Problèmes du Projet Euler avec Python 3

Franck CHAMBON

17 mai 2011

Problèmes issus de Project Euler .net

Spoiler Alert : une partie des solutions est dévoilée.

Les solutions sont proposées en Python 3.

Notations :

Dans les problèmes, on respectera les conventions suivantes :
- Les entiers naturels ne comprennent pas zéro : $\mathbb{N}^*$
- Supérieur et inférieur se comprennent au sens strict.
- Une somme possède au moins deux termes.
- Les triangles ne sont pas dégénérés (aplatis).

Table des matières

I] Problèmes 1 à 50

1 Somme des multiples de 3 ou de 5 inférieurs à 1000

Les entiers naturels inférieurs à 10 qui sont des multiples de 3 ou de 5, sont : 3, 5, 6 et 9, dont la somme est 23.

Trouver la somme de tous les multiples de 3 ou de 5 inférieurs à 1000.

2 Somme des termes pairs de la suite de Fibonacci

Chaque nouveau terme dans la suite de Fibonacci est généré en ajoutant les deux termes précédents. En commençant avec 1 et 2, les 10 premiers termes sont les suivants :

$$1,\quad 2,\quad 3,\quad 5,\quad 8,\quad 13,\quad 21,\quad 34,\quad 55,\quad 89,\quad \ldots$$

Trouver la somme de tous les termes pairs de la suite de Fibonacci n'excédant pas 4 000 000.

3 Plus grand facteur premier de 600 851 475 143

Les facteurs premiers du nombre 13195 sont 5, 7, 13 et 29. Le plus grand est 29.

Trouver le plus grand facteur premier de 600 851 475 143.

4 Nombre palindrome produit de deux nombres à 3 chiffres

Un nombre palindrome est un nombre qui peut être lu dans les deux sens, comme 456654. Le plus grand palindrome qui soit un produit de deux nombres à 2 chiffres est $9009 = 91 \times 99$.

Trouver le plus grand palindrome produit de deux nombres à 3 chiffres.

5 Plus petit entier divisible par les entiers de 1 à 20 ?

2520 est le plus petit entier divisible par chacun des entiers de 1 à 10.

Trouver le plus petit entier divisible par chacun des entiers de 1 à 20.

6 Différence entre la somme des carrés et le carré de la somme

La somme des carrés des dix premiers entiers naturels est : $1^2+2^2+...+10^2 = 385$.
Alors que le carré de la somme est : $(1+2+...+10)^2 = 55^2 = 3025$.
D'où la différence entre la somme des carrés des dix premiers entiers naturels et le carré de leur somme : $3025 - 385 = 2640$.

Trouver la différence entre la somme des carrés des cent premiers entiers naturels et le carré de leur somme.

---

7 Le 10 001^e nombre premier

La liste des dix premiers « nombres premiers » est : 2, 3, 5, 7, 11, 13, 17, 19, 23, 29.
On remarque que le 6^e nombre premier est 13.

Trouver le 10 001^e nombre premier.

---

8 Le plus grand produit

```
73167176531330624919225119674426574742355349194934
96983520312774506326239578318016984801869478851843
85861560789112949495459501737958331952853208805511
12540698747158523863050715693290963295227443043557
66896648950445244523161731856403098711121722383113
62229893423380308135336276614282806444486645238749
30358907296290491560440772390713810515859307960866
70172427121883998797908792274921901699720888093776
65727333001053367881220235421809751254540594752243
52584907711670556013604839586446706324415722155397
53697817977846174064955149290862569321978468622482
83972241375657056057490261407972968652414535100474
82166370484403199890008895243450658541227588666881
16427171479924442928230863465674813919123162824586
17866458359124566529476545682848912883142607690042
24219022671055626321111109370544217506941658960408
07198403850962455444362981230987879927244284909188
84580156166097919133875499200524063689912560717606
05886116467109405077541002256983155200055935729725
71636269561882670428252483600823257530420752963450
```

Trouver le plus grand produit de cinq chiffres consécutifs dans la liste de 1000 chiffres ci-dessus

---

9 L'unique triplet pythagoricien de somme 1000

Un triplet pythagoricien est un ensemble de trois entiers naturels non nuls, $a < b < c$, pour lesquels : $a^2 + b^2 = c^2$. Par exemple : $3^2 + 4^2 = 9 + 16 = 25 = 5^2$. Il existe exactement un triplet de Pythagore tel que $a + b + c = 1000$.

Trouver le produit des nombre a, b et c : $a \times b \times c$.

---

10 Somme des nombres premiers inférieurs à 2 000 000

La somme des nombres premiers inférieurs à 10 est $2 + 3 + 5 + 7 = 17$.

Trouver la somme de tous les nombres premiers inférieurs à deux millions.

---

11 Plus grand produit de quatre nombres dans une grille

Dans la grille de 20×20 ci-dessous, quatre nombres le long d'une ligne diagonale ont été marqués en gras.

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
```

```
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

Le produit de ces nombres est : $26 \times 63 \times 78 \times 14 = 1\,788\,696$.

Trouver le plus grand produit de quatre nombres adjacents dans cette grille 20×20 dans n'importe quelle direction (horizontale, verticale ou diagonale).

---

12 Nombre triangulaire ayant plus de cinq cents diviseurs

La suite des nombres triangulaires est générée en ajoutant les nombres naturels de 1 à n. Ainsi, le 7e nombre triangulaire est : $1+2+3+4+5+6+7 = 28$.
Les dix premiers termes sont donc : 1, 3, 6, 10, 15, 21, 28, 36, 45, 55, ...
En énumérant la liste des diviseurs des sept premiers nombres triangulaires :

$1 \mapsto 1$
$3 \mapsto 1, 3$
$6 \mapsto 1, 2, 3, 6$
$10 \mapsto 1, 2, 5, 10$
$15 \mapsto 1, 3, 5, 15$
$21 \mapsto 1, 3, 7, 21$
$28 \mapsto 1, 2, 4, 7, 14, 28$

Nous pouvons voir que 28 est le premier nombre triangulaire qui a plus de cinq diviseurs.

Trouver le premier nombre triangulaire qui a plus de cinq cents diviseurs.

---

13 Les dix premiers chiffres d'une somme

Dans le fichier `PE-013-data.txt` (clic-droit, ouvrir le lien), chaque ligne représente un nombre.

Trouver les dix premiers chiffres de la somme des nombres.

---

14 Durée de vol à Syracuse

Une suite de Syracuse est définie par son premier terme et la récurrence :

$$\begin{cases} \text{(si } u_n \text{ est pair)} & u_{n+1} = \dfrac{u_n}{2} \\ \text{(si } u_n \text{ est impair)} & u_{n+1} = 3u_n + 1 \end{cases}$$

En utilisant la règle ci-dessus et en commençant par 13, on génère la suite :

$$13 \mapsto 40 \mapsto 20 \mapsto 10 \mapsto 5 \mapsto 16 \mapsto 8 \mapsto 4 \mapsto 2 \mapsto 1$$

Cette suite,débutant avec 13 et finissant avec 1, contient 10 termes.
Bien que cela n'ait pas encore été démontré (Conjecture de Syracuse ou de Collatz), on remarque que toutes les suites finissent par 1, quel que soit le nombre de départ.

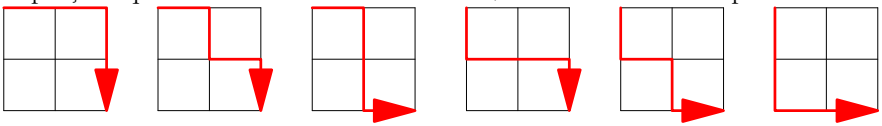
Trouver le nombre de départ inférieur à un million pour lequel la suite formée est la plus longue.

Remarque : Après le départ, les termes sont autorisés à aller à un million ou plus.

---

15 Nombre de routes en ville

Du coin supérieur gauche d'une grille 2×2 au coin inférieur droit, en ne se déplaçant que vers la droite ou vers le bas, il existe 6 chemins possibles.



Trouver le nombre de chemins possibles sur une grille 20×20.

---

16 La somme des chiffres de 2^{1000}

$2^{15} = 32768$ et la somme de ses chiffres est $3 + 2 + 7 + 6 + 8 = 26$.

Trouver la somme des chiffres de 2^{1000}.

---

17 Écrire tous les nombres de 1 à 1000 en anglais

Si les nombres de 1 à 5 sont écrits sous forme de mots en anglais :

one, *two*, *three*, *four*, *five*.

Il y a $3 + 3 + 5 + 4 + 4 = 19$ lettres au total.

Trouver le nombre de lettres utilisées pour écrire en anglais tous les nombres de 1 à 1000 (*one thousand*).

Remarque : Il ne faut pas compter les espaces ou les tirets. Par exemple, 342 (*three hundred and forty-two*) contient 23 lettres et 115 (*one hundred and fifteen*) contient 20 lettres.
L'utilisation de *and* à l'écriture d'un nombre est en conformité avec l'usage britannique.

---

18 Somme maximale d'un trajet dans un triangle

3
7 4
2 **4** 6
8 5 **9** 3

En commençant par le sommet du triangle ci-dessus et en se déplaçant vers des numéros adjacents sur la ligne en-dessous, le total maximal de haut en bas est de $3 + 7 + 4 + 9 = 23$.

75
95 64
17 47 82
18 35 87 10
20 04 82 47 65
19 01 23 75 03 34
88 02 77 73 07 63 67
99 65 04 28 06 16 70 92
41 41 26 56 83 40 80 70 33
41 48 72 33 47 32 37 16 94 29
53 71 44 65 25 43 91 52 97 51 14
70 11 33 28 77 73 17 78 39 68 17 57
91 71 52 38 17 14 91 43 58 50 27 29 48
63 66 04 68 89 53 67 30 73 16 69 87 40 31
04 62 98 27 23 09 70 98 73 93 38 53 60 04 23

Trouver le total maximal de haut en bas du triangle ci-dessus.
Remarque : Dans le triangle ci-dessus il n'y a que 16384 routes, donc il est possible de le résoudre par la force brute, ce qui n'est pas le cas avec l'exercice 67 dont le triangle contient 100 lignes (1×10^{12} routes) et donc nécessite une méthode plus intelligente !

---

19 Dimanches tombés le premier jour du mois au xx^{e} s.

On vous donne les informations suivantes, mais si vous préférez vous pouvez faire quelques recherches pour vous :
- Le 1er janvier 1900 était un lundi.
- Septembre, avril, juin, novembre : ont 30 jours chacun et les autres ont 31 jours, sauf février qui a 28 jours sauf dans les années bissextiles où il en a 29.
- Les années bissextiles sont multiples de 4 mais pas de 100 sauf celle de 400.

Trouver le nombre de dimanches qui sont tombés sur le premier jour du mois au cours du xx^{e} siècle (1 janvier 1901 — 31 décembre 2000).

---

20 Somme des chiffres du nombre 100!

$n!$ signifie $n \times (n-1) \times \cdots \times 3 \times 2 \times 1$. Par exemple, $10! = 10 \times 9 \times \ldots \times 3 \times 2 \times 1 = 3628800$, et la somme des chiffres de 10! est $3+6+2+8+8+0+0 = 27$.

Trouver la somme des chiffres du nombre 100!

---

21 Nombres amicaux inférieurs à 10 000

Soit $d(n)$ la somme des diviseurs propres de n. Si $d(A) = B$ et $d(B) = A$, avec $A \neq B$, alors A et B forment une paire amicale et chacun des nombres A et B est appelé un nombre amical.

Par exemple, les diviseurs propres de 220 sont : 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 et 110 ; donc $d(220) = 284$. Les diviseurs propres de 284 sont : 1, 2, 4, 71 et 142, de sorte que $d(284) = 220$, donc 220 et 284 sont amicaux.

Trouver la somme de tous les nombres amicaux inférieurs à 10 000.

---

22 Somme des poids des mots

Le fichier `PE-022-data.txt` contient plus de cinq mille noms. Commencer d'abord, par les trier dans l'ordre alphabétique. Remplacer ensuite chaque caractère par sa valeur alphabétique et faire la somme pour chaque mot. Multiplier ensuite, la somme par la position du mot dans la liste pour obtenir le « poids du mot ».
Par exemple, lorsque la liste est triée en ordre alphabétique, COLIN, qui vaut $3 + 15 + 12 + 9 + 14 = 53$, est classé 938^{e} dans la liste.
Donc, COLIN aura un « poids de mot » de $938 \times 53 = 49714$.

Trouver la somme de tous les « poids de mot » du fichier.

---

23 Nombres égaux à la somme de deux nombres abondants

Un nombre parfait est un nombre dont la somme des diviseurs propres est exactement égale à ce nombre. Par exemple, la somme des diviseurs propres de 28 est : $1+2+4+7+14 = 28$, ce qui signifie que 28 est un nombre parfait.
Un nombre n est appelé déficient si la somme de ses diviseurs propres est inférieure à n et il est appelé abondant si cette somme est supérieure à n.

Sachant que 12 est le plus petit nombre abondant, $1 + 2 + 3 + 4 + 6 = 16$, le plus petit nombre qui puisse s'écrire comme la somme de deux nombres abondants est 24.
Par l'analyse mathématique, il a été démontré que tous les entiers supérieurs à 28 123 peuvent être écrits comme la somme de deux nombres abondants. Toutefois, cette limite supérieure ne peut pas être diminuée davantage par l'analyse, même si on sait que le plus grand nombre qui ne peut être exprimé comme la somme de deux nombres abondants est inférieur à cette limite.

Trouver la somme de tous les entiers positifs qui ne peuvent pas être écrits comme somme de deux nombres abondants.

---

24 Millionième permutation lexicographique des dix chiffres

Une permutation est un arrangement ordonné d'objets. Par exemple, 3124 est une permutation possible des chiffres 1, 2, 3 et 4. Si toutes les permutations sont énumérées dans l'ordre alphabétique ou numérique, nous l'appelons l'ordre lexicographique. Les permutations lexicographiques de 0, 1 et 2 sont :

012 021 102 120 201 210

Trouver la millionième permutation lexicographique des chiffres 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9.

---

25 Terme de la suite de Fibonacci de plus de 1000 chiffres

La suite de Fibonacci est définie par la relation de récurrence :

$$\text{Pour } n > 2, \quad F_n = F_{n-1} + F_{n-2} \quad \text{, où } F_1 = 1 \text{ et } F_2 = 1.$$

Les 12 premiers termes sont donc les suivants :

i	1	2	3	4	5	6	7	8	9	10	11	12
F_i	1	1	2	3	5	8	13	21	34	55	89	144

Le terme F_{12}, est le premier terme qui contient trois chiffres.

Trouver le premier terme dans la suite de Fibonacci qui contient 1000 chiffres.

---

26 d pour lequel $1/d$ contient le plus long cycle récurrent

Une fraction unitaire possède 1 comme numérateur. La représentation décimale des fractions avec les dénominateurs 2 à 10 est :

Fraction	1/2	1/3	1/4	1/5	1/6	1/7	1/8	1/9	1/10
Écr. déc.	0,5	0,0(3)	0,25	0,2	0,1(6)	0,0(142857)	0,125	0,0(1)	0,1

Où $0{,}1(6) = 0{,}166\,666\ldots$, dispose d'un cycle de chiffres récurrents égale à 1. Il peut être vu que 1/7 a un cycle récurrent de 6 chiffres, donc le cycle récurrent le plus long pour la fraction $1/d$ quand $d < 10$ est pour $d = 7$.

Trouver la valeur de $d < 1000$ pour laquelle $1/d$ contient le cycle récurrent le plus long dans sa partie décimale.

---

27 Formules quadratiques produisant des nombres premiers

Euler a publié la formule quadratique remarquable : $n^2 + n + 41$. Il s'avère que cette formule peut produire 40 nombres premiers pour les valeurs consécutives n de 0 à 39. Toutefois, lorsque $n = 40$, $40^2 + 40 + 41 = 40(40 + 1) + 41$ est divisible par 41, et certainement lorsque $n = 41$, $41^2 + 41 + 41$ est clairement divisible par 41.
Avec l'utilisation des ordinateurs, on a trouvé l'incroyable formule $n^2 - 79n + 1601$, qui produit 80 nombres premiers pour les valeurs consécutives n de 0 à 79. Le produit des coefficients, -79 et 1601, est -126479.
Considérant une formule de la forme : $n^2 + An + B$, avec $|A| < 1000$ et $|B| < 1000$ où $|n|$ est la valeur absolue de n, par exemple $|11| = 11$ et $|-4| = 4$.

Trouver le produit des coefficients A et B, pour l'expression du second degré qui produit le nombre maximum de nombres premiers pour les valeurs consécutives en commençant avec $n = 0$.

---

28 Somme des diagonales sur une spirale

En commençant avec 1 et en se déplaçant dans le sens horaire, une grille de 5 par 5 est constituée comme suit :

```
21 22 23 24 25
20 07 08 09 10
19 06 01 02 11
18 05 04 03 12
17 16 15 14 13
```

On peut vérifier que la somme des nombres sur les diagonales est de 101.

Trouver la somme des nombres sur les diagonales d'une grille de 1001 par 1001 formée de la même manière.

---

29 Termes distincts dans la suite générée par a^b

Considérons toutes les combinaisons entières de a^b pour $2 \leqslant a \leqslant 5$ et $2 \leqslant b \leqslant 5$:

$$\begin{array}{llll} 2^2 = 4, & 2^3 = 8, & 2^4 = 16, & 2^5 = 32 \\ 3^2 = 9, & 3^3 = 27, & 3^4 = 81, & 3^5 = 243 \\ 4^2 = 16, & 4^3 = 64, & 4^4 = 256, & 4^5 = 1024 \\ 5^2 = 25, & 5^3 = 125, & 5^4 = 625, & 5^5 = 3125 \end{array}$$

Si elles sont ensuite placées dans l'ordre numérique, en enlevant les répétitions, nous obtenons la séquence suivante de 15 termes distincts :

$$4, 8, 9, 16, 25, 27, 32, 64, 81, 125, 243, 256, 625, 1024, 3125$$

Trouver le nombre de termes distincts dans la séquence générée par a^b pour $2 \leqslant a \leqslant 100$ et $2 \leqslant b \leqslant 100$.

---

30 Égaux à la somme des puissances cinquième de leurs chiffres

Étonnamment, il y a seulement trois nombres qui peuvent s'écrire comme la somme des puissances quatrième de leurs chiffres :
$1634 = 1^4 + 6^4 + 3^4 + 4^4 \quad 8208 = 8^4 + 2^4 + 0^4 + 8^4 \quad 9474 = 9^4 + 4^4 + 7^4 + 4^4$
Comme $1 = 1^4$ n'est pas une somme, il n'est pas inclus. La somme de ces nombres est $1634 + 8208 + 9474 = 19316$.

Trouver la somme de tous les nombres qui peuvent s'écrire comme la somme des puissances cinquième de leurs chiffres.

---

31 Façons d'obtenir 2 € avec de la monnaie

En Europe, la monnaie est constituée d'euros (€), et centimes (c), et il existe huit pièces en circulation : 1c, 2c, 5c, 10c, 20c, 50c, 1 € (100c) et 2 € (200c). Il est possible d'avoir 2 € de la manière suivante :

$$1\times 1\,€ + 1\times 50c + 2\times 20c + 1\times 5c + 1\times 2c + 3\times 1c$$

Trouver le nombre de façons d'obtenir 2€ avec n'importe quel nombre de pièces.

---

32 Somme des produits pan-digitaux

Nous dirons qu'un nombre à n chiffres est pan-digital s'il utilise tous les chiffres de 1 à n exactement une fois. Par exemple, le nombre à 5 chiffres, 15234, est de 1 à 5 pan-digital.
L'ensemble des deux nombres 39 et 186 et leur produit 7254 est de 1 à 9 pan-digital $(39 \times 186 = 7254) \mapsto 391867254$.

Trouver la somme de tous les produits dont l'ensemble multiplicande – multiplicateur – produit et un nombre 1 à 9 pan-digital.

Remarque : Certains produits peuvent être obtenus de plus d'une façon, alors assurez-vous de ne les inclure qu'une fois dans votre somme.

---

33 Fausses simplifications de fraction

La fraction $\frac{49}{98}$ est curieuse, en enlevant le 9 comme dans une fausse simplification, on obtient $\frac{4}{8}$ qui est parfaitement égale à $\frac{49}{98}$. Nous considérerons les fractions comme $\frac{30}{50} = \frac{3}{5}$, des exemples triviaux, et dans $\frac{49}{98} = \frac{4}{8}$, un exemple non trivial.
Il y a exactement quatre exemples non triviaux de ce type de fraction inférieure à 1 en valeur, et contenant deux chiffres au numérateur et au dénominateur. On fait le produit de ces 4 fractions et on la rend irréductible.

Trouver la valeur du dénominateur de cette fraction.

---

34 Égaux à la somme des factorielles de leurs chiffres

On remarque que le nombre 145 est égal à la somme des factorielles de ses chiffres, en effet $1! + 4! + 5! = 1 + 24 + 120 = 145$.

Trouver la somme de tous les nombres égaux à la somme des factorielles de leurs chiffres.

Remarque : Ici, $1! = 1$ et $2! = 2$ sont exclus, ce ne sont pas des sommes.

---

35 Nombres premiers circulaires inférieurs à un million

Le nombre 197, est appelé un nombre premier circulaire parce que toutes les rotations de ses chiffres : 197, 971 et 719, donnent des nombres premiers.
Il y a treize nombres premiers circulaires inférieurs à 100 :

2, 3, 5, 7, 11, 13, 17, 31, 37, 71, 73, 79 et 97.

Trouver la quantité de nombres premiers circulaires inférieurs à un million.

---

36 Palindromes en base 10 et en base 2

Le nombre (décimal), $585_{10} = 1001001001_2$ (binaire), est un palindrome dans les bases 2 et 10.

Trouver la somme de tous les nombres inférieurs à un million, qui sont des palindromes en base 10 et en base 2.

Remarque : on ne compte pas les zéros inutiles.

---

37 Des nombres premiers intéressants

Le nombre 3797 possède une propriété intéressante. Il est premier, et il est possible de supprimer ses chiffres de gauche à droite, il demeure premier à chaque étape : 3797, 797, 97, et 7. De même, de droite à gauche : 3797, 379, 37, et 3.

Trouver Trouver la somme des onze nombres premiers qui ont cette même propriété.

Remarque : 2, 3, 5 et 7 ne font pas partie des onze termes.

---

38 Plus grand nombre 1–9-pan-digital spécial

Prenons le nombre 192 et multiplions-le par 1, 2 ou 3 :

$$192 \times 1 = 192$$
$$192 \times 2 = 384$$
$$192 \times 3 = 576$$

La concaténation de chaque produit, donne un nombre 1–9-pan-digital : 192384576. Nous appellerons le produit 192384576 concaténé de 192 et $(1,2,3)$.
De la même manière, en partant de 9 multiplié par $(1,2,3,4,5)$, on obtient 918273645, produit concaténé de 9 par $(1,2,3,4,5)$.

Trouver le plus grand 1–9-pan-digital formé à partir de la concaténation des produits d'un entier par $(1,2,...,n)$ avec $n > 1$.

---

39 Périmètre de triangles rectangles

Si p est le périmètre d'un triangle rectangle de côtés entiers (a,b,c), il y a exactement trois solutions pour $p = 120$. 1 :{20, 48, 52}, 2 :{24, 45, 51} et 3 :{30, 40, 50}.

Trouver la valeur de $p \leqslant 1000$ donnant le nombre maximal de solutions.

---

40 n^e chiffre de la partie décimale d'un nombre irrationnel

La fraction décimale irrationnelle suivante est créée par la concaténation des entiers positifs :

$$0,12345678910\mathbf{1}112131415161718192021\ldots$$

On peut constater que le 12^e chiffre de la partie décimale est 1.
Si d_n représente le n^e chiffre de la partie décimale, on note

$$P = d_1 \times d_{10} \times d_{100} \times d_{1000} \times d_{10000} \times d_{100000} \times d_{1000000}$$

Trouver la valeur de P

---

41 Le plus grand nombre premier pan-digital

On dit qu'un nombre à n-chiffre est pan-digital s'il est composé de tous les chiffres de 1 à n exactement une fois.
Par exemple, 2143 est un nombre pan-digital à 4 chiffres et il est également premier.

Trouver le plus grand nombre premier pan-digital.

---

42 Mots triangulaires dans un fichier

Le n^e nombre triangulaire est donné par la formule : $T(n) = \frac{n(n+1)}{2}$, donc les dix premiers nombres triangulaires sont : 1, 3, 6, 10, 15, 21, 28, 36, 45, 55 ...
En remplaçant chaque lettre d'un mot par sa position en ordre alphabétique et en faisant la somme, nous formons une valeur pour le mot. Par exemple, le mot SKY a pour valeur $19 + 11 + 25 = 55 = T(10)$. Si la valeur du mot est un nombre triangulaire nous appellerons ce mot « un mot triangulaire ».
En utilisant le fichier `PE-042-data.txt` qui contient près de deux mille mots anglais,

Trouver le nombre de mots triangulaires.

---

43 Les nombres pan-digitaux à 10 chiffres intéressants

Le nombre 1406357289, est un nombre pan-digital à 10 chiffres, mais il a aussi une propriété intéressante. Si d_1 est le 1^{er} chiffre, d_2 le 2^e chiffre, et ainsi de suite, on peut noter que :

- $d_2d_3d_4 = 406$ est divisible par 2.
- $d_3d_4d_5 = 063$ est divisible par 3.
- $d_4d_5d_6 = 635$ est divisible par 5.
- $d_5d_6d_7 = 357$ est divisible par 7.
- $d_6d_7d_8 = 572$ est divisible par 11.
- $d_7d_8d_9 = 728$ est divisible par 13.
- $d_8d_9d_10 = 289$ est divisible par 17.

Trouver la somme de tous les pan-digitaux à 10 chiffres ayant cette propriété.

---

44 Paire convenable de nombres pentagonaux

Les nombres pentagonaux sont générés par la formule suivante : $P_n = \frac{n(3n-1)}{2}$.
Les dix premiers nombres pentagonaux sont :

$$1, 5, 12, 22, 35, 51, 70, 92, 117, 145, ...$$

On voit que $P_4 + P_7 = 22 + 70 = 92 = P_8$. Cependant, leur différence, $70 - 22 = 48$, n'est pas pentagonale.
Il existe une paire de nombres pentagonaux, P_j et P_k, tels que leur somme et leur différence soit pentagonale, avec $D = |P_k - P_j|$ minimale.

Trouver la valeur de D.

Modification : chercher plutôt pour que la somme soit minimale.

---

45 Nombre triangulaire également pentagonal et hexagonal

Les nombres triangulaires, pentagonaux, hexagonaux sont générés par les formules :

Triangulaires	$T(n) = n(n+1)/2$	: 1, 3, 6, 10, 15, ...
Pentagonaux	$P(n) = n(3n-1)/2$	: 1, 5, 12, 22, 35, ...
Hexagonaux	$H(n) = n(2n-1)$	: 1, 6, 15, 28, 45, ...

On peut vérifier que : $T(285) = P(165) = H(143) = 40755$.

Trouver le prochain nombre triangulaire qui soit aussi pentagonal et hexagonal.

---

46 Impair égal à un nombre premier et un double de carré

Il a été proposé par Christian Goldbach que tout nombre impair composé peut s'écrire comme la somme d'un nombre premier et du double d'un carré.

$$\begin{aligned} 9 &= 7 + 2 \times 1^2 \\ 15 &= 7 + 2 \times 2^2 \\ 21 &= 3 + 2 \times 3^2 \\ 25 &= 7 + 2 \times 3^2 \\ 27 &= 19 + 2 \times 2^2 \\ 33 &= 31 + 2 \times 1^2 \end{aligned}$$

Il s'avère que la conjecture était fausse.

Trouver le plus petit nombre impair composé qui ne peut s'écrire comme la somme d'un nombre premier du double d'un carré.

---

47 Consécutifs et même nombre de facteurs premiers

Les deux premiers nombres consécutifs qui ont deux facteurs premiers distincts sont :

$$14 = 2 \times 7$$
$$15 = 3 \times 5$$

Les trois premiers nombres consécutifs avec trois facteurs premiers distincts sont :

$$644 = 2^2 \times 7 \times 23$$
$$645 = 3 \times 5 \times 43$$
$$646 = 2 \times 17 \times 19$$

Trouver le plus petit des quatre premiers nombres consécutifs ayant quatre facteurs premiers distincts chacun.

---

48 Derniers chiffres de $1^1 + 2^2 + ... 1000^{1000}$

La somme : $1^1 + 2^2 + \cdots + 10^{10}$ est égale à : 10 405 071 317.

Trouver les 10 derniers chiffres de la somme $1^1 + 2^2 + \cdots + 1000^{1000}$

---

49 Égal à la concaténation des trois termes d'une suite

La suite arithmétique : 1487, 4817, 8147, dans laquelle chacun des termes est incrémenté de 3330, a deux propriétés :
– Chacun des trois termes est premier.
– Chacun des nombres à 4 chiffres et une permutation de l'autre.
Il y a une autre suite ayant cette propriété (dans laquelle chacun des termes a 4 chiffres, et incrémentation de 3330).

Trouver le nombre à 12 chiffres formé par concaténation des trois termes de cette suite.

---

50 Nombre premier ayant la plus longue somme

Le nombre premier 41, peut être écrit comme la somme de six nombres premiers consécutifs :

$$41 = 2 + 3 + 5 + 7 + 11 + 13$$

C'est la plus longue somme de nombres premiers consécutifs formant un nombre premier inférieur à cent.
La plus longue somme de nombres premiers consécutifs formant un nombre premier inférieur à mille, contient 21 termes, et est égal à 953.

Trouver le nombre premier inférieur à un million qui peut s'écrire comme la plus longue somme de nombres premiers consécutifs.

II] Proposition de solution

1 Somme des multiples de 3 ou de 5 inférieurs à 1000

Les entiers naturels inférieurs à 10 qui sont des multiples de 3 ou de 5, sont : 3, 5, 6 et 9, dont la somme est 23.
Trouver la somme de tous les multiples de 3 ou de 5 inférieurs à 1000.

Solution :

On peut faire un test pour chaque entier entre 1 et 1000.

```
S=0
for i in range(1,1000):
  if i%3==0 or i%5==0: S+=i
print(S)
# variante : somme d'une liste générée sous conditions
print( sum([x for x in range(1, 1000) if x%3==0 or x%5==0]) )
```

On peut aussi considérer qu'il s'agit des multiples de 3 plus les multiples de 5, moins les multiples de PPCM(3,5) = 15.
La somme de ces multiples, est liée aux nombres triangulaires.

$$3+6+9+12+\cdots+999 = 3(1+2+3+\cdots+333) = 3\times T(333)$$

Le calcul du PGCD se fait avec l'algorithme d'Euclide, vu au collège.

```
def T(n): return n*(n+1)//2
def pgcd(a, b):
  if b: return pgcd(b, a%b)
  return a
def ppcm(a, b): return a*b//pgcd(a, b)

def PE001(a=3, b=5, m=1000):
  m -=1 # la limite est exclue
  c = ppcm(a, b)
  return a*T(m//a) + b*T(m//b) - c*T(m//c)
print(PE001())
```

Remarque : avec cette méthode, on trouve très rapidement la réponse avec $a = 987456$, $b = 123456$ et $m = 10^{12}$.

2 Somme des termes pairs de la suite de Fibonacci

Chaque nouveau terme dans la suite de Fibonacci est généré en ajoutant les deux termes précédents. En commençant avec 1 et 2, les 10 premiers termes sont les suivants :

1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

Trouver la somme de tous les termes pairs de la suite de Fibonacci n'excédant pas 4 000 000.

Solution :

Une première approche avec affectation simultanée.

```
a, b, S = 1, 2, 0
while a<=M:
    a, b  =  b, b+a
    if a%2==0: S+=a
print(S)
```

Mais en observant les premiers termes, on constate que les termes pairs reviennent 3-périodiquement, ce qui permet d'accélérer le calcul et d'éviter de nombreux tests.

Preuve : avec u_n impair et u_{n+1} pair,

$u_{n+2} = u_{n+1} + u_n$, est impair et

$u_{n+3} = u_{n+2} + u_{n+1} = 2 \times u_{n+1} + u_n$, est impair et

$u_{n+4} = u_{n+3} + u_{n+2} = 3 \times u_{n+1} + 2 \times u_n$, est pair ainsi

```
def PE002(M=4000000):
  a, b, S  =  1, 2, 0
  while b<=M:
    a, b, S  =  a+2*b, 2*a+3*b, b+S
  return S
print(PE002())
```

Pour accélérer encore pour certaines valeurs, on peut partir de 0, 1 pour Fibonacci, et considérer $\begin{pmatrix} 1 & 2 & 0 \\ 2 & 3 & 0 \\ 1 & 0 & 1 \end{pmatrix}$, matrice à diagonaliser.

On déduit que $S_n \simeq \texttt{int}\left(\dfrac{(2+\sqrt{5})^n}{2\sqrt{5}}\right)$, valeur exacte pour $n \leqslant 24$

3 Plus grand facteur premier de 600 851 475 143

Les facteurs premiers du nombre 13195 sont 5, 7, 13 et 29. Le plus grand est 29. Trouver le plus grand facteur premier de 600 851 475 143.

Solution :

Sans disposer de liste de nombres premiers, on commence par diviser n par 2 autant que possible, puis par 3 et les nombres p impairs successifs. Si on arrive à $p^2 > n$, alors n est premier.

```
def PE003(n=600851475143):
  while n%2==0: n//=2
  if n==1: return 2
  p=3
  while p*p<=n:
    if n%p==0: n//=p
    else: p+=2
  return n

print(PE003())
```

Si on dispose de `primeSieve` (voir Pb 8), un générateur de nombre premiers, alors on peut aller plus vite.

```
def PE003(n=600851475143):
  "plus grand facteur premier de n>1"
  prime = primeSieve(n**0.5)
  i, p  = 0, prime[0]
  while True:
    while n%p==0: n//=p
    if n==1: return p
    if p*p>n: return n
    i+=1 ; p=prime[i]

print(PE003())
```

4 Nombre palindrome produit de deux nombres à 3 chiffres

Un nombre palindrome est un nombre qui peut être lu dans les deux sens, comme 456654. Le plus grand palindrome qui soit un produit de deux nombres à 2 chiffres est $9009 = 91 \times 99$.

Trouver le plus grand palindrome produit de deux nombres à 3 chiffres.

Solution :

En base 10, le plus simple pour savoir si un nombre n est un palindrome, est de le trans-typer en chaîne avec `str(n)`, et d'inverser la liste.
`truc[::-1]` signifie « truc : listé du début à la fin, mais à l'envers ».

```
def isPalindromic(truc): return truc==truc[::-1]

res , n  =  0, 3 # n : trois chiffres
for i in range(10**(n-1), 10**n):
   for j in range(10**(n-1), 10**n):
       if isPalindromic(str(i*j)):
           if res<i*j: res=i*j
print(res)
```

La (les) solution maximale $M(i_M, j_M)$ se trouve sur une hyperbole d'équation $xy = i_M j_M$, on peut restreindre la recherche à un demi-coin (par symétrie) situé dans le demi-plan d'équation $x + y \geqslant 2 \times y_P$ pour tout P solution.

```
def PE004(n=3):
  res = [0, 0]
  for tot in range(2*10**n - 2, 2*10**(n-1) - 1, -1):
    for i in range(10**n-1, (tot-1)//2, -1):
      if res[0]<i*(tot-i):
        if isPalindromic(str(i*(tot-i))):
          res[0]=i*(tot-i)
          res[1]=max(res[1], 2*(tot-i))
    if tot<=res[1]: return res[0]
print(PE004())
```

pour $n = 9$ on a 999 900 665 566 009 999
Figures à venir.....

5 Plus petit entier divisible par les entiers de 1 à 20 ?

2520 est le plus petit entier divisible par chacun des entiers de 1 à 10.
Trouver le plus petit entier divisible par chacun des entiers de 1 à 20.

Solution :

Une calculatrice suffit ici.
$P = 1 \times 2 \times 3 \times (\cancel{2} \times 2) \times 5 \times (\cancel{2} \times \cancel{3}) \times 7 \times (2 \times \cancel{2} \times \cancel{2}) \times (\cancel{3} \times 3) \times (\cancel{2} \times \cancel{5}) \times 11 \times (\cancel{2} \times \cancel{2} \times \cancel{3}) \times 13 \times (\cancel{2} \times \cancel{7}) \times (\cancel{3} \times \cancel{5}) \times (2 \times \cancel{2} \times \cancel{2} \times \cancel{2}) \times 17 \times (\cancel{3} \times \cancel{3} \times \cancel{2}) \times 19 \times (\cancel{2} \times \cancel{2} \times \cancel{5})$

```
>>> 2*3*2*5*7*2*3*11*13*2*17*19
```

Mais pour généraliser à n.

```
def PE005(n=20):
  prime=primeSieve(n)
  pi, res  =  len(prime), 1
  f = [0]*pi
  tab = [factor(i) for i in range(2, n+1)]
  for i in range(2, n+1):
    for pa in tab[i-2]:
      f[prime.index(pa[0])] = \
      max(f[prime.index(pa[0])], pa[1])
  for j in range(pi):
    res*=prime[j]**f[j]
  return res
print(PE005())
```

Où on utilise `primeSieve` qui fournit les nombres premiers utiles ici ; ceux inférieurs à n. Ensuite, on enregistre la décomposition en facteurs premiers (grâce à `factor`), on prend alors le maximum des coefficients associés à chaque nombre premier. Ce qui nous permet de construire notre résultat.
Remarque : pour $n = 100$, on obtient en un instant
69 720 375 229 712 477 164 533 808 935 312 303 556 800

`primeSieve, factor, ...` sont dans un module à préparer.

6 Différence entre la somme des carrés et le carré de la somme

La somme des carrés des dix premiers entiers naturels est : $1^2 + 2^2 + ... + 10^2 = 385$.
Alors que le carré de la somme est : $(1 + 2 + ... + 10)^2 = 55^2 = 3025$.
D'où la différence entre la somme des carrés des dix premiers entiers naturels et le carré de leur somme : $3025 - 385 = 2640$.
Trouver la différence entre la somme des carrés des cent premiers entiers naturels et le carré de leur somme.

Solution :

Une première approche naïve

```
s1, s2, n = 0, 0, 100
for i in range(1, n+1):
  s1 = s1+i
  s2 = s2+i*i
print(s1*s1 - s2)
```

Ou de manière plus compacte

```
def PE006(suite=range(1,100+1)):
  return sum(suite)**2 - sum([x**2 for x in suite])
print(PE006())
```

On dispose aussi de formules :

$$\left(\sum_{k=1}^{k=n} k\right)^2 = \left(\frac{n(n+1)}{2}\right)^2 \qquad \text{et} \qquad \sum_{k=1}^{k=n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

Ainsi, $\left(\sum_{k=1}^{k=n} k\right)^2 - \sum_{k=1}^{k=n} k^2 = \frac{n(n+1)(3n^2-n-2)}{12}$

```
n = 100
print(n*(n+1)*(3*n*n-n-2)//12)
```

7 Le 10 001e nombre premier

La liste des dix premiers « nombres premiers » est : 2, 3, 5, 7, 11, 13, 17, 19, 23, 29.
On remarque que le 6e nombre premier est 13.
Trouver le 10 001e nombre premier.

Solution :

Une approche avec le crible d'Ératosthène, version modifiée par Euler.
On a besoin d'une approximation de $p(n)$, le n^e nombre premier pour mettre une limite au crible. On sait que $p(n) \approx n\ln(n)$ et $p(n) \leqslant 1{,}7n\ln(n)$.

```
def primeSieve(limit):
  prime = list(range(limit+1))
  prime[1], p = 0, 2
  while p*p<=limit:
    prime[2*p::p]=[0]*len(prime[2*p::p])
    p+=1
    while prime[p]==0: p+=1
  return [p for p in prime if p!=0]

from math import log
def PE007(n=10001):
  prime=primeSieve(int(1.7*n*log(n)))
  return(prime[n-1])
print(PE007())
```

On crée d'abord une **vraie** liste avec `list(range(limit+1))`, qui va de 0 à `limit` inclus. On remplace par zéro tout nombre qui n'est pas premier ; 1 pour commencer.

On part donc de $p = 2$, premier nombre premier.

On met à 0 les multiples de p à partir du second : `prime[2*p::p]` représente justement ces multiples, la liste de $2p$ jusqu'à la fin, par pas de p.

On incrémente p jusqu'à trouver un nouveau nombre premier ; et on poursuis le crible.

Enfin, on filtre les résultats, `[p for p in prime if p!=0]` représente la liste des nombres du tableau non égaux à zéros, ils n'ont pas été criblés, ce sont les nombres premiers jusqu'à `limit`.

8 Le plus grand produit

```
73167176531330624919225119674426574742355349194934
96983520312774506326239578318016984801869478851843
85861560789112949495459501737958331952853208805511
12540698747158523863050715693290963295227443043557
66896648950445244523161731856403098711121722383113
62229893423380308135336276614282806444486645238749
30358907296290491560440772390713810515859307960866
70172427121883998797908792274921901699720888093776
65727333001053367881220235421809751254540594752243
52584907711670556013604839586446706324415722155397
53697817977846174064955149290862569321978468622482
83972241375657056057490261407972968652414535100474
82166370484403199890008895243450658541227588666881
16427171479924442928230863465674813919123162824586
17866458359124566529476545682848912883142607690042
24219022671055626321111109370544217506941658960408
07198403850962455444362981230987879927244284909188
84580156166097919133875499200524063689912560717606
05886116467109405077541002256983155200055935729725
71636269561882670428252483600823257530420752963450
```

Trouver le plus grand produit de cinq chiffres consécutifs dans la liste de 1000 chiffres ci-dessus

Solution :

Penser à mettre en chaîne et faire une itération de longueur la chaîne moins g, ici $g = 5$.

```
N="731671765313306249192251196744265747423553491949345\
.......
71636269561882670428252483600823257530420752963450"

def PE008(N, g=5):
  res = 0
  for i in range(len(N)-g):
    p = 1
    for j in range(g): p*=int(N[i+j])
    res = max(res, p)
  return res
print(PE008(N))
```

9 L'unique triplet pythagoricien de somme 1000

Un triplet pythagoricien est un ensemble de trois entiers naturels non nuls, $a < b < c$, pour lesquels : $a^2 + b^2 = c^2$. Par exemple : $3^2 + 4^2 = 9 + 16 = 25 = 5^2$. Il existe exactement un triplet de Pythagore tel que $a + b + c = 1000$.
Trouver le produit des nombre a, b et c : $a \times b \times c$.

Solution :

On note p le demi-périmètre. Les côtés mesurent moins de p. La découverte d'une solution (réputée unique) fait sortir de la fonction.

```
def PE009(p=500):
  for a in range(1,p):
      for b in range(p-1, 0, -1):
          if a**2 + b**2==(2*p-b-a)**2:
              return(a*b*(2*p-b-a))
# Variante
def PE009(p=500):
#  for k in divisor(p):
    k=1 # ici k=1 fonctionne directement
    mMin=int((p/2/k)**0.5)
    mMax=int((p/k)**0.5)
    for m in range(mMax, mMin-1, -1):
      if p%(k*m)==0 :
        n=p//(k*m)-m
        return k**3*(m*m-n*n)*(2*m*n)*(n*n+m*m)

print(PE009())
```

Variante bien plus rapide.

Théorème : Les triplets pythagoriciens, sont les triplets :
$a = k(m^2 - n^2)$, $b = k(2mn)$, $c = k(m^2 + n^2)$ avec k, m, $n \in \mathbb{N}^*$ et $m > n$.
Pour $k = 1$, on a les triplets primitifs.

Remarques : Le demi-périmètre est $p = \dfrac{a+b+c}{2} = km(m+n)$ entier.

Par inégalité triangulaire $a < p$, donc $km^2 < p$, donc $m < \sqrt{\dfrac{p}{k}}$

On a $m > n$, donc $p < km \times 2m$, donc $m > \sqrt{\dfrac{p}{2k}}$

10 Somme des nombres premiers inférieurs à 2 000 000

La somme des nombres premiers inférieurs à 10 est $2+3+5+7=17$.
Trouver la somme de tous les nombres premiers inférieurs à deux millions.

Solution :

Au problème 7, on a construit une fonction `primeSieve`. Nous stockons nos fonctions utiles dans un fichier `arith.py`.

```
from arith import primeSieve
def PE009(limit=2000000):
  return(sum(primeSieve(limit)))
print(PE009())
```

Autres fonctions utiles quand, faute de temps et/ou de place, on ne dispose pas d'une liste complète des nombres premiers inférieurs à une limite.

```
def isPrime(n):
  " valable sans initialisation "
  n=max(n, -n)
  if n == 2 or n == 3: return True
  if n < 2 or n%2 == 0: return False
  if n < 9: return True
  if n%3 == 0: return False
  r = int(n**0.5)
  k = 5
  while k <= r:
    if n%k == 0: return False
    if n%(k+2) == 0: return False
    k +=6
  return True

def nextPrime(n):
  " valable pour n>2, impair"
  n+=2
  while not(isprime(n)): n+=2
  return n
```

11 Plus grand produit de quatre nombres dans une grille

Dans la grille de 20×20 ci-dessous, quatre nombres le long d'une ligne diagonale ont été marqués en gras.

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

Le produit de ces nombres est : $26 \times 63 \times 78 \times 14 = 1\,788\,696$.
Trouver le plus grand produit de quatre nombres adjacents dans cette grille 20×20 dans n'importe quelle direction (horizontale, verticale ou diagonale).

Solution :

```
nums = [
[8,2,22,97,38,15,0,40,0,75,4,5,7,78,52,12,50,77,91,8],
......
[1,70,54,71,83,51,54,69,16,92,33,48,61,43,52,1,89,19,67,48]
 ]
def produit(n, i, j, p, vI, vJ):
  """produit de p termes de n,
  en partant de [i, j] dans la direction vI, vJ"""
  res=1
  for k in range(p): res*=n[i+k*vI][j+k*vJ]
  return res
```

```
def PE011(n, poids=4):
  "Valable même pour un tableau non carré"
  res = float("-infinity") # moins l'infini, c'est le mini
  ligns, colos = len(n), len(n[0])
  for i in range(ligns):
    for j in range(colos-poids):
      h = produit(n, i, j, poids, 0, 1)
      if i<ligns-poids:
        d1 = produit(n, i, j, poids, 1, 1)
        d2 = produit(n, i, j+poids-1, poids, 1, -1)
      res = max(res, h, d1, d2)
  for j in range(colos):
    for i in range(ligns-poids):
      res = max(res, produit(n, i, j, poids, 1, 0))
  return res

print(PE011(nums))
```

`v` pour produit sur les verticales, `h` pour horizontales, `d1` et `d2` les deux diagonales.

On peut avoir un tableau avec des nombres négatifs, mais aussi changer la quantité de nombres à considérer pour le produit.

12 Nombre triangulaire ayant plus de cinq cents diviseurs

La suite des nombres triangulaires est générée en ajoutant les nombres naturels de 1 à n. Ainsi, le 7e nombre triangulaire est : $1+2+3+4+5+6+7 = 28$.
Les dix premiers termes sont donc : 1, 3, 6, 10, 15, 21, 28, 36, 45, 55, ...
En énumérant la liste des diviseurs des sept premiers nombres triangulaires :

$$\begin{aligned}
1 &\mapsto 1\\
3 &\mapsto 1,3\\
6 &\mapsto 1,2,3,6\\
10 &\mapsto 1,2,5,10\\
15 &\mapsto 1,3,5,15\\
21 &\mapsto 1,3,7,21\\
28 &\mapsto 1,2,4,7,14,28
\end{aligned}$$

Nous pouvons voir que 28 est le premier nombre triangulaire qui a plus de cinq diviseurs.
Trouver le premier nombre triangulaire qui a plus de cinq cents diviseurs.

Solution :

Supposons construite la fonction `nbDivisor` qui renvoie le nombre de diviseurs. On peut faire une boucle simple. Mais on remarque que n et $n+1$ sont toujours premiers entre eux. Or, si p et q sont premiers entre eux, alors $\texttt{nbDivisor}(pq) = \texttt{nbDivisor}(p) \times \texttt{nbDivisor}(q)$.

```
def PE011(M=500):
  n=1
  while True:
    n+=1
    if nbDivisor(n*(n+1)//2)>M: return n*(n+1)//2

# Variante bien plus rapide
def PE011(M=500):
  n=1
  while True:
    n+=1 # n devient pair
    if nbDivisor(n//2)*nbDivisor(n+1)>M: return n*(n+1)//2
    n+=1 # n devient impair
    if nbDivisor(n)*nbDivisor((n+1)//2)>M: return n*(n+1)//2

print(PE011())
```

Pour construire `nbDivisor`, on peut faire la liste des diviseurs et en faire le compte, ici on en profite pour faire la somme.

```
def divisor(n):
  "Nombre, somme des diviseurs pour n>1"
  N=2 # Il y a 1 et n comme diviseurs
  S=1+n # voilà leur somme
  rootN = int(n**0.5)
  for c in range(2, rootN+1):
    if n%c==0:
      N+= 2
      S+= c + n//c
  if rootN**2==n: # à ne pas compter deux fois
    N-= 1
    S-= rootN
  return N, S
```

Si la décomposition en facteurs premiers de n est $\prod_{i\in I} p_i^{\alpha_i}$, alors

$$\texttt{nbDivisor}(n) = \prod_{i\in I} \alpha_i + 1 \qquad \text{et} \qquad \texttt{sumDivisor}(n) = \prod_{i\in I} \frac{p_i^{\alpha_i+1}-1}{p_i-1}$$

Supposons construite la fonction `factor` qui donne la décomposition en facteurs premiers. Exemple : `factor(45)` donne `[[3, 2], [5, 1]]`

```
def divisor(n):
  "Nombre, somme des diviseurs pour n>0"
  N = S = 1
  for pa in factor(n):
    N*=pa[1]+1 # alphai +1
    S=S*(pa[0]**(pa[1]+1)-1)//(pa[0]-1)
  return N, S
```

Dans la pratique, il est plus long de passer par la décomposition en facteurs premiers, sauf si elle est connue ou particulière.

13 Les dix premiers chiffres d'une somme

Dans le fichier PE-013-data.txt (clic-droit, ouvrir le lien), chaque ligne représente un nombre.
Trouver les dix premiers chiffres de la somme des nombres.

Solution :

Méthode pour ouvrir un fichier et remplir un tableau avec les données : des caractères.
Pour la somme, transtypage en entier, puis sommation, puis transtypage en chaîne pour le découpage. On pourrait encore transtyper pour avoir un résultat numérique.

```
def PE013(fichier = 'PE-013-data.txt', n=10):
  tab = open(fichier).read().split()
  return str(sum( [int(x[:n+1]) for x in tab] ))[:n]
# ou, si les nombres sont de taille variable
#  return str(sum( [int(x) for x in tab] ))[:n]

print(PE013())
```

Python 3 gère sans aucun problème les entiers de toute taille, la seule limite est la mémoire de l'ordinateur. Avec un autre langage, c'est un autre défi : gérer les entiers plus longs qu'une certaine limite.

14 Durée de vol à Syracuse

Une suite de Syracuse est définie par son premier terme et la récurrence :

$$\begin{cases} (\text{si } u_n \text{ est pair}) & u_{n+1} = \dfrac{u_n}{2} \\ (\text{si } u_n \text{ est impair}) & u_{n+1} = 3u_n + 1 \end{cases}$$

En utilisant la règle ci-dessus et en commençant par 13, on génère la suite :

$$13 \mapsto 40 \mapsto 20 \mapsto 10 \mapsto 5 \mapsto 16 \mapsto 8 \mapsto 4 \mapsto 2 \mapsto 1$$

Cette suite,débutant avec 13 et finissant avec 1, contient 10 termes.
Bien que cela n'ait pas encore été démontré (Conjecture de Syracuse ou de Collatz), on remarque que toutes les suites finissent par 1, quel que soit le nombre de départ. Trouver le nombre de départ inférieur à un million pour lequel la suite formée est la plus longue.

Remarque : Après le départ, les termes sont autorisés à aller à un million ou plus.

Solution :

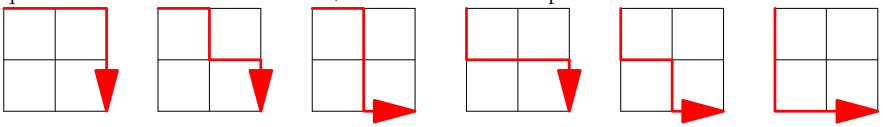
On utilise la structure des dictionnaires pour mémoriser les temps de vol à partir d'un nombre, ce qui évite de le recalculer plusieurs fois. Si n est impair, alors on peut avancer de deux étapes d'un coup. Dans `volMax`, on stocke deux choses : le point de départ optimal et son temps de vol.

```
syracuse = {1: 1}
def construit(n):
  if not n in syracuse:
    if n%2:
      syracuse[n] = construit((3*n+1)//2)+2)
    else:
      syracuse[n] = construit(n//2) + 1)
  return syracuse[n]

def PE014(liMax=10**6):
  volMax=[0, 0]
  for i in range(3, liMax-1, 2):
    temps=construit(i)
    if temps>volMax[1]: volMax=[i, temps]
  return volMax[0]

print(PE014())
```

Nous n'avons testé que les nombres i impairs, en effet, on conjecture que `tempsVol`$(2i+1) \geqslant$ `tempsVol`$(2i)$. À prouver.

15 Nombre de routes en ville

Du coin supérieur gauche d'une grille 2×2 au coin inférieur droit, en ne se déplaçant que vers la droite ou vers le bas, il existe 6 chemins possibles.



Trouver le nombre de chemins possibles sur une grille 20×20.

Solution :

Il suffit de calculer C_{40}^{20}.
La longueur du trajet est 40, il faut choisir 20 au sud parmi 40.

$$N = \frac{21 \times 22 \times 23 \times 24 \times 25 \times \cdots \times 37 \times 38 \times 39 \times 40}{1 \times 2 \times 3 \times 4 \times 5 \times \cdots \times 17 \times 18 \times 19 \times 20}$$

Après simplification, il reste :

$$N = 23 \times 29 \times 31 \times 11 \times 7 \times 6 \times 37 \times 39 \times 10$$

```
>>> print(2**2 * 3**2 * 5 * 7 * 11 * 13 * 23 * 29 * 31 * 37)
```

Pour une grille $n \times k$, il nous faut calculer C_{n+k}^{k}.

```
def binomial(n, k):
  p = 1
  for t in range(min(k, n-k)):
    p = p*(n-t)//(1+t)
  return p

def PE015(n=20, k=20):
  return binomial(n+k, n)

print(PE015())
```

16 La somme des chiffres de 2^{1000}

$2^{15} = 32768$ et la somme de ses chiffres est $3+2+7+6+8 = 26$.
Trouver la somme des chiffres de 2^{1000}.

Solution :

On pourrait faire du transtypage, mais nous serions limité à la base 10.

```
>>> print(sum(int(chiffre)for chiffre in str(2**1000))
```

On peut aussi créer une fonction qui retourne la liste des chiffres dans une base.

```
def chiffres(n, base=10):
    " liste inversée des chiffres de n "
    liste=[]
    while n:
        n, r = divmod(n, base)
        liste.append(r)
    return liste

def PE016(N=1000):
  return sum(chiffres(2**N))
```

`divmod(n, m)` renvoie le quotient et le reste de la division de n par m. C'est plus rapide que `n//m ; n%m`.

La réciproque pourra être utile.

```
def nombre(liste, base=10):
  return sum([c*base**i for i, c in enumerate(liste)])
```

`enumerate(liste)` renvoie un élément précédé de son index, ce qui permet d'éviter une écriture lourde avec une boucle et une initialisation.

17 Écrire tous les nombres de 1 à 1000 en anglais

Si les nombres de 1 à 5 sont écrits sous forme de mots en anglais :

one, *two*, *three*, *four*, *five*.

Il y a $3+3+5+4+4=19$ lettres au total.

Trouver le nombre de lettres utilisées pour écrire en anglais tous les nombres de 1 à 1000 (*one thousand*).

Remarque : Il ne faut pas compter les espaces ou les tirets. Par exemple, 342 (*three hundred and forty-two*) contient 23 lettres et 115 (*one hundred and fifteen*) contient 20 lettres.

L'utilisation de *and* à l'écriture d'un nombre est en conformité avec l'usage britannique.

Solution :

Solution proposée peu élégante. On doit faire mieux.

```
mot = {0: '', 1: 'one', 2: 'two', 3: 'three', 4: 'four',
       5: 'five',6: 'six',7: 'seven',8: 'eight',9: 'nine',
      10: 'ten',11: 'eleven',12: 'twelve',13: 'thirteen',
      14: 'fourteen', 15: 'fifteen', 16: 'sixteen',
      17: 'seventeen', 18: 'eighteen', 19: 'nineteen',
      20: 'twenty', 30: 'thirty', 40: 'forty', 50: 'fifty',
      60: 'sixty',70: 'seventy',80: 'eighty',90: 'ninety'}
def juskcent():
  s=''
  for n in range(1, 100):
    if n<21:
      s+=mot[n]
    else:
      s+=mot[n//10*10]+mot[n%10]
  return len(s)
# ou alors en une ligne
#sum([(len(mot[n]) if n<21 else len(mot[n//10*10]+mot[n%10]))
      for n in range(1, 100) ] )
def PE017():
  s = ini = juskcent()
  for i in range(1, 10):
    s+= len(mot[i]+'hundred') # x hundred,
    s+= 99*len(mot[i]+'hundredand') + ini
  s+=len('onethousand')
  return s
print(PE017())
```

18 Somme maximale d'un trajet dans un triangle

3
7 4
2 **4** 6
8 5 **9** 3

En commençant par le sommet du triangle ci-dessus et en se déplaçant vers des numéros adjacents sur la ligne en-dessous, le total maximal de haut en bas est de $3+7+4+9=23$.

75
95 64
17 47 82
18 35 87 10
20 04 82 47 65
19 01 23 75 03 34
88 02 77 73 07 63 67
99 65 04 28 06 16 70 92
41 41 26 56 83 40 80 70 33
41 48 72 33 47 32 37 16 94 29
53 71 44 65 25 43 91 52 97 51 14
70 11 33 28 77 73 17 78 39 68 17 57
91 71 52 38 17 14 91 43 58 50 27 29 48
63 66 04 68 89 53 67 30 73 16 69 87 40 31
04 62 98 27 23 09 70 98 73 93 38 53 60 04 23

Trouver le total maximal de haut en bas du triangle ci-dessus.

Remarque : Dans le triangle ci-dessus il n'y a que 16384 routes, donc il est possible de le résoudre par la force brute, ce qui n'est pas le cas avec l'exercice 67 dont le triangle contient 100 lignes (1×10^{12} routes) et donc nécessite une méthode plus intelligente !

Solution :

On fait un tableau de liste de longueur variable. En partant du bas, on cherche la meilleure somme sur la ligne au-dessus et on remplace par cette valeur. Arrivé en haut, la meilleure somme est disponible.

```
table = [
[75],
[95, 64],
[17, 47, 82],
.......................
[4,62,98,27,23,9,70,98,73,93,38,53,60,4,23] ]
def PE018(table=table):
  n=len(table)
  for i in range(1,n):
    for j in range(n-i):
      table[n-i-1][j] += max(table[n-i][j:j+2])
  return table[0][0]
print(PE018())
```

19 Dimanches tombés le premier jour du mois au xx^{e} s.

On vous donne les informations suivantes, mais si vous préférez vous pouvez faire quelques recherches pour vous :
- Le 1er janvier 1900 était un lundi.
- Septembre, avril, juin, novembre : ont 30 jours chacun et les autres ont 31 jours, sauf février qui a 28 jours sauf dans les années bissextiles où il en a 29.
- Les années bissextiles sont multiples de 4 mais pas de 100 sauf celle de 400.

Trouver le nombre de dimanches qui sont tombés sur le premier jour du mois au cours du xx^{e} siècle (1 janvier 1901 — 31 décembre 2000).

Solution :

```
def isBiss(n):
  if n%4: return False
  if n%400==0: return True
  if n%100==0: return False
  return True

mois = {
1: ['janvier', 31],   2: ['février', 28],   3: ['mars', 31],
4: ['avril', 30],     5: ['mai', 31],       6: ['juin', 30],
7: ['juillet', 31],  8:['aout', 31],   9: ['septembre', 30],
10: ['octobre', 31],11:['novembre',30],12:['décembre', 31]}
def PE019():
  j, m, a = 1, 1, 1900  # lundi, 1er janvier 1900
  res=0
  for i in range(1,12+1):
    jmois=mois[i][1]
    if isBiss(a) and i==2: jmois+=1
    j= (j+jmois)%7 # jour le 1er du mois suivant
  if j==0: res+=1  #1 janv 1901 : dimanche ?
  for a in range(1901, 2001):
    for i in range(1,12+1):
      jmois=mois[i][1]
      if isBiss(a) and i==2: jmois+=1
      j= (j+jmois)%7 # jour le 1er du mois suivant
      if j==0: res+=1
  if j==0: res-=1  #1 janv 2001 : dimanche ?
  return res
print(PE019())
```

20 Somme des chiffres du nombre 100!

$n!$ signifie $n \times (n-1) \times \cdots \times 3 \times 2 \times 1$. Par exemple, $10! = 10 \times 9 \times \ldots \times 3 \times 2 \times 1 = 3628800$, et la somme des chiffres de 10! est $3+6+2+8+8+0+0 = 27$.
Trouver la somme des chiffres du nombre 100!

Solution :

On a déjà vu `chiffres`(n) qui renvoie les chiffres d'un entier n. Ici Python 3 offre une grande facilité en acceptant les entiers de taille seulement limitée par la mémoire de l'ordinateur. On peut donc passer en force.

```
def facto(n):
    if n==1: return 1
    return n*facto(n-1)

def PE020(n=100):
  return sum(chiffres(facto(n)))

print(PE020())
```

21 Nombres amicaux inférieurs à 10 000

Soit $d(n)$ la somme des diviseurs propres de n. Si $d(A) = B$ et $d(B) = A$, avec $A \neq B$, alors A et B forment une paire amicale et chacun des nombres A et B est appelé un nombre amical.

Par exemple, les diviseurs propres de 220 sont : 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 et 110 ; donc $d(220) = 284$. Les diviseurs propres de 284 sont : 1, 2, 4, 71 et 142, de sorte que $d(284) = 220$, donc 220 et 284 sont amicaux.
Trouver la somme de tous les nombres amicaux inférieurs à 10 000.

Solution :

On modifie `sumDivisor` vue précédemment pour exclure n des diviseurs propres de n.

```
def d(n):
    "Somme des diviseurs propres"
    S=1
    rootN = int(n**0.5)
    for c in range(2, rootN+1):
        if n%c==0:
            S += c + n//c
    if rootN**2==n: S-=rootN
    return S

def PE021(limit=10**4):
  res=0
  for a in range(1, limit):
    b=d(a)
    if (a<b) and (d(b)==a):
      if b<limit:
        res += a+b
      else:
        res += a
  return res
print(PE021())
```

L'oubli de la vérification de `b<limit` est ici sans conséquence, mais l'énoncé manque de clarté pour le coup.

Remarque : Pour un nombre n, trois cas se présentent : $d(n) < n$, on dit que n est déficient. $d(n) = n$, on dit que n est parfait. $d(n) > n$, on dit que n est abondant.

22 Somme des poids des mots

Le fichier `PE-022-data.txt` contient plus de cinq mille noms. Commencer d'abord, par les trier dans l'ordre alphabétique. Remplacer ensuite chaque caractère par sa valeur alphabétique et faire la somme pour chaque mot. Multiplier ensuite, la somme par la position du mot dans la liste pour obtenir le « poids du mot ».
Par exemple, lorsque la liste est triée en ordre alphabétique, COLIN, qui vaut $3 + 15 + 12 + 9 + 14 = 53$, est classé 938^{e} dans la liste.
Donc, COLIN aura un « poids de mot » de $938 \times 53 = 49714$.
Trouver la somme de tous les « poids de mot » du fichier.

Solution :

Quel que soit l'encodage du fichier, les lettres de l'alphabet seront dans l'ordre. Ainsi, `ord(lettre)-ord('A')+1` revoie bien l'ordinal de `lettre`, ordinal qui commence à 1.
Python possède des outils de tri puissants, pourquoi s'en priver ici. L'étude des algorithmes de tri reste importante.

Attention, dans `(1+i)*somme(mot)`, il faut bien penser au `+1`, pour passer de l'index de mot à son ordinal.

```
def somme(mot):
  return sum(ord(lettre)-ord('A')+1 for lettre in mot)


def PE022(fichier='PE-022-data.txt'):
  mots = open(fichier).read().replace('"', '').split(',')
  mots.sort()
#  if ((mots.index("COLIN")+1)*somme("COLIN") != 49714):
        break
  return sum((1+i)*somme(mot) for i, mot in enumerate(mots))

print(PE022())
```

On peut penser à tester si `COLIN` donne bien 49714.

23 Nombres égaux à la somme de deux nombres abondants

Un nombre parfait est un nombre dont la somme des diviseurs propres est exactement égale à ce nombre. Par exemple, la somme des diviseurs propres de 28 est : $1+2+4+7+14=28$, ce qui signifie que 28 est un nombre parfait.
Un nombre n est appelé déficient si la somme de ses diviseurs propres est inférieure à n et il est appelé abondant si cette somme est supérieure à n.
Sachant que 12 est le plus petit nombre abondant, $1+2+3+4+6=16$, le plus petit nombre qui puisse s'écrire comme la somme de deux nombres abondants est 24.
Par l'analyse mathématique, il a été démontré que tous les entiers supérieurs à 28 123 peuvent être écrits comme la somme de deux nombres abondants. Toutefois, cette limite supérieure ne peut pas être diminuée davantage par l'analyse, même si on sait que le plus grand nombre qui ne peut être exprimé comme la somme de deux nombres abondants est inférieur à cette limite.
Trouver la somme de tous les entiers positifs qui ne peuvent pas être écrits comme somme de deux nombres abondants.

Solution :

Avec la fonction `d`(n) vue précédemment. On construit l'ensemble des nombres abondants, et pendant la construction, pour que ce soit plus rapide, on vérifie si notre nombre est candidat.

```
def PE023(limit=28123): # 20161 est la meilleure limite
  abondant=set()
  res=0
  for n in range(1, limit+1):
    if sumDivisor(n)>n: abondant.add(n)
    if not any( (n-a in abondant) for a in abondant ):
      res+=n
  return res

print(PE023())
```

24 Millionième permutation lexicographique des dix chiffres

Une permutation est un arrangement ordonné d'objets. Par exemple, 3124 est une permutation possible des chiffres 1, 2, 3 et 4. Si toutes les permutations sont énumérées dans l'ordre alphabétique ou numérique, nous l'appelons l'ordre lexicographique. Les permutations lexicographiques de 0, 1 et 2 sont :

012 021 102 120 201 210

Trouver la millionième permutation lexicographique des chiffres 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9.

Solution :

Combien de permutations vont commencer par 0 ? Il y en aura autant pour 1 etc. Avec une division euclidienne, on trouve le premier chiffre et le reste à parcourir. On boucle.

```
def facto(n):
  if n<2: return 1
  return n*facto(n-1)

def PE024(mot='0123456789', n=10**6):
  n-=1 # il faut ajuster ordinal - cardinal
  mot=list(mot) # pour pouvoir utiliser les listes
  res=[]
  reste = n%facto(len(mot))
  while mot!=[]:
    taille=len(mot)
    quotient, reste = divmod(reste, facto(taille-1))
    res.append(mot[quotient])
    del(mot[quotient])
  return res

print(PE024())

def perm(n, s): # version récursive, même principe
  "Donne la nè perm de s"
  if len(s)==1: return s
  q, r = divmod(n, facto(len(s)-1))
  return s[q] + perm(r, s[:q] + s[q+1:])
```

25 Terme de la suite de Fibonacci de plus de 1000 chiffres

La suite de Fibonacci est définie par la relation de récurrence :

$$\text{Pour } n > 2, \quad F_n = F_{n-1} + F_{n-2} \quad \text{, où } F_1 = 1 \text{ et } F_2 = 1.$$

Les 12 premiers termes sont donc les suivants :

i	1	2	3	4	5	6	7	8	9	10	11	12
F_i	1	1	2	3	5	8	13	21	34	55	89	144

Le terme F_{12}, est le premier terme qui contient trois chiffres.
Trouver le premier terme dans la suite de Fibonacci qui contient 1000 chiffres.

Solution :

```
def PE025(N=1000):
  a, b, i = 0, 1, 1
  while b<=10**(N-1):
    a, b, i = b, a+b, i+1
  return i

# Variante
from math import log, ceil, sqrt
phi = (1+sqrt(5))/2

def PE025(N=1000):
 return ceil((N-1 + log(5, 10)/2) / log(phi, 10))
print(PE025())
```

Avec la formule de Binet : $F_n = \frac{(1+\sqrt{5})^n - (1-\sqrt{5})^n}{2^n\sqrt{5}} \simeq \frac{\phi^n}{\sqrt{5}}$ où $\phi = \frac{1+\sqrt{5}}{2}$.
Pour n grand, on a :
$10^{d-1} \leqslant F_n < 10^d$
F_n a d chiffres signifie $n = \texttt{ceil}\frac{(d-1+(\log_{10} 5)/2)}{\log_{10}\phi}$

C'est très rapide, mais attention aux erreurs d'arrondis qui se cumulent à cause de ϕ quand n devient grand.

La première méthode est fiable.

26 d pour lequel $1/d$ contient le plus long cycle récurrent

Une fraction unitaire possède 1 comme numérateur. La représentation décimale des fractions avec les dénominateurs 2 à 10 est :

Fraction	1/2	1/3	1/4	1/5	1/6	1/7	1/8	1/9	1/10
Écr. déc.	0,5	0,0(3)	0,25	0,2	0,1(6)	0,0(142857)	0,125	0,0(1)	0,1

Où $0{,}1(6) = 0{,}166\,666\ldots$, dispose d'un cycle de chiffres récurrents égale à 1.
Il peut être vu que 1/7 a un cycle récurrent de 6 chiffres, donc le cycle récurrent le plus long pour la fraction $1/d$ quand $d < 10$ est pour $d = 7$.
Trouver la valeur de $d < 1000$ pour laquelle $1/d$ contient le cycle récurrent le plus long dans sa partie décimale.

Solution :

En base 10, d multiple de 2 ou de 5 donnera un cycle qui ne sera pas le meilleur. On peut l'éliminer.

```
def cycle(d, base=10):
#    if d%2 == 0 or d%5 == 0 : return 0 # base 10
    a , S, i  =  base, {}, 0
    while (a not in S):
        S[a] = i
        i += 1
        q, r = divmod(a, d)
        a = base*r
    return i-S[a]

def PE026(limit=1000):
  lmax = dmax = 0
  for d in range(limit-1, 1, -1):
    l=cycle(d)
    if l>lmax : lmax, dmax = l, d
    if d<=lmax : return dmax # Killer

print(PE026())
```

Le cycle de $1/d$ est de longueur inférieure à d, donc on peut arrêter la recherche dès que $d \leqslant d_{\max}$

27 Formules quadratiques produisant des nombres premiers

Euler a publié la formule quadratique remarquable : $n^2 + n + 41$. Il s'avère que cette formule peut produire 40 nombres premiers pour les valeurs consécutives n de 0 à 39. Toutefois, lorsque $n = 40$, $40^2 + 40 + 41 = 40(40 + 1) + 41$ est divisible par 41, et certainement lorsque $n = 41$, $41^2 + 41 + 41$ est clairement divisible par 41.
Avec l'utilisation des ordinateurs, on a trouvé l'incroyable formule $n^2 - 79n + 1601$, qui produit 80 nombres premiers pour les valeurs consécutives n de 0 à 79. Le produit des coefficients, -79 et 1601, est -126479.
Considérant une formule de la forme : $n^2 + An + B$, avec $|A| < 1000$ et $|B| < 1000$ où $|n|$ est la valeur absolue de n, par exemple $|11| = 11$ et $|-4| = 4$.
Trouver le produit des coefficients A et B, pour l'expression du second degré qui produit le nombre maximum de nombres premiers pour les valeurs consécutives en commençant avec $n = 0$.

Solution :

Pour $n = 0$, on doit avoir $B = p$ premier. On souhaite prouver A impair. Supposons A pair. Pour $n = 3$, on doit avoir $9 + 3A + p = p'$ alors $2 \in \{p, p'\}$. Pour $n = 5$, on doit avoir $25 + 5A + p = p''$, et donc encore $2 \in \{p, p''\}$. Et encore pour $n = 7$, mais on ne peut pas avoir trois fois 2 en résultat, le polynôme serait constant ; donc $p = 2$. Mais pour $n = 2$, $4 + 2A + 2$ est pair et premier, donc $2A = -4$ et pour $n = 4$, $16 + 4A + 2$ encore premier pair, donc $4A = -16$. Absurde. A est donc impair.

```
def lng(A, B):
  for n in range(B+1):
    if not isPrime(n*n+A*n+B): return n

def PE027(limit=1000):
  maxi = 0
  prime=primeSieve(limit)
  for b in prime:
      for a in range(-limit+limit%2+1, limit, 2):
        l=lng(a, b)
        if l>maxi:
          maxi, pmax = l, a*b
  return pmax

print(PE027())
```

28 Somme des diagonales sur une spirale

En commençant avec 1 et en se déplaçant dans le sens horaire, une grille de 5 par 5 est constituée comme suit :

21 22 23 24 **25**
20 **07** 08 **09** 10
19 06 **01** 02 11
18 **05** 04 **03** 12
17 16 15 14 **13**

On peut vérifier que la somme des nombres sur les diagonales est de 101.
Trouver la somme des nombres sur les diagonales d'une grille de 1001 par 1001 formée de la même manière.

Solution :

On enlève 1 à chaque case, on fait la somme des termes à la verticale, on obtient un total projeté : `36 10 0 10 36`, symétrique, progression polynomiale de degré deux, d'expression : $2(2k+8k^2)$ pour k allant de 1 à 500.
Or, on connait les sommes :

$$\sum_{k=1}^{k=n} k = \frac{n(n+1)}{2} \qquad \text{et} \qquad \sum_{k=1}^{k=n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

On déduit, après factorisation :

$$S = 4n+1+\sum_{k=1}^{k=n} 4k+16k^2 = \frac{2(n+1)(3+7n+8n^2)}{3} - 1$$

```
def PE028(n=500): # pour une grille 1001
  return 2*(n+1)*(3+7*n+8*n*n)//3-1

print(PE028())
```

Un algorithme de ballade sur la spirale aurait été plus couteux en temps.

29 Termes distincts dans la suite générée par a^b

Considérons toutes les combinaisons entières de a^b pour $2 \leqslant a \leqslant 5$ et $2 \leqslant b \leqslant 5$:

$$\begin{array}{llll} 2^2 = 4, & 2^3 = 8, & 2^4 = 16, & 2^5 = 32 \\ 3^2 = 9, & 3^3 = 27, & 3^4 = 81, & 3^5 = 243 \\ 4^2 = 16, & 4^3 = 64, & 4^4 = 256, & 4^5 = 1024 \\ 5^2 = 25, & 5^3 = 125, & 5^4 = 625, & 5^5 = 3125 \end{array}$$

Si elles sont ensuite placées dans l'ordre numérique, en enlevant les répétitions, nous obtenons la séquence suivante de 15 termes distincts :

$$4, 8, 9, 16, 25, 27, 32, 64, 81, 125, 243, 256, 625, 1024, 3125$$

Trouver le nombre de termes distincts dans la séquence générée par a^b pour $2 \leqslant a \leqslant 100$ et $2 \leqslant b \leqslant 100$.

Solution :

Force brute, on utilise la structure d'ensemble de Python qui évite de décompter les doublons.
On fait une boucle sur le produit cartésien $[\![2; limit]\!]^2$.

```
from itertools import *
def PE029(limit=100):
  lstpow=set()
  for a, b in product(range(2, limit+1), repeat=2):
      lstpow.add(a**b)
  return len(lstpow)

print(PE029())
```

30 Égaux à la somme des puissances cinquième de leurs chiffres

Étonnamment, il y a seulement trois nombres qui peuvent s'écrire comme la somme des puissances quatrième de leurs chiffres :
$1634 = 1^4 + 6^4 + 3^4 + 4^4$ $\quad$ $8208 = 8^4 + 2^4 + 0^4 + 8^4$ $\quad$ $9474 = 9^4 + 4^4 + 7^4 + 4^4$
Comme $1 = 1^4$ n'est pas une somme, il n'est pas inclus. La somme de ces nombres est $1634 + 8208 + 9474 = 19316$.
Trouver la somme de tous les nombres qui peuvent s'écrire comme la somme des puissances cinquième de leurs chiffres.

Solution :

$9^5 \times 7 < 10^6$, donc il y a au maximum 6 chiffres.
De plus $2^5 + 9^5 \times 5 < 3 \times 10^5$, donc limite plus fine.

On réécrit `chiffres` que nous avons vu précédemment.

```
def sumChifExp(n, base=10, k=5):
  s=0
  while n:
    n, r = divmod(n, base)
    s+=r**k
  return s

def PE030(k=5):
  return sum(n for n in range(10, 3*10**5)\
  if sumChifExp(n, k=k)==n)

print(PE030())
```

`sum(n for n in ensemble if condition)` est élégant pour sommer les éléments d'un ensemble remplissant une condition.

`sum(1 for n in ensemble if condition)` est utile pour les compter.

31 Façons d'obtenir 2 € avec de la monnaie

En Europe, la monnaie est constituée d'euros (€), et centimes (c), et il existe huit pièces en circulation : 1c, 2c, 5c, 10c, 20c, 50c, 1 € (100c) et 2 € (200c).
Il est possible d'avoir 2 € de la manière suivante :

$$1\times1\,€ + 1\times50c + 2\times20c + 1\times5c + 1\times2c + 3\times1c$$

Trouver le nombre de façons d'obtenir 2 € avec n'importe quel nombre de pièces.

Solution :

Une première version avec utilisation d'un dictionnaire pour mémoriser, une deuxième méthode plus astucieuse encore.

```
def paye(p, n):
  if n==1: return 1
  if not (p, n) in nbFacon:
    c = p//piece[n-1]
    s=0
    for i in range(c+1):
      s+=paye(p-i*piece[n-1], n-1)
    nbFacon[(p, n)] = s
  return nbFacon[(p, n)]

piece=[1, 2, 5, 10, 20, 50, 100, 200]
def PE031(n=200, piece=piece):
  global nbPiece, nbFacon
  nbPiece=len(piece)
  nbFacon={}
  return paye(200, nbPiece)

print(PE031())
#------------variante plus rapide
n = 200
facons = [1]+[0]*n
for p in piece:
  for i in range(p, n+1):
    facons[i] += facons[i-p]

print(facons[n])
```

32 Somme des produits pan-digitaux

Nous dirons qu'un nombre à n chiffres est pan-digital s'il utilise tous les chiffres de 1 à n exactement une fois. Par exemple, le nombre à 5 chiffres, 15234, est de 1 à 5 pan-digital.
L'ensemble des deux nombres 39 et 186 et leur produit 7254 est de 1 à 9 pan-digital $(39 \times 186 = 7254) \mapsto 391867254$.
Trouver la somme de tous les produits dont l'ensemble multiplicande – multiplicateur – produit et un nombre 1 à 9 pan-digital.

Remarque : Certains produits peuvent être obtenus de plus d'une façon, alors assurez-vous de ne les inclure qu'une fois dans votre somme.

Solution :

Le produit $a \times b$ a moins de 5 chiffres, avec $a < b$, donc $1 < a < 100$

Si $2 \leqslant a \leqslant 9$, a n'a qu'un chiffre, alors $1234 \leqslant b \leqslant \frac{9876}{a}$.

Si $10 \leqslant a \leqslant 99$, a n'a que deux chiffres, alors $123 \leqslant b \leqslant \frac{9876}{a}$.

```
def is9Pen(n, m):
  return  sorted(str(n)+str(m)+str(n*m))\
           == list('123456789')

def PE032():
  pen=set()
  for a in range(2, 10):
    for b in range(1234, 9876//a+1):
      if is9Pen(a, b): pen.add(a*b)
  for a in range(10, 100):
    for b in range(123, 9876//a+1):
      if is9Pen(a, b): pen.add(a*b)
  return sum(pen)

print(PE032())
```

33 Fausses simplifications de fraction

La fraction $\frac{49}{98}$ est curieuse, en enlevant le 9 comme dans une fausse simplification, on obtient $\frac{4}{8}$ qui est parfaitement égale à $\frac{49}{98}$. Nous considérerons les fractions comme $\frac{30}{50} = \frac{3}{5}$, des exemples triviaux, et dans $\frac{49}{98} = \frac{4}{8}$, un exemple non trivial.
Il y a exactement quatre exemples non triviaux de ce type de fraction inférieure à 1 en valeur, et contenant deux chiffres au numérateur et au dénominateur. On fait le produit de ces 4 fractions et on la rend irréductible.
Trouver la valeur du dénominateur de cette fraction.

Solution :

L'égalité de deux fractions $\frac{a}{b} = \frac{c}{d}$ avec $b \neq 0$ et $d \neq 0$ est équivalente à $ad = bc$ (le véritable produit en croix).

```
def PE033():
  sol=[]
  for a in range(1,10):
    for b in range(1,a):
      for n in range(1, 10):
        if (10*n+a)*b==(10*b+n)*a:
          sol.append([a,b])
#  if len(sol)!=4: print("Erreur")
  P=[1, 1]
  for c in sol:
    P[0]*=c[0]
    P[1]*=c[1]
  return P[0]//pgcd(P[0], P[1])

print(PE033())
```

34 Égaux à la somme des factorielles de leurs chiffres

On remarque que le nombre 145 est égal à la somme des factorielles de ses chiffres, en effet $1! + 4! + 5! = 1 + 24 + 120 = 145$.
Trouver la somme de tous les nombres égaux à la somme des factorielles de leurs chiffres.

Remarque : Ici, $1! = 1$ et $2! = 2$ sont exclus, ce ne sont pas des sommes.

Solution :

On n'utilise que les premières factorielles, on les liste.
On utilise la fonction `chiffres` déjà vue.

```
base = 10
factoChiff = [1]
for i in range(1, base):
  factoChiff.append(i*factoChiff[-1])
# en base 10
#factoChiff=[1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880]

def PE034(base=10):
  return sum(n for n in range(base, 5*10**4)\
  if sum(factoChiff[c] for c in chiffres(n))==n)
print(PE034())
```

On peut démontrer que n est entre 10 et 188888, ou entre 364009 et 488889 ou entre 731099 et 839988, (en base 10).

Conjecture (non démontrée) : n est entre 10 et 50000.

35 Nombres premiers circulaires inférieurs à un million

Le nombre 197, est appelé un nombre premier circulaire parce que toutes les rotations de ses chiffres : 197, 971 et 719, donnent des nombres premiers.
Il y a treize nombres premiers circulaires inférieurs à 100 :

2, 3, 5, 7, 11, 13, 17, 31, 37, 71, 73, 79 et 97.

Trouver la quantité de nombres premiers circulaires inférieurs à un million.

Solution :

Un premier circulant ne peut pas contenir de chiffre parmi 0, 2, 4, 5, 6, 8.
Sauf pour les nombres premiers 2 et 5.

Le transtypage est utile pour effectuer le test. On utilise `isPrime` et `primeSieve` vus précédemment pour les nombres premiers.

`l[i:]+l[:i]` correspond à une rotation (de `i` crans) de la liste `l` ; la fin , plus le début.

```
def circule(p):
  if p==2 or p==5: return True
  l=str(p)
  if any((c in l) for c in '024568'): return False
  for i in range(1, len(l)):
    if not(isPrime(int(l[i:]+l[:i]))): return False
  return True

def PE035(limit=10**6):
  prime = primeSieve(limit)
  return sum(1 for p in prime if circule(p))

print(PE035())
```

36 Palindromes en base 10 et en base 2

Le nombre (décimal), $585_{10} = 1001001001_2$ (binaire), est un palindrome dans les bases 2 et 10.
Trouver la somme de tous les nombres inférieurs à un million, qui sont des palindromes en base 10 et en base 2.

Remarque : on ne compte pas les zéros inutiles.

Solution :

Les nombres pairs finissent par 0 en binaire, donc sont hors jeu. De manière générale. Les multiples de b_1 ou de b_2 finissent par 0 dans l'une des bases, donc ne sont pas des palindromes, on les exclut rapidement.

La suite de **and** s'arrête dès qu'un des termes est **False**. On les met donc dans l'ordre d'arrêt le plus probable ou de rapidité.

On réécrit `isPal` en s'inspirant de `chiffres`.

```
def isPal(n, base):
  sym, sav = 0, n
  while n:
    n, res = divmod(n, base)
    sym = sym*base + res
  return sav==sym

def PE036(limit=10**6, b1=2, b2=10):
  return sum(n for n in range(1, limit)\
  if n%b1 and n%b2 and isPal(n, b2) and isPal(n, b1) )

print(PE036())
```

On peut faire une excellente amélioration. Au lieu de tester tous les nombres dans deux bases, on peut créer tous les palindromes d'une base, (avec XYZ donne XYZYX et XYZZYX par exemple) et tester dans l'autre base.

37 Des nombres premiers intéressants

Le nombre 3797 possède une propriété intéressante. Il est premier, et il est possible de supprimer ses chiffres de gauche à droite, il demeure premier à chaque étape : 3797, 797, 97, et 7. De même, de droite à gauche : 3797, 379, 37, et 3.
Trouver Trouver la somme des onze nombres premiers qui ont cette même propriété.

Remarque : 2, 3, 5 et 7 ne font pas partie des onze termes.

Solution :

```
from outils import isPrime, nextPrime, chiffres

def interesting(p, base=10):
  c, n, m = base, p//base, p%base
  while n:
    if not(isPrime(n)):
      return False
    if not(isPrime(m)):
      return False
    c*=base ; n//=base ; m=p%c
  return True

def PE037(fin=11):
  res = [23, 37, 53, 73]
  # liste jusqu'à cent, facile.
  fin -= 4
  p = 100
  perdu=(0, 2, 4, 5, 6, 8)
  while fin:
   p = nextPrime(p)
   if p%10 in (3, 7): # fini par 3 ou 7
     if isPrime(p//100) and isPrime(p%100):
      if not any(c in perdu for c in chiffres(p)):
          if interesting(p):
              res.append(p)
              fin-=1
  return sum(res)

print(PE037())
```

38 Plus grand nombre 1–9-pan-digital spécial

Prenons le nombre 192 et multiplions-le par 1, 2 ou 3 :

$$192 \times 1 = 192$$
$$192 \times 2 = 384$$
$$192 \times 3 = 576$$

La concaténation de chaque produit, donne un nombre 1–9-pan-digital : 192384576.
Nous appellerons le produit 192384576 concaténé de 192 et $(1,2,3)$.
De la même manière, en partant de 9 multiplié par $(1,2,3,4,5)$, on obtient 918273645, produit concaténé de 9 par $(1,2,3,4,5)$.
Trouver le plus grand 1–9-pan-digital formé à partir de la concaténation des produits d'un entier par $(1,2,...,n)$ avec $n > 1$.

Solution :

Soit d le nombre de chiffre du nombre de départ s.
Avec $d = 1$, on a obtenu 918 273 645, le maximum c'est lui ou plus. Notre nombre s commence donc par 9 : $s = 9\cdots$.
Si $d = 2$, on aurait $9a18b27c$, si $d = 3$, on aurait $9ab18cd$ ou $9ab18cd27ef$, impossible : trop ou pas assez de chiffres.
Si $d = 4$, on aurait $9abc18def$, c'est possible. $d > 4$ est impossible.
Donc $n = 5$ ou $n = 2$, cherchons le maximum pour $n = 2$, $s = 9abc$. Et comparons avec 918 273 645.

```
def PE038():
  for n in range(9876, 9122, -1):
    chif = str(n)+str(2*n)
    if len(chif)==9:
      if not '123456789'.strip(chif):
        return max(918273645, int(chif))
  return 918273645 # inutile ici

print(PE038())
```

Dans une autre situation, on pourra se servir de la fonction :

```
def isPandigital(n, s=9):
  n=str(n)
  return len(n)==s and \
  not '1234567890ABCDEF'[:s].strip(n)
```

39 Périmètre de triangles rectangles

Si p est le périmètre d'un triangle rectangle de côtés entiers (a,b,c), il y a exactement trois solutions pour $p = 120$. 1 :$\{20,48,52\}$, 2 :$\{24,45,51\}$ et 3 :$\{30,40,50\}$.
Trouver la valeur de $p \leqslant 1000$ donnant le nombre maximal de solutions.

Solution :

Pour un triplet (a,b,c), le théorème de Pythagore s'écrit $a^2+b^2=c^2$, avec $p=a+b+c$, ce qui signifie : $a^2+b^2=(p-a-b)^2=p^2+a^2+b^2-2ap-2bp+ab$, et $p^2-2ap=2bp-ab$, et enfin $b=\dfrac{p(p-2a)}{2p-a}$ est entier. Réciproquement, si p, a et $\dfrac{p(p-2a)}{2p-a}$ sont entiers alors (a,b,c) est un triplet pythagoricien.
On sait aussi que p est un nombre pair, donc on peut incrémenter p par pas de 2. De plus si p possède n solutions, alors $2p$ aura les mêmes ou plus ; on peut donc restreindre la recherche à $[\![\frac{limit}{2};limit]\!]$ qui sera donné par `range(limit//4*2, limit+1, 2)`.

Enfin, pour ne compter qu'une fois les solutions, on incrémente a de 1, jusqu'à ce que le triangle soit isocèle, auquel cas $a+a+a\sqrt{2}=p$, d'où la condition d'arrêt à $\dfrac{p}{2+\sqrt{2}}$.

```
def PE039(limit=1000):
  nMax = 0
  for p in range(limit//4*2, limit+1, 2):
    n = 0
    for a in range(1, int(p/(2+2**0.5))+1):
      if p*(p-2*a)%(2*(p-a))==0: n+=1
    if n>nMax: (nMax, res)=(n, p)
  return res

print(PE039())
```

40 n^e chiffre de la partie décimale d'un nombre irrationnel

La fraction décimale irrationnelle suivante est créée par la concaténation des entiers positifs :

$$0,12345678910\mathbf{1}112131415161718192021\ldots$$

On peut constater que le 12^e chiffre de la partie décimale est 1.
Si d_n représente le n^e chiffre de la partie décimale, on note

$$P = d_1 \times d_{10} \times d_{100} \times d_{1000} \times d_{10000} \times d_{100000} \times d_{1000000}$$

Trouver la valeur de P

Solution :

On ne garde pas en mémoire, et peut s'adapter à une autre suite $0,a_1a_2\ldots$ où $a_i = i$ ou bien autre chose.
On enregistre la position `ouvsuiv` du chiffre suivant. Elle augmente de `len(str(a(i)))`. Si on dépasse un seuil, d'une puissance de dix, alors on regarde quel est le chiffre en place, et on passe au seuil suivant.

```
def a(i): return i  # notre suite a_i

def PE040(a=a, e=6):
  ouvsuiv = 1
  i=0
  puiss=0
  res=1 # un produit vide
  while puiss<e+1:
    i+=1
    ouvsuiv+=len(str(a(i)))
    decalage=ouvsuiv-10**puiss
    if decalage>0:
      res*=(a(i)//(10**(decalage-1)))%10
      puiss+=1
  return res

print(PE040())
```

41 Le plus grand nombre premier pan-digital

On dit qu'un nombre à n-chiffre est pan-digital s'il est composé de tous les chiffres de 1 à n exactement une fois.
Par exemple, 2143 est un nombre pan-digital à 4 chiffres et il est également premier.
Trouver le plus grand nombre premier pan-digital.

Solution :

12, 123, 12345, 123456, 12345678 et 123456789 sont divisibles par trois. La somme des chiffres est conservée par permutation, donc seules les permutations de 1234 et 1234567 peuvent être premières.
On commence par la fin, chance.

```
def isPandigital(n):
  n=str(n)
  return not '123456789'[:len(n)].strip(n)


def PE041():
  for p in range(7654321, 1242, -2):
    if isPrime(p):  # 2 tests peut-être à échanger !!!
      if isPandigital(p): return p

print(PE041())
```

42 Mots triangulaires dans un fichier

Le n^e nombre triangulaire est donné par la formule : $T(n) = \frac{n(n+1)}{2}$, donc les dix premiers nombres triangulaires sont : 1, 3, 6, 10, 15, 21, 28, 36, 45, 55 ...
En remplaçant chaque lettre d'un mot par sa position en ordre alphabétique et en faisant la somme, nous formons une valeur pour le mot. Par exemple, le mot SKY a pour valeur $19+11+25 = 55 = T(10)$. Si la valeur du mot est un nombre triangulaire nous appellerons ce mot « un mot triangulaire ».
En utilisant le fichier PE-042-data.txt qui contient près de deux mille mots anglais,
Trouver le nombre de mots triangulaires.

Solution :
Plusieurs options.

```
phrase = open('data.txt').read().replace('"', '').split(',')

alph=["*", "A", "B", "C", ⋯, "Y", "Z"]

def poids(mot):
  return sum(alph.index(car) for car in mot)
#              Ou plus rapide
def poids(mot):
  return sum(ord(car) - ord('A') + 1 for car in mot)

nbTrig = set( n*(n+1)/2 for n in range(1, 100) )
#         Ou
def isTrig(n):
  return ((8*n+1)**0.5-1)/2%1==0

def PE42():  # au choix
  return sum(1 for mot in phrase if isTrig(poids(mot)))
#  return sum(1 for mot in phrase if poids(mot) in nbTrig)
```

Avec $n \in \mathbb{N}$,
$T(n) = y$ signifie $n^2 + n - y = 0$, et $y = \frac{\sqrt{8n+1}-1}{2} \in \mathbb{N}$.
D'où, le `((8*n+1)**0.5-1)/2%1==0`.

43 Les nombres pan-digitaux à 10 chiffres intéressants

Le nombre 1406357289, est un nombre pan-digital à 10 chiffres, mais il a aussi une propriété intéressante. Si d_1 est le 1er chiffre, d_2 le 2e chiffre, et ainsi de suite, on peut noter que :

- $d_2d_3d_4 = 406$ est divisible par 2.
- $d_3d_4d_5 = 063$ est divisible par 3.
- $d_4d_5d_6 = 635$ est divisible par 5.
- $d_5d_6d_7 = 357$ est divisible par 7.
- $d_6d_7d_8 = 572$ est divisible par 11.
- $d_7d_8d_9 = 728$ est divisible par 13.
- $d_8d_9d_10 = 289$ est divisible par 17.

Trouver la somme de tous les pan-digitaux à 10 chiffres ayant cette propriété.

Solution :

Un peu de réduction du problème.
Info 1 : $d_1 \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Info 2 : d_4 est pair, $d_4 \in \{0, 2, 4, 6, 8\}$

Info 3 : $d3 + d4 + d5 \equiv 0[3]$

Info 4 : $d_6 \in \{0, 5\}$, mais

Info 5 : si $d_6 = 0$, les multiples de 11 : 011, 022, ... sont interdits. $d_6 = 5$

Info 6 : $d5d6d7$ divisible par 7. Plus tard !

Info 7 : $5d_7d_8$ est divisible par 11, donc c'est dans {506, 517, 528, 539, 561, 572, 583, 594} avec l'info 5, il reste $d7d8 \in \{06, 17, 28, 39, 61, 72, 83, 94\}$

Info 8 : Les multiples acceptables de 13 qui commencent par {06, 17, 28, 39, 61, 72, 83, 94}, sont $d_7d_8d_9 \in \{286, 390, 728, 832\}$ donc $d_8d_9 \in \{86, 90, 28, 32\}$.

Info 9 : $86x$, $90x$, $28x$ ou $32x$ est divisible par 17, donc $d_8d_9d_{10} \in \{867, 901, 289\}$. 323 est interdit (double 3). $d_7d_8d_9 \in \{286, 390, 728\}$, donc

On a le résumé, $d_6d_7d_8d_9d_{10} \in \{52867, 53901, 57289\}$ mais avec l'info 6, on a $d_5d_6d_7$ multiple de 7 qui se finit par 52, 53 ou 57.

```
>>> for c in range(152, 958):
    if c%7==0 and (c%100 in (52, 53, 57)): print(c, end="")
```

on a $d_5d_6d_7 \in \{252, 357, 553, 952\}$, mais 252 et 553 sont interdits pour cause de doublon. Il reste $d_5d_6d_7d_8d_9d_{10} \in \{357289 , 952867\}$.

Pour finir, on fait un algorithme.

```
P=[2,3,5,7,11,13,17]

def isInter(n):
  for i in range(7):
    if ((n//10**(6-i))%1000)%P[i]:
      return False
  if not isPandigital(n, 10):
      return False
  return True

def PE043():
  s=0
  for c in range(1023, 6543+1):
    candidat = c*10**6+357289
    if isInter(candidat): s+=candidat
    candidat = c*10**6+952867
    if isInter(candidat): s+=candidat
  return s

print(PE043())
```

On peut accélérer en faisant les permutations ('146') + '357289' et les permutations ('134') + '952867'.

44 Paire convenable de nombres pentagonaux

Les nombres pentagonaux sont générés par la formule suivante : $P_n = \frac{n(3n-1)}{2}$.
Les dix premiers nombres pentagonaux sont :

$$1, 5, 12, 22, 35, 51, 70, 92, 117, 145, \ldots$$

On voit que $P_4 + P_7 = 22 + 70 = 92 = P_8$. Cependant, leur différence, $70 - 22 = 48$, n'est pas pentagonale.
Il existe une paire de nombres pentagonaux, P_j et P_k, tels que leur somme et leur différence soit pentagonale, avec $D = |P_k - P_j|$ minimale.
Trouver la valeur de D.

Modification : chercher plutôt pour que la somme soit minimale.

Solution :

Avec l'énoncé modifié. On construit l'ensemble des pentagonaux, et pendant la construction P_l est notre somme, on regarde si les différences $P_k = P_l - P_j$ et $P_i = P_l - 2Pj$ sont pentagonales. Si oui, on a $P_j + P_k = P_l$ et $P_k - P_j = P_i = D$ qui sont pentagonales avec P_l minimale.
On peut démontrer que $j < \frac{l}{\sqrt{2}}$, en effet $P_l - 2P_j > 0$ (puis majoration sur le trinôme).

```
def P(n):
  return n*(3*n-1)//2

def PE044():
  Pen=set()
  l=0
  while True:
    l+=1
    Pen.add(P(l))
    for j in range(int(l/2**0.5)):
      if (P(l)-P(j) in Pen) and (P(l)-2*P(j) in Pen):
        return P(l)-2*P(j)

print(PE044())
```

Si on travaille avec l'énoncé original, bien plus dur **sans tricher**, pour le même résultat. Le calcul est même un peu plus rapide.
On ne doit pas présupposer de bornes quelconques sans les démontrer.

D'abord comme pour les nombres triangulaires, on peut faire un test en résolvant $P_n = x$, qui donne : $n = \dfrac{1+\sqrt{1+24x}}{6}$ qui doit être entier.

P_n se construit aussi par récurrence : ($P_0 = 0$, et $P_{i+1} = P_i + 3i + 1$).
On part de $D = P_i$ minimal, puis on teste pour tout $k = j + d$ en faisant varier d dans $\mathbb{N}^*$ (à réduire). On note : $P_j + D = P_k$ et $P_j + P_k = 2P_j + D$, on a $i < j < k < l$.

Or $D = P_k - P_j = P_{j+d} - Pj = \cdots = 3jd + P_d$, on déduit que $j = \dfrac{D - P_d}{3d}$ qui doit être entier.

Il existe un tel j, si au moins avec $j = 1$, on a $3d + P_d \leqslant P_i = D$, par croissance de P, on a $d < i$, d'où le `d in range(1, i)`.

Enfin $P_l = 2P_j + D = j(3j-1) + D$, dernier rempart.

```
def isPen(x): # P(n)=x ?
  sixn=(24*x+1)**0.5+1
  if sixn%6==0:
    return int(sixn/6)  # n, True
  return 0  # 0, False

def PE044(i=0, D=0):
  while True:
    D+=3*i+1 ; i+=1
    for d in range(1, i):
      j, r = divmod(D-P(d), 3*d)
      if (not r) and isPen(j*(3*j-1)+D):
        return D

print(PE044())
```

45 Nombre triangulaire également pentagonal et hexagonal

Les nombres triangulaires, pentagonaux, hexagonaux sont générés par les formules :

Triangulaires	$T(n) = n(n+1)/2$	: 1, 3, 6, 10, 15, ...
Pentagonaux	$P(n) = n(3n-1)/2$	: 1, 5, 12, 22, 35, ...
Hexagonaux	$H(n) = n(2n-1)$	: 1, 6, 15, 28, 45, ...

On peut vérifier que : $T(285) = P(165) = H(143) = 40755$.
Trouver le prochain nombre triangulaire qui soit aussi pentagonal et hexagonal.

Solution :

$H(\mathbb{N}^*)$ est inclus dans $T(\mathbb{N}^*)$, en effet $T(2n-1) = H(n)$.
On a déjà vu un test pour $P(n) = x$.

```
def isPen(x):
  return   ((1 + 24*x)**0.5+1)/6%1 == 0

def PE045(n=144):
  while True:
    if isPen(n*(2*n-1)): return n*(2*n-1)
    n+=1

print(PE045())
```

46 Impair égal à un nombre premier et un double de carré

Il a été proposé par Christian Goldbach que tout nombre impair composé peut s'écrire comme la somme d'un nombre premier et du double d'un carré.

$$\begin{aligned} 9 &= 7 + 2\times 1^2 \\ 15 &= 7 + 2\times 2^2 \\ 21 &= 3 + 2\times 3^2 \\ 25 &= 7 + 2\times 3^2 \\ 27 &= 19 + 2\times 2^2 \\ 33 &= 31 + 2\times 1^2 \end{aligned}$$

Il s'avère que la conjecture était fausse.
Trouver le plus petit nombre impair composé qui ne peut s'écrire comme la somme d'un nombre premier du double d'un carré.

Solution :

On reconstruit l'ensemble des nombres premiers, et pendant la construction, si l'impair est composé, on lui applique un test.
On voit une autre façon de coder le crible d'Ératostène modifié par Euler.

```
def PE046():
  n = 5
  prime = set()
  while True:
    if all( n%p for p in prime ): prime.add(n)
    else:
      if not any( (n-2*i*i) in prime for i in\
      range(1, 1+int(((n-3)//2)**0.5)) ):
        return n
    n += 2

print(PE046())
```

47 Consécutifs et même nombre de facteurs premiers

Les deux premiers nombres consécutifs qui ont deux facteurs premiers distincts sont :

$$14 = 2 \times 7$$
$$15 = 3 \times 5$$

Les trois premiers nombres consécutifs avec trois facteurs premiers distincts sont :

$$644 = 2^2 \times 7 \times 23$$
$$645 = 3 \times 5 \times 43$$
$$646 = 2 \times 17 \times 19$$

Trouver le plus petit des quatre premiers nombres consécutifs ayant quatre facteurs premiers distincts chacun.

Solution :

Une première méthode en s'inspirant du problème précédent s'avère trop longue.

La deuxième méthode utilise `factor` qui donne la décomposition d'un nombre en facteurs premiers. C'est ici plus rapide.

```
def PE047(lng=3, nbf=3):
  n = 2
  prime = set()
  cpt = 0
  while True:
    x=sum(1 for p in prime if n%p==0)
    if x==0: prime.add(n)
    if x==nbf: cpt+=1
    else: cpt=0
    if cpt==lng: return n-lng+1
    n += 1

print("Avec 3 :", PE047())
```

La variante effective :

```
from arith import *
from math import log
def PE047(lng=4, nbf=4):
```

```
    prime = primeSieve(int(1.7*nbf*log(nbf)))
    n=1
    for i in range(nbf): n*=prime[i]
    #n=2*3*5*7 #pour nbf=4, n est le premier candidat.
    lngi = 1
    while lngi != lng:
      n += 1
      if len(factor(n)) == nbf: lngi += 1
      else: lngi = 0
    return n-lng+1

  print(PE047())
```

48 Derniers chiffres de $1^1 + 2^2 + ... 1000^{1000}$

La somme : $1^1 + 2^2 + \cdots + 10^{10}$ est égale à : 10 405 071 317.
Trouver les 10 derniers chiffres de la somme $1^1 + 2^2 + \cdots + 1000^{1000}$

Solution :

Il faut penser ici à utiliser la puissance modulaire directement implantée dans Python. En effet, `pow(a, n, p)` renvoie le même résultat que `a**n%p`, mais de manière bien plus rapide.

```
def PE048(limit=1000, nbChiff=10):
 return sum(pow(i, i, 10**nbChiff)\
 for i in range(1, limit+1)) % (10**nbChiff)

print(PE048())
```

49 Égal à la concaténation des trois termes d'une suite

La suite arithmétique : 1487, 4817, 8147, dans laquelle chacun des termes est incrémenté de 3330, a deux propriétés :
– Chacun des trois termes est premier.
– Chacun des nombres à 4 chiffres et une permutation de l'autre.
Il y a une autre suite ayant cette propriété (dans laquelle chacun des termes a 4 chiffres, et incrémentation de 3330).
Trouver le nombre à 12 chiffres formé par concaténation des trois termes de cette suite.

Solution :

Pour savoir si un nombre est une permutation d'un autre, on peut (en base 10) transityper en `str`, et trier pour comparer. Dans une autre autre base, on peut faire de même avec `chiffres(a, base)`.

```
def isPerm(a, b):
  return sorted(str(a)) == sorted(str(b))

def PE049(dec=3330):
  a = 1489  # impair
  while True:
    b, c = a+dec, a+2*dec
    if isPrime(a) and isPrime(b) and isPrime(c) \
    and isPerm(a,b) and isPerm(b,c):
      return int(str(a)+str(b)+str(c))
    a += 2

print(PE049())
```

50 Nombre premier ayant la plus longue somme

Le nombre premier 41, peut être écrit comme la somme de six nombres premiers consécutifs :

$$41 = 2+3+5+7+11+13$$

C'est la plus longue somme de nombres premiers consécutifs formant un nombre premier inférieur à cent.
La plus longue somme de nombres premiers consécutifs formant un nombre premier inférieur à mille, contient 21 termes, et est égal à 953.
Trouver le nombre premier inférieur à un million qui peut s'écrire comme la plus longue somme de nombres premiers consécutifs.

Solution :

On note notre limite $M = 10^6$. On va faire la somme cumulée des nombres premiers jusqu'à obtenir deux termes au moins égaux à M, (c'est suffisant).
On souhaite avoir :

$$p_i + p_{i+1} + \cdots + p_{j-1} + p_j = S(p_j) - S(p_{i-1}) = p_k$$

avec $p_k < M$ et $j - i$ le plus grand possible.
On note $p_0 = 0$, $p_1 = 2$ de sorte que $2+3+5+7 = S(p_4) - S(p_0)$.
On va construire une liste $S(p_i)$, puis calculer $S(p_j) - S(p_i) = p_k$ premier ?
On l'obtient avec `cumulePrime` qui renvoie cette liste avec sa longueur.

```
def PE050(M=10**6):
  prSum, larg = cumulePrime(M)

  res = 1
  for i in range(larg):
    for j in range(i+res, larg):
      pk = prSum[j] - prSum[i]
      if (pk<M and j-i>res and isPrime(pk)):
        res, pMax = j-i, pk
  return pMax, res
```

La question délicate est d'estimer le plus grand nombre premier utile.
`primeSieve(pMax)` avec $p_{\text{Max}} = M$ est suffisant, mais on pourra faire mieux.
On peut établir que $p_{\text{Max}} < M^{0.666}$; résultat basé sur $p(n) \approx n\ln(n)$.

$$S(p_{\text{Max}}) = \sum_{p\in\mathbb{P}}^{p\leqslant p_{\text{Max}}} p \approx \int_2^{p_{\text{Max}}} \frac{x}{\ln x}\,\mathrm{d}x \quad > \quad M$$

```
def cumulePrime(M):
  """ Somme cumulée de nombre premiers,
   avec deux termes au moins égaux à M """
  pMax = int(M**0.666) # 1ère estimation
  prime = primeSieve(pMax)
  larg = len(prime)
  for i in range(larg-2):
    pk = prime[i+1]+prime[i]
    prime[i+1]=pk
    if pk>=M:
      prime[i+2]+=prime[i+1] # un de plus
      return prime[:i+3], i+3 # et hop !

  larg-=1 # Si l'estimation était mauvaise
  p=prime.pop() # lui était intact
  while prime[-1]<M:
    prime.append(prime[-1]+p)
    p=nextPrime(p)
    larg+=1
  prime.append(prime[-1]+p) # un de plus
  return prime, larg+1

print(PE050())
```

Justifications de l'estimation. On peut voir que $p_k \lessapprox M$.
Et aussi que $pk = S(j) - S(i) \approx S(j)$, car $i \ll j$. Ainsi $S(j) \approx M$.

$$S(p_{\text{Max}}) \approx \int_2^{p_{\text{Max}}} \frac{x}{\ln x}\,\mathrm{d}x \quad \gtrapprox \quad M$$

Et $p_{\text{Max}} < M^{0.666}$ convient, et on peut faire encore mieux.

La meilleure approche pour des valeurs de M plus élevée est de s'appuyer sur un fichier contenant la liste des premiers nombres premiers pour ne pas avoir à réinventer la roue à chaque fois.