

# Corrigé sujet 12 - Année : 2023

Sujet 12 - 2022 ↴

## 1. Exercice 1

Script Python

```
1 class ABR:
2     def __init__(self, g0, v0, d0):
3         self.gauche = g0
4         self.cle = v0
5         self.droit = d0
6
7     def __repr__(self):
8         if self is None:
9             return ''
10        else:
11            return '(' + (self.gauche).__repr__() + ',' + str(self.cle) + ',' +
12            (self.droit).__repr__() + ')'
13
14 n0 = ABR(None, 0, None)
15 n3 = ABR(None, 3, None)
16 n2 = ABR(None, 2, n3)
17 abr1 = ABR(n0, 1, n2)
18
19
20 def ajoute(cle,a):
21     if a == None:
22         return ABR(None,cle,None)
23     elif a.cle>cle:
24         sag = ajoute(cle,a.gauche)
25         return ABR(sag,a.cle,a.droit)
26     elif a.cle<cle:
27         sad = ajoute(cle,a.droit)
28         return ABR(a.gauche,a.cle,sad)
29     else:
30         return a
```

### Attention

- Le sujet est difficile et aborde diverses notions du programme (arbre binaire de recherche, récursivité, programmation objet).
- De façon très inhabituelle pour une exercice 1, une portion de code est fournie, c'est le code de la classe `ABR`, avec la méthode d'affichage. A noter que la `if` de cette méthode n'est jamais executé, en effet `None` n'est pas une instance de la classe `ABR`, par contre `None` a déjà une méthode `__repr__` (ce qui arrête la récursivité)
- La fonction à écrire `ajoute` devrait être une méthode de classe `ABR` et pas une fonction externe, elle devrait donc modifier l'arbre donné en paramètre et pas en créer un nouveau.

## 2. Exercice 2

### Script Python

```

1 def empaqueter(liste_masses, c):
2     n = len(liste_masses)
3     nb_boites = 0
4     boites = [0]*n
5     for masse in liste_masses : #(1)
6         i=0
7         while i <= nb_boites and boites[i] + masse > c: #(2)
8             i = i + 1
9             if i == nb_boites + 1: #(3)
10                 nb_boites = nb_boites + 1
11                 boites[i] = boites[i] + masse
12             return nb_boites + 1 #(4)

```

- Parcours de la liste des masses
- Tant qu'on a pas atteint la première boîte vide et que la masse ne rentre pas on avance dans la liste de boîtes.
- La masse s'insère dans une boîte vide, donc le nombre de boîte utilisée augmente
- Les boîtes sont comptées à partir de zéro, donc on ajoute un au nombre

### Attention

Dans cette correction la fonction renvoie 1 pour si la liste de masse est vide, si on veut que la fonction renvoie `0`, on doit soit introduire un `if` dans le `return` (hors programme), soit modifier le code donné.