2 Afficher de l'information

Comme nous l'avons vu au cours précédent, les objets géographiques sont définis à la fois par leurs géométries, mais aussi par leurs propriétés. Le thème de ce cours sera de lire et de représenter les propriétés des objets géographiques.

En nous basant sur les fonctions de lecture de fichiers MIF que nous avons construites au cours précédent, nous allons maintenant lire les propriétés des objets.

2.1 Lectures des propriétés

Les propriétés des objets peuvent être vues comme des couples *(nom, valeur)*. Par exemple, un département français peut avoir les propriétés nommées *Nom_Departement* et *Code_Departement* avec, respectivement, les valeurs *Calvados* et *14*. La structure de données PYTHONque nous allons utiliser pour représenter les propriétés sera donc le dictionnaire. Dans les données que nous allons utiliser, les noms et les valeurs sont dans deux fichiers différents :

— les noms de propriétés sont stockées dans l'entête du fichier MIF ;

— les valeurs sont stockées dans le fichier MID associé.

Dans un premier temps, il faut modifier la fonction de lecture des fichiers MIF pour qu'elle permette de lire les noms de propriétés. Ensuite, il faut ajouter une fonction pour lire les valeurs dans le fichier MID.

Parfois, les propriétés peuvent être définies avec un type et une unité. Pour ce travail, nous ne tiendrons pas compte de ces paramètres.

Lecture des noms de propriétés

Le début de la partie du fichier contenant noms de propriétés est identifiée par la ligne commençant par *Columns*. La deuxième information de cette ligne est le nombre de propriétés.

Chaque propriété est ensuite codée sur une ligne. La première information de chaque ligne est son nom et la seconde information est son type.

Pour simplifier le programme, le type des propriétés ne sera pas utilisé. Nous considérerons qu'il s'agit toujours de propriétés textuelles.

```
1  version 300
2  Charset "WindowsLatin1"
3  CoordSys Earth Projection 3, 1002, "m", 0.000000000000, 46.800000000000,
4  45.898918964419, 47.696014502038, 600000.000, 2200000.000
5  Columns 9
6  Code_International char (200)
7  Code_Region char (200)
8  Nom_Region char (200)
9  Code_Departement_County_UA char (200)
10 Nom_Departement_County_UA char (200)
11 Statut_Administratif char (200)
12 Superficie_km2 char (200)
13 Population_Totale_Sans_Double_Compte char (200)
14 Densite char (200)
15 Data
16 Region 62
17 13
18 176700.00 2448800.00
19 176800.00 2448900.00
20 176900.00 2448900.00
21 176900.00 2449000.00
22 176900.00 2449100.00
23 177000.00 2449100.00
```

FIGURE 10 – Début d'un fichier MIF

Lecture des valeurs des propriétés

Les valeurs des propriétés sont stockées dans le fichier MID associé. A chaque objet géographique du fichier MIF correspond une ligne dans le fichier MID. Chaque ligne est composée de différentes valeurs des propriétés de

l'objet séparées par des tabulations. Ces valeurs sont ordonnées de la même façon que dans le fichier MIF.

Pour lire ce genre fichier, le plus simple est d'utiliser la classe *reader* du module *csv*. Cette classe permet d'itérer sur les lignes du fichier en retournant la liste des valeurs de chaque ligne.

```
>>> print csv.reader.__doc__
    csv_reader = reader(iterable [, dialect='excel']
                        [optional keyword args])
    for row in csv_reader:
        process(row)

The "iterable" argument can be any object that returns a line
of input for each iteration, such as a file object or a list.  The
optional "dialect" parameter is discussed below.  The function
also accepts optional keyword arguments which override settings
provided by the dialect.

The returned object is an iterator.  Each iteration returns a row
of the CSV file (which can span multiple input lines)

>>> s = '"albert"\t"10"\t"12"\n"bernard"\t"11"\t"15"\n"cedric"\t"13"\t"9"'

>>> print s
"albert"        "10"    "12"
"bernard"       "11"    "15"
"cedric"        "13"    "9"

>>> f = StringIO.StringIO(s)

>>> for l in csv.reader(f, delimiter="\t", lineterminator="\n"):
>>>     print l
>>>
['albert', '10', '12']
['bernard', '11', '15']
['cedric', '13', '9']
```

FIGURE 11 – Aide de la classe "reader" du module csv

```
"4EUFR" "53"    "Bretagne"      "22"    "C\^otes d'Armor" "D\'epartement"
"6878"  "550610"        "80.0"
"4EUFR" "11"    "Ile-de-France" "92"    "Hauts-de-Seine"        "D\'epartement"
"176"   "1462347"       "8309.0"
"4EUFR" "52"    "Pays de la Loire"      "49"    "Maine-et-Loire"        "D\'epartement"
"7166"  "742703"        "104.0"
"4EUFR" "11"    "Ile-de-France" "77"    "Seine-et-Marne"        "D\'epartement"
"5915"  "1220221"       "206.0"
```

FIGURE 12 – Début d'un fichier MID

Exercice de TD. Modifiez les fonctions de lecture de fichiers MIF/MID de façon à ajouter aux objets géographiques des propriétés.

2.2 Le clustering

À partir des valeurs des propriétés, nous allons modifier la couleur des zones. Pour cela, il faut pouvoir attribuer une couleur à chaque zone. Il existe différentes méthode pour attribuer une couleur en fonction d'un attribut. Nous utiliserons la méthode des quartiles et nous découperons les données en quatre classes différentes (A, B, C, D).

Cette méthode se base sur un calcul de médianes. C'est à dire une valeur q telle que pour liste donnée il y ait autant de valeurs plus petites que de valeurs plus grandes que q.

Les quatre groupes sont définis tels que :

— $objgeo \in A$ ssi $val < q1$

— $objgeo \in B$ ssi $q1 \leq val < q2$

— $objgeo \in C$ ssi $q2 \leq val < q3$

— $objgeo \in D$ ssi $q3 \leq val$

En considérant que *val = objgeom.getProp(nomProp)*

Exercice de TD. Définissez une classe *Quartiles* qui permet de faire un découpage en quatre groupes. Le constructeur de la classe prendra en paramètre une liste d'objets géographiques, le nom de la propriété, et une liste de couleurs. Cette classe possédera une méthode *getCouleur* qui prendra en argument un objet géographique et qui retournera sa couleur.

Exercice de TP.

- Implémentez la lecture de fichiers MID et la classe Quartiles.
- Générez une carte de la zone transmanche représentant les densités de population. Vous utiliserez les fichiers *transmanche3* disponibles son le wiki. Les quatre couleurs devront être en dégradé, par exemple : marron, rouge, orange et jaune. Vous pouvez utiliser la balise "g" du svg pour factoriser des propriétés de différents polygone (par exemple leur couleur de remplissage).
- Ajoutez une méthode *toSVG* à la classe *Quartiles* qui dessine une légende des valeurs à partir des informations de la classe *Quartiles*.