I) Introduction

On a vu des objets en python de type int, float, str et bool et on a souvent manipuler une variable contenant une valeur. On peut facilement voir qu'il serait utile d'avoir des variables contenant plusieurs valeurs : des coordonnées, une liste de noms, une liste de notes, etc... À partir du moment où on parle d'une collection de valeur il vient le problème de l'organisation de ces valeurs, l'ordre, etc. On a donc plusieurs types d'objets pour manipuler une collection de valeur. On les appelle des types construits.

II) Tuples

II.1) Objets modifiables ou persistant

> ***Définition*:**
>
> On dira qu'un objet de type construit est modifiable ("mutable" en anglais) si on peut remplacer une des valeurs par une nouvelle valeur, changer la longueur de l'objet, etc. On dit qu'un objet est persistant ou immuable ("immutable" en anglais) s'il n'est pas modifiable une fois construit.

Intérêt de l'immuabilité

- plus rapide d'accès dans la mémoire
- nécessite moins de précaution d'emploi
- l'interpréteur peut gérer l'allocation mémoire de l'objet plus finement, notamment en partageant la représentation mémoire d'objet immuable qui interviennent plusieurs fois.

II.2) Définition des tuples et utilisation en Python

> ***Définition*:**
>
> Un **tuple** ou **p-uplet** ("tuple" en anglais) est une collection immuable de valeur. Cette collection est **indexée**, c'est-à-dire qu'à chaque valeur est attribuée sa position dans la collection.

En Python : Les tuples sont de type tuple en python et sont définis en séparant les valeurs par une virgule et en entourant la collection de valeur par des parenthèses

Exemple de création

In [1]:
```
a = (3,89,-4) #Création d'un tuple à 3 éléments
b = ()        #Création d'un tuple vide
c = (2,)      #Création d'un tuple à 1 élément (attention à la virgule)
```

Opérateur + : Pour concaténer deux tuples, on peut utiliser l'opérateur "+"

In [2]:
```
a = (3,89,-4)
c = (2,)
d = a + c
print(d)
```

```
(3, 89, -4, 2)
```

Opérateur "*" : On peut créer un tuple par répétition d'un tuple avec l'opérateur "*"

In [3]:
```
a = (3,89,-4)
e = a*3
print(e)
```

```
(3, 89, -4, 3, 89, -4, 3, 89, -4)
```

Tester l'appartenance

In [ ]:
```
a = (3,89,-4)
3 in a
2 in a
```

Position et longueur :

In [ ]:
```
a = (3,89,-4)
a[0] #Première valeur (en position 0)
a[2] #Troisième valeur (en position 2)
len(a) #Longueur de a
a[-1] #Dernière valeur de a
```

Remarque : Les parenthèses

Lorsqu'il n'y a pas d'ambiguités, Python reconnait un t-uplet sans les parenthèses.

In [1]:
```
a = 3,4,5
b = 3,
```

In [ ]:
```
type(a)
```

In [ ]:
```
type(b)
```

On peut mettre un t-uplet dans un t-uplet, il est alors nécessaire d'utiliser les parenthèses

In [ ]:
```
t = 3,4,(2,1),5
```

In [ ]:
```
type(t)
```

In [ ]:
```
print(t)
```

Exemple d'utilisation : les affectations multiples

In [ ]:
```
def coord_fonction_carre(a):
    return (a,a**2)

x,y = coord_fonction_carre(5)
```

> ***Exercice*:**
>
> 1. Que renvoie les instructions suivantes :
>
> - ```
>   a = 3
>   type(a)
>   ```
> - ```
>   a = "Bonjour"
>   type(a)
>   ```
> - ```
>   a = (2)
>   type(a)
>   ```
> - ```
>   a = (2,)
>   type(a)
>   ```
> - ```
>   a = (3,2,30)
>   type(a)
>   ```
> - ```
>   a = (3,"fdsf",3,"Bonjour")
>   type(a)
>   ```
>
> 2. Ecrire la ligne de code permettant d'affecter à la variable "couleur" un tuple contenant le triplet RGB de la couleur Rouge
> 3. On considère la variable triplet qui est un tuple à 3 éléments : (3,22,45)
>   - Quelle instruction renvoie la deuxième valeur (donc 22) de triplet
>   - Peut on remplacer le 3 par un 2 dans ce tuple ?
>   - Peut on ajouter une quatrième valeur dans ce tuple ?

III) Listes

III.1) Généralités

III.1.1) Définir une liste

> ***Définition*:**

> Les listes pythons sont des **variables** dans lesquelles on peut mettre plusieurs **variables**.

Exemple

L = [1, 2, 3, 4, 5, 6] permet de définir une liste dans python. Cette liste comporte 6 éléments. Chacun des éléments sont séparés par des virgules. Le tout est encadré par des crochets

In [2]:
```
L = [1,2,3,4,5,6]
L
```

Out[2]:
```
[1, 2, 3, 4, 5, 6]
```

Exemple L = [] permet de définir une liste vide dans python. Cette liste comporte 0 élément.

- Une liste étant une variable on peut mettre des listes dans des listes. C'est très utile lorsqu'on cherche à faire des tableaux.
- Une liste est du type 'list'

III.2) Ce qu'il y a dans les listes

III.2.1) Le premier indice

1. Une liste étant une variable on peut mettre des listes dans des listes. C'est très utile lorsqu'on cherche à faire des tableaux.
2. Une liste est du type 'list'

Exemple

Le premier élément de la liste `L` est donnée par `L[0]`. Le 4e par `L[3]`.

In [3]:
```
L = [1,2,3,4,5,6]
print(L[0],L[3])
```

```
1 4
```

II.2.2) Accéder à des éléments d'une liste

> ***Exercice:*** On pose `L = [1,2,3,4,5,6]`
>
> 1. Que retourne `L[0]` ?
> 2. Que devez vous taper dans la console pour obtenur l'élément 8 ?
> 3. Que devez vous taper dans la console pour obtenir l'élément "c"?

`L[-1]` retourne le dernier élément de la liste `L`. Tester cette commande. Que retourne `L[-2]` ?

In [ ]:
```
# Modifie cette ligne
```

II.2.3) Modifier une liste

Une liste est un objet mutable. On peut donc modifier les éléments d'une liste. Nous pouvons modifier le 3e élément en faisant :

In [4]:
```
L[2] = "J'ai changé"
print(L)
```

```
[1, 2, "J'ai changé", 4, 5, 6]
```

Le premier élément a toujours l'indice 0.

II.2.4) Longueur de liste

Il est possible de connaître le nombre d'éléments d'une liste. Pour cela on utilise la fonction : `len`. `len(L)` retourne le nombre de variable que contient la liste `L`.

In [5]:
```
L = [1,2,3,4]
len(L)
```

Out[5]:
```
4
```

II.3) Manipulation de listes

II.3.1) Concaténation

Il est possible de concaténer deux listes (i.e. les coller l'une après l'autre). Notons `L` et `V` deux listes. On peut créer une nouvelle liste `W` en concaténant les listes `L` et `V`. Pour concaténer, on utilise le "+" : `W = L + V`.

In [6]:
```
L = [1,2,3]
V = [4,6,7]
W = L + V
W
```

Out[6]:
```
[1, 2, 3, 4, 6, 7]
```

II.3.2) Ajouter un élément

Il existe différentes méthodes pour ajouter un élément à une liste.

1. La première méthode (la plus commune) est append (·). Cela permet d'ajouter un élément à droite (à la fin) de la liste. Continuons avec la même liste `L = [1, 2, 3, 4, 5, 6]`. Ici `L.append(7)` va ajouter un 7 à la fin de la liste.

In [7]:
```
L = [1,2,3,4,5,6]
L.append(7)
L
```

Out[7]:
```
[1, 2, 3, 4, 5, 6, 7]
```

2. La seconde méthode est de concaténer avec une liste à un élément.

In [8]:
```
# Fait le
```

3. La méthode `insert(i, .)` ajoute un élément `.` à une liste existante à la position `i`. Tous les éléments d'indice supérieurs à `i` sont décalés vers la droite.

In [ ]:
```
L = [1,2,4]
L.insert(1,4)
L
```

II.3.3) Retirer un élément

On a ajouté un élément, modifié un élément, voyons maintenant comment retirer un élément d'une liste.

- **Supprimer une entrée avec un index**

In [10]:
```
L = ["a","b","c"]
del L[1]
L
```

Out[10]:
```
['a', 'c']
```

- **Supprimer une entrée avec sa valeur**

Attentio on ne supprime alors que l'entré la plus à gauche avec cette valeur.

In [11]:
```
L = [1,2,3,3,5,2]
L.remove(2)
L
```

Out[11]:
```
[1, 3, 3, 5, 2]
```

III.4) Programmer avec les listes

III.4.1) Parcourir une liste

Pour essayer de comprendre ce que signifie parcourir une liste, nous allons travailler sur un exemple : une liste de personnage "célèbre". Considérons la liste suivante : `L = ["Asterix","Obélix","Idéfix","Panoramix","James Bond","Harry Potter"]`. Faisons afficher ces personnages un à un.

In [ ]:
```
L = ["Asterix","Obélix","Idéfix","Panoramix","James Bond","Harry Potter"]
for personnage in L:
    print(personnage)
```

> ***Exercice:*** Considérons une seconde liste (en plus de celle précédente) : `B = ["la marmite de potion magique", "la marmite de jus de citrouille", "l'eau"]`. Voici un petit programme qui montre que ces personnages ne sont pas toujours adroits. Il manque un point à chacune de ces phrases, modifier le programme précédent pour avoir ce point.

In [14]:
```
L = ["Asterix","Obélix","Idéfix","Panoramix","James Bond","Harry Potter"]
B = ["la marmite de potion magique", "la marmite de jus de citrouille", "l'eau"]
import random as rd
for personnage in L:
    print(f" {personnage} est tombé dans {B[rd.randint(1,2)]}")
```

```
Asterix est tombé dans la marmite de jus de citrouille
Obélix est tombé dans l'eau
Idéfix est tombé dans la marmite de jus de citrouille
Panoramix est tombé dans l'eau
James Bond est tombé dans l'eau
Harry Potter est tombé dans l'eau
```

III.5) Parcourir une liste avec `range`

Pour parcourir les éléments d'une liste, nous pouvons utiliser l'indexation de la liste. Voici un exemple.

In [15]:
```
L = [3,2,4,1,5]
for k in range(len(L)):
    print(L[k])
```

```
3
2
4
1
5
```

> ***Exercice:*** Déterminer le nombre d'éléments d'une liste à l'aide d'une boucle `for`

III.6) Méthode `pop`

La méthode `.pop()` supprime et renvoie le dernier élément d'une liste existante. La méthode `.pop(·)` avec l'argument optionnel index <code· supprime et renvoie l'élément à la position `.`

> ***Exercice:***
>
> 1. Créer une liste des jours de la semaine
> 2. Par deux méthodes différentes, retirer le dernier élément de la liste
> 3. Supprimer et renvoyer le premier élément de la liste.

IV) Exercices

> ***Exercice:*** Écrivez un programme qui réalise les différentes instructions suivantes successivement :
>
> - crée une liste contenant 7 entiers ;
> - affiche le deuxième élément de la liste ;
> - affecte la valeur 8 au troisième élément de la liste ;
> - affecte la moyenne des deux premiers éléments de la liste au dernier élément de la liste ;
> - affiche 10 fois de suite l'avant-dernier élément de la liste.

In [ ]:

> ***Exercice:*** Réalisez une fonction Python qui prend une liste en entrée et renvoie en sortie la liste renversée (par exemple, pour l'entrée [2,7,8,1], la fonction renvoie [1,8,7,2]).

In [ ]:

> ***Exercice:*** Réalisez une fonction Python qui prend en entrée une liste de nombres et renvoie en sortie une liste contenant deux éléments : le plus petit et le plus grand élément de la liste.

In [ ]:

> ***Exercice:***

> 1. Réalisez une fonction qui calcule la somme des éléments d'une liste de nombres à l'aide d'une boucle.
> 2. Réalisez une fonction qui calcule la moyenne des éléments d'une liste à partir de cette liste.

In [ ]: