

# Arithmétique avec Python

## 1. Déterminer la liste des diviseurs d'un entier donné

Trouver la décomposition d'un nombre en facteurs premiers est facile pour un être humain, mais nécessite la connaissance de structures avancées (les **séquences**, et plus particulièrement les **dictionnaires**) dès qu'on veut les appliquer en Python.

Cependant il est assez facile de demander à Python de trouver la **liste** des diviseurs d'un nombre donné. Il faut utiliser pour cela l'algorithme suivant :

```
fonction diviseurs(n :entier)
    Créer une liste vide L
    Pour chaque nombre i dans les entiers allant de 1 à n
        Si i est un diviseur de n
            Ajouter i à L
    renvoyer L
```

Cet algorithme en langage naturel peut être quasiment traduit à l'identique en Python grâce aux instructions suivantes :

```
def diviseurs(n) :
    L = list()
    for i in range(1, n+1) :
        if n%i == 0 :
            L.append(i)
    return L
```

### ? Exercice 1

#### Enoncé

1. Ouvrir **Thonny**, puis créer un fichier dans votre espace personnel nommé `arithmetique.py`.
2. Taper le code précédant dans l'éditeur.
3. Dans la zone de script (le *shell*), taper la ligne suivante :

```
diviseurs(60)
```

Quel est le résultat obtenu ?

4. Que faut-il écrire en Python pour obtenir les diviseurs de 30 ? de 262 ?
5. Combien y-a-t'il de diviseurs de 413 525 ? (On pourra utiliser la ligne `len(diviseurs(413525))`)
6. Combien y-a-t'il de diviseurs de 2 745 897 ?
7. Combien y-a-t'il de diviseurs de 234 745 896 ? Que constate-t'on ?
8. Combien y-a-t'il de diviseurs de 234 745 897 ? Que constate-t'on ? Comment s'appelle ce type de nombre entiers ?

#### Réponses

A venir !

## 2. PGCD et PPCM

### 2.1. PGCD

## Listes en Python

L'objet renvoyé par la fonction `diviseurs` est de type `list`.

Il s'agit d'un type particulier, qu'on peut imaginer comme étant une **suite de cases numérotées en partant de 0**

Par exemple la liste Python `L = [ 12 , 32 , 44 , -5 , 7 ]` correspond au schéma suivant :

<b>L</b>	<b>12</b>	<b>32</b>	<b>44</b>	<b>-5</b>	<b>7</b>
<i>n°</i>	0	1	2	3	4

et il est possible d'accéder aux méthodes suivantes :

- `L[2]` renverra le nombre 44 ;
- `L[0] * L[2]` renverra le nombre  $12 \times 44 = 528$  ;
- `L[-1]` renverra le dernier nombre de la liste donc ici 7.

En ajoutant la fonction suivante :

```
def inter(l1, l2) :
    return [e for e in l1 if e in l2]
```

on obtient la possibilité d'obtenir l'**intersection** de deux listes `l1` et `l2`, par exemple :

```
>>> inter([1, 2, 4, 9, 16], [3, 4, 8, 9, 12])
[4, 9]
```

## ? Exercice 2

### Enoncé

1. Ajouter la fonction `inter` au fichier `arithmetique.py`.
2. Que faut-il écrire en Python pour obtenir la liste des diviseurs de 60 ?
3. Que faut-il écrire en Python pour obtenir la liste des diviseurs de 132 ?
4. Que faut-il écrire en Python pour obtenir les diviseurs communs à 60 et 132 ?
5. Comment faire pour obtenir le PGCD de 60 et 132 ?
6. Faire de même pour 372 522 et 195 624.
7. Créer une fonction python `pgcd(a, b)` qui renvoie le pgcd des nombres entiers naturels `a` et `b` passés en argument.

### Réponses

A venir !

## 2.2. PPCM

### relation entre PGCD et PPCM

Soient  $a$  et  $b$  deux entiers naturels. Alors :

$$a \times b = PGCD(a; b) \times PPCM(a; b)$$

Créer une fonction `PPCM` qui renvoie le ppcm de deux entiers naturels `a` et `b`. Créer une fonction `PPCM` qui renvoie le ppcm de deux entiers naturels `a` et `b`.