1 Liste

Définition. En Python, une *liste* est une collection ordonnée d'objets, repérés par leur indice qui commence à zéro.

Propriété. On peut obtenir et modifier l'élément d'une liste `L` de rang `n` avec l'expression `L[n]`.

Exemple 1. On définit la liste `L` par :

```
L=[6, 8, -2, 9, 1]
```

Quelles sont les valeurs des expressions suivantes ?

(a) `L[1]`

(b) `L[3]`

(c) `L[0]`

(d) `L[-1]`

(e) `L[6]`

Exemple 2. Qu'affichent chacune des instructions **`print`** ?

```
L = [16, 9, 4, 2, 0]
print(L[1])
L[1] = 8
print(L[1])
print(L[2])

L = [4, 10, 3, 5]
L[3] = L[2] - 2
L[0] = L[0] + L[1]
print(L[0])
print(L[3])
```

Propriété. L'instruction **`list(range(n))`** génère la liste des n nombres entiers de 0 à $n-1$.

Exemple 3. Que génèrent les instructions suivantes ?

(a) **`list(range(5))`**

(b) **`list(range(3))`**

2 Ajout d'éléments

Propriété. Il est possible d'ajouter des éléments à la fin d'une liste `L` :

- un seul élément avec : `L.append(X)` ;
- plusieurs éléments avec l'addition : `L + [X, Y]`.

Exemple 4. Décrire *toute* la liste après l'exécution de chaque instruction.

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

Exemple 5. Décrire *toute* la liste après l'exécution de chaque instruction.

```
L = [1, 4]
L.append(3)
L = L + [5, 7, 9]
```

3 Compréhension de liste

Propriété. Une liste peut être générée par une formule, généralement à partir de `range()` :

```
[EXPRESSION for VARIABLE in LISTE]
```

ou

```
[EXPRESSION for VARIABLE in LISTE if CONDITION]
```

Exemple 6. Qu'affichent les programmes suivants ?

```
L1 = [2*i for i in range(5)]
print(L1)
```

```
L2 = [i+3 for i in range(3)]
print(L2)
```

Exemple 7.

1. Écrire un programme qui affiche tous les nombres pairs de 0 à 100.
2. Écrire un programme qui affiche les carrés des nombres de 0 à 10.

Exemple 8. Qu'affiche le programme suivant ?

```
L1 = [3*i for i in range(20) if i > 10]
print(L1)

L2 = [i**2 for i in range(10) if i%2 == 0]
print(L2)
```

4 Exercices

Exercice 1. Qu'affiche le programme suivant ?

```
s = 0
liste = [i**2 for i in range(5)]
for i in range(len(liste)):
    s = s + liste[i]
print(s)
```

Exercice 2. Compléter le programme suivant pour qu'il remplace tous les nombres de la liste supérieurs à 10 par des 0.

```
liste = [1, 12, 7, 8, 24, 2, 33]
for i in range(len(liste)):
    if liste[i] ...:
        ...
```

Exercice 3. Compléter le programme suivant pour qu'il calcule et affiche le nombre d'éléments de la liste qui sont inférieurs à 10.

```
liste = [4, 18, 16, 2, 42, 7]
compteur = ...
for i in range(len(liste)):
    if liste[i] ...:
        ...
print(...)
```

Exercice 4. Compléter le programme suivant pour qu'il cherche et affiche le plus grand élément de la liste.

```
liste = [7, -2, 8, 12, 50, 3]
maximum = ...
for i in range(len(liste)):
    if liste[i] ...:
        ...
print(...)
```

Exercice 5. Compléter la fonction suivante pour qu'elle renvoit la moyenne de la liste donnée en argument.

```
def moyenne(liste):
    somme = ...
    for i in range(...):
        ...
    return ...
```

Exercice 6. Compléter la fonction suivante pour qu'elle renvoit l'étendue de la liste donnée en argument.

```
def étendue(liste):
    minimum = ...
    maximum = ...
    for i in range(...):
        if liste[i] ... :
            ...
        if ...:
            ...
    return ...
```

Exercice 7. Compléter la fonction suivante pour qu'elle renvoit `True` si les deux listes sont égales, `False` sinon.

```
def égales(liste1, liste2):
    for i in range(len(liste1)):
        if ...:
            ...
    ...
```

Exercice 8. Compléter la fonction suivante pour qu'elle renvoit `True` si l'élément `e` est dans la liste `liste`, et `False` sinon.

```
def contient(liste, e):
    for i in range(len(liste)):
        if ...:
            ...
    ...
```

Exercice 9. On donne deux listes `L1` et `L2` de même taille. Compléter l'algorithme suivant pour qu'il compte et affiche le nombre d'éléments communs, à la même place.

```
liste1 = [7, 9, 7, 3, 0, 1]
liste2 = [7, 4, 0, 3, 2, 0]
compteur = ...
for i in range(len(liste1)):
   if liste1[i] ...:
       ...
print(...)
```

Exercice 10. Compléter la fonction suivante pour qu'elle renvoit `True` si la liste `liste` est triée par ordre croissant, et `False` sinon.

```
def triée(liste):
    for i in range(...):
        if liste[i] ...:
            ...
    ...
```