# Algorithmique et Programmation Impérative

## TD2 (séances 2 et 3) – Suites & Récursivité

*Nous rappelons que l'objectif de ce module est d'apprendre les concepts de l'algorithmique et non pas un langage de programmation précis. Cependant, nous allons utiliser Python pour les TDs puisque c'est un langage que vous avez appris à l'ENSIBS.*

### I. Itérations : Suite d'entiers

1. Écrivez un programme *Sequence* qui lit une suite de 10 entiers positifs et affiche la moyenne de ces entiers en utilisant une boucle for
2. Réécrivez ce programme avec une boucle *while*. Laquelle de ces deux boucles est la plus adaptée ?
3. Modifier ce programme pour qu'il lise une suite d'entiers positifs terminée par -1 (on ne connaît pas à l'avance le nombre d'entiers saisis) et affiche la moyenne de ces entiers, en utilisant la boucle *while*
4. Complétez ce programme pour qu'il affiche le maximum et le minimum de la suite
5. Complétez ce programme pour qu'il affiche le nombre d'occurrences du premier entier dans la suite
6. Complétez ce programme pour qu'il affiche le nombre de sous-suites strictement croissantes les plus longues possibles de la suite (e.g. la suite 12 4 16 4 4 8 12 contient 4 sous-suites strictement croissantes : (12) (4 16) (4) (4 8 12))
7. Modifiez ce programme pour qu'il lise la suite d'entiers, puis affiche le menu affiché ci-dessous, liste le choix de l'utilisateur, et l'exécute, jusqu'à ce que l'utilisateur sélectionne 'q' , en choisissant le type de boucle (*for* ou *while* la plus adaptée). Le menu proposé :
    a. afficher la moyenne
    b. afficher le minimum
    c. afficher le maximum
    d. afficher le nombre d'occurrences du 1er entier
    e. afficher le nombre de monotonies
    q. quitter

## II. Dessins

1. Écrivez un programme qui, à l'aide de signe * , affiche un carré de coté n (n est donné par l'utilisateur) :

```
* * *
* * *
* * *
```

2. Pareil pour le dessin suivant :

```
*
* *
* * *
```

3. Pareil pour le dessin suivant :

```
    *
  * * *
* * * * *
```

4. Pareil pour le dessin suivant :

```
    *
  * * *
* * * * *
  * * *
    *
```

5. Pareil pour le dessin suivant :

```
* * *
*   *
* * *
```

6. Pareil pour le dessin suivant :

```
  * *
    * *
      * *
```

## III. Manipulation sur un nombre entier

1. Écrivez un programme qui demande à l'utilisateur un nombre entier, et calcule l'entier correspondant à ce nombre inversé (exemple 5314 devient 4135)
2. Complétez ce programme pour qu'il vérifie si ce nombre est premier
3. Complétez ce programme pour qu'il donne la décomposition du nombre en facteurs premiers (exemple 45 donne 3 3 5)
4. Complétez ce programme pour qu'il demande un deuxième entier et calcule le PGCD des deux entiers en utilisant la méthode d'Euclide (méthode par divisions successives), fondée sur la propriété suivante :
   - Si b est un diviseur de a, alors PGCD (a, b) = b
   - pour a > b, PGCD (a, b) = PGCD (b, a%b)

5. Question subsidiaire : L'algorithme d'Euclide étendu

   Le théorème de Bézout établit que le PGDC d de deux entiers a et b est une combinaison linéaire (à coefficient entiers) de a et b, i.e. $d = au + bv$

   Une modification simple de l'algorithme d'Euclide (qu'on appelle alors algorithme d'Euclide étendu) permet de calculer ces coefficients u et v. Remarquons tout d'abord que l'algorithme d'Euclide calcule une suite d'entiers définie par une récurrence à deux termes :
   - $a_0 = a$ , $a_1 = b$
   - $a_{n-1} = q_n a_n + a_{n+1}$ autrement dit : $a_{n+1} = -q_n a_n + a_{n-1}$
   - Si l'on pose les récurrences : $u_{n+1} = -q_n u_n + u_{n-1}$ et $v_{n+1} = -q_n v_n + v_{n-1}$ avec les conditions initiales suivantes :
     - $u_0 = 1$ , $v_0 = 0$
     - $u_1 = 0$ , $v_1 = 1$
   - **Montrez par récurrence que, pour tout n, $a_n = a u_n + b v_n$**

     Dès que $a_{n+1} = 0$, on obtient ainsi $a_n = PGCD(a, b)$ avec $u_n$ et $v_n$ les coefficients entiers de l'identité de Bézout

- Exemple : a = 96 et b = 81

| n | $a_n$ | a | $u_n$ | b | $v_n$ |
|---|---|---|---|---|---|
| 0 | 96 | = 96 * | 1 | + 81 * | 0 |
| 1 | 81 | = 96 * | 0 | + 81 * | 1 |
| 2 | 15 = 96 - 81 | | | | |
| | | = 96 * | (1 - 0) | + 81 * | (0 - 1) |
| | | = 96 * | 1 | + 81 * | -1 |
| 3 | 6 = 81 - 5 * 15 | | | | |
| | | = 96 * | (0 - 5 * 1) | + 81 * | (1 - 5 * (-1)) |
| | | = 96 * | -5 | + 81 * | 6 |
| 4 | 3 = 15 - 2 * 6 | | | | |
| | | = 96 * | (1 - 2 * (-5)) | + 81 * | (-1 - 2 * 6) |
| | | = 96 * | 11 | + 81 * | -13 |

- **Réalisez une fonction qui calcule le PGCD de deux nombres a et b donnés en entrée ainsi que les coefficients de l'identité de Bézout**