TP6 : Les collections
(2h à 4h)

Consignes :

⇒ Ce travail est INDIVIDUEL
⇒ L'IDE à utiliser est PyCharm
⇒ Il est fortement recommandé d'ouvrir le support du CM5 durant toute la séance du TP6
⇒ Toutes les fonctions sont à écrire dans un fichier séparés appelé **functions.py**
⇒ Tous les tests des fonctions seront à faire dans le fichier **main.py**
⇒ Les instructions **print()** et **input()** sont interdites dans les fonctions sauf si le rôle même de la fonction est de permettre l'affichage ou la saisie d'informations utilisateur.

Acquis d'apprentissage :

A la fin de ce TP, chaque étudiant doit être capable :
- De manipuler les tuples
- De manipuler les sets
- De manipuler les dictionnaires
- D'identifier la structure à utiliser en fonction du problème.

Niveau débutant

Exercice 1

1. Écrire en langage Python une fonction qui prend en paramètre un tuple et qui le convertit en chaîne de caractères.
2. Tester la fonction dans le programme principal

Exercice 2

1. Écrire en langage Python une fonction qui prend en paramètre un tuple et qui retourne son 4ème élément, ainsi que son 4ème élément en partant de la fin.
2. Tester la fonction dans le programme principal

Exercice 3

1. Écrire en langage Python une fonction qui reçoit une chaîne de caractères contenant des nombres séparés par des virgules en paramètre, et qui génère puis retourne une liste et un tuple contenant les nombres composants cette chaine.

   Exemple :

   Entrée :
   ```
   S = « 34,67,55,33,12,98 »
   ```
   Sortie :
   ```
   ['34', '67', '55', '33', '12', '98']
   ('34', '67', '55', '33', '12', '98')
   ```
2. Tester la fonction dans le programme principal.

Exercice 4

1. Écrire en Python une fonction qui prend en paramètre un nombre **n** et qui génère un dictionnaire qui contient des nombres entre **1** et **n** sous la forme (x, x^2).

   Exemple :

   Entrée :

   n=5

   Sortie :

   {1 : 1, 2: 4, 3: 9, 4: 16, 5: 25}

2. Tester la fonction dans le programme principal.

Exercice 5

1. Écrire en Python une fonction qui prend en paramètres un dictionnaire et qui retourne la somme ainsi que le produit de toutes ses valeurs.
2. Tester la fonction dans le programme principal.

Niveau intermédiaire

Exercice 1

1. Construire un dictionnaire « Semestre2 » des 6 premiers mois de l'année 2022 avec comme valeurs le nombre de jours respectif.
2. Afficher la liste des mois et celle des jours.
3. Faire un test si l'année est bissextile et mettre à jour le nombre de jour pour le mois de février.
4. Construire le dictionnaire « Semestre1 » qui commence du 4 septembre pour une durée de 6 mois avec comme valeurs le nombre de jours respectifs.
5. Ajouter dans un dictionnaire « AnneeScolaire » les éléments du « Semestre1 » et ceux du « Semestre 2 »
6. Créer le dictionnaire « VacanceScolaire » qui associe à chaque mois le nombre de jours de vacances selon le calendrier des vacances scolaires de l'année 2021-2022
7. Créer manuellement le dictionnaire « weekend » qui associe à chaque mois le nombre de jours de weekend
8. Créer le dictionnaire « JoursTravail » qui associe à chaque mois le nombre de jours de travail effectif.

Exercice 2

Écrire puis tester une fonction qui combine deux dictionnaires en un seul en additionnant les valeurs ayant les mêmes clefs.

Exemple :

Entrée : :

d1 = {'a': 100, 'b': 200, 'c':300}

d2 = {'a': 300, 'b': 200, 'd':400}

Sortie :

{'a': 400, 'b': 400, 'd': 400, 'c': 300}

Exercice 3

Considérons le dictionnaire imbriqué suivant qui est composé de clés de type entier et de valeurs qui sont à leur tour de type dictionnaire :

```
personnes = {1 : {'nom': 'John', 'age': '27', genre: 'Homme'},
             2 : {'nom': 'Marie', 'age': '22', 'genre': 'Femme'}
            }
```

1. Ajoutez deux nouvelles personnes au dictionnaire « personnes »
2. Modifiez le nom de la personne ayant pour clé 1 de **John** à **Jonathan**
3. Supprimez la personne ayant pour clé 2.
4. Affichez toutes les paires de clés - valeurs du dictionnaire « personnes » à l'aide d'une boucle for. Assurez-vous de bien formater votre affichage.

Exercice 4

Écrire puis tester une fonction qui prend en paramètres deux chaînes de caractères et qui indique si l'une est une anagramme de l'autre ou pas.

Indication :

Une chaîne de caractères est dite anagramme d'une autre si elle est composée des mêmes lettres mais à des positions différentes.

Niveau avancé

Exercice 1

Imaginez que vous travaillez actuellement dans une entreprise informatique et que vous recevez la liste d'employés suivante.

Nom	Âge	Département
Jean Mackee	38	Vente
Lisa Crawford	29	Management
Sujan Patel	33	RH

Votre responsable vous demande d'utiliser Python pour stocker ces données en vue d'une utilisation ultérieure par l'entreprise.

Pour répondre à cette demande :

1. Utiliser une liste de dictionnaires pour stocker ces 3 employés.
2. Proposer à l'utilisateur un menu lui permettant :
    a. D'afficher l'ensemble des employés de la liste sous la forme suivante :

```
Nom : Jean Mackee
Âge : 38
Département : Vente
--------------------
Nom : Lisa Crawford
Âge : 29
Département : Management
--------------------
Nom : Sujan Patel
Âge : 33
Département : RH
--------------------
```

    b. D'afficher les informations détaillées d'un employé dont le nom est donné par l'utilisateur (penser à utiliser une fonction)
    c. D'afficher l'ensemble des employés ayant un âge inférieur à une valeur **a** donnée par l'utilisateur.

Exercice 2

Écrire puis tester une fonction qui prend en paramètres une liste de tuples (nom, age, taille) et qui l'ordonne dans l'ordre croissant. La liste de tuples est donnée par l'utilisateur, et le critère de tri est le suivant :

(a) d'abord le nom (ordre alphabétique) ;

(b) puis l'âge ;

(c) puis la taille.

Exemple :

Entrée :

Tom,19,80

John,20,90

Jony,17,91

Jony,17,93

Json,21,85

Sortie :

[('John', '20', '90'), ('Jony', '17', '91'), ('Jony', '17', '93'), ('Json', '21', '85'), ('Tom', '19', '80')]

Exercice 3

Voici une base de données reportant le nombre de ventes réalisées par des employés dans quatre différentes régions géographiques.

	N	S	E	O
John	3056	8463	8441	2694
Tom	4832	6786	4737	3612
Anne	5239	4802	5820	1859
Fiona	3904	3645	8821	2451

1. En utilisant un dictionnaire imbriqué (dictionnaire de dictionnaire), représenter les informations du tableau ci-dessus.
2. Écrire une fonction qui à partir d'un nom d'employé et d'une région, elle retourne le nombre de ventes réalisées.
3. Ajouter une fonction qui permet à partir d'un nom d'employé et d'une région de modifier le nombre de ventes associé.
4. Ajouter une fonction qui permet d'afficher toutes les ventes réalisées dans une région géographique donnée.

Exercice 4

Partie I

Le but ici est de créer un moyen d'indexation des mots d'un livre ou d'un document. Pour cela, nous souhaitons implémenter un dictionnaire **Index** où les clés représentent les mots et les valeurs seront la liste des numéros de pages dans lesquelles figurent les mots. Les numéros de pages doivent figurer de façon unique pour chaque mot.

1. Écrire une fonction **Add** qui prend en paramètre un mot et un numéro de page et qui permet de l'ajouter à l'index lorsque cela est possible.
2. Écrire une fonction **remove** qui prend en paramètres un mot et qui permet de le supprimer (ainsi que toutes les pages dans lesquelles il figure) de l'index.
3. Écrire une fonction **search** qui prend en paramètre un mot et qui permet de retourner toutes les pages dans lesquelles il figure.
4. Écrire une fonction **display** qui affiche le contenu de l'index trié par ordre alphabétique et dont les numéros de pages sont triés dans l'ordre croissant.

Exemple d'affichage :

```
Liste : 8, 50
```

```
Python : 12,25
Tuple : 30
```

5. Tester les 4 fonctions précédentes dans un programme principal.

Partie II

À partir de l'index précédent, nous souhaitons générer un index inverse **reverse_index** qui permet d'associer à chaque numéro de page la liste des mots qu'elle contient.

1. Écrire une fonction **create_reverse_index** qui prend en paramètre un index et qui permet de créer un reverse_index.

2. Écrire une fonction **display_reverse_index** qui permet d'afficher le contenu de de reverse_index par ordre croissant sur les numéros de page, l'ensemble de mots pour chaque page doit être affiché par ordre alphabétique.

   Exemple d'affichage de cette méthode :

```
10 : Python
12 : Python tuple
15 : dictionnaire
25 : Langage Python
45 : addition type variables
```