Listes en Python – exercices

Mathématiques – Première spécialité

Logique et raisonnement · exercices ★ à ★★★, pièges et QCM

Nom : _____ Date : _____

Durée : environ 1 h. *Calculatrice interdite, sauf mention contraire. Les exercices vont de ★ (automatismes) à ★★★ (niveau d'un contrôle exigeant). Fais les séries dans l'ordre ; si une série ★ est réussie sans erreur, passe à la suite. Corrige-toi avec le corrigé, puis refais chaque calcul faux sans regarder.*

Les pièges de cette fiche

- Les indices commencent à 0 : `L[1]` est le **deuxième** élément.
- Le dernier indice est `len(L) - 1` : `L[len(L)]` provoque une erreur.
- `range(a, b)` s'arrête à $b - 1$: pour aller jusqu'à n, on écrit `range(n + 1)`.
- `L.append(x)` modifie `L` et ne renvoie rien : jamais `L = L.append(x)`.
- Liste vide, somme, compteur : on les initialise **avant** la boucle, jamais dedans.
- `=` affecte (stocke) une valeur, `==` compare (dans un `if` ou une liste en compréhension).

Exercice 1 – Indices et longueur ★

On a écrit `L = [12, 5, 8, 21, 3, 17]`. Que vaut chaque expression ?

a) `L[0]`

b) `L[3]`

c) `len(L)`

d) `L[len(L) - 1]`

e) `L[6]`

f) `L[1] + L[4]`

g) `L[2] * L[5]`

h) `9 in L`

Exercice 2 – Que contient la liste ? ★

Donne le contenu de la liste à la fin de chaque programme.

a)

```
T = []
for k in range(5):
    T.append(2 * k)
```

b)

```
T = [1]
for k in range(4):
    T.append(3 * T[k])
```

c)

```
L = [4, 7, 1]
L.append(9)
L[0] = 5
```

d)

```
L = [6, 2, 8, 2]
L.remove(2)
```

e)

```
L = [3, 1, 4, 1, 5]
del L[1]
```

f)

```
T = []
for x in [2, -1, 5, 0]:
    T.append(x * x)
```

g)

```
T = []
for k in range(1, 4):
    T.append(k)
    T.append(-k)
```

h)

```
L = [5, 8, 1]
for i in range(len(L)):
    L[i] = L[i] + 10
```

Exercice 3 – Listes en compréhension ★

Donne la liste obtenue.

a) `[k * k for k in range(6)]`

b) `[3 * n - 2 for n in range(1, 6)]`

c) `[d for d in range(1, 21) if 20 % d == 0]`

d) `[x for x in range(-3, 4) if x * x > 4]`

e) `[2 ** n for n in range(7)]`

f) `[k for k in range(50) if k % 7 == 0]`

g) `[x * x - 3 * x for x in [-1, 0, 2, 5]]`

h) `[n for n in range(1, 30) if n % 2 == 1 and n % 3 == 0]`

Exercice 4 – Écrire une fonction qui parcourt une liste ★★

Écris la fonction décrite, puis donne le résultat du test.

a) `somme(L)` : la somme des éléments de `L`. Teste avec `[4, -2, 7, 1]`.

b) `moyenne(L)` : la moyenne des éléments de `L` (non vide). Teste avec `[12, 15, 9]`.

c) `maxi(L)` : le plus grand élément de `L` (non vide). Teste avec `[-5, -2, -9]`.

d) `compte(L, s)` : le nombre d'éléments de `L` supérieurs ou égaux à `s`. Teste avec `[8, 13, 10, 4, 17]` et $s = 10$.

e) `positifs(L)` : la liste des éléments strictement positifs de `L`, écrite en compréhension. Teste avec `[3, -1, 0, 6, -4]`.

Exercice 5 – Termes d'une suite ★★

Réponds à chaque question.

a) Écris en compréhension la liste `U` des termes u_0 à u_5 de la suite $u_n = 5n - 3$, puis donne-la.

b) $u_0 = 2$ et $u_{n+1} = 3u_n - 1$. Écris un programme qui range u_0 à u_4 dans une liste `U`, puis donne-la.

c) (v_n) est géométrique de premier terme $v_0 = 1000$ et de raison 0,5. Écris un programme qui range v_0 à v_4 dans une liste, puis donne-la.

d) Avec la liste `U` de la question a), calcule $u_0 + u_1 + \cdots + u_5$ avec une boucle.

e) $w_n = n^2 - 10n$. Que vaut `c` à la fin de ce programme ? Que représente-t-il ?

```
W = [n * n - 10 * n for n in range(20)]
c = len([w for w in W if w < 0])
```

Exercice 6 – Programmes à trous ★★

Complète les `...` pour que le programme fasse ce qui est annoncé : a) liste des multiples de 3 de 3 à 30 ; b) somme $1^2 + 2^2 + \cdots + n^2$ pour $n = 6$; c) produit des éléments de `L` ; d) liste des entiers de 1 à 40 qui se terminent par 7.

a)

```
T = []
for k in range(1, ...):
    T.append(...)
```

b)

```
n = 6
s = ...
for k in range(1, ...):
    s = ...
```

c)

```
L = [3, -2, 5, 1, 4]
p = ...
for x in L:
    p = ...
```

d)

```
D = [d for d in range(1, ...) if ...]
```

Exercice 7 – Les notes ★★★

Les notes d'un élève sont rangées dans une liste : `notes = [12, 8.5, 15, 9, 17, 11, 6.5, 14]`.

1. Que valent `len(notes)`, `notes[2]` et `notes[7]` ?
2. Écris un programme qui calcule la moyenne `m` des notes, et donne-la.
3. Écris en compréhension la liste `bonnes` des notes d'au moins 10. Combien y en a-t-il ?
4. Écris un programme qui trouve l'**indice** `i_max` de la meilleure note.

Exercice 8 – Tester une conjecture ★★★

On conjecture que, pour tout entier naturel n, le nombre $n^2 + n + 41$ est premier. La fonction ci-dessus renvoie `True` si l'entier `k` est premier, `False` sinon.

1. Explique pourquoi `premier(k)` renvoie `False` dès qu'elle trouve un diviseur.
2. Écris en compréhension la liste `C` des entiers n de 0 à 49 pour lesquels $n^2 + n + 41$ n'est **pas** premier.

3. La conjecture est-elle vraie ? Justifie par un calcul.
4. Si la liste C avait été vide, aurait-on démontré la conjecture ?

```
def premier(k):
    if k < 2:
        return False
    for d in range(2, k):
        if k % d == 0:
            return False
    return True
```

Exercice 9 – Trouve l'erreur ★★

Chaque programme ne fait pas ce qui est annoncé. Trouve l'erreur, puis corrige.

a)

```
L = [4, 9, 2]
x = L[3]
```

But : mettre le dernier élément dans x.

b)

```
L = [5, 3, 8, 6]
S = 0
for i in range(1, len(L)):
    S = S + L[i]
```

But : la somme des éléments de L.

c)

```
T = []
for k in range(5):
    T = T.append(k * k)
```

But : la liste des carrés de 0 à 4.

d)

```
L = [14, 8, 21, 11]
for x in L:
    c = 0
    if x > 10:
        c = c + 1
```

But : compter les éléments plus grands que 10.

e)

```
L = [-4, -1, -7]
m = 0
for x in L:
    if x > m:
        m = x
```

But : le plus grand élément de L.

f)

```
P = [k for k in range(1, 10) if k % 2 = 0]
```

But : les nombres pairs de 1 à 9.

QCM – Six questions comme en contrôle

Une seule bonne réponse par question. Note ta réponse, puis vérifie dans le corrigé.

1. Avec `L = [7, 3, 9, 4, 6]`, `L[2]` vaut :

   A 3 **B** 4 **C** 9 **D** 2

2. `len([2, 5, 5, 8])` vaut :
   A `4` **B** `3` **C** `8` **D** `5`
3. `[k * 2 for k in range(1, 4)]` vaut :
   A `[2, 4, 6, 8]` **B** `[0, 2, 4]` **C** `[1, 2, 3, 4, 5, 6]` **D** `[2, 4, 6]`
4. Avec `L = [10, 20, 30]`, `L[len(L)]` donne :
   A `30` **B** une erreur **C** `10` **D** `3`
5. Après `T = [1, 2]`, `T.append(5)` et `T.append(1)`, `T` vaut :
   A `[1, 2, 6]` **B** `[5, 1, 1, 2]` **C** `[1, 2, 5]` **D** `[1, 2, 5, 1]`
6. Pour parcourir tous les indices d'une liste `L`, on écrit :
   A `for i in range(len(L)):` **B** `for i in range(1, len(L)):`
   C `for i in range(len(L) + 1):` **D** `for i in range(L):`

Ce document est la propriété intellectuelle de Ben, professeur particulier (benprofparticulier.fr), protégée par le Code de la propriété intellectuelle (art. L. 111-1 et L. 122-4). Il est mis à disposition pour un usage personnel de révision. Toute reproduction, diffusion, mise en ligne ou revente, totale ou partielle, sans autorisation écrite est interdite et constitue une contrefaçon (art. L. 335-2).