1101001011010101000101011101001101011 01sin$^2\,x+\cos^2\,x=1$0010101011010010101101000011110001010000111001$e^{i\pi}+1=0$1010110100101010110100101011010101011010110101**Tilleuil**0101011010101010111111101101001010110101001011010101010111011010$\sum_{n=1}^{\infty}\frac{1}{n^2}=\frac{\pi^2}{6}$1010110100101011010110100101010110100101010110101000101100101010**Desbonnez**1101010101001101101010$f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$0101101011011010100101011010010110110100101101101101**InforMathématique**0101000101101101001011011010010101
informathematiqueptijmd@gmail.com1010010101101010101010010101101 0**www.informathematique.be**

Factoriellement vôtre

1. Des zéros à la fin du nombre $n!$, combien y en a-t-il ?

Rappelons que $n!$ est le produit des n premiers naturels (non nuls, évidemment). Ces nombre deviennent très vite gigantesques et une calculatrice n'est très vite plus capable de les afficher. Il n'est même plus possible de les nommer... L'informatique devient nécessaire pour le calcul puis l'affichage qui accompagne. Les mathématiques peuvent expliquer, par exemple, le nombre de zéros qui apparaitront à la fin de ces grands nombres, sans les calculer.

On comprend aisément qu'un zéro apparaît suite à une multiplication par 10 ($= 2 \times 5$). Observons le tableau des premières factorielles :

n	$n!$
4	24
5	120
9	362880
10	3628800
14	87178291200 (1)
15	1307674368000
19	121645100408832000
20	2432902008176640000
24	620448401733239439360000
25	15511210043330985984000000

Un premier zéro apparaît pour 5! et reste jusqu'à 9! ; à partir de 10!, un deuxième zéro, qui reste jusqu'à 14! ; à partir de 15!, un troisième, qui reste jusqu'à 19! ; à partir de 20!, un quatrième, qui reste jusqu'à 24!.
À partir de 25!, deux zéros supplémentaires.

Explications

Il est clair qu'on rencontre plus de multiples de 2 que de multiples de 5, et donc seuls les multiples de 5 sont à prendre en compte.
À chaque multiple de 5 apparaît un zéro supplémentaire. En calculant le nombre de multiples de 5 dans n ($\frac{n}{5}$), on a une première estimation du nombre de zéros engendrés.

Pour 25 (5×5), également multiple de 5, ce sont deux zéros qui s'ajoutent : le premier a déjà été comptabilisé par $\frac{n}{5}$, mais il y en a un deuxième, pour le second 5. Un zéro supplémentaire s'ajoutera donc pour chaque multiple de 25, il y en a $\frac{n}{25}$.

(1). Ici s'arrête ma calculatrice, *Python* prend le relai.

Ainsi, pour 100!, on aura $\frac{100}{5} + \frac{100}{25}$ zéros, soit 24.

Vérification : 100! = 93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916864000000000000000000000000 [(2)]. Le compte est bon.

Si n est suffisamment grand, on aura encore un zéro supplémentaire pour les multiples de 125 ($\frac{n}{125}$), puis encore un pour les multiples de 5^4, et ainsi de suite.

Remarque : si une division n'est pas entière, on ne garde que la partie entière.

Exemple : combien de zéros à la fin du nombre 217! ?

$\frac{217}{5} = 43.4$; $\frac{217}{25} = 8.68$; $\frac{217}{125} = 1.736$. On y trouvera donc $43 + 8 + 1 = 52$ zéros.

Vérification : 217! = 21744341135827053912861801359536045325470741138412801209661349413381277516299968827502528265486850752203192140673494841353806544677872704517426784233265702656201573667372889921633491710486887984632676648593648757901855518138946324129802052083666719357821651943732647462742999625314221407462413313272455783383441584377804320870378978237783563475166017959188706099200000000000000000000000000000000000000000000000000. Le compte est bon !

Bien sûr, il faut faire confiance… au rédacteur et à l'ordinateur.

Voici deux codes *Python* pour calculer une factorielle ; le premier utilise le produit des n premiers entiers, le second utilise une fonction qui fait directement le calcul (mais qui nécessite le chargement de la bibliothèque *math*).

```
# CalculFactorielle.py

n = 217

fact = 1
for i in range (1, n+1):
  fact = fact * i

print(n,"! = ", fact)
```

```
# calculFactorielle2.py

import math

nombre = 100

print(math.factorial(nombre))
```

Pour le calcul du nombre de zéros à la fin de la factorielle on peut utiliser soit le tableur (c'est plus visuel pour les détails des résultats des divisions par les puissances successives de 5), soit aussi la programmation en *Python*.

Dans les deux situations, on a calculé le nombre de zéros à la fin de 100 000!. Difficile de vous montrer cette factorielle pour vérification, car elle fait plus de 70 pages…

(2). Merci *Python*.

Par tableur

	A	B	C
1	Nombre de zéros dans n!		
2	n =	100000	
3	réponse :		24999
4	1	5	20000
5	2	25	4000
6	3	125	800
7	4	625	160
8	5	3125	32
9	6	15625	6
10	7	78125	1
11	8	390625	0
12	9	1953125	0
13			

Dans la colonne `A`, les exposants de 5, dans la colonne `B` les puissances de 5, dans la colonne `C` la partie entière de la division de n par la puissance de 5 correspondant, et enfin dans la cellule `C3` la somme des cellules de la colonne `C`.

On peut facilement étendre le calcul à une valeur de n supérieure à 100 000.

Par programmation en *Python*

```python
# NombreDeZeros.py

from math import *

n = 100000
print(f"{"Nombre de zéros à la fin du nombre "}{n}{"!"}")

zeros = 0
k = 1
while 5**k <= n :
  zeros = zeros + n // 5**k
  k += 1
print("il y a ", zeros, "zéros")
```

```
Nombre de zéros à la fin du nombre 100000!
il y a  24999 zéros
```

De ce qui précède, on peut déduire un cas particulier d'une formule de Legendre :

$$\text{nombre de zéros à la fin de } n! = \sum_{i\geqslant 1} \left\lfloor \frac{n}{5^i} \right\rfloor$$

où $\lfloor\cdot\rfloor$ signifie *partie entière du nombre* entre crochets, qui est donc nulle lorsque ce nombre est strictement inférieur à 1.

2. Décomposition de $n!$ en facteurs premiers

Les facteurs premiers p intervenant dans la factorisation sont nécessairement tels que $p \leqslant n$. Il reste à trouver les exposants.

Comme pour le calcul précédent du nombre de zéros, où l'on divise n par les puissances successives de 5, on divisera ici n par les puissances successives de p, $\forall p \leqslant n$, en ne gardant que la partie entière du résultat.

Pour chaque valeur de p, on additionne les parties entières pour obtenir l'exposant de p dans la décomposition de $n!$ en facteurs premiers.

$$\text{L'exposant de } p \text{ dans la décomposition de } n! = \left\lfloor \frac{n}{p} \right\rfloor + \left\lfloor \frac{n}{p^2} \right\rfloor + \left\lfloor \frac{n}{p^3} \right\rfloor + \cdots$$

Exemple pour 10!

Les nombres premiers inférieurs à 10 sont 2, 3, 5 et 7. On va diviser 10 par les puissances successives de ces 4 nombres premiers.

$$\begin{aligned}
\tfrac{10}{2^1} = 5 \quad &\longrightarrow \quad 5 \\
\tfrac{10}{2^2} = 2.5 \quad &\longrightarrow \quad 2 \\
\tfrac{10}{2^3} = 1.25 \quad &\longrightarrow \quad 1 \quad \text{total} : 8 \longrightarrow \text{facteur } 2^8
\end{aligned}$$

$$\begin{aligned}
\tfrac{10}{3^1} = 3.33\ldots \quad &\longrightarrow \quad 3 \\
\tfrac{10}{3^2} = 1.1\ldots \quad &\longrightarrow \quad 1 \quad \text{total} : 4 \longrightarrow \text{facteur } 3^4
\end{aligned}$$

$$\tfrac{10}{5^1} = 2\ldots \quad \longrightarrow \quad 2 \quad \text{total} : 2 \longrightarrow \text{facteur } 5^2$$

$$\tfrac{10}{7^1} = 1.42\ldots \quad \longrightarrow \quad 1 \quad \text{total} : 1 \longrightarrow \text{facteur } 7^1$$

On a donc : $10! = 2^8 \cdot 3^4 \cdot 5^2 \cdot 7^1$

Comme toujours en informathématique, on peut trouver de l'aide informatique pour les calculs.

Par tableur

Décomposition de 10!, puis 50! et 100!.

	A	B	C	D	E	F	G
1	Décomposition de n ! en facteurs premiers						
2	n =	10					
3	nbres premiers p	2	3	5	7	11	13
4	puissances de p						
5	1	5	3	2	1		
6	2	2	1				
7	3	1					
8	4						
9	5						
10	6						
11	7						
12	8						
13	9						
14	10						
15	exposant de p	8	4	2	1		
16							

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Décomposition de n ! en facteurs premiers															
2	n =	50														
3	nbres premiers p	2	3	5	7	11	13	17	19	23	29	31	37	41	43	47
4	puissances de p															
5	1	25	16	10	7	4	3	2	2	2	1	1	1	1	1	1
6	2	12	5	2	1											
7	3	6	1													
8	4	3														
9	5	1														
10	6															
11	7															
12	8															
13	9															
14	10															
15	exposant de p	47	22	12	8	4	3	2	2	2	1	1	1	1	1	1
16																

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	Décomposition de n ! en facteurs premiers																									
2	n =	100																								
3	nbres premiers p	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
4	puissances de p																									
5	1	50	33	20	14	9	7	5	5	4	3	3	2	2	2	2	1	1	1	1	1	1	1	1	1	1
6	2	25	11	4	2																					
7	3	12	3																							
8	4	6	1																							
9	5	3																								
10	6	1																								
11	7																									
12	8																									
13	9																									
14	10																									
15	exposant de p	97	48	24	16	9	7	5	5	4	3	3	2	2	2	2	1	1	1	1	1	1	1	1	1	1
16																										

Par programmation *Python*

Première version :

```python
# FactorisationPremiersFactorielle.py

from math import *
from sympy import isprime

n = 25
print("Décomposition de ",n,"!", "en facteurs premiers")
print("Facteur --- Exposant")

for p in range(2, n+1):
  if isprime(p):
    k = 1
    exposant = 0
    while p**k <= n :
      exposant = exposant + int(n/p**k)
      k += 1
    print(p, "---", exposant)
```

Le résultat affiché est :

```
Décomposition de  25 ! en facteurs premiers
Facteur --- Exposant
2 --- 22
3 --- 10
5 --- 6
7 --- 3
11 --- 2
13 --- 1
17 --- 1
19 --- 1
23 --- 1
```

Seconde version, plus joli... :

```
# FactorisationPremiersFactorielle2.py

from math import *
from sympy import isprime

n = 25
print("Décomposition de ",n,"!", "en facteurs premiers")
# print("Facteur --- Exposant")

fact = [] # liste des facteurs
exp = [] # liste des exposants
compteur = 0

for p in range(2, n+1):
  if isprime(p):
    k = 1
    exposant = 0
    while p**k <= n :
      exposant = exposant + int(n/p**k)
      k += 1
    #print(p, "---", exposant)
    fact.append(p)
    exp.append(exposant)
    compteur += 1
print(n,"! =", end=" ")
for i in range(0, compteur):
  if i != compteur-1 :
    print(f"{fact[i]}{"^"}{exp[i]}{"."}",end="")
  else :
    print(f"{fact[i]}{"^"}{exp[i]}",end=" ")
```

```
Décomposition de  25 ! en facteurs premiers
25 ! = 2^22.3^10.5^6.7^3.11^2.13^1.17^1.19^1.23^1
```

Lorsque n est... grand... la décomposition de $n!$ en facteurs premiers devient très longue. Le programme *Python* qui suit permet d'en extraire une partie. On choisit un intervalle $[pmin, pmax]$, inférieur à n, dans lequel on recherche les nombres premiers ainsi que leurs exposants dans la décomposition en facteurs premiers de $n!$.

Dans l'exemple qui suit, on a recherché les puissances des nombres premiers dans l'intervalle $[4000, 4100]$ dans la factorisation de $500000!$:

```
Puissance de nombres premiers p dans l'intervalle [ 4000 , 4100 ] donné
dans la factorisation de  500000 !
4001 --- 124
4003 --- 124
4007 --- 124
4013 --- 124
4019 --- 124
4021 --- 124
4027 --- 124
4049 --- 123
4051 --- 123
4057 --- 123
4073 --- 122
4079 --- 122
4091 --- 122
4093 --- 122
4099 --- 121
```

Code *Python* :

```python
# PuissanceQuelquesFacteurs.py

from math import *
from sympy import isprime

n = 500000
pmin = 4000
pmax = 4100

print("Puissance de nombres premiers p dans l'intervalle [",pmin,",",pmax,"]",
  "donné")
print("dans la factorisation de ",n,"!")

for p in range(pmin, pmax+1):
  if isprime(p):
    k = 1
    exposant = 0
    while p**k <= n :
      exposant = exposant + n // p**k
      k += 1
    print(p, "---", exposant)
```

Pour la fin de l'histoire, 500000! est un nombre qui se termine par 124 999 zéros...
Si on compte environ 40 lignes de 80 caractères sur une telle page, cela nous fait 39 pages... de zéros.

Pour les insomniaques, il suffit de compter, non pas les moutons, mais les zéros d'une factorielle !

3. Des chiffres dans $n!$, combien y en a-t-il ?

Pour calculer la quantité de chiffres composant un nombre, on pense évidemment au logarithme décimal, noté traditionnellement log, et plus précisément à sa partie entière augmentée de 1.

Exemples :

- $\log(27) = 1.4314\cdots \to 2$
- $\log(4\,321) = 3.6356\cdots \to 4$
- $\log(987\,654\,321) = 8.9946\cdots \to 9$

Alors, comment calculer le logarithme de nombres gigantesques qu'on ne sait même pas introduire dans une calculatrice ?

Depuis peu, on sait factoriser ces nombres, et le logarithme (décimal en particulier) a deux propriétés qui vont nous être bien utiles :

- $\log(p^k) = k \cdot \log(p)$
- $\log(a \cdot b) = \log(a) + \log(b)$

Et par conséquent, puisque $25! = 2^{22} \cdot 3^{10} \cdot 5^6 \cdot 7^3 \cdot 11^2 \cdot 13^1 \cdot 17^1 \cdot 19^1 \cdot 23^1$,

$\log(25!) = 22 \cdot \log(2) + 10 \cdot \log(3) + 6 \cdot \log(5) + 3 \cdot \log(7) + 2 \cdot \log(11) + 1 \cdot \log(13) + 1 \cdot \log(17) + 1 \cdot \log(19) + 1 \cdot \log(23) = 25.190645678835075\ldots$

Il y aurait donc 26 chiffres dans 25!

Vérifions : 25! = 15511210043330985984000000. Le compte est bon.

Pour information, 100 000! est composé de 456 574 chiffres, dont 24 999 zéros à la fin. Quelle (longue) histoire !

Code Python

Il a évidemment suffit de modifier très légèrement le calcul de la factorisation, celui-ci nous donnant déjà les facteurs et les exposants.

```
# NombrDeChiffres.py
# Combien de chiffres dans n! ?

from math import *
from sympy import isprime

n = 100000
print("Combien de chiffres dans ",n,"!")

chiffres = 0 # initialisation
for p in range(2, n+1):
  if isprime(p):
    k = 1
    exposant = 0
    while p**k <= n :
      exposant = exposant + int(n/p**k)
      k += 1
    chiffres = chiffres + exposant * log10(p)
nb_chiffres = int(chiffres) + 1
print(nb_chiffres)
```

À suivre...