \equiv b \left[\frac{n}{c \wedge n}\right]$,
3. Soit $(n_1, n_2) \in \mathbb{N}^{*2}$. Alors :
$$\begin{cases} a \equiv b[n_1] \\ a \equiv b[n_2] \end{cases} \Leftrightarrow a \equiv b[n_1 \vee n_2].$$

Démonstration :

1. En reformulant :
$$\begin{cases} n|(a-b)c \\ n \wedge c = 1 \end{cases}$$
Donc d'après le théorème de Gauss, $n|a-b$.
2. Posons $d = c \wedge n$.
($\Rightarrow$) Supposons que $ac \equiv bc[n]$. Il existe donc un entier k tel que $ac - bc = kn$. Divisant par d, nous obtenons $ac' - bc' = kn'$, où $c' \wedge n' = 1$. Comme $n'|(a-b)c'$, le théorème de Gauss nous assure que $n'|a-b$.
($\Leftarrow$) $n' := \frac{n}{d}|a-b$. En multipliant par c, on obtient que $\frac{nc}{d} := nc'|c(a-b)$, et comme $n|nc'$, nous obtenons que $n|c(a-b)$.
3. ($\Rightarrow$) Par hypothèse, $n_1|a-b$ et $n_2|a-b$. Donc $n_1 \vee n_2|a-b$.
($\Leftarrow$) Comme $n_1 \vee n_2|a-b$, on a puisque $n_1|n_1 \vee n_2$ que $n_1|a-b$. De même, $n_2|a-b$.

Théorème 1-4-13 (Théorème chinois) : Soient $(a,b) \in \mathbb{Z}$ et $(n_1,n_2) \in \mathbb{N}^*$. Supposons que $n_1 \wedge n_2 = 1$. Alors le système $\begin{cases} x \equiv a[n_1] \\ x \equiv b[n_2] \end{cases}$ a une solution.

Démonstration : $n_1 \wedge n_2 = 1$, donc d'après le théorème de Bezout, il existe $(u,v) \in \mathbb{Z}^2$ tel que $un_1 + vn_2 = 1$.
On en déduit que :
— $aun_1 + avn_2 = a$, d'où $avn_2 \equiv a[n_1]$
— $bun_1 + bvn_2 = b$, d'où $bun_1 \equiv b[n_2]$
Posons $x_0 = bun_1 + avn_2$. Par construction, x_0 est solution du système. Le lecteur se convaincra (puisque n_1 et n_2 sont premiers entre eux) que les solutions du système sont de la forme $x = x_0 + kn_1n_2, \quad (k \in \mathbb{Z})$. Ceci donne par ailleurs un procédé de construction d'une solution particulière.

Corollaire 1-4-14 : Soient $(a_1, a_2, \ldots, a_k) \in \mathbb{Z}^k$ et $(n_1, n_2, \ldots, n_k) \in \mathbb{N}^*$. Supposons que les n_i soient *deux à deux* premiers entre eux. Alors le système $\begin{cases} x \equiv a_1[n_1] \\ x \equiv a_2[n_2] \\ \vdots \\ x \equiv a_k[n_k] \end{cases}$ a une solution.

Démonstration : Donnons un algorithme de construction d'une solution particulière x_0 et un script l'illustrant.
Posons $n = n_1n_2 \ldots n_k$ et $\hat{n_i} = \dfrac{n}{n_i}$.
Pour tout entier $i \in \{1, 2, \ldots, k\}$, $n_i \wedge \hat{n_i} = 1$ par hypothèse. Donc d'après le théorème de Bachet-Bezout, il existe $(u_i, v_i) \in \mathbb{Z}^2$ tel que $u_in_i + v_i\hat{n_i} = 1$ *i.e* $e_i := v_i\hat{n_i} \equiv 1[n_i]$. De plus, $e_i \equiv 0[n_j]$ si $j \neq i$ par construction.
Mais alors $x_0 = \sum_{i=0}^{k} a_ie_i$ est solution du système.
L'ensemble des solutions s'en déduit :
$$\mathcal{S} = \{x_0 + kn, \ k \in \mathbb{Z}\}$$

Nous donnons maintenant un script identique à la démonstration qui utilisera l'algorithme d'Euclide étendu vu précédemment.

```
#Entree : k entier, la liste des n_i et des a_i (1<=i<=k)
#Sortie : une solution particuliere du systeme

def Entree1(k) :  # k : nombre d'entiers premiers entre eux 2 a 2 saisis
    L = []
    for i in range(k) :
        print("n_",i+1,"= ? ")
        n = int(input())
        L.append(n)
    return L    #renvoie la liste des n_i

def Entree2(k) :
    L = []
    for i in range(k) :
        print("a_",i+1,"= ? ")
        n = int(input())
        L.append(n)
    return L    #renvoie la liste des a_i

def EuclideEtendu(a,b) : #fonction indispensable ici
    if a%b == 0 :
        return [0,1]
    else :
        u,U = 1,0
        v,V = 0,1
        q,r = a//b,a%b
        while r != 0 :
            u,U = U,u-q*U
            v,V = V,v-q*V
            a,b = b,r
            r = a%b
            q = a//b
    return [U,V]

def Solution(liste1,liste2) : #liste1 : les n_i, liste2 : les a_i
    P = 1
    for el in liste1 :    #renvoie n = n_1...n_k
        P *= el
    L = []                #liste des s_i:=a_i*v_i*n/n_i
    for i in range(k) :
        m = int(P/liste1[i])
        s = liste2[i]*EuclideEtendu(liste1[i],m)[1]*m
        L.append(s)
    return sum(L)

k = int(input("Nombre de congruences ? "))
print("Une solution particuliere est : ",Solution(Entree1(k),Entree2(k)))
```

Nous n'avons pas fait de test de vérification que les n_i sont premiers entre eux deux à deux. Nous laissons ceci à la sagacité du lecteur.

■ Vous pouvez faire ici l' exercice 8.

2 Exercices

2.1 Énoncés des exercices

Exercice 0 : Théorie et pratique.

1. Démontrer le corollaire 1-1-4 et adapter le premier script du cours,
2. Démontrer le corollaire 1-1-5 et adapter le script précédent.

Exercice 1 : Déterminer un nombre entier n de trois chiffres tel que n soit multiple de 5 et de 14 et que la somme des chiffres de n soit égale à 14 (resp. à 10).

Exercice 2 : Deux questions indépendantes.

1. Déterminer à l'aide d'un script Python tous les diviseurs d'un entier (relatif) N saisi par l'utilisateur ainsi que la somme des diviseurs positifs de N.
2. Déterminer un entier $n > 0$ le plus petit possible, tel que le nombre $N = 88\ldots888$ écrit avec n chiffres 8 dans le système décimal, soit multiple de 4264.

Exercice 3 : Écrire un script en Python qui demande à l'utilisateur de saisir une base $b \geq 2$ ($b \leq 10$), deux entiers a et b écrits dans cette base (un test de bonne saisie sera effectué), et qui renvoie la somme de ces entiers, écrits en base b.
Application : calculer $\overline{23}^{(5)} + \overline{32}^{(5)}$, $\overline{234}^{(6)} + \overline{323}^{(6)}$.
En utilisant la conversion entre bases vue dans le cours, calculer $\overline{23}^{(7)} + \overline{34}^{(5)}$ en base 8.

Exercice 4 : Un nombre n s'écrit avec trois chiffres en base 9, puis avec les trois mêmes chiffres, dans un ordre différent, en base 13. Quel est ce (ou ces) nombre(s) n ? Donnez son écriture dans les bases 10, 9 et 13.

Exercice 5 : Prouver que pour tout entier naturel n :

1. $4^{3n} - 4^n$ est divisible par 5
2. $2^{4n+2} + 2^{4n+1} - 1$ est un multiple de 5
3. $4^n + 15n - 1$ est un multiple de 9
4. $n^2(n^4 - 1)$ est divisible par 60. Peut-on faire mieux *i.e* trouver un diviseur commun à tous les nombres $n^2(n^4 - 1)$ qui soit plus grand que 60 ? Vous pouvez si vous le souhaitez, vous aider de Python ou utiliser le petit théorème de Fermat énoncé plus loin.

Exercice 6 : Trouver le plus grand entier n qui soit divisible par tous les entiers inférieurs ou égaux à $\sqrt[3]{n}$.

Exercice 7 : Écrire un script Python qui demande à l'utilisateur de saisir N entiers supérieurs ou égaux à 2 et qui renvoie leur PGCD.

Exercice 8 : Théorie et ...théorie !

1. Prouver qu'il existe n nombres consécutifs qui ne sont pas des puissances parfaites. Indication : on pensera à la suite des nombres premiers et au fait que si p est un nombre premier, alors $x \equiv p[p^2]$ implique que x ne peut être une puissance parfaite.

2. Généralisation du théorème chinois : Soient $n_1, n_2, \ldots, n_k$ k entiers strictement positifs, et $a_1, a_2, \ldots, a_k$ des entiers quelconques. Alors le système $(S)\begin{cases} x \equiv a_1[n_1] \\ x \equiv a_2[n_2] \\ \vdots \\ x \equiv a_k[n_k] \end{cases}$ a une solution si et seulement si pour tous i et j, $a_i \equiv a_j[n_i \wedge n_j]$.
Si tel est le cas, il existe un entier a tel que $x \equiv a[n_1 \vee n_2 \vee \cdots \vee n_k]$.

2.2 Solutions des exercices

Exercice 0 : Généralisation progressive.

1. Ici, $(a, b) \in \mathbb{Z} \times \mathbb{N}^*$.
Si $a \geq 0$, c'est le cas précédent.
Si $a \in \mathbb{Z} \setminus \mathbb{N}$, alors $|a| = -a > 0$. D'où $a + b|a| = (b-1)|a| > 0$. Nous pouvons donc effectuer la division euclidienne de $a+b|a|$ par b comme dans le théorème. Il existe alors un unique couple d'entiers naturels (q', r') tel que $a + b|a| = bq' + r'$, où $0 \leq r' < b$. Ainsi, $a = b(q' - |a|) + r'$. Le couple $(q, r) = (q' - |a|, r')$ répond à la question. L'unicité est comme dans le théorème.
Ce qui amène au script alternatif :

```
#Entree : un entier relatif a et un entier naturel b ( b non nul )
#Sortie : le quotient dans la division euclidienne de a par b
def division2(a,b) :
    if a >= 0 :
        c = a            #on part de a
        if c < b :
            return 0
        else :
            q = 0        #le compteur de pas
            while c >= 0 :
                c -= b    #on enleve b a chaque pas
                q += 1   #increment du compteur
            return q-1
    else :
        c = (b-1)*abs(a)
        q = 0
        while c >= 0 :
            c -= b
            q += 1
        return q+a-1     #soit q-|a|+1
```

2. Si $b > 0$, c'est la cas précédent ; si $b < 0$, on effectue la division euclidienne de a par $-b$: $a = -bq + r$, $0 \leq r < -b = |b|$ *i.e* $a = b(-q) + r$.
Ce qui amène au script alternatif :

```
def division3(a,b) :
    if b > 0 :
        return division2(a,b)
    else:
        return -division2(a,-b)
```

Exercice 1 : Une réflexion préalable s'impose pour simplifier le script au maximum.

```python
def nombreMystere(somme) :
    Liste = []
    #Le nb est multiple de 5 et de 14 (donc de 2) donc se termine par 0
    L = [100*i+10*j for i in range(1,10) for j in range(10) if i+j==somme]
    for el in L :
        if el%14 == 0 :
            Liste.append(el)
    return Liste

print(nombreMystere(14))
#renvoie [770]
print(nombreMystere(10))
#renvoie [280, 910]
```

Exercice 2 : Pour la question 1, donnons directement un programme. Au lecteur de le simplifier.

```python
#Question 1
def Diviseurs(N) :   #Sans listes definies par comprehension
    Div = []
    for d in range(1, N+1) :
        if N%d == 0 :
            Div.append(d)
            Div.append(-d)
    return Div

def SommeDiviseurs(Liste) :
    return sum([Liste[i] for i in range(len(Liste)) if Liste[i]>0])

N = int(input("Saisir un entier N : "))
print("Diviseurs de N : ",Diviseurs(N))
print("Somme des diviseurs positifs de N :",SommeDiviseurs(Diviseurs(N)))
```

La question 2 demande juste une petite précision : le nombre $N = 88\ldots888$ écrit avec n chiffres 8 dans le système décimal s'écrit : $N = 8 \times 10^{n-1} + 8 \times 10^{n-2} + \cdots + 8 \times 10 + 8$. Le processus pour passer de N avec n 8 à N avec $n+1$ 8 est donc : $N \longleftarrow 10N + 8$. Sachant ceci, nous pouvons programmer le script :

```python
#Question 2
def NombreInconnu() :
    N,n = 8,1
    while N%4264 != 0 :
        N = 10*N + 8
        n += 1
    return n

print(NombreInconnu())
#on trouve n=30
```

Nous traiterons cet exercice d'une manière théorique en se servant de la décomposition en facteurs premiers. Remarquons que $4264 = 2^3 \times 13 \times 41 = 8 \times 13 \times 41$. Affaire à suivre ...

Exercice 3 : Nous allons commencer par le test de bonne saisie d'un nombre dans une base $b \geq 2$ donnée. Un tel nombre ne peut avoir de chiffres supérieurs ou égaux à b dans son écriture. Par exemple, saisir 16243 n'a pas de sens en base inférieure ou égale à 6.

```
def Saisie(b,n) :
    mot = str(n)
    condition = [0<=int(mot[i])<b for i in range(len(mot))]
    return condition
```

Passons maintenant à l'addition :

```
def addition(b,n1,n2) :  #on convertit d'abord n1 et n2 en listes
    ch1 = str(n1)          #en passant par un intermediaire str
    L1 = [int(ch1[i]) for i in range(len(ch1))]
    ch2 = str(n2)
    L2 = [int(ch2[i]) for i in range(len(ch2))]
    if len(L1)<len(L2) : #de sorte que L1 et L2 aient meme longueur
        L1 = [0 for i in range(len(L2)-len(L1))]+L1
    else :
        L2 = [0 for i in range(len(L1)-len(L2))]+L2
    i = len(L1)-1          #on part des unites de L1 et L2
    Somme = [0 for i in range(len(L1))]
    retenue = 0
    for i in range(len(L1)-1,-1,-1) :
        Somme[i] = L1[i]+L2[i] + retenue
        if Somme[i] >= b :  #Pour que Somme[i] soit bien defini
            Somme[i] -= b
            retenue = 1
        else :
            retenue = 0
    if retenue == 0 :       #La derniere des retenues eventuelles
        return Somme    #Somme renvoyee sous forme de liste
    else :
        return [1] + Somme
```

Exercice 4 : Commençons par écrire le nombre n recherché en base 9 : $n = \overline{ABC}^{(9)}$, avec $1 \leq A \leq 8$ et $0 \leq B, C \leq 8$. On cherche toutes les permutations σ différentes de l'identité telles que $n = \overline{\sigma(A)\sigma(B)\sigma(C)}^{(13)}$.
Formellement, $n = C + 9B + 81A = \sigma(C) + 13\sigma(B) + 169\sigma(A)$. Nous allons nous servir de la fonction conversionBase10b(n,b) du cours.

```
def mystere() :
    L = []
    for A in range(1,9) :    #A ne peut etre nul
        for B in range(9) :
            for C in range(9) :
                n = 81*A+9*B+C
                perm = {(A,C,B),(B,A,C),(B,C,A),(C,A,B),(C,B,A)}
                for el in perm :
                    if n == 169*el[0]+13*el[1]+el[2] :
                        L.append(n)
    return L

print("En base 10 : ", mystere())
L1 = [conversionBase10b(el,9) for el in mystere()]
L1bis = [str(el[0])+str(el[1])+str(el[2]) for el in L1]
print("En base 9 : ", [int(el) for el in L1bis])
L2 = [conversionBase10b(el,13) for el in mystere()]
L2bis = [str(el[0])+str(el[1])+str(el[2]) for el in L2]
print("En base 13 : ", [int(el) for el in L2bis])
```

Exercice 5 :

1. $4 \equiv -1[5]$, donc pour tout entier naturel n, (corollaire 1-3-6) on a :
   $4^{3n} - 4^n \equiv (-1)^{3n} - (-1)^n = 0[5]$. Donc 5 divise $4^{3n} - 4^n$ pour tout entier naturel n.
2. $\begin{cases} 2^4 \equiv 1[5] \Rightarrow 2^{4n+2} \equiv 1^n \times 4 \equiv -1[5] \\ 2^{4n+1} \equiv 2[5] \\ -1 \equiv -1[5] \end{cases}$, donc $2^{4n+2} + 2^{4n+1} - 1 \equiv -1 + 2 - 1 = 0[5]$
3. $15 \equiv 6[9]$, donc $4^n + 15n - 1 \equiv 4^n + 6n - 1[9]$. Utilisons Python pour évaluer le reste de la division euclidienne des entiers $4^n + 6n - 1$ par 9. En vertu du corollaire 1-3-5, il suffit de considérer $n = 0, 1, \ldots, 8$.

   ```
   def Modulo() :
       return [(4**n+6*n-1)%9 for n in range(9)]

   print(Modulo())
   ```

   La liste de sortie est une liste uniquement composée de 0, ce qui achève la preuve.
4. $A_n := n^2(n^4 - 1) = n^2(n^2 - 1)(n^2 + 1) = n^2(n - 1)(n + 1)(n^2 + 1)$.
   Remarquons que l'écriture $A_n = (n^2-1)n^2(n^2+1)$ constituée de trois entiers consécutifs nous assure que A_n est un multiple de 3. Comme, $60 = 3 \times 4 \times 5$, et que $3, 4, 5$ sont premiers deux à deux (*cf* section nombres premiers), il reste à vérifier que $A_n \equiv 0[4]$ et $A_n \equiv 0[5]$.
   Comme $A_n = n(n^5 - n)$, A_n est multiple de $n^5 - n$, donc d'après le petit théorème de Fermat, A_n est divisible par 5.
   De même, $A_n = n(n^4 - n)$, donc A_n est divisible par 4. Ce qui achève la preuve.

Exercice 6 : Nous pouvons reformuler cet énoncé, puisque nous travaillons uniquement avec des entiers par :
Déterminer le plus grand entier naturel n divisible par tous les entiers naturels non nuls inférieurs ou égaux à $E(\sqrt[3]{n})$, où $E(x)$ désigne la partie entière du nombre réel x.

Le script Python qui suit nous donne l'idée du résultat à trouver : 420.

```
def recherche(N) :
    from math import floor  #partie entiere d'un reel
    L = []
    for n in range(1,N) :
        D = [i for i in range(1,floor(n**(1/3))+1)] #E(racine cubique de n)
        test = [n%el==0 for el in D]
        if test == [True for i in range(len(test))]: #la condition demandee
            L.append(n)
    return L

N = int(input("Borne sup de recherche ? : "))
print(recherche(N))

#Avec N=1000, on trouve 420
#On teste N =10 000 : meme chose !
#N = 100 000 : idem !
#Il semblerait que la reponse soit 420
```

Nous souhaitons prouver que l'ensemble $\boxed{\mathcal{A} := \{n \in \mathbb{N}^* \ ; \forall d \in [\![1; E(\sqrt[3]{n})]\!], \ d|n\}}$ est majoré et que sa borne supérieure est 420.

Déjà, $\mathcal{A} \neq \emptyset$ car $1 \in \mathcal{A}$.
De plus, la partie "$\forall d \in [\![1; E(\sqrt[3]{n})]\!],\ d|n$" nous fait inévitablement penser au PPCM de ces entiers d ; d'où l'idée d'introduire la suite $u_n := \text{PPCM}(1, 2, \ldots, n)$.

Ainsi :

$$\boxed{\mathcal{A} := \{n \in \mathbb{N}^* \ ;\ u_{E(\sqrt[3]{n})} | n\}}$$

Cette dernière ré-écriture de l'ensemble $\mathcal{A}$ nous amène inévitablement à constater quelques propriétés de la suite (u_n) :

1. $u_1 \mid u_2 \mid u_3 \mid \ldots$ En particulier, u est croissante,
2. Si p est premier, alors $u_p = pu_{p-1}$,
3. Excepté u_1 qui vaut 1, tous les u_k sont pairs.

Il est prouvé (*cf* O.Bordellès - Thèmes d'arithmétique - Ellipses) que pour $n \geq 2,\ u_n \geq 2^{n-2}$, ce qui résout notre problème. En effet :

Remarque fondamentale : si l'on prouve qu'*à partir d'un certain entier* n, $n \notin \mathcal{A}$, alors ce sera gagné ! Et c'est là où notre script initial intervient : il réduira les cas jusqu'à $n = 420$. Mais lourde artillerie quand même ! Nous qui aimons la beauté ...

Reprenant donc le cours de nos pensées, la condition "plus petit que $E(\sqrt[3]{n})$" fait penser à "racine cubique", donc "cube" ! D'où l'idée de partitionner $\mathbb{N}^*$ en intervalles faisant intervenir des cubes d'entiers : $\mathbb{N}^* = \bigsqcup_{k=1}^{\infty} A_k$, où $A_k := [\![k^3; (k+1)^3 - 1]\!]$.

Il fallait donc continuer l'exploration. La suite $v_n := u_{E(\sqrt[3]{n})}$ semblait jouer un rôle particulier et il était question de la comparer à la suite de terme générique $w_n := n$.

Confrontant cette idée de partition de $\mathbb{N}^*$ avec la suite (u_n), mais aussi au fait que $420 = 2^2 \times 3 \times 5 \times 7 = u_7$, que $u_8 = 2^3 \times 3 \times 5 \times 7$ (bref utiliser la décomposition en facteurs premiers des termes de (u_n)), nous résumons ceci dans le tableau suivant :

k	A_k	v_n si $n \in A_k$	dans A_k, v_n divise ...
1	$[\![1; 7]\!]$	1	1, 2, 3, 4, 5, 6, 7
2	$[\![8; 26]\!]$	2	8, 10, 12, 14, 16, 18, 20, 22, 24, 26
3	$[\![27; 63]\!]$	$6 = 2 \times 3$	30, 36, 42, 48, 54, 60
4	$[\![64; 124]\!]$	$12 = 2^2 \times 3$	72, 84, 96, 108, 120
5	$[\![125; 215]\!]$	$60 = 2^2 \times 3 \times 5$	180
6	$[\![216; 342]\!]$	$60 = 2^2 \times 3 \times 5$	240, 300
7	$[\![343; 511]\!]$	$420 = 2^2 \times 3 \times 5 \times 7$	420
8	$[\![512; 728]\!]$	$840 = 2^3 \times 3 \times 5 \times 7$	aucun n de A_k
9	$[\![729; 999]\!]$	$2520 = 2^3 \times 3^2 \times 5 \times 7$	aucun n de A_k
10	$[\![1000; 1330]\!]$	$2520 = 2^3 \times 3^2 \times 5 \times 7$	aucun n de A_k
11	$[\![1331; 1727]\!]$	$27720 = 2^3 \times 3^2 \times 5 \times 7 \times 11$	aucun n de A_k

Il est question de mettre en avant la notion de PPCM, très présente dans cet énoncé. En effet, $v_n = 1 \vee 2 \vee \cdots \vee k_n$, où k_n est l'unique entier naturel non nul tel que $k_n^3 \leq n < (k_n + 1)^3$.

Rappel : Soit $(a, b) \in \mathbb{N}^{*2}$. Alors : $(a \wedge b)(a \vee b) = ab$.

Ce dernier résultat admet une généralisation au cas de plusieurs entiers. Citons celui où ils sont quatre :

Lemme : Soit $(a, b, c, d) \in \mathbb{N}^{*4}$ tous distincts. Notons :

1. $M = a \vee b \vee c \vee d$ leur PPCM,
2. $D_1 = a \wedge b \wedge c \wedge d$ leur PGCD,
3. $D_2 = (a \wedge b)(a \wedge c)(a \wedge d)(b \wedge c)(b \wedge d)(c \wedge d)$ le produit des six PGCD de ces quatre nombres pris deux à deux,
4. $D_3 = (a \wedge b \wedge c)(a \wedge b \wedge d)(a \wedge c \wedge d)(b \wedge c \wedge d)$ le produit des quatre PGCD des ces quatre nombres pris trois à trois.

Alors : $MD_1D_2 = PD_3$, soit $M = \dfrac{PD_3}{D_1D_2}$.

La démonstration est laissée au lecteur. On pourra par exemple considérer la valuation p-adique $v_p(x)$ dans la décomposition en facteurs premiers d'un entier x et se ramener, quitte à les permuter au cas où $v_p(a) \leq v_p(b) \leq v_p(c) \leq v_p(d)$.

Soit n assez grand (nous préciserons par la suite). Posons $a = k_n - 3,\ b = k_n - 2,\ c = k_n - 1$ et $d = k_n$. Comme a et b sont deux entiers consécutifs, $a \wedge b = 1$. De même, $b \wedge c = c \wedge d = 1$. D'où $D_1 = (a \wedge b) \wedge (c \wedge d) = 1 \wedge 1 = 1$.
$D_2 = (a \wedge c)(a \wedge d)(b \wedge d)$ et $D_3 = 1$ comme D_1.
Ainsi, d'après le lemme,

$$M = \frac{abcd}{(a \wedge c)(a \wedge d)(b \wedge d)}$$

<u>*Cas 1*</u> : k_n est impair. Ainsi, $a = k_n - 3$ est pair (donc c aussi est pair).
De plus, b et d sont deux nombres impairs consécutifs, donc premiers entre eux.
Comme a et c sont deux nombres pairs consécutifs, $a \wedge c = 2$.
On en déduit que $M = \dfrac{abcd}{2(a \wedge d)}$. D'où :

$$a \vee b \vee c \vee d = \frac{(k_n - 3)(k_n - 2)(k_n - 1)k_n}{2[(k_n - 3) \wedge k_n]}$$

Le dénominateur de cette dernière fraction nous amène à étudier $a \wedge (a + 3) = (a + 3) \wedge 3$.
Facilement, si $a \equiv 0[3]$, alors $(a + 3) \wedge 3 = 3$, sinon $(a + 3) \wedge 3 = 1$.
Remarque : Quoi qu'il en soit,

$$\frac{(k_n - 3)(k_n - 2)(k_n - 1)k_n}{6} \text{ divise } a \vee b \vee c \vee d$$

.
Or $a \vee b \vee c \vee d$ divise v_n. D'où $v_n \geq \dfrac{(k_n - 3)(k_n - 2)(k_n - 1)k_n}{6}$.
Mais pour x assez grand ($x > \alpha$, où $\alpha \approx 12,7027$), on a $(x - 3)(x - 2)(x - 1)x > 6 \times (x + 1)^3$ (étude de fonction comme à la preuve 1). Remplaçant x par k_n pour $k_n \geq 13$, nous obtenons que $v_n \geq \dfrac{(k_n - 3)(k_n - 2)(k_n - 1)k_n}{6} > (k_n + 1)^3 > n$.
Ce qui achève la preuve.
<u>*Cas 2*</u> : k_n pair. Se traite à l'identique.

Exercice 7 : Le mieux est de prendre un point de vue récursif.

```
def Euclide(a,b) :
    while a%b != 0 :
        a,b = b,a%b   #servons-nous de l'affectation parallele
    return b

def PGCDn(N,liste) : #La recursivite en action !
    d = Euclide(liste[0],liste[1])
    for el in liste[2:] :
        d = Euclide(d,el)
    return d

#Programme principal
n = int(input("Combien d'entiers a saisir ? "))
L = []
for i in range(1,n+1) :
    print("a_",i,"= ? ")
    a = int(input())
    L.append(a)

print("Le PGCD de ces entiers est : ",PGCDn(n,L))
```

Exercice 8 : Laissé à la sagacité du lecteur / de la lectrice.

3 Compléments utiles

Définition : Soit x un nombre réel. On appelle *partie entière* de x, et l'on note $E(x)$ ou $[x]$ le plus grand entier inférieur ou égal à x.
On appelle *partie décimale* de x le réel noté $\{x\}$ et défini par $\{x\} = x - [x]$.
Exemple : $[3.45] = 3$; $[-3,45] = -4$
$\{3,45\} = 3,45 - 3 = 0,45$; $\{-3,45\} = -3,45 + 4 = 0,55$

Propriétés : Nous nous limiterons aux plus classiques :

1. Pour tout réel x, $[x] \leq x < [x] + 1$,
2. x est un entier si et seulement si $[x] = x$ (ou $\{x\} = 0$)
3. (a) Si $x \in \mathbb{Z}$, alors $[-x] = -[x]$,
   (b) Si $x \in \mathbb{R} \setminus \mathbb{Z}$, alors $[-x] = -[x] - 1$
4. Si $(x, y) \in \mathbb{R}^2$, alors : $[x] + [y] \leq [x + y] \leq [x] + [y] + 1$

Scripts alternatifs : Sur les thèmes de divisibilité, de congruences, etc.

Division euclidienne entre deux entiers naturels :

```
#division euclidienne entre entiers naturels
#script alternatif avec des listes
def division(a,b) :
    if a < b :
        return [0,a]
    else :
        q,r = 0,a
        while r >= 0 :
            q += 1
            r = a-b*q
        return [q-1,a-b*(q-1)] #Attention au decalage

a = int(input("Saisir un entier naturel a : "))
b = int(input("Saisir un entier naturel non nul b : "))
print("a=",division(a,b)[0],"*",b,"+",division(a,b)[1])
```

4 Bibliographie

[1] Daniel Perrin - mathématiques d'école - Cassini (2005)
[2] Gilles Bailly-Maitre - Arithmétique et cryptologie - Ellipses (2021)
[3] Dany Jack Mercier - Codes correcteurs d'erreurs - CSIPP (2014)
[4] Daniel Duverney - Théorie des nombres - Dunod (1998)