

# Théorie des nombres [th01] - Utilitaires

Karine Zampieri, Stéphane Rivière

Unisciel

sciel

algotprog

UNIVERSITÉ  
HAUTE-ALSACE

Version 22 mai 2018

## Table des matières

<b>1</b>	<b>Utilitaires Théorie des nombres</b>	<b>2</b>
1.1	Fonction miroir (renversé d'un entier) . . . . .	2
1.2	Fonction nchiffres (nombre de chiffres) . . . . .	4
1.3	Fonction palindromique (prédicat de palindromie) . . . . .	5
1.4	Fonction persistance (persistance d'un entier) . . . . .	6
1.5	Fonction poidsAmis (prédicat de même poids) . . . . .	7
1.6	Fonction produitChiffres (produit des chiffres) . . . . .	8
1.7	Fonction sommeChiffres (somme des chiffres) . . . . .	9
1.8	Opérations de base (sur un entier) . . . . .	10

## Python - Utilitaires Théorie des nombres (Solution)



**Mots-Clés** Théorie des nombres ■

**Requis** Axiomatique impérative sauf Fichiers ■



### Objectif

Ce module contient un ensemble d'utilitaires de « Théorie des nombres » sous forme de problèmes externalisés.

# 1 Utilitaires Théorie des nombres

## 1.1 Fonction miroir (renversé d'un entier)



### Définition

Le **renversé d'un entier** (dit aussi **miroir**) est la lecture de la gauche vers la droite de ses chiffres. Exemples :

- Le renversé de 2035 est 5302.
- Celui de 8954070 est 704598.
- Et celui de 14500 est 541.



Comment réaliser cette opération pour un entier  $n$  ?  
Prenez par exemple 12306.

### Aide méthodologique

Le tableau ci-dessous montre la méthode pour l'exemple :

Valeurs successives de l'entier $n$	Restes successifs	Miroirs successifs
12306	6	6
1230	0	60
123	3	603
12	2	6032
1	1	60321

### Solution simple

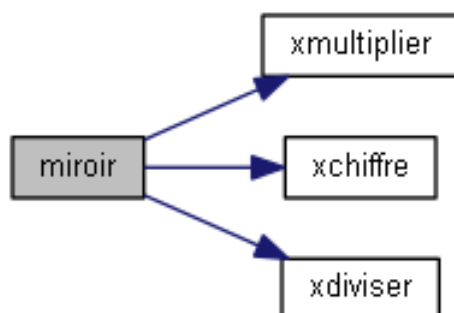
Soit  $r$  l'entier miroir, initialement nul.

Tant que  $n$  n'est pas nul :

- On fait entrer un zéro dans  $r$  (en le multipliant par 10).
- On récupère le dernier chiffre de  $n$  (par le modulo 10).
- Et on l'ajoute dans  $r$ .
- Enfin on perd le dernier chiffre de  $n$  (par une division par 10).



Écrivez une fonction `miroir(n)` qui calcule et renvoie le renversé d'un entier  $n$  (supposé positif).



**Solution simple**

On récupère un par un (opération `xdiviser`) le chiffre de poids faible de l'entier (opération `xchiffre`) et on l'insère en tant que poids faible (opération `xmultiplier`) au résultat à retourner.

**Solution détaillée**

Voici les étapes :

- Initialisez le résultat `rs` à zéro.
- Puis tant que `n` n'est pas nul :
  - Faites entrer un zéro dans `rs` (opération `xmultiplier`).
  - Récupérez le chiffre de poids faible de `n` (opération `xchiffre`).
  - Et ajoutez-le dans le `rs`.
  - Enfin perdez le chiffre de poids faible de `n` (opération `xdiviser`).



Validez votre fonction avec la solution.

**Solution Python**

@[UtilsTH.py]

```
def miroir(n):  
    """ Renversé d'un entier  
  
    :param n: un entier positif  
    :return: le renversé de n  
    """  
  
    rs = 0  
    while n != 0:  
        rs *= 10  
        rs += n % 10  
        n //= 10  
    return rs
```

## 1.2 Fonction `nchiffres` (nombre de chiffres)



Écrivez une fonction `nchiffres(n)` qui calcule et renvoie le nombre de chiffres d'un entier `n` (supposé strictement positif). Exemples :

```
nchiffres(2613609) ==> 7
nchiffres(10000) ==> 5
```

### Solution simple

L'idée est d'isoler les chiffres successivement, jusqu'à ce que l'entier `n` vaut zéro, traduisant que tous les chiffres de `n` ont été analysés. Le nombre d'itérations, initialement nul, et incrémentez à chaque tour de boucle divisant `n` par 10, équivaut au nombre de chiffres de `n`.



Modifiez votre fonction de sorte qu'elle renvoie le nombre de chiffres d'un entier `n` positif ou nul, c.-à-d. :

```
nchiffres(0) ==> 1
```

### Solution simple

On modifie l'initialisation du compteur à un ou zéro selon la valeur de `n`.



Validez votre fonction avec la solution.

### Solution Python

@[UtilsTH.py]

```
def nchiffres(n):
    """ Nombre de chiffres d'un entier

    :param n: un entier positif
    :return: le nombre (=taille) de chiffres de n
    """
    rs = (1 if n == 0 else 0)
    while n != 0:
        rs += 1
        n //= 10
    return rs
```

### 1.3 Fonction palindromique (prédicat de palindromie)



#### Définition

Un **entier** est dit **palindromique** s'il est égal à son renversé.

Exemple : Les entiers 12021, 134431 et 1047401 sont palindromiques.



Écrivez une fonction `palindromique(n)` qui teste et renvoie **Vrai** si un entier `n` (supposé positif) est palindromique, **Faux** sinon.



Validez votre fonction avec la solution.

#### Solution Python

@[UtilsTH.py]

```
def palindromique(n):  
    """ Prédicat de palindromie d'un entier  
  
    :param n: un entier positif  
    :return: Vrai si n est palindromique, Faux sinon  
    """  
    return (n == miroir(n))
```

## 1.4 Fonction persistance (persistance d'un entier)



### Définition

La **persistance d'un entier** positif est le nombre d'itérations nécessaires jusqu'à réduire le produit de ses chiffres à un seul chiffre.

Exemple : La persistance de 6788 vaut 6 car :

1.  $6 \times 7 \times 8 \times 8 = 2688$
2.  $2 \times 6 \times 8 \times 8 = 768$
3.  $7 \times 6 \times 8 = 336$
4.  $3 \times 3 \times 6 = 54$
5.  $5 \times 4 = 20$
6.  $2 \times 0 = 0$

Il a fallu six étapes pour sa réduction à un entier compris entre 0 et 9.



Écrivez une fonction `persistance(n)` qui calcule et renvoie la persistance d'un entier `n` (supposé positif).



### Solution simple

On utilise la fonction `produitChiffres` et on compte le nombre d'itérations à effectuer jusqu'à le réduire à un unique chiffre, c.-à-d. tant que  $n \geq 10$ .



Validez votre fonction avec la solution.

### Solution Python

@[UtilsTH.py]

```

def persistance(n):
    """ Persistance d'un entier

    :param n: un entier positif
    :return: la persistance de n
    """
    niters = 0
    while n >= 10:
        niters += 1
        n = produitChiffres(n)
    return niters
  
```

## 1.5 Fonction poidsAmis (prédicat de même poids)



### Définition

Deux entiers  $a$  et  $b$  ont **même poids** si la somme de leurs chiffres est identique.  
Exemple : 24 et 11121 ont même poids, la somme des chiffres étant 6.



Écrivez une fonction `poidsAmis(a,b)` qui teste et renvoie `Vrai` si deux entiers  $a$  et  $b$  (supposés positifs) ont même poids, `Faux` sinon.



### Solution simple

Soient  $s1$  la somme des chiffres de  $a$  et  $s2$  celle de  $b$ .  
Alors  $a$  et  $b$  ont même poids si  $s1$  vaut  $s2$ .



Validez votre fonction avec la solution.

### Solution Python

@[UtilsTH.py]

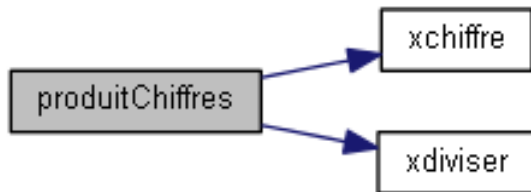
```
def poidsAmis(a, b):  
    """ Prédicat de poids amis de deux entiers  
  
    :param a: un entier positif  
    :param b: un entier positif  
    :return: Vrai si a et b ont meme poids, Faux sinon  
    """  
    return (sommeChiffres(a) == sommeChiffres(b))
```

## 1.6 Fonction produitChiffres (produit des chiffres)



Écrivez une fonction `produitChiffres(n)` qui calcule et renvoie le produit des chiffres d'un entier `n` (supposé positif). Exemple :

```
produitChiffres(82419) ==> 9*1*4*2*8 = 576
```



### Solution simple

On récupère un par un le chiffre de poids faible de l'entier `n` (opération `xchiffre`), on le multiplie au résultat à retourner, puis on perd le chiffre traité dans `n` (opération `xdiviser`).

**Attention !** C'est un produit d'où l'initialisation à 1 (si `n` n'est pas nul).



Validez votre fonction avec la solution.

### Solution Python

@[UtilsTH.py]

```
def produitChiffres(n):  
    """ Produit des chiffres d'un entier  
  
    :param n: un entier positif  
    :return: le produit des chiffres de n  
    """  
  
    rs = (0 if n == 0 else 1)  
    while n != 0:  
        rs *= n % 10  
        n //= 10  
    return rs
```

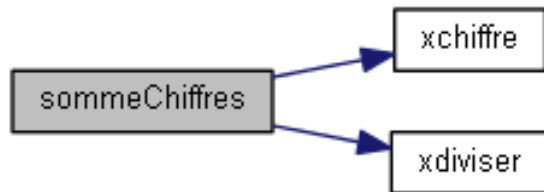


## 1.7 Fonction sommeChiffres (somme des chiffres)



Écrivez une fonction `sommeChiffres(n)` qui calcule et renvoie la somme des chiffres d'un entier `n` (supposé positif). Exemple :

```
sommeChiffres(213609) ==> 9+0+6+3+1+2 = 21
```



### Solution simple

La fonction doit traverser l'entier (boucle `TantQue`) et :

- Récupérer le dernier chiffre de `n` : opération `xchiffre`
- Perdre le dernier chiffre de `n` : opération `xdiviser`

### Solution détaillée

On effectue les opérations suivantes :

- Initialisation du résultat à zéro (c'est une somme).
- Puis tant que tous les chiffres de l'entier `n` n'ont pas été traités :
  - Récupération du chiffre de poids faible de `n` (par le modulo).
  - Sommaton au résultat à retourner.
  - Puis perte du chiffre traité dans `n` (par la division).



Validez votre fonction avec la solution.

### Solution Python @[UtilsTH.py]

```
def sommeChiffres(n):
    """ Somme des chiffres d'un entier

    :param n: un entier positif
    :return: la somme des chiffres de n
    """
    rs = 0
    while n != 0:
        rs += n % 10
        n //= 10
    return rs
```

## 1.8 Opérations de base (sur un entier)

### Terminologie

Dans le système décimal, un entier naturel  $N$  est représenté par une séquence de **chiffres**  $a_n a_{n-1} \cdots a_k \cdots a_0$  dont la valeur est :

$$\begin{aligned} N &= a_n \times 10^n + a_{n-1} \times 10^{n-1} + \dots + a_1 \times 10 + a_0 \\ &= \sum_{k=0}^n a_k \times 10^k \end{aligned}$$

avec  $0 \leq a_k < 10$ ,  $0 \leq k \leq n$  et  $a_n \neq 0$ . L'entier  $a_0$  est appelé le **chiffre de poids faible** de  $N$  et  $a_n$  son **chiffre de poids fort**.

### Exemple

Pour  $N = 8107$  :

- $n = 3$ ,  $a_3 = 8$ ,  $a_2 = 1$ ,  $a_1 = 0$ ,  $a_0 = 7$
- $N = 8 \times 10^3 + 1 \times 10^2 + 0 \times 10^1 + 7 \times 10^0 = 8000 + 100 + 7$
- 7 est le chiffre de poids faible de  $N$
- 8 est le chiffre de poids fort de  $N$



Associez chacun des éléments (à gauche) à son opération (à droite).

Etiquette	Cible
Décale les chiffres d'un entier vers la gauche d'une position Exemples : 30129 ==> 301290 et 210 ==> 2100	Par le modulo 10
Chiffre de poids faible d'un entier Exemples : 30129 ==> 9 et 210 ==> 0	En le divisant par 10
Décale les chiffres d'un entier vers la droite d'une position Exemples : 30129 ==> 3012 et 210 ==> 21	En le multipliant par 10

### Solution

- Le modulo 10 (reste de la division euclidienne par 10) récupère le chiffre de poids faible d'un entier.
- La division par 10 décale les chiffres d'un entier vers la droite d'une position.
- Alors que la multiplication par 10 décale les chiffres d'un entier vers la gauche d'une position.