RND. Types construits (dictionnaires). Exercices

Exercice 1 : Vrai / Faux

	VRAI	FAUX
Soit *d* un dictionnaire dont les clés sont les noms des pays et les valeurs leur capitale. Si la France n'est pas dans le dictionnaire, l'instruction suivante : d['France'] = 'Paris' provoque une erreur.		
Affirmation : Une clé d'un dictionnaire peut être une liste.		
Soit d'un dictionnaire vide. L'instruction v = d['un'] provoque une erreur.		
Affirmation : Une valeur d'un dictionnaire peut être un dictionnaire.		
L'instruction get(*clé, valeur par défaut)* permet d'éviter de stopper le programme si la *clé* n'existe pas.		
Les accolades permettent à Python de savoir qu'un objet est un dictionnaire.		

Exercice 2 : QCM

Pour chaque question, une seule réponse est correcte parmi les quatre proposées.

Question 1 : Parmi les termes suivants, lequel n'est pas une méthode d'un dictionnaire ?

1. data.
2. items.
3. keys.
4. values.

Question 2 : Qu'affiche le programme suivant ?

```
d = {"if" : "si" , "yes" : "oui" , "no" : "non"}

for c in d :
    print(c)
```

1. if yes no.
2. si oui non.
3. ('if' : 'si') , ('yes' : 'oui') , ('no' : 'non')
4. L'écriture de l'instruction provoque une erreur.

Question 3 : Comment peut-on accéder à la valeur associée à une clé dans un dictionnaire ?

1. Il faut parcourir le dictionnaire avec une boucle à la recherche de la clé.
2. On peut y accéder directement à partir de la clé.
3. On ne peut pas accéder à une valeur contenue dans un dictionnaire à partir d'une clé.
4. Il faut d'abord déchiffrer la clé pour accéder à un dictionnaire.

Question 4 : Après avoir défini : d = { 'tigre': 'félin', 'tortue': 'reptile', 'renard': 'canidé' }, laquelle des quatre expressions suivantes est correcte ?

1. d['4']
2. d['reptile']
3. d['tortue' : 'reptile']
4. d['tortue']

Question 5 : On exécute le script suivant :

```
inventaire = {'pommes': 430, 'bananes': 312,'oranges' : 274, 'poires' : 137}

stock = 0
for fruits in inventaire :
    if fruits != "bananes" :
        stock = stock + inventaire[fruits]
```

Quel contient la variable stock à la fin de l'exécution ?

1. { 430, 274, 137 }.
2. 841.
3. 312.
4. { 'pommes' , 'oranges' , 'poires' }

A noter : Pour les exercices suivants, on pourra vérifier les réponses en programmant les algorithmes sur « Jupyter »

Exercice 3 :

Au zoo de Beauval, il y a 5 éléphants d'Asie, 17 écureuils d'Asie, 2 pandas d'Asie' etc.
Au zoo de la flèche, il y a 4 ours d'Europe, 7 tigres d'Asie, 11 girafes d'Afrique etc.
On représente ces inventaires à l'aide de deux dictionnaires :

```
zooBeauval = { 'éléphant' : ('Asie',15) , 'écureuil' : ('Asie', 17) ,
               'panda' : ('Asie' , 2) , 'hippopotame' : ('Afrique', 7) ,
               'girafe' : ('Afrique', 4) }

zooFlèche = { 'ours' : ('Europe',4) , 'tigre' : ('Asie', 7) ,
              'girafe' : ('Afrique' , 11) , 'hippopotame' : ('Afrique', 3) }
```

1/ **Ecrire** une fonction qui prend en **paramètre** un **zoo** et qui **renvoie** le **nom de l'animal le plus représenté**.
2/ **Ecrire** un programme donnant les **animaux en commun** aux deux zoos.
3/ **Ecrire** une fonction qui prend en **paramètre** un **zoo** et un **continent** et qui **renvoie** le **nombre total d'animaux originaires** ce **continent**.

Exercice 4 :

On modélise dans un dictionnaire les notes des élèves obtenues dans certaines spécialités comme ceci :

```
notes = { 'Emma' : ('Math',16) , 'Lucas' : ('NSI', 17) ,
          'Manon' : ('SVT' , 12) , 'Enzo' : ('NSI', 14) ,
          'Chloé' : ('SVT', 14) , 'Nicolas' : ('Math' , 15) }
```

1/ **Ecrire** l'instruction qui ajoute l'élève **Léa** qui a obtenu **14** en **NSI**.
2/ **Ecrire** un programme donnant **l'élève** qui a obtenu la **meilleure note toutes matières confondues**.
3/ Ecrire la fonction *TriParMatieres*(*notes : dict*) qui donne le résultat suivant :

```
notes = { 'Emma' : ('Math',16) , 'Lucas' : ('NSI', 17) ,
          'Manon' : ('SVT' , 12) , 'Enzo' : ('NSI', 14) ,
          'Chloé' : ('SVT', 14) , 'Nicolas' : ('Math' , 15) ,
          'Léa' : ('NSI' , 14) }

print(TriParMatiere(notes))

# Résultat attendu
{ 'Math' : [16,15] , 'NSI' : [17,14,14] , 'SVT' : [12,14] }
```

Exercice 5 : **GPS et dictionnaire**

On a construit un dictionnaire ayant pour clés des couples contenant les coordonnées GPS de villes et pour les villes correspondantes. On trouve les coordonnées sur Internet par exemple. Les données sont sous forme décimale en degré.

```
positions = {}   # Dictionnaire vide
positions[(48.853585, 2.301490)] = "Paris"
positions[(11.611358, 43.147752)] = "Djibouti"
positions[(37.023113, -8.996601)] = "Fortaleza"
positions[(7.677989, -5.025387)] = "Bouaké"
```

On suppose avoir reçu une photo prise avec un smartphone et on voit les coordonnées GPS de l'appareil.

Question : Ecrire une fonction prenant en **paramètres** un **couple de coordonnées** GPS et le **dictionnaire** construit et renvoyant le **nom** du lieu correspondant.
On tolère une précision de 1/10000 de degré.
Exemple : Si les coordonnées sont (37.02311 , -8.9966) le fonction doit retourner '*Fortaleza*'.

Exercice 6 : **Comparaison liste de listes à deux éléments et dictionnaire**

La fonction *time* permet de mesurer la durée d'exécution d'un programme. Voici son mode d'utilisation :

```
from time import time

st = time()    # On récupère l'heure à l'instant initial

################### Programme à tester ###############

print(time() - st)    # Durée d'exécution du programme
```

1/ **Ecrire** et exécuter le programme suivant en notant la durée d'exécution

Pour la liste de listes

```
from time import time
from random import randint

def recherche(liste,hasard) :
    if hasard in liste :
        return liste[1]

st = time()    # On récupère l'heure à l'instant initial

liste = [[i,i] for i in range(1000001)]

for i in range(50) :  # 50 recherches aléatoires
    hasard = randint(0,1000000)  # On prend un nombre au hasard entre 0 et 10000000
    valeur = recherche(liste,hasard)  # On cherche la valeur dans la liste

print(time() - st)    # Durée d'exécution du programme
```

Pour le dictionnaire

```
from time import time
from random import randint

def recherche(dico,hasard) :
    return dico.get(hasard)

st = time()    # On récupère l'heure à l'instant initial

dico = { i : i for i in range(1000001)}

for i in range(50) :  # 50 recherches aléatoires
    hasard = randint(0,1000000)  # On prend un nombre au hasard entre 0 et 10000000
    valeur = recherche(dico,hasard)  # On cherche la valeur dans le dictionnaire

print(time() - st)    # Durée d'exécution du programme
```

2/ **Conclure** sur l'efficacité d'un dictionnaire ou d'une liste pour des recherches multiples.