# 06 - Listes - 1ere Partie

November 29, 2023

Listes Python - Introduction & Opérations élémentaires

***Adapté de C. Poulmaire / AEIF***

## 1 Exercice 1 : évolution dans le temps du prix d'un article

Dans cet exercice, on considère un article coûtant initialement 50 euros et augmentant de 3,5 euros par an. Nous allons étudier l'évolution du prix au cours du temps.

1. On considère le code Python suivant :

[ ]:
```
def afficheTousLesPrix(a,n):
    for i in range(1,n+1):
        # ligne à compléter
        a = a + 3.5
    return a

afficheTousLesPrix(50,5)
```

Compléter ce code de façon à ce que, lors de l'exécution de la cellule, s'affichent tous les prix successifs de l'article sur *"n"* ans, y compris le prix initial (on doit donc avoir *n + 1* valeurs !).

On saisit à présent l'instruction `prix = afficheTousLesPrix(50,5)`

Quel est le contenu de la variable `prix` ?

ECRIRE LA REPONSE ICI

2. Cette fonction est-elle satisfaisantes pour étudier l'évolution du prix de l'article au cours du temps ?

ECRIRE LA REPONSE ICI

On considère à présent le code Python suivant :

[ ]:
```
def tousLesPrix(a,n):
    prix = [a]
    for i in range(1,n+1):
        a = a + 3.5
        prix.append(a)
    return prix
```

```
tousLesPrix(50,5)
```

3. Exécuter cette cellule. Que renvoie l'instruction `tousLesPrix(50,5)` ?

ECRIRE LA REPONSE ICI

4. Ecrire dans la cellule ci-dessous une instruction Python donnant le type de la donnée renvoyée par l'instruction `tousLesPrix(50,5)` - reconnaissez-vous ce type ?

[ ]:
```
# Ecrire l'instruction Python ici
```

5. Rendez-vous à présent sur Python Tutor; copiez / collez le code ci-dessus et regardez comment se construit cette liste.

- comment se construit la liste `prix` ?
- à quoi sert l'instruction `prix.append(a)` ?
- comment sont repérés les éléments de la liste `prix` ?

ECRIRE LA REPONSE ICI

4. Exécuter la cellule ci-dessous.

[8]:
```
listePrix = tousLesPrix(50,5)
len(listePrix)
```

- Exécuter la cellule ci-dessus. Que renvoie la fonction `len` ?

ECRIRE LA REPONSE ICI

5. Ecrire dans les deux cellules ci-dessous une instruction renvoyant le premier élément et une instruction renvoyant le dernier élément de la liste `listePrix`.

[ ]:
```
# Ecrire ici une instruction Python renvoyant le premier élément de listePrix␣
↪(pour rappel du cours, le format est Liste[indice])
```

[ ]:
```
# Ecrire ici une instruction Python renvoyant le dernier élément de listePrix
```

- Exécuter la cellule ci-dessous. Qu'obtient-on ? Pourquoi ?

[ ]:
```
listePrix[6]
```

ECRIRE LA REPONSE ICI

## 2 Exercice 2 : définition d'une liste par extension

1. Compléter le code suivant de telle sorte que la liste `L` définie à la première ligne devienne `[1,5,8,-3]`.

[ ]:
```
L = [1,5,8]
L.append(...)
```

2. Compléter le code suivant de telle sorte que la liste `L` définie à la première ligne devienne `[2,5,4,2]`.

[ ]:
```
L = [2,5]
L.append(...)
L.append(...)
```

3. Compléter le code suivant de telle sorte que la liste `L` définie à la première ligne devienne `[0,1,4,9,16,25]`.

[ ]:
```
L = []
for i in range(6):
    L.append(...)
```

## 3 Exercice 3 : nombre de licenciés d'un club de triathlon

Un club de triathlon compte 1500 licenciés. Chaque année, il perd 5% de ses licenciés et enregistre 400 nouveaux inscrits.

Compléter la fonction suivante pour qu'elle renvoie une liste contenant l'évolution du nombre de licenciés sur un nombre `nbAnnees` d'années.

[ ]:
```
def nbLicenciesTriathlon(nbAnnees):
    listeNbLicencies = [...]
    for i in range(...):
        listeNbLicencies.append(int(...)) # on calcule un nombre entier de␣
 ↪licenciés !
    return listeNbLicencies
```

Les quatre cellules suivantes permettent de tester votre fonction. Elles doivent renvoyer `True` lors de leur exécution.

[ ]:
```
nbLicenciesTriathlon(0) == [1500]
```

[ ]:
```
nbLicenciesTriathlon(1) == [1500, 1825]
```

[ ]:
```
nbLicenciesTriathlon(4) == [1500, 1825, 2133, 2426, 2704]
```

[ ]:
```
nbLicenciesTriathlon(8) == [1500, 1825, 2133, 2426, 2704, 2968, 3219, 3458,␣
 ↪3685]
```

## 4 Exercice 4 : échange d'éléments dans une liste

La fonction `echange` ci-dessous prend en paramètre une liste non vide `L` et deux entiers `i` et `j`, et renvoie la liste `L` dans laquelle les éléments d'indices `i` et `j` ont été échangés.

Avant d'effectuer l'échange des deux éléments, la fonction effectue trois tests d'assertion : - le premier vérifie que `L` est bien une liste non vide ; - le deuxième vérifie que l'entier `i` est bien un nombre compatible avec un indice de `L` ; - le troisième vérifie que l'entier `j` est bien un nombre compatible avec un indice de `L`.

Compléter cette fonction `echange`.

[ ]:
```
def echange(L,i,j):
    temp = L[i]
    L[i] = ...
    L[j] = ...
    return ...
```

Les trois cellules suivantes permettent de tester votre fonction. Elles doivent renvoyer `True` lors de leur exécution.

[ ]:
```
echange([8,9],0,1) == [9,8]
```

[ ]:
```
echange([1,2,3,5,4,6,7],3,4) == [1,2,3,4,5,6,7]
```

[ ]:
```
echange(['A','C','B','D'],1,2) == ['A','B','C','D']
```

## 5 Exercice 5 : opérateurs de concaténation et de duplication

Déterminer ce que renvoie chacune des instructions suivantes puis vérifier en exécutant les cellules :

[ ]:
```
[1,2,3] + [4,5]
```

[ ]:
```
[-1,2,3] + [0]
```

[ ]:
```
[5,4,3] + []
```

[ ]:
```
[] + [-1,2]
```

[ ]:
```
[5,4,3] + 5
```

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

[ ]:
```
[5]*2
```

[ ]:
```
6*[0]
```

[ ]:
```
[0,8,9] + [6]*3
```

## 6 Exercice 6 : insertion d'élément dans une liste

1. La fonction `insertionDebutListe` ci-dessous prend en paramètre une liste `L` et une donnée `element`, et renvoie une liste dans laquelle `element` a été inséré au début de `L` à l'aide de l'opérateur de concaténation.

Compléter cette fonction.

[ ]:
```
def insertionDebutListe(L,element):
    return ... + L
```

Les trois cellules ci-dessous sont un jeu de tests pour tester votre fonction; elles doivent toutes renvoyer True.

[ ]:
```
insertionDebutListe([],5) == [5]
```

[ ]:
```
insertionDebutListe([4],0) == [0,4]
```

[ ]:
```
insertionDebutListe([2,1,5],-3) == [-3,2,1,5]
```

2. En s'aidant de la fonction `insertionDebutListe` écrite ci-dessus, écrire une fonction `insertionFinListe` qui permet d'insérer un élément à la fin d'une liste.

[10]:
```
# Ecrire ici le code de la fonction insertionFinListe
```

Les trois cellules ci-dessous sont un jeu de tests pour tester votre fonction; elles doivent toutes renvoyer True.

[ ]:
```
insertionFinListe([],5) == [5]
```

[ ]:
```
insertionFinListe([4],0) == [4,0]
```

[ ]:
```
insertionFinListe([2,1,5],-3) == [2,1,5,-3]
```

3. Reprenez votre fonction `insertionDebutListe` ci-dessus et complétez-la pour qu'elle n'insère l'élément demandé *que* s'il n'est pas déjà présent dans la liste (ainsi, un appel à `insertionDebutListe([1, 2],0)` renverra `[0, 1, 2]` tandis qu'un appel à `insertionDebutListe([1, 2],1)` renverra `[1, 2]` - donc la liste inchangée).

[ ]:
```
# Ecrire ici le code de la fonction insertionFinListe
```

Les trois cellules ci-dessous sont un jeu de tests pour tester votre fonction; elles doivent toutes renvoyer True.

[ ]:
```
insertionDebutListe([1, 2],9) == [9, 1, 2]
```

[ ]:
```
insertionDebutListe([1, 2],1) == [1, 2]
```

[ ]:
```
insertionDebutListe([1, 2],2) == [1, 2]
```