

### EXERCICE 3 (6 points)

*Cet exercice traite de programmation orientée objet en Python et d'algorithmique.*

Un pays est composé de différentes régions. Deux régions sont voisines si elles ont au moins une frontière en commun. L'objectif est d'attribuer une couleur à chaque région sur la carte du pays sans que deux régions voisines aient la même couleur et en utilisant le moins de couleurs possibles.

La figure 1 ci-dessous donne un exemple de résultat de coloration des régions de la France métropolitaine.

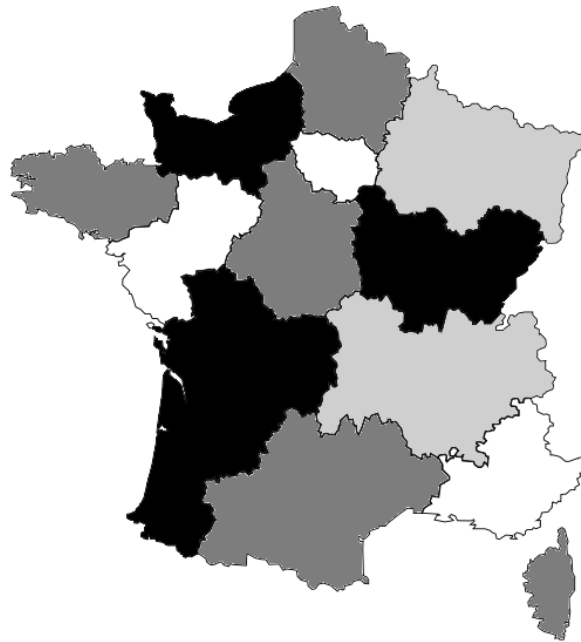


Figure 1 – Carte colorisée des régions de France métropolitaine

On rappelle quelques fonctions et méthodes des tableaux (le type `list` en Python) qui pourront être utilisées dans cet exercice :

- `len(tab)` : renvoie le nombre d'éléments du tableau `tab` ;
- `tab.append(elt)` : ajoute l'élément `elt` en fin de tableau `tab` ;
- `tab.remove(elt)` : enlève la première occurrence de `elt` de `tab` si `elt` est dans `tab`. Provoque une erreur sinon.

Exemple :

- `len([1, 3, 12, 24, 3])` renvoie 5 ;
- avec `tab = [1, 3, 12, 24, 3]`, l'instruction `tab.append(7)` modifie `tab` en `[1, 3, 12, 24, 3, 7]` ;
- avec `tab = [1, 3, 12, 24, 3]`, l'instruction `tab.remove(3)` modifie `tab` en `[1, 12, 24, 3]`.

Les deux parties de cet exercice forment un ensemble. Cependant, il n'est pas nécessaire d'avoir répondu à une question pour aborder la suivante. En particulier, on pourra utiliser les méthodes des questions précédentes même quand elles n'ont pas été codées.

Pour chaque question, toute trace de réflexion sera prise en compte.

## Partie 1

On considère la classe `Region` qui modélise une région sur une carte et dont le début de l'implémentation est :

```
1 class Region:
2     '''Modélise une région d'un pays sur une carte.'''
3     def __init__(self, nom_region):
4         '''
5         initialise une région
6         : param nom_region (str) le nom de la région
7         '''
8         self.nom = nom_region
9         # tableau des régions voisines, vide au départ
10        self.tab_voisines = []
11        # tableau des couleurs disponibles pour colorier
12        la région
13        self.tab_couleurs_disponibles = ['rouge', 'vert',
14        'bleu', 'jaune', 'orange', 'marron']
15        # couleur attribuée à la région et non encore
16        choisie au départ
17        self.couleur_attribuee = None
```

1. Associer, en vous appuyant sur l'extrait de code précédent, les noms `nom`, `tab_voisines`, `tab_couleurs_disponibles` et `couleur_attribuee` au terme qui leur correspond parmi : *objet*, *attribut*, *méthode* ou *classe*.

2. Indiquer le type du paramètre `nom_region` de la méthode `__init__` de la classe `Region`.

3. Donner une instruction permettant de créer une instance nommée `ge` de la classe `Region` correspondant à la région dont le nom est « Grand Est ».

4. Recopier et compléter la ligne 6 de la méthode de la classe `Region` ci-dessous :

```
1 def renvoie_premiere_couleur_disponible(self):
2     '''
3     Renvoie la première couleur du tableau des couleurs
4     disponibles supposé non vide.
5     : return (str)
6     '''
7     return ...
```

**5. Recopier et compléter la ligne 6 de la méthode de la classe `Region` ci-dessous :**

```
1 def renvoie_nb_voisines(self) :
2     '''
3     Renvoie le nombre de régions voisines.
4     : return (int)
5     '''
6     return ...
```

**6. Compléter la méthode de la classe `Region` ci-dessous à partir de la ligne 6 :**

```
1 def est_coloriee(self):
2     '''
3     Renvoie True si une couleur a été attribuée à cette
4     région et False sinon.
5     : return (bool)
6     '''
7     ...
```

**7. Compléter la méthode de la classe `Region` ci-dessous à partir de la ligne 8 :**

```
1 def retire_couleur(self, couleur):
2     '''
3     Retire couleur du tableau de couleurs disponibles de
4     la région si elle est dans ce tableau. Ne fait rien
5     sinon.
6     : param couleur (str)
7     : ne renvoie rien
8     : effet de bord sur le tableau des couleurs
9     disponibles
10    '''
11    ...
```

**8. Compléter la méthode de la classe `Region` ci-dessous, à partir de la ligne 7, en utilisant une boucle :**

```
1 def est_voisine(self, region):
2     '''
3     Renvoie True si la region passée en paramètre est une
4     voisine et False sinon.
5     : param region (Region)
6     : return (bool)
7     '''
8     ...
```

## Partie 2

Dans cette partie :

- on considère qu'on dispose d'un ensemble d'instances de la classe `Region` pour lesquelles l'attribut `tab_voisines` a été renseigné ;
- on pourra utiliser les méthodes de la classe `Region` évoquées dans les questions de la partie 1 :
  - `renvoie_premiere_couleur_disponible`
  - `renvoie_nb_voisines`
  - `est_coloriee`
  - `retire_couleur`
  - `est_voisine`

On a créé une classe `Pays` :

- cette classe modélise la carte d'un pays composé de régions ;
- l'unique attribut `tab_regions` de cette classe est un tableau (type `list` en Python) dont les éléments sont des instances de la classe `Region`.

9. Recopier et compléter la méthode de la classe `Pays` ci-dessous à partir de la ligne 7 :

```
1 def renvoie_tab_regions_non_coloriees(self):
2     '''
3     Renvoie un tableau dont les éléments sont les régions
4     du pays sans couleur attribuée.
5     : return (list) tableau d'instances de la classe
6     Region
7     '''
8     ...
```

10. On considère la méthode de la classe `Pays` ci-dessous.

```
1 def renvoie_max(self):
2     nb_voisines_max = -1
3     region_max = None
4     for reg in self.renvoye_tab_regions_non_coloriees():
5         if reg.renvoye_nb_voisines() > nb_voisines_max:
6             nb_voisines_max = reg.renvoye_nb_voisines()
7             region_max = reg
8     return region_max
```

- Expliquer dans quel cas cette méthode renvoie `None`.
- Indiquer, dans le cas où cette méthode ne renvoie pas `None`, les deux particularités de la région renvoyée.

**11.** Coder la méthode `colorie(self)` de la classe `Pays` qui choisit une couleur pour chaque région du pays de la façon suivante :

- On récupère la région non coloriée qui possède le plus de voisines.
- Tant que cette région existe :
  - ♦ La couleur attribuée à cette région est la première couleur disponible dans son tableau de couleurs disponibles.
  - ♦ Pour chaque région voisine de la région :
    - si la couleur choisie est présente dans le tableau des couleurs disponibles de la région voisine alors on la retire de ce tableau.
  - ♦ On récupère à nouveau la région non coloriée qui possède le plus de voisines.