## TP08 TuplesTableauxExercices

TD n°8: Les types
construits: tuples et
tableaux

Thème 4:
Langages et
Programmation

EXERCICES

### Exercice 5.1

On considère la variable repertoire suivante.

```
In []: repertoire = [('Michel', '0210101010'), ('Marjorie', '02
```

- 1. Quel est le type de la variable repertoire ?
- 2. Quel est le type de la variable repertoire[0] ?
- 3. Qu'affiche l'instruction repertoire[1][1]?
- 4. Quelle instruction permet d'accéder au numéro de téléphone de Michel ?
- 5. Quel est le type de la variable repertoire[2][0] ?

#### Réponse:

- 1.
- 2.
- 3.
- 4.
- 5.

#### ▼ Solution

- La variable repertoire est de type list
- La variable repertoire[0] est de type tuple
- L'instruction repertoire[1][1] affiche 0211111111
- Pour accéder au numéro de téléphone de Michel, il faut utiliser l'instruction repertoire[0][1]
- La variable repertoire[2][0] est de type str

### Exercice 5.2

Répondez aux questions suivantes **puis** exécutez la cellule pour vérifier.

1. Quel est le tableau construit par les instructions suivantes ?

```
In [ ]: t1 = [0]*10
```

```
for i in range(10):
t1[i] = i
```

1. Quel est le tableau construit par les instructions suivantes ?

```
In []: t2 = [i+1 for i in range(8)]
```

1. Quel est le tableau construit par les instructions suivantes ?

```
In []: t3 = [i*i for i in range(5, 10)]
```

1. Quel est le tableau construit par les instructions suivantes ?

```
In []: t4 = [0]*50

for i in range(len(t4)):

t4[i] = i % 2
```

## Exercice 5.3

Complétez la fonction suivante pour qu'elle échange les valeurs d'index i et j du tableau tab.

#### Exemple

```
T = [3, 8, 12, 9, 56, 14, 22, 7, 13, 41] echange(T, 2, 4)
```

```
T
[3, 8, 56, 9, 12, 14, 22, 7, 13, 41]
```

```
In []: def echange(tab,i,j):
    pass
    return tab

# Tests
T = [3, 8, 12, 9, 56, 14, 22, 7, 13, 41]
    echange(T, 2, 4)
T
```

```
▼
Solution
```

```
```python def echange(tab,i,j): temp=tab[j] tab[j]=tab[i] tab[i]=temp return tab ```
```

## Exercice 5.4

On considère le tableau t suivant.

```
In []: t = [0, 2, 4, 6, 8, 10]
```

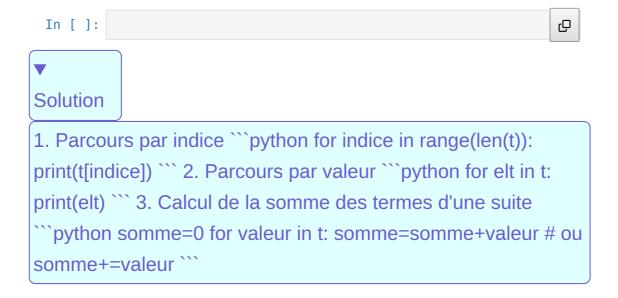
1. Ecrivez un programme qui parcourt *par indice* le tableau et affiche tous ses éléments.

```
In [ ]:
```

1. Ecrivez un programme qui parcourt *par valeur* le tableau et affiche ses éléments.

In [ ]:

1. Ecrire un programme qui calcule la somme des élèments du tableau.

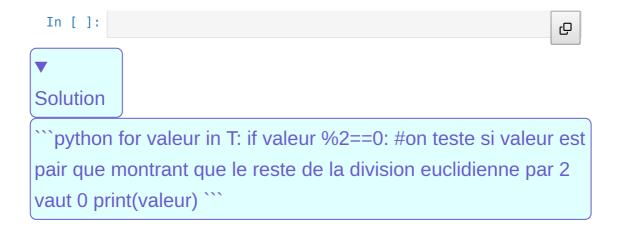


## Exercice 5.5:

On considère le tableau défini par compréhension suivant.

```
In []: T = [i for i in range(15)]
```

- 1. Que contient le tableau T ?
- 1. Ecrivez un programme qui parcourt le tableau T et qui affiche uniquement les multiples de 2 qu'il contient.



## Exercice 5.6:

On rappelle que l'on peut construire un tableau en utilisant une boucle for (en ayant au préalable créé un tableau de la bonne taille avec une valeur arbitraire) ou en le définissant par compréhension.

Utilisez les deux méthodes pour construire les tableaux demandés.

- 1. Un tableau qui contient tous les entiers compris entre 0 et 1000.
- 2. Un tableau qui contient tous les entiers compris entre 5 et 15.
- 3. Un tableau qui contient tous les nombres pairs compris entre 0 et 50.
- Un tableau qui contient les carrés de tous les entiers compris entre 1 et 100 (c'est-à-dire 1, 4, 9, 16, etc.)

```
In [ ]: # Q1 avec méthode 1
  Q
         # Q1 avec méthode 2
In [ ]:
  Q
         # Q2 avec méthode 1
In [ ]:
  0
In [ ]:
         # Q2 avec méthode 2
  Q
         # Q3 avec méthode 1
In [ ]:
  O
         # Q3 avec méthode 2
In [ ]:
  Q
In [ ]:
         # Q4 avec méthode 1
  Q
In [ ]:
         # Q4 avec méthode 2
  Q
```

## Solution

```python #Q1 Méthode 1 tab1=[] #initialisation for k in range(0,1001): tab1.append(k) ``` ```python #Q1 Méthode 2 par compréhension tab1=[k for k in range(0,1001)] ``` ```python #Q2 Méthode 1 tab2=[] #initialisation for k in range(5,16): tab2.append(k) ``` ```python #Q2 Méthode 2 par compréhension tab2=[k for k in range(5,16)] ``` ```python #Q3 Méthode 1 tab3=[] #initialisation for k in range(0,51,2): tab3.append(k) ``` ```python #Q3 Méthode 2 par compréhension tab3=[k for k in range(0,51,2)] ``` ou ```python #Q3 Méthode 2 par compréhension tab3=[k for k in range(51) if k%2==0] ``` ```python #Q4 Méthode 1 tab4=[] #initialisation for k in range(1,101): tab4.append(k\*\*2) ``` ```python #Q4 Méthode 2 par compréhension tab4=[k\*\*2 for k in range(1,101)] ```

## Exercice 5.7 : Moyenne annuelle

Compléter la fonction moyenne qui prend en argument un tableau de notes et qui renvoie la moyenne de ce tableau.

```
notes=[12,10,15,18,8]
moyenne(notes)
>>>12.6

In []: def moyenne(notes):
...
...
Solution

**`python def moyenne(notes): somme=0 nb_notes=len(notes)
for note in notes: somme+=note moy=somme/nb_notes return
moy ```
```

## Exercice 5.8 : Moyenne coefficientée

Voici les premières notes de l'année obtenues par un élève en NSI :



Note	coefficient
15	1
13	2
8	1
18	1

- 1. Ces notes et coefficients sont d'abord stockées dans deux tableaux comme ci-dessous.
- a. Quelle instruction permet d'accéder la troisième note?
- b. Quelle instruction permet d'accéder au coefficient de la deuxième note ?

```
In []: notes = [15, 13, 8, 18] coeff = [1, 2, 1, 1]
```

#### Réponse:

1. Ecrire une fonction moyenneCoeff(notes, coeffs) qui prend en paramètres deux tableaux et qui renvoie la moyenne coefficientée correspondante.

```
In []: def moyenneCoeff(notes,coeffs):
sommenotes=0
```

```
sommecoeffs=0
for i in range(...):
...
```

## ▼ Solution

```python def moyenne(notes,coeffs): sommenotes=0
sommecoeffs=0 nb2notes=len(notes) for indice in
range(nb2notes): sommenotes+=notes[indice]
sommecoeffs+=coeffs[indice] moy=sommenotes/sommecoeffs
return moy ```

1. Ces notes et coefficient sont maintenant stockées dans un tableau de tableaux.

Quelle instruction permet d'accéder à la troisième paire note-coefficient ?

Quelle instruction permet d'accéder la troisième note ? Quelle instruction permet d'accéder au coefficient de la deuxième note ?

```
In []: notes_coeff = [[15, 1], [13, 2], [8, 1], [18, 1]]
```

#### Réponse:

1. Ecrire une fonction moyenneCoeff(tab) qui prend en paramètre un tableau et qui renvoie la moyenne coefficientée correspondante.

```
In [ ]:
```



""python def moyenneCoeff(tab): sommenotes=0
sommecoeffs=0 nb2notes=len(tab) for indice in
range(nb2notes): sommenotes+=tab[indice][0]\*tab[indice][1]
#les notes ont toujours pour position 0 dans le tableau extrait et
les coeffs ont pour position1 sommecoeffs+=tab[indice][1]
moy=sommenotes/sommecoeffs return moy ""

## Exercice 5.9:

Répondez aux questions suivantes **puis** exécutez la cellule pour vérifier.

1. Quel est le tableau construit par compréhension de la façon suivante ?

```
In []: tab = [[j**2 for j in range(5)] for i in range(3)] tab
```

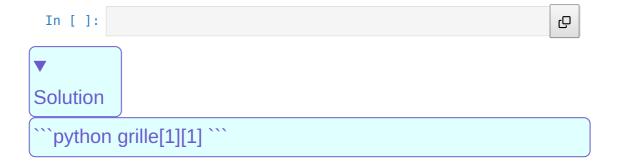
1. Quel est le tableau construit par compréhension de la façon suivante ?

```
In []: tab = [[2*i + j for j in range(2)] for i in range(4)] tab
```

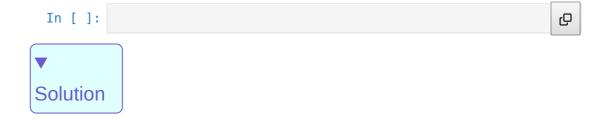
## Exercice 5.10:

On a mémorise dans le tableau grille l'état d'un jeu de morpion. Voici le contenu du jeu après deux tours.

C'est au joueur « O » de jouer et il compte mettre un
 '0' au centre de la grille. Quelle instruction faut-il écrire ? Vérifiez.



1. Vous avez du constater qu'en affichant le tableau, celui ne s'affiche pas comme on souhaiterait (comme la grille du jeu). Pour pallier à ce souci, il suffit de parcourir le tableau ligne par ligne et afficher cette ligne. Ecrivez le programme correspondant.



```python for ligne in grille: print(ligne) ```

#### 1. Approfondissement

Pour ceux qui le désire, écrire un programme permettant de joueur au morpion.

In []:

## Parcours tableaux à plusieurs dimensions

## Exercice 5.11:

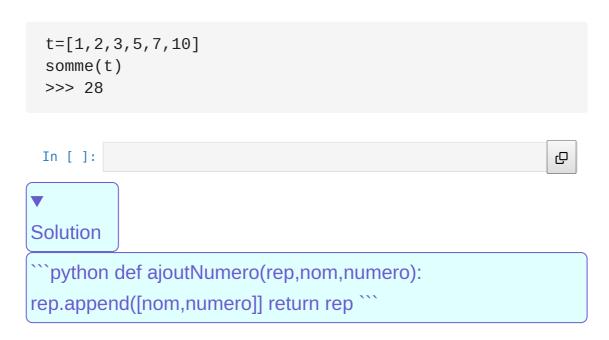
Ecrire une fonction ajoutNumero(rep, nom, numero) qui prend pour paramètre un répertoire, un nomn et un numero de téléphone et qui permet d'enregistrer un nouveau nom ainsi que son numéro dans le répertoire

```
In []: repertoire = [['Michel', '0210101010'], ['Marjorie', '02
```

#### Exercice 5.11:

Ecrivez une fonction somme(t) qui renvoie la somme des éléments du tableau d'entiers t.

#### Exemple:



## Exercice 5.12 : cadavre exquis

Le cadavre exquis est un « jeu qui consiste à faire composer une phrase, ou un dessin, par plusieurs personnes sans qu'aucune d'elles ne puisse tenir compte de la collaboration ou des collaborations précédentes.»

Remarque : la méthode choice du module random permet de tirer au sort un élément dans un itérable non vide (et donc dans un tableau non vide). Vous pouvez

exécuter plusieurs fois la cellule de code ci-dessous pour tester :

```
In []: from random import choice

tab = ['chat', 'chien', 'cheval', 'vache', 'cochon']
animal = choice(tab)
animal
```

Compléter la fonction cadavre\_exquis ci-dessous qui prend en argument :

- un tableau de noms non vide,
- un tableau de verbes non vide,
- un tableau de compléments non vide,

et qui renvoie une phrase aléatoire construite selon le principe du cadavre exquis : en piochant un nom dans le tableau de noms, puis en piochant un verbe dans le tableau de verbes et enfin en piochant un complément dans le tableau de compléments.

```
In []: from random import choice

def cadavre_exquis(noms, verbes, complements):
    nom = ...
    verbe = ...
    complement = ...
    phrase = ...
    return ...
```



```
""python from random import choice def cadavre_exquis(noms, verbes, complements): nom = noms verbe = verbes complement = complements phrase = choice(nom)+choice(verbe)+choice(complement) return phrase
```

#### Question 2:

Tester votre fonction en complétant l'exemple cidessous.

```
In []: n = ['Le chasseur ', 'La présidente ', 'Le lapin ', 'Pil control ver le la la la lapin ', 'Pil control ver le lapin ', 'Vener ce la la la lapin ', 'Vener ce la lapin ', 'Ila la lapin ', 'Pil control ver la lapin
```

## Exercice 5.13 : Travail sur les listes en compréhension

Créez une liste Liste1 de 15 entiers aléatoires compris entre 0 et 20 à l'aide d'une boucle.

Aide : Nous allons utiliser le module random qui génère des nombres pseudo-alétoires.

Commencez par importer la fonction randint du module avec l'instruction :

from random import randint à placer en début de cellule.

L'instruction randint(a,b) retournera un entier alétoire compris entre a et b (bornes comprises)

#### Vers la documentation du module

```
In []: #un exemple :
    from random import randint
    Liste1 = []

    for k in range(15):
        alea=randint(0,20)
        Liste1.append(alea)
    print(Liste1)
```

Créez une liste Liste2 de 30 entiers aléatoires compris entre 0 et 20 à l'aide d'une compréhension de liste.



Créez une fonction moyenne qui prend en paramètre une liste de nombres et qui renvoie la moyenne de cette liste.

Testez votre fonction à l'aide des assert de la cellule suivante.

```
In []: def moyenne(liste):
    pass
    return moyenne

Solution

""python def moyenne(liste): somme=0 for elt in liste:
somme+=elt moy=somme/len(liste) return moy ""

In []: # Méthode de TEST
    assert moyenne(Liste1) == sum(Liste1)/len(Liste1)
    assert moyenne(Liste2) == sum(Liste2)/len(Liste2)
```

Un méchant professeur veut baisser le notes de ses élèves de 10 %

# Créez une liste Liste3 obtenue en diminuant de 10 % les nombres de la liste Liste1 avec une compréhension de liste

Testez votre fonction à l'aide des assert de la cellule suivante.



Le même professeur cherche à écrire des appréciation automatiques en fonction de la note.

Il souhaite voir écrit "Pas terrible" si la note est inférieure ou égale à 10 et "Pas mal" sinon.

## Créez une liste Appreciations à partir de la liste Liste1 à l'aide d'une comprehension de liste

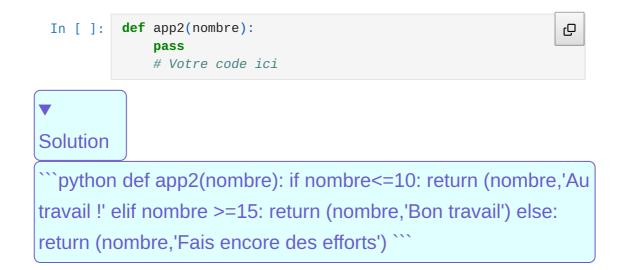
Aide: [<expression> if <condition> else
 <autre\_expression> for <element> in <iterable>]



Le professeur souhaite maintenant créer une fonction app2 qui prend en paramètre un nombre et qui renvoie un tuple constitué de ce nombre et d'une appréciation. L'appréciation sera :

- "Bon travail" si la note est supérieure ou égale à 15;
- "Fais encore des efforts" si la note est entre 10 et 15 (non compris);
- "Au travail!" si la note est inférieure ou égale à 10.

#### Ecrivez la fonction ci-dessous



Créez une liste Appreciations2 à partir de la liste Liste2 contenant les notes et les appréciation à l'aide d'une compréhension de liste

```
In []: Appreciations2 = [app2(nb) for nb in Liste2] print(Appreciations2)
```

## Exercice 5.14 : Travail sur les listes de listes.

Les carrés magiques : prolongement de l'exercice vu dans la leçon.

Première étape - Créer une fonction Gen\_carre qui prend en paramètre un entier naturel non nul \$n\$, et qui renvoie un tableau carré (liste de liste)

### contenant les entiers de 1 à \$n^2\$ entiers "mélangés"

```
Par exemple: Gen_carre(3) porra retourner [[2,7,6], [9,5,1], [4,3,8]]
```

Vous allez utiliser le module random et en particulier la fonction shuffle qui prend en paramètre une liste et qui la mélange "sur place". Par exemple liste = [0,1,2,3] suivi de shuffle(liste) va mélanger les éléments de liste et, par exemple, on obtiendra liste = [3,0,1,2].

```
In []: from random import shuffle, randint

def Gen_carre(n):

In []: carre3 = Gen_carre(3)
print(carre3)
```

Deuxième étape : Ecrire une fonction somme\_ligne qui prend en paramètres un tableau carré d'entiers et un entier \$n\$ et qui retourne la somme de la ligne d'index \$n\$

Troisième étape : Ecrire une fonction verif\_lignes qui prend en paramètres un tableau carré d'entiers et qui retourne un booléen indiquant si toutes les lignes ont la même somme.

#### Quatrième étape : Faire de même avec les colonnes.

Vous avez la liberté pour construire les fonctions que vous souhaitez, ajoutez des docstrings et des tests à vos fonctions.

```
In []: def somme_colonne(carre : list,index_colonne : int):
    assert somme_colonne(carre3,0) == 15
    assert somme_colonne(carre3,1) == 15
    assert somme_colonne(carre3,2) == 15
    assert somme_colonne(carre4,0) == 35
    assert somme_colonne(carre4,1) == 34

    def verif_colonnes(carre : list):
```

```
assert verif_colonnes(carre3) == True
assert verif_colonnes(carre4) == False
```

## Cinquième étape : Vérifier de même les deux diagonales

```
In []: def verif_diag(carre : list):

assert verif_diag(carre3) == True
assert verif_diag(carre4) == False
```

Dernière étape : Ecrire une fonction qui prend en paramètre un tableau carré d'entiers et qui retourne un booléen indiquant si le carré est magique

```
In []: def verif_carre(carre):

assert verif_carre(carre3) == True
assert verif_carre(carre4) == False
```

Bonus : à l'aide de votre fonction Gen\_carre, trouvez des carrés magiques d'ordre 3, puis 4.

Conseil: soyez patient!

```
In []: # Voici une exemple de solution (Aucune inquiétude, je def carre_magique_ordre3():

carre_magique_ordre3()
```