# RND. Types Construits (dictionnaires). Exercices. Corrigés

***Exercice 1* : Vrai / Faux**

*Q.1 : FAUX. Cela ajoute le couple ('France', 'Paris') dans le dictionnaire.*
*Q.2 : FAUX. Une clé est non mutable, or une liste l'est. C'est donc impossible.*
*Q.3 : VRAI. Voir cours.*
*Q.4 : VRAI. On notera que ce cas est même très fréquent.*
*Q.5 : VRAI. Voir cours.*
*Q.6 : VRAI. Voir cours.*

***Exercice 2* : QCM**

*Q.1 : Réponse 1. Voir cours.*
*Q.2 : Réponse 1. Par défaut, les itérations se font sur les clés d'un dictionnaire.*
*Q.3 : Réponse 2. Il suffit d'écrire nomdudico['clé'].*
*Q.4 : Réponse 4. Voir cours.*
*Q.5 : Réponse 2. On additionne bien les valeurs de chaque fruit (sauf les bananes).*

***Exercice 3* :**

*1/ Un exemple de programme :*

```
def PlusGrandNombre(zoo) :
    nomMax = ""     # On prend des valeurs vides ou nulles
    nombreMax = 0

    for (nom,(continent,nombre)) in zoo.items() :
        if nombreMax < nombre :  # Si le nombre d'animaux trouvés est supérieur à nombreMax,
            nombreMax = nombre   # il devient nombreMax.
            nomMax = nom

    return nomMax

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)}


print(PlusGrandNombre(zooBeauval))
print(PlusGrandNombre(zooFlèche))
```

```
écureuil
girafe
```

*2/ Un exemple de programme*

```
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)}


print(zooBeauval.keys() & zooFlèche.keys())
```

```
{'hippopotame', 'girafe'}
```

*3/ Un exemple de programme*

```
def NombreTotal(zoo,continent) :
    somme = 0

    for (origine,nombre) in zoo.values() :
        if origine == continent :
            somme = somme + nombre

    return somme



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)}


print(NombreTotal(zooBeauval,'Asie'))
print(NombreTotal(zooFlèche,'Afrique'))
```

```
34
14
```

***Exercice 4** :*

*1/ Voici l'instruction*

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

notes['Léa'] = ('NSI',14)

print(notes)
```

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

*2/ Un exemple de programme*

```
def MeilleureNote(notes) :
    bestNote = 0       # Initialisation à 0 ou vide
    bestEleve = ""
    for nom, (matière,note) in notes.items() :  # On parcourt le dictionnaire
        if note > bestNote : # Si la note trouvée est supérieure à bestnote
            bestNote = note  # elle devient bestnote
            bestEleve = nom

    return bestEleve


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

print(MeilleureNote(notes))
```

```
Lucas
```

*3/ Un exemple de programme*

```
def TriParMatiere(notes) :
    notesTriees = {}   # Dictionnaire allant recueillir les notes par matière
    for (nom,(matière,note)) in notes.items() :
        if matière in notesTriees :           # Si il y a déjà une note, on l'ajoute
            notesTriees[matière].append(note)  # à la liste
        else :                                # Sinon, on ajoute un couple (matière,note)
            notesTriees[matière] = [note]

    return notesTriees

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))
```

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

***Exercice 5* : GPS et dictionnaire**

*Un exemple de programme*

```
def TrouverLieu(positions,gps) :
    for k in positions.keys() :  # On itère sur les clés
        if ( abs(k[0] - gps[0]) and abs(k[0] - gps[0]) ) < 1/10000 :
            return positions[k] # Si l'écart en valeur absolue entre
                                # la position et le gps correspond à 1/10000 près

positions = {}
positions[(48.853585 , 2.301490)] = "Paris"
positions[(11.611358 , 43.147752)] = "Djibouti"
positions[(37.0223113 , -8.996601)] = "Fortelaza"
positions[(7.677989 , -5.025387)] = "Bouaké"

print(TrouverLieu(positions, (37.022311 , -8.9966)))
```

```
Fortelaza
```

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

*2/ Il  y a un rapport d'au moins 10 entre le temps de recherche dans un dictionnaire et une liste de listes à deux éléments. Un dictionnaire est clairement plus adapté pour des recherches.*

*A noter : Cela est dû au fait qu'un dictionnaire est constamment ordonné et donc une recherche de type « dichotomique » peut être effectué et est très efficace (d'une complexité de l'ordre de $\log_2$(nombre éléments)).*