Tuples listes et dictionnaires

I- Tuples et listes

Jusqu'à présent, nous n'avons utilisé des variables que pour stocker une information, mais il est possible de stocker plusieurs informations différentes dans un même objet. En python il existe 3 types d'objets le permettant et il est possible de voir que l'interpréteur les différencie à l'aide de la commande type( ) :

1. Créer 3 variables :

```
a = (11, 6, 9, 7, 3, 1) #parenthèse, séparation des nombres par une virgule
b = [11, 6, 9, 7, 3, 1] #crochets
c = {"un":11,"deux":6,"trois":9,"quatre":7,"cinq":3,"six":1}
```

2. Indiquer ce qu'indique la commande type( ) pour chacune de ces variables :

a :

b :

c :

3. Indiquer le retour de chaque commande

a[2] :

b[2] :

c["deux"] :

b[7] :

c[1] :

4. Executez la commande b[3]=0, puis affichez b. Que remarquez-vous ?

5. Faites de même avec a, que se passe-t-il ?

6. Que faut-il faire pour avoir un résultat similaire avec c ?

II- Opérations sur les tuples et les listes

En mathématiques et en physique on utilise couramment des vecteurs pour représenter des grandeurs telles qu'une vitesse, une accélération ou une force.

On note généralement $\vec{v}=[x,y,z]$. On a l'habitude d'utiliser dans ce cas une multiplication $k\cdot\vec{v}=[k\cdot x,k\cdot y,k\cdot z]$ ou une addition $\vec{v}+\vec{u}=[x_u+x_v,y_u+y_v,z_u+z_v]$

7. créer 2 "vecteurs" u et v :

```
u=[1,2,3]
v=[9,8,7]
```

8. Indiquer le résultat des commandes

3*u :

u + v :

Conclusion :

III- Création d'une liste en compréhension

Il est possible de créer une liste de plusieurs manières. La plus simple a été vue au 1. Lorsque les items de la listes suivent un ordre logique particulier, il est possible de créer une liste en **compréhension**.

Exemple : → Je souhaite créer une liste contenant les 10 multiples de 7 :

```
table_7=[7*i for i in range(1,11)]
```

Il est même possible de créer un tableau (c'est à dire une liste contenant des listes)

```
table_multiplication=[[i*j for j in range(1,11)] for i in range(1,11)]
```

9. A l'aide de la création d'une liste en compréhension, créer une liste qui serait le résultat des opérations 3*u et u+v "classiques"

10. Créer 2 fonctions mult(nombre,vecteur) et add(nombre,vecteur) qui permettraient de retourner les résultats précédents à l'aide des modèles suivants :

```
def mult(nombre,vecteur) :
  res=[…à compléter…]
  return res
```

```
def add(vec1,vec2) :
  res=[…à compléter…]
  return res
```