



TD1 – Fonctions et recherches dans les séquences

Rappel. Une fonction se définit avec `def`. Les paramètres reçoivent les valeurs fournies lors de l'appel ; `return` transmet le résultat et termine la fonction. `print` affiche, mais ne renvoie pas de résultat.

```
def double(x):  
    return 2*x  
  
y = double(5)      # y vaut 10
```

1 | Définir et appeler une fonction

Question 1 : Sans exécuter le code, donner les valeurs de `a` et `b` après les instructions suivantes. Expliquer pourquoi `a` ne contient pas le même objet que `b`.

```
def carre(x):  
    print(x*x)  
  
def carre_retour(x):  
    return x*x  
  
a = carre(4)  
b = carre_retour(4)
```

Question 2 : Écrire une fonction `est_pair(n)`, qui renvoie un booléen valant `True` si l'entier `n` est pair. Donner les valeurs renvoyées par les appels `est_pair(8)` et `est_pair(9)`.

2 | Lire et écrire des fonctions

On suppose que `n` est un entier naturel et que `M` est un nombre positif.

Question 3 : Donner le rôle et un exemple d'appel de chacune des quatre fonctions suivantes. Pour `f4`, préciser ce que devient le résultat si `M = 0`.

```
def f1(x):  
    if x >= 0:  
        return x  
    else:  
        return -x  
  
def f2(n):  
    s = 0  
    i = 0  
    while i < n:  
        i += 1  
        s += i**2  
    return s  
  
def f3(n):  
    p = 1  
    for i in range(1, n + 1):
```

```

        p = p*i
    return p

def f4(M):
    s = 0
    i = 0
    while s < M:
        i += 1
        s += i**2
    return i

```

Question 4 : Écrire une fonction `somme(L)` qui renvoie la somme des nombres de la liste. Que renvoie `somme([])` ?

Question 5 : Écrire `moyenne(L)` en réutilisant `somme(L)`. On suppose que `L` n'est pas vide. Donner la valeur renvoyée pour `[8, 10, 12]`.

3 | Chercher une valeur dans une liste

On considère une liste d'entiers `L`. Pour les questions suivantes, on parcourt explicitement la liste : ne pas utiliser `in`, `index`, `count`, `min`, ni `max`.

Question 6 : Écrire une fonction `contient(L, cible)`, qui renvoie `True` si `cible` apparaît dans `L`, et `False` sinon. *Pour cela, parcourir les éléments de la liste : « pour chaque élément... »*. Déterminer les résultats de `contient([4, -2, 7, -2], -2)` et `contient([], 3)`.

Question 7 : Écrire une fonction `premier_indice(L, cible)`, qui renvoie l'indice de la première apparition de `cible`, ou `-1` si la valeur est absente. Prévoir les résultats pour `[4, -2, 7, -2]` avec les cibles `-2`, `5` et pour la liste vide.

4 | Chercher dans une chaîne de caractères

En tant que séquence, une chaîne se parcourt comme une liste. On cherche ici un **caractère unique** et on distingue majuscules et minuscules. Ne pas utiliser `in`, `find`, ni `count`.

Question 8 : Écrire `nombre_occurrences(texte, caractere)`, qui renvoie le nombre d'apparitions du caractère dans la chaîne. Donner le résultat pour `("banane", "a")` et pour une chaîne vide.

Question 9 : Écrire `premiere_position(texte, caractere)`, qui renvoie l'indice de la première apparition du caractère ou `-1` s'il est absent. Préciser les résultats pour `("banane", "n")`, `("banane", "z")` et `("", "a")`.

Pour aller plus loin. Adapter `premiere_position` pour chercher une sous-chaîne entière, comme `"nane"` dans `"banane"`. Faire apparaître la dernière position de départ possible.

5 | Manipuler des listes

Question 10 : Pour chacune des listes ci-dessous, écrire une expression qui extrait l'élément demandé : le nombre 2, le nombre 3 dans la sous-liste, puis la lettre "m".

```

L1 = [4, 5, 3, 2, 1]
L2 = [4, "soleil", [1, 3], 0]
L3 = [5.1, 3, "informatique", 89.5]

```

Question 11 : Construire trois listes : les entiers de 0 à 15 inclus ; les entiers pairs de 10 à 100 inclus ; puis les nombres strictement inférieurs à 150 de la forme $3k + 2$ avec $k \in \mathbb{N}$. Donner le premier et le dernier élément de chacune.

Question 12 : Écrire une suite d'instructions qui affiche une seule fois `True` si `x` figure dans `L`, et

False sinon.

Question 13 : Les listes L et M ont la même longueur. Construire une nouvelle liste N dont l'élément d'indice i est la somme de L[i] et M[i]. Donner N pour L = [4, 5, 6] et M = [2, 3, 4].

6 | Trouver le minimum et le maximum

Dans cette partie, la liste est supposée non vide.

Question 14 : Pour L = [-8, -3, -12], expliquer pourquoi le programme ci-dessous donne un résultat faux, puis corriger son initialisation.

```
maximum = 0
for valeur in L:
    if valeur > maximum:
        maximum = valeur
```

Question 15 : Écrire une fonction maximum(L) qui renvoie la plus grande valeur d'une liste non vide.

Question 16 : Écrire une fonction extremes(L) qui renvoie les deux valeurs (minimum, maximum) après un seul parcours de la liste. Donner les résultats renvoyés pour [4, 1, 9, 1], [-8, -3, -12] et [6].

Question 17 : Écrire une fonction maximumS(L) qui renvoie la liste contenant la plus grande valeur et son nombre d'occurrence, on pourra utiliser la fonction maximum(L). Donner les résultats renvoyés pour [4, 5, 6, 1, 6] et [-8, -3, -12].