1. function_1(tableau1, tableau2)

```
def function_1(tableau1, tableau2):
    presentDansDeuxListes = 0
    for nombre1 in tableau1:
        for nombre2 in tableau2:
            if nombre1 == nombre2:
                presentDansDeuxListes += 1
                break
    return presentDansDeuxListes
```

Analyse :

- La première boucle itère sur tous les éléments de `tableau1` (nn éléments).
- La deuxième boucle imbriquée itère sur tous les éléments de `tableau2` (mm éléments), mais elle s'arrête dès qu'une correspondance est trouvée (`break`).
- Dans le pire des cas, chaque élément de `tableau1` sera comparé à tous les éléments de `tableau2` (aucune correspondance).

Complexité : O(n⋅m)O(n \cdot m), où n=len(tableau1)n = \text{len}(tableau1) et m=len(tableau2)m = \text{len}(tableau2).

2. function_2(x)

```
def function_2(x):
    valeur = 0
    while x > 0:
        valeur = valeur + x
        x -= 1
    return valeur
```

Analyse :

- La boucle `while` diminue la valeur de `x` jusqu'à ce qu'elle atteigne 0.
- La boucle s'exécute exactement xx fois.

Complexité : O(x)O(x).

3. function_3(x)

```
def function_3(x):
    valeur = 0
    if x < 0:
        valeur = -x
    elif x > 0:
        pass
    return valeur
```

Analyse :

- La fonction contient uniquement des blocs conditionnels simples (`if` et `elif`), chacun étant évalué en temps constant.
- Il n'y a pas de boucle ni de récursivité.

Complexité : O(1)O(1) (constante).

Résumé des complexités :

1. **function_1** : O(n · m)O(n \cdot m)
2. **function_2** : O(x)O(x)
3. **function_3** : O(1)O(1)