TRAVAUX PRATIQUES n° 13
Dictionnaires

Dans ce TP, on apprend à créer et manipuler des **dictionnaires** Python : association clé ↦ valeur, test d'appartenance (`cle in dico`), parcours (`for cle in dico`), et applications (comptage, traduction, analyse de texte).

Exercice 1 (*Épicerie*) : Dans cet exercice, nous allons créer et manipuler un dictionnaire permettant de stocker les prix des différents articles d'une épicerie.

1) Créer un dictionnaire `dictionnaire_de_prix` dont les clés sont les noms des produits et les valeurs sont les prix de ces produits en se basant sur le tableau suivant :

Produit	Prix en euros
Beurre	2.61
Café	5.29
Céréales	2.99
Jus de fruits	2.51
Lait	5.46
Oeufs	1.49
Pâtes	1.70
Riz	2.39

2) Écrire une fonction `disponibilite(produit, dictionnaire_de_prix)` qui renvoie `True` si `produit` est une clé de `dictionnaire_de_prix` et `False` sinon.
3) Écrire une fonction `prix_moyen(dictionnaire_de_prix)` calculant le prix moyen des produits disponibles.
4) Écrire une fonction `fourchette_prix(a, b, dictionnaire_de_prix)` qui renvoie la liste des produits dont le prix est compris entre `a` et `b` **inclus**.
5) Écrire une fonction `prix_achats_1(panier, dictionnaire_de_prix)` qui, étant donné une liste de produits `panier`, renvoie le prix total du panier. On suppose que tous les produits du panier sont disponibles dans le dictionnaire.

   Par exemple : `prix_achats_1(["Jus de fruits", "Pâtes"], dictionnaire_de_prix)` renvoie `4.21`.
6) Écrire une fonction `prix_achats_2(panier, dictionnaire_de_prix)` qui, étant donné un dictionnaire de produits `panier` sous la forme `produit: quantité`, renvoie le prix total du panier. On suppose encore que tous les produits du panier sont disponibles.

   Par exemple : `prix_achats_2({"Jus de fruits": 2, "Pâtes": 3}, dictionnaire_de_prix)` renvoie `10.12`.

Exercice 2 (*Occurrences*) : Écrire une fonction `occurrences(L)` qui, étant donné une liste de nombres `L`, renvoie un dictionnaire dont les clés sont les éléments de la liste `L` et les valeurs sont le nombre de fois où l'élément apparaît dans la liste.

Par exemple : `occurrences([4, 3, 2, 5, 2, 3, 2, 1])` renvoie `{1: 1, 2: 3, 3: 2, 4: 1, 5: 1}`.

Exercice 3 (*Scrabble*) : Le dictionnaire `scrabble` dans le fichier « TP 13.py » contient les points des lettres au scrabble français.

1) Écrire une fonction `points_1(mot)` qui calcule le nombre de points du mot (on suppose que `mot` est écrit en majuscules).
2) Écrire une fonction `points_2(mot, lettres_compte_double)` qui calcule le nombre de points du mot en comptant double les lettres dont les indices sont dans la liste `lettres_compte_double`.

   Par exemple : `points_2("BCPST", [1, 3])` renvoie `15` points.

Exercice 4 (*QCM*) : Les réponses correctes d'un QCM de Maths sont stockées dans un dictionnaire nommé `bonnes_reponses`.

Les clés sont des chaînes de caractères de la forme `"Q1"`. Les valeurs possibles sont des chaînes de caractères correspondant aux quatre réponses `"a"`, `"b"`, `"c"`, `"d"`.

Par exemple, `bonnes_reponses` peut être de la forme `{"Q1":"c", "Q2":"a", "Q3":"d", "Q4":"c", "Q5":"b"}`.

Les réponses données par un élève sont stockées dans un dictionnaire `reponses_eleve` sous la même forme que `bonnes_reponses`. Lorsque l'élève n'a pas répondu à une question, il n'y a pas de clé correspondant au nom de la question.

Par exemple, `reponses_eleve` peut être de la forme `{"Q1":"b", "Q2":"a", "Q3":"d", "Q5":"a"}`.

La notation d'un QCM de Maths est la suivante : 3 points par réponse correcte, -1 point par réponse incorrecte et 0 si l'on n'a pas répondu.

Écrire une fonction `note(reponses_eleve, bonnes_reponses)` qui calcule et renvoie la note de l'élève au QCM.

Exercice 5 (*Morse*) : Le dictionnaire `code_morse` dans le fichier « TP 13.py » permet de traduire chaque lettre en Morse.

1) Écrire une fonction `traduction_francais_morse(texte)` qui traduit un texte écrit en majuscule en Morse en mettant 2 espaces entre chaque lettre et 7 espaces entre chaque mot.
2) Écrire une fonction `traduction_morse_francais(code)` qui traduit un code Morse en français avec les mêmes règles entre chaque lettre et entre chaque mot.
3) Traduire le code Morse `message` fourni dans le fichier « TP 13.py ».

Exercice 6 (*Le retour des misérables*) : Placez le fichier « Les misérables.txt » du TP 9 **dans le même dossier** que votre script Python, avec le même `chdir(r"...")` qu'au TP 9 : il vous servira à tester vos fonctions.

1) Écrire une fonction `occurrences_lettres(texte)` qui renvoie le dictionnaire contenant les occurrences de chaque lettre du texte. On regroupera les lettres majuscules et minuscules qui sont identiques et on ignorera tous les autres caractères (espaces, ponctuation, etc.).
2) Écrire une fonction `pourcentage_lettres(texte)` qui renvoie le dictionnaire contenant les fréquences de chaque lettre du texte (proportion parmi les lettres uniquement).
3) Écrire une fonction `tri_par_occurrences(D)` qui prend en paramètre un dictionnaire `D` et qui renvoie la liste des clés triées par ordre décroissant de leur valeur. Appliquer alors cette fonction en combinaison de la fonction `occurrences_lettres` sur le texte « Les misérables.txt ».