FST Mulhouse.

Licence 1 Maths-Info

Mathématiques : ARITHMETIQUE-ALGEBRE

Elisabeth REMM

Chapitre 2

Les entiers, $\mathbb{N}$, $\mathbb{Z}$, arithmétique.

Table des matières

1. L'ensemble $\mathbb{N}$ et le raisonnement par récurrence

Rappelons que $\mathbb{N}$ est l'ensemble des entiers positifs ou nuls. Une de ses propriétés caractéristiques est que si $n \in \mathbb{N}$, son successeur $n+1$ appartient aussi à $\mathbb{N}$. Une autre propriété, moins évidente à priori mais tout aussi importante est la suivante. Si A est une partie infinie ou pas de $\mathbb{N}$, elle admet un plus petit élément a_0, c'est-à-dire vérifiant

$$\forall a \in A,\ a_0 \leq a.$$

Cette propriété est à la base du raisonnement par récurrence, indispensable lorsqu'on veut vérifier une formule indexée par un entier positif. Comme exemple d'une telle formule indexée par un entier positif, citons

$$\forall n \in \mathbb{N}, n \neq 0, \quad 1+2+3+\cdots+(n-1)+n = \frac{n(n+1)}{2}.$$

Il y a deux manières de présenter ce raisonnement par récurrence.

1.1. Le raisonnement par récurrence : première présentation.

Théorème 1. *Soit $P(n)$ une assertion ou une formule proposée pour tout $n \geq 1$. Si on démontre*

(1) (Initialisation) L'assertion $P(1)$ est vraie (c'est-à-dire la formule donnée est vraie pour $n = 1$),

(2) (Hérédité) Pour tout entier $n_0 \geq 1$ donné si l'assertion $P(n_0)$ est vraie, alors l'assertion $P(n_0+1)$ est aussi vraie,

alors nous pouvons conclure que l'assertion $P(n)$ est vraie pour tout $n \geq 1$.

Notons que si l'initialisation se fait non pas pour $n = 1$ mais pour $n = 2$ ou 3 ou $\cdots$, et si l'hérédité est prouvée pour $n_0 \geq 2$ ou 3 ou $\cdots$, alors l'assertion ne sera valable que pour tout $n \geq 2$ ou 3 ou $\cdots$.

Exemple. Démontrons la propriété

$$\forall n \geq 1,\ 1+2+3+\cdots+(n-1)+n = \frac{n(n+1)}{2}.$$

par un raisonnement par récurrence. Pour cela il est nécessaire de bien détailler les étapes. Ici l'assertion $P(n)$ est

$$P(n): \quad 1+2+3+\cdots+(n-1)+n = \frac{n(n+1)}{2}.$$

(1) Initialisation. Si $n = 1$, la formule s'écrit

$$1 = \frac{1.2}{2}.$$

Cette formule est vraie donc $P(1)$ est vérifiée.

(2) Hérédité. Supposons que la formule $P(n)$ soit vérifiée pour un entier n donné, $n \geq 1$ (et surtout ne pas écrire que $P(n)$ est vérifiée pour tout n, ce qui signifierait que l'on suppose la formule toujours juste et rend absurde le raisonnement suivant). Ainsi pour cet entier n donné, on a

$$1+2+3+\cdots+(n-1)+n = \frac{n(n+1)}{2}.$$

Démontrons la formule pour l'entier $n+1$. Calculons

$$1+2+\cdots+n+(n+1).$$

D'après l'hypothèse de récurrence, on en déduit

$$\begin{aligned} 1+2+3+\cdots+n+(n+1) &= \frac{n(n+1)}{2}+(n+1) \\ &= \frac{n(n+1)+2(n+1)}{2} \\ &= \frac{(n+1)(n+2)}{2}. \end{aligned}$$

Ainsi $P(n+1)$ est vraie.

(3) Conclusion. On en déduit que la propriété

$$1+2+3+\cdots+(n-1)+n=\frac{n(n+1)}{2}$$

est vraie quel que soit $n \geq 1$.

1.2. **Le raisonnement par récurrence : deuxième présentation.**

Théorème 2. *Soit $P(n)$ une assertion ou une formule proposée pour tout $n \geq 0$ (ici on a choisi 0 mais cela n'a pas d'importance sauf dans la conclusion). Si on démontre*

(1) (Initialisation) L'assertion $P(0)$ est vraie (c'est-à-dire la formule donnée est vraie pour $n = 0$),

(2) (Hérédité) Pour tout entier $n \geq 0$ donné, si l'assertion $P(k)$ est vraie pour tout entier k tel que $0 \leq k \leq n$, alors l'assertion $P(n+1)$ est aussi vraie,

alors nous pouvons conclure que l'assertion $P(n)$ est vraie pour tout $n \geq 0$.

2. La division euclidienne. Divisibilité. PGCG

Dans tout ce qui suit, on appellera entier tout élément de $\mathbb{Z}$. Les éléments de $\mathbb{N}$ seront dits entiers positifs.

2.1. **La division euclidienne.**

Théorème 3. *Etant donnés deux entiers a et b positifs et $b \neq 0$, il existe un couple unique d'entiers positifs ou nuls q et r avec*

$$0 \leq r < b$$

tels que

$$a = bq + r.$$

Démonstration. La démonstration se fait par récurrence sur l'entier a. Elle pourra se faire en exercice. Ce résultat ne concerne que les entiers positifs. Toutefois il s'étend aux entiers relatifs et permettra dans la suite de décrire une structure algébrique de $\mathbb{Z}$.

Théorème 4. *Etant donnés deux entiers a et b, $a \in \mathbb{Z}$ et $b \in \mathbb{N}^*$, il existe un couple unique d'entiers q et r avec $q \in \mathbb{Z}$ et $r \in \mathbb{N}$ vérifiant*

$$0 \leq r < b$$

tels que

$$a = bq + r.$$

2.2. **Divisibilité. Nombres premiers.** Nous allons revenir brièvement sur les notions de divisibilité, de diviseurs, de nombres premiers et montrer comment trouver tous les diviseurs d'un entier donné à l'aide de PYTHON.

Définition 1. *Soient n et d deux entiers (dans $\mathbb{Z}$) non nuls. Nous dirons que d divise n ou que d est un diviseur de n, s'il existe un entier q tel que*

$$n = dq.$$

Nous dirons également que dans ce cas, n est un multiple de d.

Trouver tous les diviseurs d'un nombre donné n'est pas une affaire immédiate. Dans la dernière section, nous donnons un petit programme sous PYTHON permettant de vérifier si un entier donné est un diviseur d'un autre entier donné et un autre programme déterminant tous les diviseurs d'un entier donné. Par contre, la chose est mieux structurée lorsqu'on s'intéresse aux multiples d'un entier donné. Pour un entier d non nul donné, notons par $J(d)$ l'ensemble des multiples de d. Cet ensemble est non vide car il contient d. De plus, si $n \in J(d)$, alors tout multiple de n est aussi dans $J(d)$. De même si n_1 et n_2 sont des multiples de d, alors $n_1 + n_2$ est aussi dans $J(d)$. Revenons à la notion de diviseur.

Pour simplifier cette procédure, nous allons introduire la notion de nombres premiers.

Définition 2. *Un nombre premier est un entier p supérieur ou égal à 2 qui n'a d'autres diviseurs que 1 et lui même, autrement dit l'équation $p = mn$ où m et n sont des entiers positifs, alors $m = 1$ ou $n = 1$.*

Les premiers nombres premiers sont

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 43, \cdots$$

Le crible d'Eratosthène est le procédé le plus connu qui permet de trouver tous les nombres premiers inférieurs à un certain entier naturel donné n. L'algorithme procède par élimination. On écrit tous les entiers positifs de 2 à n. On commence par 2 qui lui est premier. On en déduit que tous ses multiples, les nombres pairs plus grands que 2 ne le sont pas. On les barre donc dans le tableau. On passe au suivant non barré, qui est alors un nombre premier. A ce niveau c'est l'entier 3 qui est donc premier. On barre tous ses multiples plus grands que 3. On continue ainsi jusqu'à l'entier le plus proche de $\sqrt{n}$ car pour tout entier supérieur à cette borne, les non barrés sont forcément premiers. En effet si on a une décomposition du type $p = d_1 d_2$, soit d_1 soit d_2 est inférieur ou égal à $\sqrt{p}$. A la fin il ne reste que les entiers qui ne sont multiples d'aucun entier, et qui sont donc les nombres premiers plus petits que n.

2.3. **PGCD : plus grand commun diviseur.** Etant donnés deux entiers non nuls m et n, on appelle diviseur commun à m et n tout entier d non nul qui divise à la fois m et n. Un plus grand diviseur commun à m et n et un diviseur commun d positif tel que si d_1 est un autre diviseur commun, alors d_1 divise d. Il est clair qu'un plus grand commun diviseur de m et n existe toujours et est unique. On l'appellera le PGCD de m et n et on écrira souvent $\mathrm{PGCD}(m, n)$.

Dans cette définition les entiers m et n ne sont pas forcément positifs. Mais nous remarquons que, si $m, n \in \mathbb{Z}$,

$$PGCD(m, n) = PGCD(m, -n) = PGCD(-m, n) = PGCD(-m, -n).$$

Nous pouvons donc nous ramener au calcul du PGCD de deux entiers m et n positifs (et non nuls). Toutefois, nous pourrons nous accorder sur $PGCD(0, n) = n$ dès que $n \neq 0$.

Remarquons aussi que si n divise m alors $PGCD(m, n) = n$.

Pour calculer ce PGCD nous pouvons utiliser l'algorithme d'Euclide qui se traduit comme suit :

Soient m et n deux entiers positifs non nuls. On peut supposer $m > n$.

(1) On effectue la division euclidienne de m par n. Soit r_1 le reste de cette division. Il vérifie $0 \leq r_1 < n$.

(2) On effectue la division euclidienne de n par r_1. Soit r_2 le reste de cette division. Il vérifie $0 \leq r_2 < r_1$.

(3) On effectue la division euclidienne de r_1 par r_2. Soit r_3 le reste de cette division. Il vérifie $0 \leq r_3 < r_2$.

(4) On réitière cette procédure. Comme la suite des restes obtenus est strictement décroissante et que tous ces restes sont des entiers, il existe une étape pour laquelle le reste obtenu est nul.

(5) Le PGCD de m et n est le dernier reste non nul obtenu.

Du point de vue algorithmique, la procédure est facile à écrire :
— On effectue la division euclidienne de m par n. Soit r le reste de cette division.
— On remplace m par n et n par r.
— Tant que le reste est non nul, on réitère le procédé.
— Le PGCD est le dernier reste non nul.
Cet algorithme sera utilisé en dernière partie pour déterminer en utilisant le langage PYTHON le PGCD de deux entiers. Il nous reste toutefois à vérifier que l'algorithme d'Euclide fournit bien le PGCD. Cette démonstration repose sur la remarque suivante :

Lemme 1. *Supposons* $1 \leq n \leq m$. *Alors*

$$\mathrm{PGCD}(m, n) = \mathrm{PGCD}(m - n, n).$$

Démonstration. En effet si d est un diviseur commun à m et n, alors

$$m = dm_1, \ n = dn_1$$

et donc

$$m - n = dm_1 - dn_1 = d(m_1 - n_1)$$

ce qui montre que d est un diviseur commun à $m-n$ et n. Inversement, supposons que d soit un diviseur commun à $m-n$ et n, on a alors

$$m-n=dq,\ \ n=dn_1$$

d'où

$$m=dq+n=dq+dn_1=d(q+n_1)$$

et donc d est aussi un diviseur commun à m et n. Ainsi les diviseurs communs à m et n sont les mêmes que les diviseurs communs à $m-n$ et n et donc

$$\mathrm{PGCD}(m,n)=\mathrm{PGCD}(m-n,n).$$

Ce lemme, qui fournit également une méthode algorithmique, mais un peu longue, pour le calcul du PGCD, permet de démontrer la proposition suivante, à la base de l'algorithme d'Euclide :

Proposition 1. *Soit m et n des entiers non nuls, avec $m \geq n$. Soit*

$$m=nq+r,\quad r\leq n$$

la division euclidienne de m par n. Alors

$$\mathrm{PGCD}(m,n)=\mathrm{PGCD}(n,r).$$

Démonstration. Il suffit d'appliquer le lemme précédent après avoir remarqué que $r=m-nq$.

Exemple. Nous allons déterminer le PGCD de 3045 et 300 en utilisant tout d'abord le lemme précédent puis l'algorithme d'Euclide.

- Première méthode :

$$\begin{aligned}
\mathrm{PGCD}(3045,300) &= \mathrm{PGCD}(2745,300)\\
&= \mathrm{PGCD}(2445,300)\\
&= \mathrm{PGCD}(2145,300)\\
&= \mathrm{PGCD}(1845,300)\\
&= \mathrm{PGCD}(1545,300)\\
&= \mathrm{PGCD}(1245,300)\\
&= \mathrm{PGCD}(945,300)\\
&= \mathrm{PGCD}(645,300)\\
&= \mathrm{PGCD}(345,300)\\
&= \mathrm{PGCD}(45,300)\\
&= \mathrm{PGCD}(255,45)\\
&= \mathrm{PGCD}(210,45)\\
&= \mathrm{PGCD}(165,45)\\
&= \mathrm{PGCD}(120,45)\\
&= \mathrm{PGCD}(75,45)\\
&= \mathrm{PGCD}(30,45)\\
&= \mathrm{PGCD}(15,30)\\
&= \mathrm{PGCD}(15,15)\\
&= 15
\end{aligned}$$

- Deuxième méthode (algorithme d'Euclide)

$$\begin{array}{ll} 3045 = 300 \times 10 + 45 & \mathrm{PGCD}(3045, 300) = \mathrm{PGCD}(300, 45) \\ 300 = 45 \times 6 + 30 & \mathrm{PGCD}(300, 45) = \mathrm{PGCD}(45, 30) \\ 45 = 30 \times 1 + 15 & \mathrm{PGCD}(45, 30) = \mathrm{PGCD}(30, 15) \\ 30 = 15 \times 2 + 0 & \mathrm{PGCD}(3045, 300) = 15 \end{array}$$

car 15 est le dernier reste non nul.

On se convainc rapidement que l'algorithme d'Euclide est "plus efficace". Plus précisément, le nombre d'étapes pour avoir la réponse en utilisant cet algorithme est inférieur à cinq fois le nombre de chiffres nécessaire à écrire le plus petit des deux entiers. Dans notre exemple, il faut deux chiffres pour écrire 300 et donc le nombre d'étapes est inférieur à 10 (en fait il en faut 4).

Définition 3. *Deux entiers m et n sont dits premiers entre eux si* $\mathrm{PGCD}(m, n) = 1$.

2.4. **Le théorème de Bézout.** C'est certainement le théorème le plus important de cette partie arithmétique.

Théorème 5. *Soient m et n deux entiers non nuls ($m, n \in \mathbb{Z}^*$) et soit $d = \mathrm{PGCD}(m, n)$. Il existe alors deux entiers $u, v \in \mathbb{Z}^*$ tels que*

$$mu + nv = d.$$

Démonstration. Elle est une conséquence directe de l'algorithme d'Euclide. Celui-ci se présente comme une suite d'identités. On remonte "à l'envers" cette suite et l'identité de Bézout en découle. Au lieu de s'étendre sur le cas général, développons cette procédure sur un exemple, l'illustration de la méthode sera plus accessible.

Exemple. Soient les entiers 3045 et 300. Ecrire l'identité de Bézout relative à ces deux nombres. Nous avons, dans le paragraphe précédent, calculé le PGCD de ces deux nombres :

$$\mathrm{PGCD}(3045, 300) = 15.$$

Reprenons l'algorithme d'Euclide "par le bas". La dernière identité, donnat le PGCD était

$$45 = 30 \times 1 + 15.$$

On en déduit

$$d = 15 = 45 - 30 \times 1.$$

L'identité précédent cette dernière était

$$300 = 45 \times 6 + 30.$$

On en déduit

$$30 = 300 - 45 \times 6$$

D'où

$$d = 15 = 45 - 30 \times 1 = 45 - (300 - 45 \times 6) = -300 + 45 + 45 \times 6$$

d'où

$$d = 15 = -300 + 45 \times 7.$$

(on peut vérifier que cette identité est bien juste.) On remonte encore l'algorithme d'Euclide et on considère l'identité

$$3045 = 300 \times 10 + 45.$$

On en déduit

$$45 = 3045 - 300 \times 10$$

et on remplace 45 dans la dernière relation donnant d :

$$d = 15 = -300 + 45 \times 7 = -300 + (3045 - 300 \times 10) \times 7 = 3045 - 300 - 300 \times 70.$$

Ainsi

$$d = 15 = 3045 \times 7 - 300 \times 71.$$

(On peut vérifier que cette identité est bien vraie). On a bien une identité de la forme

$$d = 3045 \times u + 300 \times v$$

avec

$$u = 7, \quad v = -71.$$

3. Décomposition d'un entier en produit de nombres premiers

3.1. Lemme de Gauss.

Théorème 6. *Soit p un nombre premier et soient m et n deux entiers non nuls tels que p divise le produit mn. Alors p divise m ou p divise n.*

Démonstration. Par hypothèse, il existe un entier q tel que $mn = pq$. Supposons que p ne divise pas m. Comme p est un nombre premier, ses seuls diviseurs sont p et 1 et donc le PGCD de m et p est 1. L'identité de Bézout relative à ces deux entiers s'écrit donc

$$1 = pu + mv.$$

On en déduit

$$n = npu + nmv.$$

Mais p divise le produit nm. Ainsi il existe un entier q tel que $n = pq$. D'où

$$n = npu + pqv = p(nu + qv).$$

On en déduit que p est un diviseur de n.

3.2. Décomposition d'un entier en produit de nombres premiers.

Théorème 7. *Tout entier positif $n \geq 2$ s'écrit comme un produit de nombres premiers (non nécessairement distincts)*

$$n = p_1 p_2 \cdots p_k.$$

Cette décomposition est unique à l'ordre des facteurs près.

Démonstration. Si n est premier, $n = n$ est sa décomposition. Supposons qu'il existe au moins un entier non premier qui n'admette pas de décomposition. Soit m le plus petit de ces nombres. Comme il n'est pas premier, il existe des entiers d_1 et d_2 plus grands ou égaux à 2 tels que $m = d_1 d_2$. Mais d_1 et d_2 sont plus petit strictement que m. Par hypothèse, ils admettent une décomposition en nombre premier et donc m aussi. Ce qui est contraire à notre hypothèse qui est donc absurde. On en déduit l'existence d'une telle décomposition pour tout entier plus grand ou égal à 2. Il nous reste à montrer que cette décomposition est unique. Soit

$$p_1 p_2 \cdots p_k = q_1 q_2 \cdots q_s$$

deux décompositions de n. Alors le nombre premier p_1 divise le produit $q_1 q_2 \cdots q_s$. D'après le lemme de Gauss, soit il divise q_1 et dans ce cas $p_1 = q_1$, soit il divise $q_2 \cdots q_s$. En procédant

par récurrence on en déduit qu'il existe un entier i tel que $p_1 = q_i$. En reindexant les facteurs on peut toujours supposer que $p_1 = q_1$ et après simplification il reste

$$p_2 \cdots p_k = q_2 \cdots q_s.$$

En réitérant ce raisonnement, on déduit

$$k = s$$

et quitte à changer l'ordre des facteurs

$$p_i = q_i,\ i = 1, \cdots, k.$$

Dans la décomposition précédente, il est raisonnable d'exprimer ce produit en regroupant tous les facteurs égaux et en ordonnant les entiers premiers p_i par ordre croissant. dans ce cas la décomposition s'écrit :

$$n = p_1^{s_1} p_2^{s_2} \cdots p_r^{s_r}$$

avec $p_1 < p_2 < \cdots < p_r$ et $s_i > 0$ pour $i = 1, \cdots, r$.

Example.

$$2016 = 2^5 \cdot 3^2 \cdot 7.$$

3.3. **Applications. Calcul du PGCD et du PPCM.** Cette décomposition d'un entier en facteurs premiers permet de déterminer rapidement tous ses diviseurs.

Proposition 2. *Soit $n = p_1^{s_1} p_2^{s_2} \cdots p_r^{s_r}$ un nombre entier strictement positif, avec $s_1, \cdots, s_r \geq 1$. Alors les diviseurs de n sont les entiers de la forme*

$$p_1^{l_1} p_2^{l_2} \cdots p_r^{l_r}$$

avec $0 \leq l_1 \leq s_1,\ 0 \leq l_2 \leq s_2, \cdots, 0 \leq l_r \leq s_r$.

En particulier, cette proposition permet de compter les diviseurs d'un nombre. Soit $n = p_1^{l_1} p_2^{l_2} \cdots p_r^{l_r}$ ce nombre. Pour chaque nombre premier p_i intervenant dans la décomposition, choisissons un entier l_i vérifiant $l_i \leq s_i$. Nous avons donc $s_1 + 1$ choix possible de cet exposant l_i. Ainsi le nombre de diviseurs de $n = p_1^{l_1} p_2^{l_2} \cdots p_r^{l_r}$ est égal à

$$\pi(n) = (s_1 + 1)(s_2 + 1) \cdots (s_r + 1).$$

Examples. Le nombre de diviseurs de 2^{36} est égal à 37. Le nombre de diviseurs de $n = 2^6.3^7$ est $\pi(n) = 7 \times 8 = 56$.

Etant donnés deux entiers positifs m et n, leurs décompositions en nombres premiers permettent de calculer facilement le PGCD de ces deux nombres. Pour cela nous allons extrapoler cette écrite de décomposition en imposant que tous les facteurs premiers de m et n apparaissent dans ces décompositions avec, bien entendu, le facteur p_u^0 si ce nombre premier n'est pas dans la décomposition originelle. L'avantage, uniquement pour l'écriture, est que les deux décompositions font apparaître les mêmes facteurs.

Proposition 3. *Soient $m = p_1^{s_1} p_2^{s_2} \cdots p_r^{s_r}$ et $n = p_1^{v_1} p_2^{v_2} \cdots p_r^{v_r}$ deux entiers avec , $0 \leq s_i$ et $0 \leq v_i$. Alors*

$$\mathrm{PGCD}(m, n) = p_1^{w_1} p_2^{w_2} \cdots p_r^{w_r}$$

avec

$$w_i = \min(s_i, v_i),\ i = 1, \cdots, r.$$

Exemple. Calculons le PGCD de 441 et 777. Les décompositions en facteurs premiers sont

$$441 = 3^2 \times 7^2, \quad 777 = 3 \times 7 \times 37$$

Ainsi

$$\mathrm{PGCD}(441, 777) = 3 \times 7 = 21.$$

Cette décomposition est aussi utile pour déterminer le plus petit commun multiple (PPCM) de deux nombres. Il est clair, qu'étant donnés deux entiers m et n, le produit mn est un multiple commun. Mais il n'est pas en général le plus petit de ces multiples.

Proposition 4. *Soient $m = p_1^{s_1} p_2^{s_2} \cdots p_r^{s_r}$ et $n = p_1^{v_1} p_2^{v_2} \cdots p_r^{v_r}$ deux entiers strictement positifs avec $0 \leq s_i$ et $0 \leq v_i$. Alors*

$$\mathrm{PPCM}(m, n) = p_1^{w_1} p_2^{w_2} \cdots p_r^{w_r}$$

avec

$$w_i = \max(s_i, v_i), \ \ i = 1, \cdots, r.$$

Exemple. Calculons le PPCM de 441 et 777. Les décompositions en facteurs premiers sont

$$441 = 3^2 \times 7^2, \quad 777 = 3 \times 7 \times 37$$

Ainsi

$$\mathrm{PPCM}(441, 777) = 3^2 \times 7^2 \times 37 = 16317.$$

Comme conséquence, on pourra démontrer, à titre d'exercice, le résultat suivant.

Proposition 5. *Soient m et n deux entiers strictement positifs. Alors*

$$\mathrm{PPCM}(m, n) \times \mathrm{PGCD}(m, n) = mn.$$

4. Sur les nombres premiers

Malgrè une définition fort simple de la notion de nombre premier, un nombre premier est un entier positif plus grand ou égal à 2, n'ayant comme diviseurs que 1 et lui même, on ne connaît que très peu de chose sur l'ensemble de ces entiers : leur construction, leur répartition dans $\mathbb{N}$. Nous allons, dans cette section, rappeler les rares résultats classiques connus.

4.1. **Il y a une infinité de nombres premiers.** A titre d'information, les vingt-cinq nombres premiers inférieurs à 100 sont :

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97.$$

Il est assez facile de montrer ceci. Par contre, récemment, on a montré que le nombre

$$2^{77232917} - 1$$

était premier. La démonstration est un peu plus rude.

L'existence d'une infinité de nombres premiers est déjà prouvée par Euclide. On peut la lire dans ses Éléments (proposition 20 du Livre IX). La démonstration d'Euclide est la suivante : soit une famille finie $p_1, \cdots, p_r$ de nombres premiers et considérons le nombre $N = p_1 p_2 \cdots p_r + 1$. Par construction, quand on effectue la division de ce nombre par l'un des p_i on obtient un reste égal à 1. Ce nombre n'admet donc aucun des p_i comme diviseur. S'il n'existait qu'un nombre fini de nombres premiers, par exemple $\{p_1, \cdots, p_r\}$, tout nombre plus grand que tous ces p_i ne serait donc pas premier. Or on vient de voir que $N = p_1 p_2 \cdots p_r + 1$ n'avait aucun p_i comme

diviseur. Il est donc premier et plus grand que chaque p_i. D'où la contradiction. Il existe ainsi une infinité de nombres premiers.

4.2. **La fonction de compte des nombres premiers.** Etant donné un entier positif n on note par $\Pi(n)$ le nombre de nombres premiers inférieurs à n. On n'a pas hélas de formule algébrique concernant cette fonction. On peut toutefois l'étendre en une fonction d'une variable réelle en considérant pour $x \leq 2, x \in \mathbb{R}$ la fonction $\Pi(x)$ égale au nombre de nombres premiers inférieurs à x. On a pour cette fonction le résultat suivant :

Quand $x \to +\infty$, la fonction $\Pi(x)$ est équivalente à la fonction $\dfrac{x}{\ln x}$.

Cela donne une idée sur la proportion des nombres premiers inférieurs à x quand x devient grand.

5. Comment programmer l'arithmétique avec PYTHON

5.1. **Un bref aperçu de PYTHON.** PYTHON est un langage de programmation certainement largement utilisé dans les lycées. Son code est facile à lire et permet du calcul mathématique assez poussé. Un autre avantage et non des moindres, il est gratuit. Pour l'installer, il suffit de se rendre sur le site

http ://www.python.org

et télécharger la dernière version. Une fois l'installation terminée, l'icône **IDLE** donne accès à une fenêtre de programmation et d'exécution portant le nom de **shell Python**. Dans cette fenêtre l'invite des commandes est

>>>

Les opérations essentielles pour l'arithmétique sont

+ pour l'addition:
- pour la soustraction:
*** pour la multiplication:**
// pour la division euclidienne:
% pour le reste de la division euclidienne :
/ pour la division ordinaire:
**** pour la puissance :**

Comme dans ce chapitre nous allons travailler avec des entiers, le langage doit reconnaître les nombres réels et les nombres entiers. Un entier est un nombre sans virgule, un réel avec virgule. Notons la notation **2.3 au lieu de 2,3** pour les réels. Ainsi lorsque nous rentrons l'expression **a** dans le programme, il peut s'agir soit d'un mot d'une lettre, soit d'un nombre réel soit un nombre entier. Python sait reconnaitre de quoi il s'agit (son étiquette ou label). Pour vérifier au sein d'un programme on peut utiliser

(1) >>> type(a) et python répond <'class 'int'> si **a** est entier, <class "float'> s'il est réel
(2) >>> int(a) transforme un réel en entier en enlevant tout ce qui est derrière la virgule
(3) >>> float(a) transforme en réel un entier en rajoutant une virgule suivie d'un 0

Enfin si **a** est un mot (une chaîne de caractères) dont tous ses composants sont des chiffres (en gros c'est un mot mais pas un nombre), alors la commande int(a) (respectivement float(a)) transforme cette chaîne en entier (respectivement réel). Rappelons que les chaînes de caractères (**string**) sont encadrées par des guillemets, par exemple '1' ou 'bonjour').

5.2. **La division euclidienne.** Voici un programme en Python permettant de calculer le quotient et le reste de la division euclidienne de deux nombres positifs entiers donnés

Programme Python division euclidienne

```
def div_eucl(a,b) :
diviseur=a
dividende=b
quotient=b%a
reste =a-(b%a)*b
resultat1= '%s : est le quotient'
resultat2= '%s : est le reste'
conclusion = la division euclidienne de %s par %s donne'
print(conclusion % (diviseur, dividende))
print(resultat1 % quotient
print(resultat2 % reste)
```

Quelques commentaires. **def** permet de définir une fonction (ici div_eucl) à deux variables a et b dont les valeurs seront demandées lors de l'exécution du programme. Le symbole %s qui apparait dans une chaine de caractères est inséré à la place d'une valeur et il est suivi de l'indication où se trouve cette valeur. L'opérateur % dans print permet de remplacer %s par le contenu des variables mentionnées dans %s.