Types construits

Exercice 1 (Vrai/Faux)
Pour chacune des affirmations suivantes, dire si elle est vraie ou fausse. Justifier si possible la réponse.

1. Le tuple `(1,2,(3,4),5)` a pour longueur 5.
2. Si `t='a','d','c'`, l'instruction `t[1]='b'` provoque une erreur.
3. l'expression `3*(1,2)` a pour valeur `(3,6)`.
4. Soit `L` la liste `[1,2,3]`. Après l'instruction `L.append([4,5])`, la liste `L` vaut `[1,2,3,4,5]`.
5. Après l'instruction :

```
L=[3*x for x in [1,2,3] if x%2!=0]
```

   la liste `L` a pour valeur `[3,9]`.
6. Soit `nombres` une liste de nombres. Pour modifier le dernier élément, on écrit : `nombres(len(nombres))=...`
7. Soit `d` un dictionnaire vide. L'instruction `v=d['un']` provoque une erreur.
8. Soit `d` un dictionnaire. L'instruction `d[(10,15)]=5` provoque une erreur.
9. Soit `d` un dictionnaire. L'instruction `d[[10,15]]=5` provoque une erreur.

Exercice 2 (QCM)
Pour chaque question, une seule réponse est valable. Indiquer laquelle.

1. On considère le tuple `t=(3,5,1)`. Qu'obtient-on après l'instruction `t[1]=4` ?
   (a) La valeur de `t` est `t=(4,5,1)`.
   (b) La valeur de `t` est `t=(3,4,5)`.
   (c) La valeur de `t` est `t=(3,5,4)`.
   (d) Une erreur
2. Après l'instruction `a = 0,2+0,3==0,5`, qu'obtient-on ?
   (a) `a` vaut `False` (à cause des flottants).
   (b) `a` vaut `True` (car l'égalité est bien vraie dans Python).
   (c) `a` vaut `(0,2,False,5)`.
   (d) Une erreur
3. Soit la liste `L=[15,17,12,23]`. Après l'instruction `L[2]=25`, la liste `L` vaut :
   (a) `[15,17,25,23]`
   (b) `[15,25,12,23]`
   (c) `[15,17,25,12,23]`
   (d) `[15,25,17,12,23]`
4. On dispose de la liste `L=[[1,2,3],[4,5,6],[7,8,9]]`. La valeur de `L[1][2]` est :
   (a) `2` (b) `4` (c) `6` (d) `8`

5. Après l'instruction `L=[[i,i+1] for i in range(2)]`, la valeur de `L` est :

   (a) `[[0,1],[1,2]]`
   (b) `[[1,2],[2,3]]`
   (c) `[0,1,1,2]`
   (d) `[1,2,2,3]`

6. On construit une matrice avec le code suivant :

```
matrice = [3*[0] for i in range(3)]
for i in range(3):
    matrice[i][i]=i+1
    matrice[0][i]=matrice[0][i]+i+1
    matrice[i][2]=matrice[i][2]+i+1
```

   Quel est le résultat obtenu ?

   (a) `[[1,2,4],[0,2,2],[0,0,6]]`
   (b) `[[2,2,4],[0,2,2],[0,0,4]]`
   (c) `[[1,2,4],[0,2,4],[0,0,6]]`
   (d) `[[2,2,4],[0,2,2],[0,0,6]]`

7. On considère le code Python suivant :

```
def f():
    liste1.append(2)
    liste2=liste1+[3]
liste1=[0,1]
liste2=[0,1]
f()
```

   Quel est le résultat obtenu ?

   (a) La liste `liste1` vaut `[0,1,2]` et la liste `liste2` vaut `[0,1,2,3]`.
   (b) La liste `liste1` vaut `[0,1,2]` et la liste `liste2` vaut `[0,1,3]`.
   (c) La liste `liste1` vaut `[0,1,2]` et la liste `liste2` vaut `[0,1]`.
   (d) La liste `liste1` vaut `[0,1]` et la liste `liste2` vaut `[0,1]`.

8. Parmi les termes suivants, lequel n'est pas une méthode d'un dictionnaire ?

   (a) `data` (b) `items` (c) `keys` (d) `values`

9. On considère le code suivant :

```
d={"if":"si","yes":"oui","no":"non"}
for c in d:
    print(c)
```

   Qu'obtient-on ?

   (a) L'affichage de `if`, `yes` et `no`.
   (b) L'affichage de `si`, `oui` et `non`.
   (c) L'affichage des couples `('if','si')`, `('yes','oui')` et `('no','non')`.
   (d) Une erreur.

Exercice 3 (Tuples)

1. Définir une fonction `vecteur(A,B)` qui prend en argument deux couples (tuples à 2 éléments) de nombres et qui retourne le couple des coordonnées du vecteur $\overrightarrow{AB}$.

   Donner une définition utilisant les indices pour accéder aux valeurs de `A` et `B`, puis une définition utilisant l'affectation multiple.

   Rappel mathématique : si $A(x_A,y_A)$ et $B(x_B,y_B)$, alors $\overrightarrow{AB}(x_B - x_A,y_B - y_A)$

```
>>> A = 5,2
>>> B = 7,-2
>>> vecteur(A,B)
(2, -4)
```

2. Définir une fonction `norme(u)` qui prend en argument un couple de coordonnées d'un vecteur `u` et qui retourne la norme de ce vecteur.
   Rappel mathématique : si $\overrightarrow{u}(x,y)$, alors sa norme est $\|\overrightarrow{u}\| = \sqrt{x^2+y^2}$.
   Aide : pour utiliser la fonction racine carrée (`sqrt`), il faut l'importer du module `math` : `from math import sqrt`
   Donner là aussi une définition utilisant les indices pour accéder aux valeurs de `u`, puis une définition utilisant l'affectation multiple.

```
>>> u = 3,4
>>> norme(u)
5.
```

3. À l'aide de la fonction précédente, comment obtenir la norme du vecteur dont les coordonnées sont (2,5) sans donner de nom à ce vecteur ?
4. À l'aide des fonctions `vecteur` et `norme`, définir une fonction `distance(A,B)` qui prend en argument deux couples de nombres et retourne la distance AB, qui vaut la norme du vecteur $\overrightarrow{AB}$.

```
>>> distance(A,B)
4.47213595499958
```

Exercice 4 (Dictionnaires)
On considère le code suivant :

```
dict_eleve = {'nom': 'Figny','prénom': 'Charline','âge': 16}
```

1. Comment accéder au nom de l'élève ? À son prénom ?
2. Comment obtenir le nombre d'éléments de ce dictionnaire ?
3. Ajouter la moyenne de Charline, qui s'élève à 12.34.
4. On vient de célébrer l'anniversaire de Charline qui a maintenant 17 ans. Changer son âge.
5. Charline ne souhaite pas que son nom soit utilisé. Supprimer son nom du dictionnaire.

Exercice 5 (Dictionnaires et un peu de tuples)

1. Créer un dictionnaire vide de deux manières différentes.
2. Créer le dictionnaire `utilisateur` avec les couples clés, valeurs suivants (attention, la plupart sont des chaînes de caractères) :

clé	valeur
nom	Jinvante
prenom	Anon
age	22
password	jhgfd54.*

3. Finalement, la clé `'password'`, en anglais, de `utilisateur` ne nous plaît pas, et on veut la changer en `'mdp'`.
   Réaliser ce changement.
   Aide : Le changement de clé n'est pas aussi facile que le changement de valeurs. Pour cela, il faut créer l'entrée avec la clé `'mdp'` avec pour valeur celle de la clé `'password'`, puis supprimer l'entrée avec la clé `'password'`.

4. Écrire une fonction `ListeCoupleVersDict` qui prend en argument une liste de couples de la forme (clé, valeur) et retourne le dictionnaire correspondant :

```
>>> entree = [('cle', 'val'), (2, 'deux'), (True, 0)]
>>> ListeCoupleVersDict(entree)
{'cle': 'val', 2: 'deux', True: 0}
```

Essayer de définir deux versions différentes dans la manière de parcourir la liste et récupérer le contenu de ses éléments.

Remarque : la fonction `dict` fait exactement ce qui est demandé, mais il s'agit de ne pas l'utiliser ici.

```
>>> dict([(1, 2), (3, 4)])
{1: 2, 3: 4}
```

Exercice 6 (Dictionnaires)

Un site de jeux vidéos enregistre les scores de ses joueurs au jeu Pacman dans un dictionnaire :

```
scores_pacman = {'erode': 34, 'honorine': 456, 'moussa': 89,
                 'octave': 542, 'marine': 12, 'patricia': 631}
```

Attention à bien tester les réponses, en particulier les fonctions à définir.

1. Ajouter le score de Sabrina qui a 542 points.

   Attention à respecter la convention : tous les prénoms sont écrits en minuscule.

2. Créer une fonction `ScoreJoueur(score_jeu, joueur)` qui prend un dictionnaire tel que le précédent et le nom d'un joueur comme 'Sabrina' et retourne son score si le joueur est inscrit ou 0 sinon.

   Utiliser au moins deux méthodes différentes pour le faire.

    Attention à nouveau à la convention d'enregistrement des noms.

   Aide : la méthode `lower` permet de transformer une chaîne de caractère en minuscules :

   ```
   >>> chaine = "AbCd"
   >>> chaine.lower()
   'abcd'
   ```

3. Créer une fonction `InscrireJoueur(score_jeu, joueur)`, où `score_jeu` est un dictionnaire tel que le précédent et `joueur` est une chaîne de caractère, permettant d'inscrire un joueur.

   Elle retourne `True` si le joueur n'est pas déjà inscrit après avoir modifié le dictionnaire pour associer un score de 0 au joueur, `False` s'il est déjà inscrit.

   Utiliser deux méthodes différentes pour le faire.

4. Définir une fonction `ListeJoueurs(score_jeu)` qui prend en argument un dictionnaire de scores et retourne la liste des joueurs.

5. Définir une fonction `MoyenneScores(score_jeu)` qui prend en argument un dictionnaire de scores et retourne la moyenne des scores.

6. Créer une fonction `AfficheScores` qui prend un dictionnaire de scores en paramètre et affiche une série de phrases telles que celle ci-dessous, séparées par des retours à la ligne :

   ```
   Le score de Erode est 34
   ...
   ```

    Attention à nouveau à la façon dont est noté le prénom, cette fois-ci avec la première lettre en majuscule.

   La méthode `capitalize` est faite pour cela :

```
>>> chaine = "aBcD"
>>> chaine.capitalize()
'Abcd'
```

7. Définir une fonction `ListeScores` qui prend un dictionnaire de scores en paramètre et n'affiche rien mais retourne une liste de phrases de la forme `'Le score de Erode est 34'`.
8. (Facultatif)
   Redéfinir la fonction `ListeScores` précédente de sorte à avoir les scores par ordre décroissant.
   Aide : On pourra utiliser entre autres :
   - une liste construite à partir des couples clé, valeurs du dictionnaire :
     ```
     >>> dico = {'a':2,'b':1}
     >>> liste = list(dico.items())
     >>> liste
     [('a', 2), ('b', 1)]
     ```
   - la fonction `sorted`, avec l'argument `reverse` permettant de trier à l'envers :
     ```
     >>> l = list(range(1,6)) # valeurs croissantes de 1 à 5
     >>> sorted(l,reverse=True)
     [5, 4, 3, 2, 1]
     ```
     et l'argument `key` qui permet d'indiquer selon quoi trier (ici selon le deuxième élément de chaque couple) :
     ```
     >>> sorted(liste,key = lambda couple:couple[1])
     [('b', 1), ('a', 2)]
     ```
   On pourra de même faire la liste triée par nom croissant.

Exercice 7 (Dictionnaires)
Le Scrabble est un jeu de société où l'on doit former des mots avec un tirage aléatoire des lettres, chaque lettre valant un certain nombre de points. Le tableau suivant donne la valeur de certaines lettres :

Lettre	A	B	C	D	E	F	G	H	I	J	K
Valeur	1	3	3	2	1	4	2	4	1	8	5

1. Définir un dictionnaire dont les clés sont les lettres et les valeurs sont la valeur des lettres au Scrabble indiquées par le tableau ci-dessus.
2. Définir une fonction qui :
   (a) Demande un mot (en utilisant la fonction `input` qui retourne un élément de type `str`).
   (b) Retourne la valeur totale du mot (on comptera 0 pour une lettre absente du dictionnaire).

    Transformer le mot donné en majuscules. La méthode `upper` est faite pour cela :
   ```
   >>> chaine = "AbCd"
   >>> chaine.upper()
   'ABCD'
   ```

Exercice 8 (Dictionnaires)
Le dictionnaire suivant indique les valeurs des lettres dans les règles françaises du Scrabble :

```
valeurs_Scrabble = {10: 'KWXYZ', 8: 'JQ', 4: 'FHV', 3: 'BCP', 2: 'DMG'}
```

sachant que les lettres qui manquent ont pour valeur 1.

1. Définir à partir de `valeurs_Scrabble` un dictionnaire similaire à celui de l'exercice 7, c'est à dire dont les clés sont les lettres et les valeurs sont leur valeur au Scrabble, sans ajouter les lettres de valeur 1.

2. Redéfinir la fonction de l'exercice 7, mais en comptant cette fois 1 pour les lettres absentes du dictionnaire.

Exercice 9 (Dictionnaires – facultatif)
On a construit un dictionnaire ayant pour clés des couples contenant les coordonnées GPS de villes et pour valeurs les villes correspondantes. On trouve les coordonnées sur Internet par exemple. Les données sont sous forme décimale en degré. On a par exemple :

```
positions = {}
positions[(48.853585, 2.301490)] = "Paris"
positions[(11.611358, 43.147752)] = "Djibouti"
positions[(37.023113, -8.996601)] = "Fortaleza de São Vicente"
positions[(7.677989, -5.025387)] = "Bouaké"
# etc.
```

Nous supposons avoir reçu une photo prise avec un smartphone par une personne en vacances. Nous voyons dans les propriétés les coordonnées GPS au moment de la prise de vue. Écrire une fonction prenant en paramètres un couple de coordonnées GPS et le dictionnaire construit et renvoyant le nom du lieu correspondant, ou `None` si aucune ville du dictionnaire ne correspond.
On tolère une précision au dix-millième de degré.
Par exemple, si les coordonnées sont `(37.02311, -8.9966)`, la fonction doit nous renvoyer : `"Fortaleza de São Vicente"`.

Exercice 10 (Dictionnaires – mini-projet de groupe)
Écrire un programme Python permettant de gérer un répertoire téléphonique sous forme d'un dictionnaire. Un premier répertoire contenant des noms et numéros de téléphone sera défini au préalable dans le code Python. Le programme doit permettre, lorsque l'on exécute une fonction `main()`, d'ajouter, modifier, supprimer une entrée, mais également de faire des recherches (par nom, par numéro). Ces actions seront faites à partir d'un « menu » affiché, où le choix est demandé à l'utilisateur, y compris celui de quitter le programme, et qui revient en boucle.
Pour aller plus loin, on pourra faire en sorte de sauvegarder le répertoire dans un fichier, de le charger en mémoire.