

TP - Classement de villes par ABR

Objectifs :

- ▷ Utiliser un arbre binaire de recherche (ABR) pour classer des villes selon différents critères.

Position	Commune	Département	
# 1	Paris	Paris	2 145 906 hab
# 2	Marseille	Bouches-du-Rhône	870 321 hab
# 3	Lyon	Rhône	522 228 hab
# 4	Toulouse	Haute-Garonne	498 003 hab

1 Travail à faire

- Récupérer le fichier `villes.csv` qui contient la liste des 200 plus grandes villes de France dans un ordre aléatoire.

Chaque ligne de ce fichier contient les 5 informations suivantes :

- ▷ Nom de la ville
- ▷ Numéro du département
- ▷ Nombre d'habitants
- ▷ Superficie (en km²)
- ▷ Rang au niveau national (critère : nombre d'habitants)

Voici à quoi ressemblent les deux premières lignes de ce fichier :

```
1 Saint-Priest ,69 ,40944 ,29.7 ,155
2 Versailles ,78 ,85761 ,26.2 ,46
```

- Créer une classe `Ville` telle que ci-après :

```
1 class Ville :
2
3     def __init__(self,liste) :
4         ''' Le constructeur'''
5         self.nom = liste[0]
6         self.departement = int(liste[1])
7         self.population = int(liste[2])
8         self.superficie = float(liste[3])
9         self.rang = int(liste[4])
10
11 # creation d'un objet:
12 liste = ['Nice', '6', '343123', '71.9', '5']
13 ville=Ville(liste)
```

- Vérifier l'accès aux différents attributs.
- Écrire une méthode `getRang(self)`, qui retourne le rang de la ville.
- Écrire une méthode `getSuperficie(self)`, qui retourne la superficie de la ville.
- Écrire une méthode `afficherVille(self)`, qui affiche les 5 données de la ville.
- Écrire une méthode `afficherNom(self)`, qui affiche le nom de la ville.

8. Utiliser la classe Noeud ci-dessous vue au chapitre précédent sur les ABR :

```

1 class Noeud:
2     '''Structure de donnees de type ABR'''
3
4     def __init__(self, liste, left=None, right=None):
5         self.ville = Ville(liste)
6         self.left = left
7         self.right = right
8
9     def __str__(self):
10        return str(self.ville.afficherVille())
11
12    def estFeuille(self):
13        if not self.left and not self.right:
14            return True
15        else:
16            return False

```

9. Rajouter la méthode `insérer(self,liste)` :

```

1 def insérer(self, liste):
2     '''Methode pour insérer une ville dans l'ABR'''
3
4     ville = Ville(liste)
5     if ville.getRang() < self.ville.getRang() :
6         if self.left is None:
7             self.left = Noeud(liste)
8         else:
9             self.left.insérer(liste)
10    else :
11        if self.right is None:
12            self.right = Noeud(liste)
13        else:
14            self.right.insérer(liste)

```

L'arbre est alors "*rangé*" suivant le rang de la ville.

10. Rajouter dans votre programme principal les lignes de code ci-dessous :

```

1 import csv
2 liste_villes=[]
3
4 with open("villes.csv",'r',encoding='utf-8') as f:
5     lecteur=csv.reader(f,delimiter=',')
6     for ligne in lecteur:
7         liste_villes.append(ligne)

```

Rappel : il faut que le fichier `villes.csv` soit dans le même dossier que votre fichier Python.

11. Vérifier que l'on a bien créé une liste `liste_villes` contenant les lignes du fichier `villes.csv`.

Questions :

- ▷ Pour obtenir un arbre équilibré, il faut choisir la racine avec un rang de 100. Pourquoi ?

Réponse :

- ▷ En quoi est-ce important pour une recherche dans l'arbre ?

Réponse :

12. Construire l'arbre avec le code ci-dessous :

```
1 liste = ["Maisons-Alfort", '94', '51091', '5.4', '100']
2 noeud = Noeud(liste)
3 print(noeud)
4
5 for el in liste_villes:
6     ville = Ville(el)
7     noeud.inserer(el)
```

13. Récupérer la méthode (ou la fonction) `parcours_infixe()` créée dans un chapitre précédent et l'adapter pour faire afficher les villes par ordre croissant de leur rang.
14. Modifier ce qu'il faut pour faire afficher les villes par ordre croissant de leur superficie.
15. En utilisant cette nouvelle construction, écrivez une méthode `rechercheMax(self)` qui renvoie la ville ayant la plus grande superficie.
16. Écrire une méthode `rechercher(self, rang)`, qui retourne la ville dont le rang est `rang`.