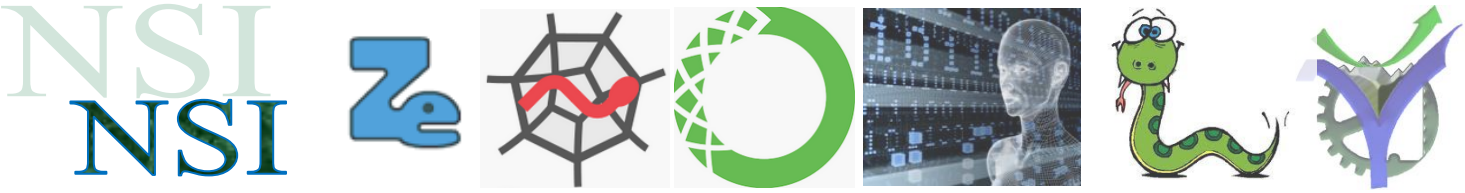


# Les types construits exercices

| **Nom :** | **Note : / 20** |
|---|---|
| | **Classe :** |

## 1 Pour s'échauffer

Avec l'aide du document NSI_TYPES_CONSTRUITS.pdf répondre aux questions ci-dessous. Le langage Python étant très cohérant. Vous verrez que beaucoup de méthodes pour accéder, itérer, modifier un élément dans un type construit sont universelles pour les autres types.

### 1.1 Quatre types construits

Q1. Lister les quatre types construits décrits dans le document.

Q2. Comment différencier les différents types construits dans le code ? Sur l'exemple ci-dessous trois paires de caractères sont utilisées indiquez ce que représente :

```
type_a = [   ]      type_b = {   }      type_c = (   )
```

type_a :

type_b :

type_c :

**1er Type construit :**

```
type_1 = [1, 2, 3,'A', 'B']
```

Q3. Comment enlever 'A' avec une seule instruction ?

Q4. Comment insérer 'D' entre 'A' et 'B' Pour obtenir `[1, 2, 3, 'A', 'D', 'B']`

Q5. Comment modifier 'D' par 'E' ?

**2ème Type construit :**

```
type_2 = ('A', 2, 3.5, 'B')
```

Q6. Comment tester si 'B' est dans notre type_2 ?

Q7. Comment connaître le nombre d'éléments de type_2 ?

**2ème Type construit :** `type_2 = ('A', 2, 3.5, 'B')`

Q8. Comment tester si 'B' est dans notre type_2 ?

Q9. Comment connaître le nombre d'éléments de type_2 ?

Q10. Comment accéder à l'élément 3.5 ?

**3ème Type construit :** `type_3 = {"A":8, 2:"B"}`

Q11. Ensemble ou dictionnaire ?

Q12. Comment lister toutes les clés ?

Q13. Comment modifier la valeur associée à la clé 'A' pour le remplacer par 10 ?

Q14. Comment supprimer l'item (2 :'B') ?

**4ème Type construit :** `type_4 = {'a', 'e', 'i', 'o', 'u', 'y'}`

Q15. Comment supprimer l'élément 'a' ? Il y a deux possibilités.

## 1.2 Pour approfondir

Utiliser une console Python pour répondre aux exercices ci-dessous :

### a) Exemple 1

La variable sequence contient une liste de lettres, éventuellement répétées, choisies parmi 'A', 'B', 'C', 'D'.
On veut créer un dictionnaire effectifs associant à chaque lettre le nombre de fois qu'elle apparaît dans la liste sequence.
Par exemple si sequence contient ['A', 'B', 'B', 'D', 'B', 'A'],
effectifs doit contenir {'A':2, 'B':3, 'C':0, 'D':1}.
Parmi les scripts suivants, lequel réalise cet objectif ?

A

```
effectifs = {'A':0, 'B':0, 'C':0, 'D':0}
for lettre in sequence:
   effectifs[lettre] = effectifs[lettre] + 1
```

B

```
effectifs = {}
for lettre in sequence:
   effectifs[lettre] = effectifs[lettre] + 1
```

C

```
effectifs = {'A':0, 'B':0, 'C':0, 'D':0}
for lettre in effectifs.keys():
   effectifs[lettre] = len([lettre in effectifs])
```

D

```
effectifs = {}
for lettre in effectifs.keys():
   effectifs[lettre] = len([lettre in effectifs])
```

## b) Exemple 2

On dispose d'une table tab constituée d'une liste de trois sous-listes contenant chacune quatre caractères.

```
tab = [  ['A', 'B', 'C', 'D'],
         ['E', 'F', 'G', 'H'],
         ['I', 'J', 'K', 'L'] ]
```

Parmi les propositions suivantes, laquelle permet de convertir cette table en une liste L contenant dans l'ordre, ligne par ligne, les 12 caractères de tab ?

```
# à la fin, on a l'égalité :
L==[ 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L' ]
```

A
```
L = []
for i in range(3):
    for j in range(4):
        L.append(tab[i][j])
```

B
```
L = []
for i in range(4):
    for j in range(3):
        L.append(tab[i][j])
```

C
```
L = []
for i in range(3):
    L.append(tab[i])
```

D
```
L = []
for i in range(4):
    L.append(tab[i])
```

## c) Exemple 3

Laquelle des expressions suivantes a-t-elle pour valeur la liste des carrés des premiers entiers qui ne sont **pas** multiples de 5 ?

A `[x*x for x in range (11) if x//5 != 0]`
B `[x*x if x%5 != 0 for x in range (11)]`
C `[x*x if x//5 != 0 for x in range (11)]`
D `[x*x for x in range (11) if x%5 != 0]`

## d) Exemple 4

L est une liste d'entiers.
On définit la fonction suivante :

```
def f(L):
    m = L[0]
    for x in L:
        if x > m:
            m = x
    return m
```

A le maximum de la liste L passée en argument
B le minimum de la liste L passée en argument
C le premier terme de la liste L passée en argument
D le dernier terme de la liste L passée en argument

Que calcule cette fonction ?

# 2 Utilisation de dictionnaires

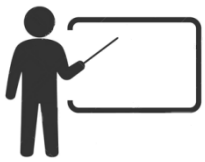
## 2.1 Le Zoo de Beauval[1]

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



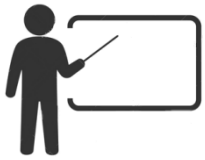
```
    Population du zoo de Beauval

    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

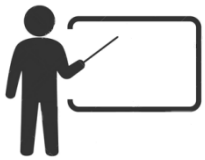
Pour comparer deux zoos différents nous allons prendre un deuxième zoo, les nouvelles données sont listées ci-dessous :

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

Script_types_construits_2. 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.

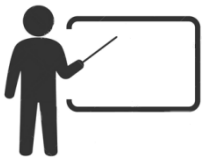


Script_types_construits_3. Compléter la fonction qui analyse un zoo et qui retourne le nom de l'espèce la plus représentée dans ce zoo.



```
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. Améliorez l'affichage des données en respectant les règles d'orthographe comme indiqué ci-dessous.



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
```

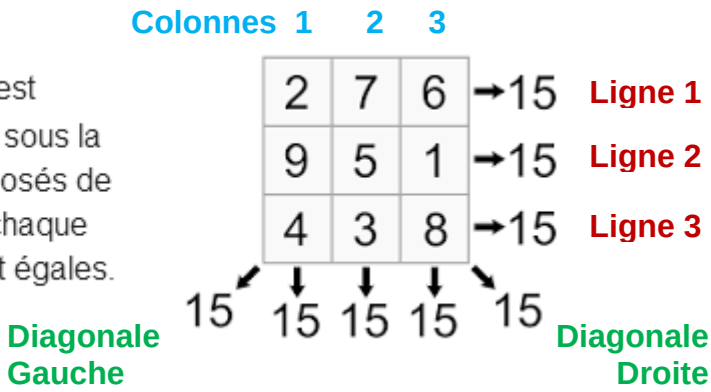
En utilisant un ensemble comme ci-dessous pour lister les cas particuliers, réaliser un script d'affichage qui répond à cette règle d'orthographe.

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

# 3 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.

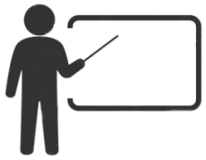


Nous modéliserons un carré magique en liste de listes et travaillerons avec deux exemples :

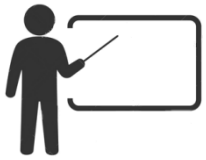
```
carre_magique = [
    [2, 7, 6],
    [9, 5, 1],
    [4, 3, 8]
    ]
```

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

Script_types_construits_5. Écrire un script qui affiche la valeur de la somme de toutes les lignes.

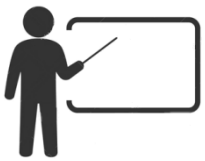


Script_types_construits_6. Modifier le script précédent pour définir une fonction qui calcule à partir d'un carré à analyser et d'un numéro de ligne la somme des éléments de la ligne. Voir ci-dessous un prototype de cette fonction.

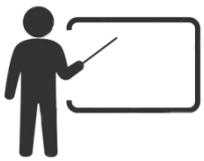


```
def somme_ligne(carre,numero_ligne):
    '''

    '''
    somme = 0

    return somme
```

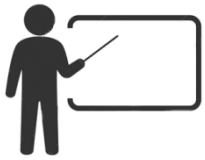
Script_types_construits_7. Écrire un script qui affiche la valeur de la somme de toutes les colonnes.



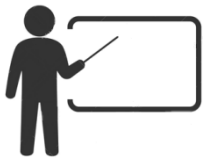
Script_types_construits_8. Modifier le script précédent pour définir une fonction qui calcule à partir d'un carré à analyser et d'un numéro de colonne la somme des éléments de la colonne. Voir ci-dessous un prototype de cette fonction.



Script_types_construits_9. Ajouter le test des deux diagonales.



**Proposer un algorithme pour décrire le test complet d'un carré magique ?**



Script_types_construits_10. Programmez votre algorithme et réalisez le test des deux carrés présentés pour voir s'ils sont magiques ou non.