# Licence 1 Informatique – 2022/2023 – S2

## Algorithmique et programmation Python

## TD 3 : Les boucles

**Rappel des consignes** :

a. Créez un dossier pour la séance.
b. Regroupez **les fichiers Python « .py » dans ce dossier** ainsi que **le fichier Word avec les algorithmes**
c. 1 exercice = 1 fichier Python
d. Le nom du fichier doit être en relation avec le numéro de l'exercice (« exercice_1.py », etc.)
e. A la fin de l'heure, vous regroupez les fichiers du dossier dans une archive (zip, 7z, rar, autre…)
f. A rendre (copier) sur votre espace Moodle.
g. Commentez abondamment votre code source Python.

**On commence par élaborer l'algorithme avant d'écrire le programme**

### Exercice 1 – Liste de nombres

**Entrée** : n1 et n2 entiers

**Sortie** : liste des nombres compris entre n1 et n2

**Calcul** :

- Vous devez vous assurer que n2 et strictement supérieur à n1, vous affichez un message d'erreur dans le cas contraire.
- Si les saisies n1 et n2 sont valides (n1 < n2), vous affichez la série de valeurs entières comprises entre n1 et n2 (ex. n1 = 8, n2 = 13, vous devez afficher 9, 10, 11, 12).

**Indications** : il est question de boucle for et d'utilisation de range dans cet exercice.

### Exercice 2 – Somme de valeurs positives

**Entrée** : un ensemble de valeurs réelles positives

**Sortie** : la somme de ces valeurs

**Calcul** : Faire saisir par l'utilisateur des valeurs réelles,

- Vous additionnez les saisies tant qu'il entre des valeurs strictement positives.
- Vous stoppez la saisie lorsqu'il aura rentré une valeur négative ou nulle. Vous affichez la somme précédemment calculée alors.

Attention, vous devez afficher la valeur 0 si aucune saisie valide n'a été effectuée.

**Indications** : Un while serait approprié pour cet exercice

## Exercice 3 – Liste des diviseurs d'un entier

**Entrée** : n (entier positif >= 2)

**Sortie** : la liste de ses diviseurs compris entre 2 et (n-1)

**Calcul** :

- Faire saisir un entier « n » par l'utilisateur, vérifiez qu'il est supérieur ou égal à 2, dans le cas contraire, affichez un message d'erreur.
- Si la saisie est valide, affichez alors la liste des diviseurs de « n » compris entre 2 et (n-1).
- Indications :
    - Si b est un diviseur de a, alors a modulo b est nul (10 modulo 2 = 0, le reste de la division de 10 par 2 est égal à 0). L'opérateur modulo sous Python est %
    - Une stratégie simple consiste à passer au crible les diviseurs potentiels de n compris entre 2 et (n-1) c.-à-d. si n = 12, alors il faudra tester 2, 3, 4, …, 11 ; et les diviseurs sont 2, 3, 4 et 6.

## Exercice 4 – Recherche de nombre premier

**Entrée** : n (entier)

**Sortie** : Indiquer que n est premier ou non

**Calcul** : Une piste simple consiste à vérifier qu'il n'existe pas de diviseurs de « n » entre 2 et (n-1), en les testant tour à tour :

- Si aucun diviseur n'est trouvé, « n » est premier ;
- Si au moins un diviseur est trouvé, alors « n » n'est pas premier.

## Exercice 5 – Recherche de nombre premier avec diviseurs

**Entrée** : n (entier)

**Sortie** : Indiquer que n est premier ou non. S'il n'est pas premier, vous devez afficher le plus petit et le plus grand diviseur de n compris entre 2 et (n-1).

**Exemple** : 14 n'est pas premier : votre programme doit afficher « 14 n'est pas premier », et il doit indiquer 2 comme plus petit diviseur, et 7 comme plus grand diviseur.

**Remarque** : A ce stade de notre progression, nous ne sommes pas censés savoir manipuler les tableaux et les listes, il faut produire une solution qui n'utilise pas ces structures.

## Exercice 6 – Nombre mystère

L'ordinateur génère un nombre mystère au hasard (entre 1 et 100) (Remarque : pour générer un nombre aléatoire, vous devez importer (**import**) le module **random**, puis initialiser le générateur avec la commande **random.seed(None)**, et enfin générer la valeur avec **random.randint(1,100)** -- cf. https://docs.python.org/2/library/random.html).

Le programme vous demande de rentrer un chiffre,

- S'il est égal au nombre mystère → **FIN du programme** avec l'affichage du nombre d'essais.
- Si plus grand, il affiche "Votre << chiffre >> est plus grand que le nombre mystère".
- Si plus petit, affiche "Votre << chiffre >> est plus petit que le nombre mystère".

Dans ces deux derniers cas, le programme doit redemander à l'utilisateur de réessayer un nouveau chiffre.

## Exercice 7 - Jeu des allumettes

L'ordinateur génère un nombre aléatoire entre 15 et 20 (inclus). Ensuite, programmez un jeu qui permet de jouer contre l'ordinateur :

- Vous retirez un certain nombre d'allumettes compris entre 1 et 3 (inclus). S'il n'en reste plus qu'une, vous avez gagné.
- Sinon, l'ordinateur retire lui aussi un certain nombre d'allumettes compris entre 1 et 3. S'il n'en reste plus qu'une, il a gagné.

Et vous itérez ainsi jusqu'il y ait un gagnant.

**Remarques** :

- A chaque étape, le programme doit afficher le nombre d'allumettes restant.
- Une autre manière de voir la victoire est de considérer que celui qui a retiré la dernière allumette a perdu.
- Essayez de définir une stratégie aussi pertinente que possible pour l'ordinateur.
- Vérifier que l'utilisateur ne triche pas en essayant de retirer un nombre d'allumettes illicite (non compris entre 1 et 3, ou un nombre plus élevé que les allumettes restantes), forcez-le à rejouer dans ce cas.