TD : Types construits — listes et tuples

Exercices 1 à 6 : à traiter sur le cahier ou sur la feuille, sans utiliser l'ordinateur.

Exercice 1 : Indices et tranches.

On définit la liste `tab = [12, 7, 30, 5, 18]`. Compléter.

`tab[0]` = `tab[3]` = `tab[-1]` = `tab[-2]` = `len(tab)` =

`tab[1:3]` = `tab[:2]` = `tab[3:]` = `7 in tab` = `8 in tab` =

Exercice 2 : Valeurs des variables.

Pour chaque script, donner la valeur des variables à la fin de son exécution.

1.

```
liste1 = [1, 2, 3]
liste1.append(4)
liste1[0] = liste1[1] + liste1[3]
```

`liste1` =

2.

```
liste1 = [5, 8, 2]
dernier = liste1.pop()
premier = liste1.pop(0)
```

`dernier` = `premier` =
`liste1` =

3.

```
liste1 = [4, 1]
liste1 = liste1.append(7)
```

`liste1` =

4.

```
liste1 = [3, 9, 1]
liste2 = liste1
liste2[1] = 0
liste1.append(6)
```

`liste1` =
`liste2` =

Exercice 3 : Listes et fonctions.

```
def f(tab):
    tab[0] = 100
    return len(tab)


tab1 = [1, 2, 3]
tab2 = tab1
tab3 = tab1[:]
n = f(tab2)
```

1. Donner les valeurs de `tab1`, `tab2`, `tab3` et `n`.
2. Expliquer pourquoi `tab1` a été modifiée alors que la fonction ne la renvoie pas.

Exercice 4 : Listes en compréhension.

1. Donner la valeur de chaque liste.
   a. `[n * n for n in range(1, 6)]`
   b. `[2 * i + 1 for i in range(5)]`
   c. `[n for n in range(10) if n % 3 == 0]`
   d. `[ch + ch for ch in "abc"]`
2. Écrire **en compréhension** :
   a. `[0, 5, 10, 15, 20, 25]`
   b. la liste des multiples de 4 compris entre 1 et 30.

Exercice 5 : Tuples.

```
def division(a, b):
    return a // b, a % b


resultat = division(17, 5)
q, r = division(29, 6)
```

1. Donner les valeurs de `resultat`, `resultat[0]`, `q`, `r` puis le type de `resultat`.
2. On définit `tuple1 = ([1, 2], [3, 4])`. Chaque instruction provoque-t-elle une erreur ? Sinon, donner la nouvelle valeur de `tuple1`. a. `tuple1[1][0] = 9` b. `tuple1[1] = [0]`

Exercice 6 : Écrire des fonctions.

Écrire chacune des fonctions suivantes, sans utiliser d'autre méthode que `.append` et `.pop`.

1. `remplace(tab, ancien, nouveau)` : **modifie** la liste `tab` en remplaçant chaque occurrence de `ancien` par `nouveau`. Elle ne renvoie rien.

```
>>> tab1 = [4, 7, 4, 1]
>>> remplace(tab1, 4, 0)
>>> tab1
[0, 7, 0, 1]
```

2. `est_symetrique(tab)` : **renvoie** `True` si la liste se lit de la même façon de gauche à droite et de droite à gauche, `False` sinon.

```
>>> est_symetrique([3, 8, 8, 3])
True
>>> est_symetrique([3, 8, 5])
False
```

3. `compte_parites(tab)` : prend une liste d'entiers et **renvoie** le tuple formé du nombre d'entiers pairs et du nombre d'entiers impairs.

```
>>> compte_parites([4, 7, 2, 9, 10])
(3, 2)
```