UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

Info. - TD : Feuille 2

Exercice 1 – listes et tranches de listes en Python

Après le code python suivant

```
1
2 l1 = [50, 42, 33, 21, 10]
3 l2 = [v-10 for v in l1]
4 l3 = list(range(100, 200, 5))
```

Q1) Expliquez quel est le type et la valeur de `l1` (autrement dit, que vaut `l1`).

Q2) Expliquez quel est le type et la valeur de `l2` (autrement dit, que vaut `l2`).

Q3) Expliquez quel est le type et la valeur de `l3` (autrement dit, que vaut `l3`).

Q4) Que vaut `l1[1]`?

Q5) Que vaut `l1[0]`?

Q6) Que vaut `l1[2]`?

Q7) Que vaut `l1[len(l1)]`?

Q8) Que vaut `l1[len(l1)-1]`?

Q9) Que vaut `l2[4]`?

Q10) Que vaut `l2[5]`?

Q11) Que vaut `l3[1]`?

Q12) Que vaut `l3[len(l3)-1]`?

Q13) Que vaut `l2[len(l3)-2]`?

Q14) Que vaut `l1[-1]`?

Q15) Que vaut `l2[-2]`?

Q16) Que vaut `l3[-3]`?

Q17) Que vaut `l2[-4]`?

Q18) Que vaut `l1[-5]`?

Q19) Que vaut `l1[-6]`?

Exercice 2 – tranches de listes en Python

Q20) Que vaut `l3[1:5:1]`?

Q21) Que vaut `l3[1:5:2]`?

Q22) Que vaut `l3[:5:4]`?

Q23) Que vaut `l3[1::4]`?

Q24) Que vaut `l3[::4]`?

Q25) Que vaut `l3[-5:-2:1]`?

Q26) Que vaut `l3[-5:-2:]`?

Q27) Que vaut `l3[-5:-2]`?

Q28) Comment extraire de `l1` toutes ses valeurs sauf la première ?

Q29) Comment extraire de `l1` toutes ses valeurs sauf les deux premières ?

Q30) Comment extraire de `l1` toutes ses valeurs sauf la dernière (indice 4) ?

Q31) Comment extraire de `l1` toutes ses valeurs sauf les deux dernières ?

Q32) Reprendre le dernier groupe de questions (qui utilise `l1`) mais avec une liste `l` quelconque (dans la longueur est inconnue, mais supérieure à 2).

Rappels : indices négatifs

L'indice dans l'accès à une liste peut être négatif : dans ce cas, on compte la position de l'élément en partant de la fin de la liste. Si `l` est une liste et `i` est négatif, on a:

```
l[i]
```

qui est équivalente à

```
l[len(l) + i]
```

Vu autrement, `l[-j]` est équivalent à `l[len(l) - j]`.

Rappels : tranches (et liste en compréhension)

Si `l` est une variable contenant une liste, et `a`, `b` et `c` trois valeurs entières alors

```
res = l[a:b:c]
```

est équivalent à

```
res = [l[i] for i in range(a, b, c)]
```

qui est de plus équivalent à

```
res = []
for i n range(a, b, c):
    res.append(l[i])
```

Compléments :

- Il est possible d'utiliser cette construction à gauche du `=` (pour remplacer les valeurs de la tranche de liste par celles d'une autre liste.
- Il est possible d'omettre le `a`, dans ce cas la tranche part du début de la liste.
- Il est possible d'omettre le `b`, dans ce cas la tranche va jusqu'à la fin de la liste.
- Il est possible d'omettre le `c`, dans ce cas le pas vaut 1 (exemples : `l[1:8:]`, `l[5::]`, `l[3::]`).
- Si le `c` est omis, il est possible d'omettre le second `:` (exemples : `l[1:8]`, `l[5:]`, `l[:3]`).
- Les indices `a` et `b` et le pas `c` peuvent être des entiers négatifs.