

Devoir n°2 : POO sur table

1. Enoncé

[Devoir n°2 : SQL !\[\]\(666e09182d4cd268646ea700ea60dcdf_img.jpg\)](#)

2. Correction

Partie 1

- 1.
2. `nom` : attribut
3. `tab_voisines` : attribut
4. `tab_couleurs_disponibles` : attribut
5. `nom_region` est de type `str` : chaîne de caractère.
6. `ge = Region ("Grand Est")`

7. Script Python

```
def renvoie_premiere_couleur_disponible(self):
    return self.tab_couleurs_disponibles[0]
```

- 8.

Script Python

```
def renvoie_nb_voisines(self) :
    return len(self.tab_voisines)
```

1. Script Python

```
def est_coloriee(self):
    if self.couleur_attribuee == None :
        return False
    else :
        return True
```

2. Script Python

```
def retire_couleur(self, couleur):
    if couleur in self.tab_couleurs_disponibles:
        self.tab_couleurs_disponibles.remove(couleur)
```

3.  Script Python

```
def est_voisine(self, region):
    for i in range(len(self.tab_voisines)) :
        if region == self.tab_voisines[i] :
            return True
    return False
```

4.  Script Python

```
def renvoie_tab_regions_non_coloriees(self):
    L=[]
    for region in self.tab_regions :
        if est_coloriee(region) == False :
            L.append(region)
    return L
```

5. a. La méthode renvoie `None` dans le cas où tout est colorié.

b. La région renvoyée est la région qui a le plus de voisines parmi celles qui ne sont pas colorierées.

6.  Script Python

```
def colorie(self):
    region_m = self.renvoie_max()
    while region_m:
        region_m.couleur_attribuee = region_m.renvoie_premiere_couleur_disponible()
        for voisine in region_m.tab_voisines:
            voisine.retire_couleur(region_m.couleur_attribuee)
        region_m = self.renvoie_max()
```