Instructions concernant ce TP 4 :

- ⇒ Le TP devra être rédigé par chaque binôme à la main. La feuille de présentation devra contenir le nom et la classe des membres du groupe.
- ⇒ Tous les programmes seront écrits en script python 3 sauf indications contraires.
- ⇒ Les feuilles seront rendues avant de commencer le prochain cours magistral. Chaque binôme devra donc se rassurer d'avoir une deuxième copie avec lui. Il serait préférable de la numériser ou de la prendre en photo. Vous en aurez besoin pour la pratique en classe.

Notation : Chaque exercice est noté sur 5 points. Note TP : 65/65 → 20/20

- ⇒ Syntaxe correcte : 2/2
- ⇒ Pertinence de l'Algorithme *(méthode de résolution)* : 3/3
- ⇒ Hors sujet : 0/5

I. Les chaines de caractères (String ou str)

Exercice 1 : Impression d'une date en clair

Ecrire un programme qui convertit une date donnée sous la forme d'un entier à 6 chiffres (jjmmaa) sous la forme d'une chaîne de caractères : jj mois–en-lettre 20aa.
Exemple : saisir 120309 et l'afficher sous la forme : 12 mars 2009

Exercice 2. Traduction d'un texte en Louchebem

Le Louchebem est "le langage des bouchers", à savoir un jargon consistant à rejeter à la fin d'un mot sa première lettre, suivie des lettres "em", et à la remplacer en tête de mot par un "l".
Exemples : "voiture" → "loiturevem" , "vache" → "lachevem" , "bonne" → "lonnebem" , etc.
Écrire un programme qui retourne la traduction en louchebem d'un mot rentré au clavier en paramètre.

Exercice 3. Palindrome

Un palindrome est un mot qui se lit aussi bien de gauche à droite que de droite à gauche :
Exemples : ELLE – A – REVER – ICI – RADAR
Écrire un programme qui retourne « la chaine de caractère " ----- " est un palindrome » si une chaîne de caractères rentré au clavier est un palindrome et dans le cas contraire « la chaine de caractère " ----- " n'est pas un palindrome ».

Exercice 4. Conjugaison

Imprimer la table de conjugaison au présent de l'indicatif d'un verbe du 2ème groupe. On dispose pour cela :

- ⇒ de la liste des pronoms personnels : PRON
- ⇒ de la liste des suffixes de la conjugaison des verbes du second groupe : SUF

Exemple : PRON	SUF	Résultat avec finir
je	is	je finis
tu	is	tu finis
il	it	il finit
nous	issons	nous finissons
vous	issez	vous finissez
ils	issent	ils finissent

II. Les Listes (list)

Exercice 5. **Moyenne des éléments d'une liste**
Écrire un programme qui calcule la moyenne des éléments d'une liste de réels donnés.

Exercice 6. **Somme des éléments de deux listes**
Écrire un programme qui construit la liste somme de deux listes de n éléments chacune. On suppose dans un premier temps que les listes sont de même longueur.
Dans une version améliorée, si une des deux listes est plus petite, on recopiera telle quelle la fin de la liste la plus longue à la fin de la liste résultat.
Exemple : Soient les listes L1 = [12, 5, 18, 21] et L2 = [2, 8, 10, 50, 62, 100]
On veut construire la liste L3 = [14, 13, 28, 71, 62, 100].

Exercice 7. **Décalage à droite**
Écrire un programme qui effectue un décalage circulaire vers la droite des éléments d'une liste sans utiliser de tableau intermédiaire.
On demande de modifier la liste avec le décalage, pas seulement de l'imprimer.
Exemple : L : 1 6 3 0 8 Devient : 8 1 6 3 0

Exercice 8. **Insertion dans une liste triée**
On dispose d'une liste d'entiers rangés par ordre croissant.
Écrire un programme qui insère un élément à sa place dans ce tableau, c'est à dire en respectant l'ordre croissant.

1. Ecrire un programme qui recherche la place de l'élément à insérer :
   a. Par une recherche séquentielle
   b. Par une recherche dichotomique
2. Ecrire le programme qui, une fois sa p

III. Les fonctions

Exercice 0. **Tester les programmes suivants :**

Dans cette partie du TP vous aurez besoin de jeter un coup d'œil sur le cours !

```
# LDA
Fonction Addition(a,b:Entier):Entier
Debut
    Retourner a + b
FinFonction

# Python
def Addition(a,b):
    return a + b

# Test
print(Addition(5,8))
# affiche 13
```

- Tester cette fonction avec T = [4,88,14,25,64,75,23,5,22,3,44]
  - Il suffit de faire ***print(maxium(T))*** après avoir bien évidement copié le code ci-dessous

```
def maximum(T):
        if len(T)==1:
                return T[0]
        m=len(T)//2
        max1=maximum(T[:m])
        max2=maximum(T[m:])
        if max1>max2:
                return max1
        return max2
```

Soient les deux fonctions suivantes :

```
def Pair(N):
        if N==1:
                return False
        return Impair(N-1)

def Impair(N):
        if N==1:
                return True
        return Pair(N-1)
```

- Tester Pair(45), Pair(42), Impair(7) et Impair(88)

Exercice 9. **Fonction « carré »**

1. Ecrire une fonction « carre_de_x » ayant pour argument un nombre x et qui retourne la valeur x^2.
2. Ecrire une fonction « carre_liste » ayant pour argument une liste L_init et qui retourne une liste L_resultat contenant le carré de chaque terme de la liste L_init.
3. Tester vos fonctions précédentes avec une liste L_init contenant tous les entiers compris entre 0 et 10 inclus, par différentes méthodes :
   1ère méthode : partir d'une liste vide et ajouter les termes avec la méthode « L.append() ».
   2ème méthode : partir d'une liste de nombres de 0 à 10 avec la transformation « list() » et la fonction « range(départ, arrivée, pas) » et ensuite modifier chaque terme de la liste.
   3ème méthode : utiliser des listes par compréhension.

Exercice 10. Fonction « moyenne » et fonction « variance »
On rappelle que la variance est la moyenne des carrés des écarts à la moyenne. C'est aussi la moyenne des carrés moins le carré de la moyenne.
Ecrire une fonction « moyenne_variance » ayant pour argument une liste de nombres, et qui retourne la moyenne et la variance de ces nombres. Pour cela, on écrira un algorithme « naïf » utilisant de l'algorithmique de base.
Dans un deuxième temps, écrire une fonction « moyenne » et une fonction « variance » ayant pour argument une liste de nombres, et qui retournent respectivement la moyenne et la variance de ces nombres. On pourra utiliser des fonctions existantes et des listes par compréhension.

Exercice 11. Fonction « trouve »
Ecrire une fonction « trouve » qui, étant donnés une liste L et un élément x, retourne le booléen True si x est l'un des éléments de L, et False sinon.

Exercice 12. Insertion dans une liste triée
On dispose d'une liste L d'entiers rangés par ordre croissant. On souhaite écrire une fonction qui insère un élément nb à sa place dans cette liste, c'est-à-dire en respectant l'ordre croissant.

1. Ecrire une fonction recherche_place_dans_liste(L, nb) qui retourne la place de l'élément à insérer :
   a) Par une recherche séquentielle
   b) Par une recherche dichotomique
2. Ecrire une fonction insert_nb_dans_liste(place) qui, une fois sa place connue, insère l'élément dans la liste et retourne la nouvelle liste.

Exercice 13. Fonction lambda
Dans Python, on peut également définir une fonction d'une autre manière que celle rappelée en introduction :
<nom_de_fonction> = lambda <nom_des_paramètres>: <expression>
Exemple : $f(x)= x^2$

```
f = lambda x: x * x # définition de la fonction
f(2) # appel de la fonction
```

On peut aussi définir une fonction à plusieurs paramètres :
Exemple : $f(x, y)= x * y$

```
f = lambda x, y: x * y # définition de la fonction
f(2, 3) # appel de la fonction
```

1. Définir une fonction « cube » qui calcule le cube d'un nombre.
2. Définir une fonction « distance_carre » qui calcule $(x, y) \rightarrow x^2 + y^2$