FEUILLE N°1 *Listes et tuples*

**Exercice 1. Questions de cours** Par chacune des structures de données suivantes, préciser son type, sa taille et le type de ses éléments. Préciser également la valeur des variables a, b et c. Pour vous aider, vous pouvez faire un schéma représentant la mémoire.

### 1.1.

```
sdd = ["Serpent", "Minitortue", "Souris", "Dragon"]
a = sdd[1]
b = sdd[1][5]
c = sdd[3]
```

### 1.2.

```
sdd = (9, 52, 25)
a = sdd[1]
b = sdd[2]%3
c = sdd[1]%3 == 1
```

### 1.3.

```
sdd = [("Abo", 9), ("Carapuce", 52), ("Rattata", 25)]
a = sdd[1]
b = sdd[1][0]
c = sdd[1][0][5]
```

### 1.4.

```
sdd = (["Abo", "Carapuce", "Rattata"], [9, 52, 25])
a = sdd[0]
b = sdd[1][0]
c = sdd[0][1][5]
```

**1.5.** Chacune des lignes suivantes comporte une erreur de syntaxe : trouver laquelle

```
1 a = "Abo
2 b = ["Abo", "Carapuce", "Rattata", "Abo")
3 c = ("Abo"  "Carapuce", "Rattata", "Abo")
```

**1.6.** Par chacune des variables suivantes, préciser si elle est mutable ou non mutable.

```
a = "Abo"
b = ["Abo", "Carapuce", "Rattata", "Abo"]
c = ("Abo", 9)
d = [("Carapuce", 52), ("Rattata", 25)]
```

## Exercice 2. Représentation de la mémoire

**2.1.** Donner une représentation de la mémoire à la ligne 4. Qu'affichera ce script ?

```
1 pokedex1 = ["Abo", "Onix", "Stari", "Paras"]
2 pokedex2 = pokedex1
3 pokedex1[3] = "Draco"
4 print(pokedex2)
```

**2.2.** Donner une représentation de la mémoire à la ligne 6. Qu'affichera ce script ?

```
1 pokedex1 = ["Abo", "Onix"]
2 pokedex2 = pokedex1
3 pokedex1.append("Draco")
4 pokedex2.pop(1)
5 pokedex1 = ["Pika"]
6 print(pokedex1)
7 print(pokedex2)
```

**2.3.** Donner une représentation de la mémoire à la ligne 9. Qu'affichera ce script ?

```
1  serpents = ["Abo", "Machoc", "Arbock"]
2  plantes = ["Chetiflor", "Boustiflor"]
3  pokedex = [serpents, plantes]
4  variable1 = pokedex[0]
5  variable2 = pokedex[0][1]
6  plantes.append("Empiflor")
7  serpents.pop(1)
8  pokemon = pokedex[0][1]
9  print(variable1)
10 print(variable2)
11 print(pokedex)
```

### Exercice 3. Lecture de code

**3.1.** On donne le code suivant. Compléter le tableau avec les valeurs des expressions à chaque passage par la ligne indiquée

```
1  def moyennePonderee(listeNotes, listeCoeff):
2          """
3          paramètres: listeNotes et listeCoeff sont deux listes de nombres
4          résultat: la moyenne pondérée des notes (float)
5          """
6          sommePonderee = 0
7          sommeCoeff = 0
8          # LA
9          for i in range(len(listeNotes)):
10                 sommePonderee+= listeNotes[i] * listeCoeff[i]
11                 sommeCoeff+= listeCoeff[i]
12                 # ICI
13         return sommePonderee/sommeCoeff
14 notes = [12, 5, 9, 23]
15 coefficients = [3, 2, 5, 1]
16 print(moyennePonderee(notes, coefficients))
```

| | i | listeNotes[i] | listeCoeff[i] | sommePonderee | sommeCoeff |
|---|---|---|---|---|---|
| LA<br><br>ICI | | | | | |

**3.2.** Que se passe-t-il si on ajoute les lignes suivantes ?

```
notes2 = [12, 5, 9, 23]
coefficients2 = [3, 2, 5, 1, 7]
moyenne = moyennePonderee(notes2, coefficients2)
print(moyenne)
```

| | i | listeNotes[i] | listeCoeff[i] | sommePonderee | sommeCoeff |
|---|---|---|---|---|---|
| LA<br><br>ICI | | | | | |

**3.3.** Que se passe-t-il si on ajoute les lignes suivantes ?

```
notes3 = [12, 5, 9, 23]
coefficients3 = [3, 2, 5]
moyenne = moyennePonderee(notes3, coefficients3)
print(moyenne)
```

| | i | listeNotes[i] | listeCoeff[i] | sommePonderee | sommeCoeff |
|---|---|---|---|---|---|
| LA<br><br>ICI | | | | | |

**3.4.** Proposez une solution pour résoudre le problème

**3.5.** **Travail à la maison** Dans son incommensurable bonté, votre enseignant a décidé de calculer la moyenne en supprimant la note la plus basse. Proposer une fonction qui calcule cette moyenne.

### Exercice 4. Petit problème : mot de passe

Anakin, jeune padawan de l'informatique, doit écrire une fonction `dialogue_mot_de_passe()` qui demande à l'utilisateur d'entrer un mot de passe (sur l'entrée standard) et qui lui indique (sur la sortie standard) si son mot de passe est correct. D'après les lois de la République, un mot de passe est correct s'il vérifie les conditions suivantes :

a) contenir au moins huit caractères ;

b) comporter au moins un chiffre ;

c) ne pas comporter d'espace


Anakin propose le code suivant :

```python
#!/bin/python3
# Codé par Anakin, jeune padawan de l'informatique
def dialogue_mot_de_passe():
    mot_de_passe_correct = False
    while not mot_de_passe_correct :
        mot_de_passe = input("Entrez votre mot de passe : ")
        # je vérifie la longueur
        if len(mot_de_passe) < 8:
            longueur_ok = False;
        else:
            longueur_ok = True
        # je vérifie s'il y a un chiffre
        chiffre_ok = False
        for lettre in mot_de_passe:
            if lettre.isdigit():
                chiffre_ok = True
        # je vérifie qu'il n'y a pas d'espace
        sans_espace = True
        for lettre in mot_de_passe:
            if lettre == " ":
                sans_espace = False
        # Je gère l'affichage
        if not longueur_ok:
            print("Votre mot de passe doit comporter au moins 8 caractères")
        elif not chiffre_ok:
            print("Votre mot de passe doit comporter au moins un chiffre")
        elif not sans_espace:
            print("Votre mot de passe ne doit pas comporter d'espace")
        else:
            mot_de_passe_correct = True
    print("Votre mot de passe est correct")
    return mot_de_passe
dialogue_mot_de_passe()
```

**4.1.** Anakin est très fier de ce qu'il a codé. Pourtant, son maître n'est pas satisfait de son travail car Anakin a oublié certaines bonnes pratiques. Les quelles ?

- □ Le nom des variables et des fonctions doit être explicite
- □ Chaque fonction réalise une tâche clairement identifiée dans sa documentation
- □ Chaque fonction doit réaliser une et une seule tâche et son code fait 20 lignes maximum, sauf dans des cas exceptionnels.
- □ Le code doit être commenté si nécessaire.

**4.2.** Aidez Anakin à améliorer son code pour les quatre bonnes pratiques précédentes soient respectées.
Vous n'oublierez pas d'ajouter un ou plusieurs tests (sous la forme de `assert`) à toutes les micro-fonctions que vous écrirez.

**4.3.** Faites évoluer votre programme de façon à respecter le décret du chancelier Palpatine qui vient d'arriver. Désormais, un mot de passe correct s'il vérifie toutes les conditions suivantes :

a) contenir au moins huit caractères ;
b) comporter au moins un chiffre ;
c) ne pas comporter d'espace ;
d) comporter au moins une majuscule ;
e) ne pas comporter deux majuscules consécutives ;
f) ne pas contenir de caractères de ponctuation.

Exemple d'utilisation :

```
>>> dialogueMotDePasse()
Entrez votre mot de passe : chou
Votre mot de passe doit comporter au moins 8 caractères
Entrez votre mot de passe : chou bouilli
Votre mot de passe doit comporter au moins un chiffre
Entrez votre mot de passe : 1 chou bouilli
Votre mot de passe ne doit pas comporter d espace
Entrez votre mot de passe : 1choubouilli
Votre mot de passe doit comporter au moins une majuscule
Entrez votre mot de passe : 1choubouilliAARRGGHH
Votre mot de passe ne doit pas comporter deux majuscules consécutives
Entrez votre mot de passe : 1ChouBouilliAargghhJeVaisCraquer!!!
Votre mot de passe ne doit pas comporter de ponctuation
Entrez votre mot de passe : 1ChouBouilliAargghhJeVaisTeRetrouverEtTeFaireLaPeau
Votre mot de passe est correct. Ce n était pas si difficile !
```

### 4.4. En bonus

Le chancelier Palpatine vous demande de faire en sorte que, lorsque le mot de passe est valide, il soit stocké dans un fichier `mdpUltraSecret.txt`. Faites évoluer votre programme pour répondre à cette demande.

## Exercice 5. Reconnaître les briques de base pour écrire des algorithmes

Voici cinq fonctions. le paramètre `pokedex` est une liste de tuples `(pokemon, poids)` où

- `pokemon` est de type **str**
- `poids` est de type **int**

```python
def les_legers(pokedex):
    legers = list()
    for (pokemon, poids) in pokedex:
        if poids < 10:
            legers.append(pokemon)
    return legers
```

```python
def le_plus_lourd(pokedex):
    (nom_lourd, poids_lourd)= pokedex[0]
    for (pokemon, poids) in pokedex:
        if poids > poids_lourd:
            poids_lourd = poids
            nom_lourd = pokemon
    return nom_lourd
```

```python
def poids_total(pokedex):
    total = 0
    for ( _ , poids) in pokedex:
        total+= poids
    return poids
```

```python
def bien_rangés(pokedex):
    for i in range(1,len(pokedex)):
        ( _ , poids1) = pokedex[i-1]
        ( _ , poids2) = pokedex[i]
        if poids1 > poids2:
            return False
    return True
```

```python
def contient_pika(pokedex):
    for (pokemon, _ ) in pokedex:
        if pokemon == "Pikachu":
            return True
    return False
```

**5.1.** Quel sera le type et la valeur des variables `a, b, c, d et e` ?

```python
mon_pokedex = [("Mew", 4), ("Mélo", 3), ("Draco", 16), ("Xatu", 15)]
a = les_legers(mon_pokedex)
b = poids_total(mon_pokedex)
c = contient_pika(mon_pokedex)
d = le_plus_lourd(mon_pokedex)
e = bien_rangés(mon_pokedex)
```

**5.2.** Expliquer, en une phrase, ce que fait chaque fonction. Proposez une documentation.

**5.3.** Écrire des tests pour chacune de ces fonctions.

**5.4.** Pour le moment, vous connaissez cinq familles de "micro-fonctions" :

- les fonctions de cumuls
- les fonctions de vérifications
  - les vérifications du type "il existe ..."
  - les vérifications du type "pour tous ..."
- Les fonctions min/max
- les filtres

Associer chaque fonction de cet exercice à sa famille.

### Exercice 6. Savoir choisir la bonne structure de données

Les membres de la famille Adams vous demande de les aider à gérer leurs listes de course. Il veulent que vous écriviez deux fonctions :

- une fonction qui calcule le montant total d'une liste de courses
- une fonction qui renvoie le nom de l'article le moins cher d'une liste de courses

Par exemple, sur la liste de Morticia , on trouve de la bave de crapeau à 24€, des oeufs de dragon pour 157€, du sang de lézard pour 17€, du ketchup à 2€ et du sel qui vaut 1€.

Pour cette liste de coures :

- le montant total s'élève à 201€
- l'article le moins cher est le sel

**6.1.** sur la liste de Gomez , on trouve des chaussures de tango à 247€, un couteau à 15€, des explosifs à 167€ et du sumac veneneux pour 27€. Pour cette liste de courses :

- Quel est le montant total ?
- Quel est l'article le moins cher ?

Anakin, à qui vous avez rendu service, veut vous aider et vous propose trois modélisations pour les données :

**modelisation n° 1**

```
courses_morticia = ["bave de crapeau", "oeufs de dragon",
                    "sang de lézard", "ketchup", "sel"]
facture_morticia = [24, 157, 17, 2, 1]
courses_gomez = ["chaussures de tango", "couteau",
                 "explosif", "sumac veneneux"]
facture_gomez = [247, 15, 167, 27]
```

**modelisation n° 2**

```
courses_morticia = ["bave de crapeau", 24, "oeufs de dragon", 157,
                    "sang de lézard", 17, "ketchup", 2, "sel", 1]
courses_gomez = ["chaussures de tango", 247, "couteau", 15,
                 "explosif", 167, "sumac veneneux", 27]
```

**modelisation n° 3**

```
courses_morticia = [("bave de crapeau", 24), ("oeufs de dragon", 157),
                    ("sang de lézard", 17), ("ketchup", 2), ("sel", 1)]
courses_gomez = [("chaussures de tango", 247), ("couteau", 15),
                 ("explosif", 167), ("sumac veneneux", 27)]
```

**6.2.** Choisissez la modélisation qui vous semble la plus adaptée, et écrivez le code des deux fonctions demandées. Bien entendu, vous n'oublierez pas la documentation et vous penserez à écrire au moins deux tests pour chaque fonction.

**6.3.** Choisissez une autre modélisation et écrivez le code des deux fonctions demandées. Bien entendu, vous n'oublierez pas la documentation et vous penserez à écrire au moins deux tests pour chaque fonction.