Les types construits exercices

Nom :

Note : / 20 / 35

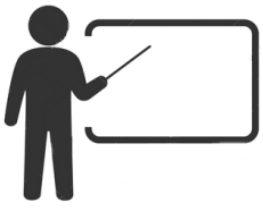
1 Utilisation de dictionnaires

1.1 Le Zoo de Beauval[1]

Nous décrivons la population du zoo de Beauval par un dictionnaire présenté ci-dessous :

Le dictionnaire zoo_Beauval

```
zoo_Beauval = {
    'éléphant' : ('Asie', 5),
    'écureuil': ('Asie', 17),
    'panda': ('Asie', 2),
    'hippopotame': ('Afrique', 7),
    'girafe': ('Afrique', 4)}
```



Script_types_construits_1. ☐ 5

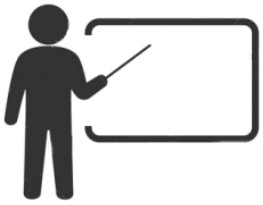
Coder le dictionnaire en Python, puis réaliser un script pour lister tous les éléments du dictionnaire, présentez le résultat comme ci-dessous :

```
    Population du zoo de Beauval

    Espèce |     Origine    | Quantité
-------------------------------------------
   écureuil |         Asie   |     17
   éléphant |         Asie   |      5
     girafe |      Afrique   |      4
 hippopotame |      Afrique   |      7
      panda |         Asie   |      2
```

[1] Sur une idée proposée dans Prépabac 1ère NSI Hatier

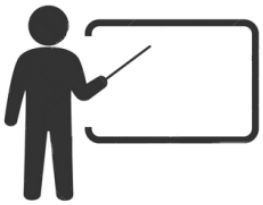
Pour comparer deux zoos différents nous allons prendre un deuxième zoo, les nouvelles données sont listées ci-dessous :

zoo_LaFleche		
ours	Europe	4
tigre	Asie	7
girafe	Afrique	11
hippopotame	Afrique	3
zèbre	Afrique	10



Script_types_construits_2. ☐ 5

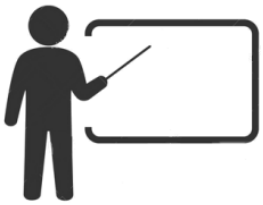
Coder le dictionnaire en Python selon le même modèle que celui du zoo de Beauval. Puis réaliser un script pour lister tous les animaux communs aux deux zoos.



Script_types_construits_3. ☐ 5

Compléter la fonction qui analyse un zoo et qui retourne le nom de l'espèce la plus représentée dans ce zoo.

```
def plus_grand_nombre(zoo):
    """
    En entrée : un dictionnaire  représentant
    les animaux du zoo.
    nom_max    : nom de l'animal le plus représenté
    nombre_max : nombre d'individus du plus représenté

    En sortie : nom_max
    """
    nom_max = None
    nombre_max = None
    for (nom, (_,nombre)) in zoo.items():

    ' à compléter'


    return nom_max

print(plus_grand_nombre(zoo_Beauval))
```



Script_types_construits_4. 5

On améliore le script précédent en affichant un texte comme ci-dessous :

```
L'animal le plus nombreux à Beauval est  écureuil
```

On remarque immédiatement une prise en compte médiocre de l'orthographe.

La règle de grammaire nous rappelle que nous devons écrire l' devant une voyelle ou un h muet. Ceci pour obtenir un affichage adapté tel que ci-dessous :

```
L'animal le plus nombreux à Beauval est l' écureuil

L'animal le plus nombreux à Beauval est le panda
```

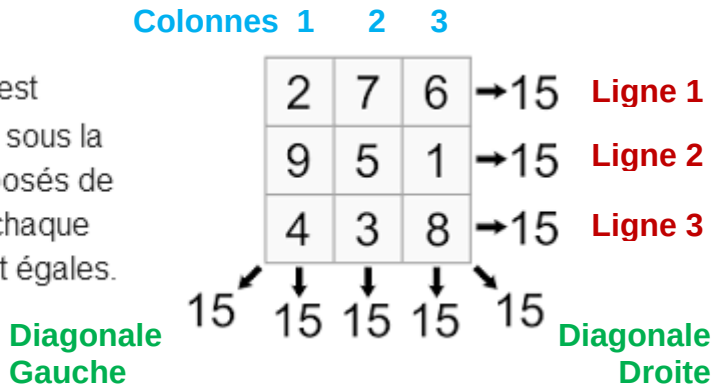
En utilisant un ensemble comme ci-dessous pour lister les cas particuliers, réaliser un script d'affichage qui répond à cette règle d'orthographe.

```
('a','e,','i','o','u','y','h','è','é','ê')
```

2 Utilisation de listes

Nous allons dans ce travail proposer une vérification automatique de carré magique. Nous rappelons la définition, attention nous nommerons les lignes et colonnes de manière usuelle et non pas 'informatique' on commencera avec la valeur 1.

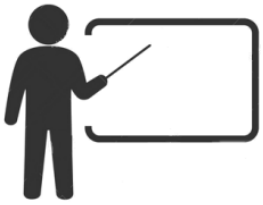
En mathématiques, un **carré magique** d'ordre n est composé de n^2 entiers strictement positifs, écrits sous la forme d'un tableau **carré**. Ces nombres sont disposés de sorte que leurs sommes sur chaque rangée, sur chaque colonne et sur chaque diagonale principale soient égales.



Nous modéliserons un carré magique en liste de listes et travaillerons avec deux exemples :

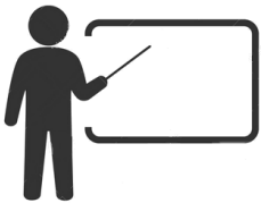
```
carre_magique = [
    [2, 7, 6],
    [9, 5, 1],
    [4, 3, 8]
    ]
```

```
carre_pas_magique = [
    [1, 2, 9],
    [5, 7, 6],
    [4, 8, 3]
    ]
```



Script_types_construits_5. ☐ 3

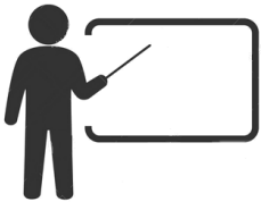
Écrire un script qui affiche la valeur de la somme de toutes les lignes.



Script_types_construits_6. ☐ 3

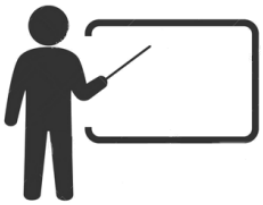
Modifier le script précédent pour définir une fonction qui calcule à partir d'un carré à analyser et d'un numéro de ligne la somme des éléments de la ligne. Voir ci-dessous un prototype de cette fonction.

```
def somme_ligne(carre,numero_ligne):
    '''

    '''
    somme = 0

    return somme
```



Script_types_construits_7. ☐ 3

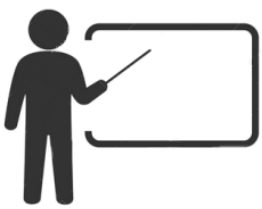
Écrire un script qui affiche la valeur de la somme de toutes les colonnes.



Script_types_construits_8. ☐ 3

Modifier le script précédent pour définir une fonction qui calcule à partir d'un carré à analyser et d'un numéro de colonne la somme des éléments de la colonne. Voir ci-dessous un prototype de cette fonction.

```
def somme_colonne(carre,numero_colonne):
    '''

    '''
    somme = 0

    return somme
```



Script_types_construits_9. ☐ 3

Regrouper toutes vos fonctions puis ajouter le test des deux diagonales et réalisez le test des deux carrés présentés pour voir si ils sont magiques ou non.