Exercice 1 *Préparation*

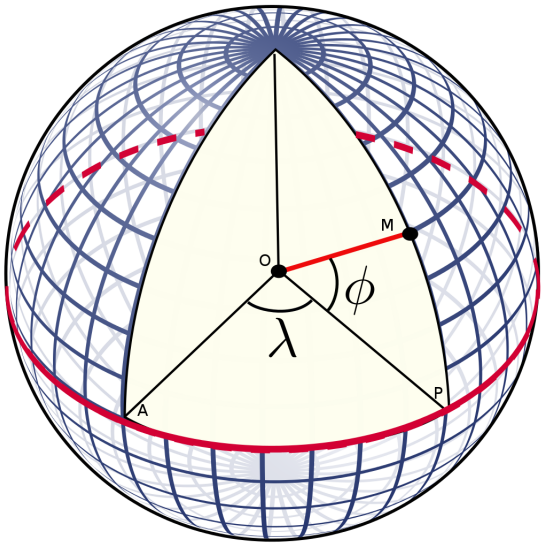
1. Copier le sous-dossier `Activite2` du dossier `ICN\Cartes` situé dans le partage `Donnees` de sa classe. Coller ce dossier dans son espace personnel.
2. `Activite2` contient un sous-dossier `ressources` avec les fichiers textes que nous allons manipuler. Ouvrir `Pyzo`, enregistrer dans `ressources` un fichier `Texte-Activite2.py`, puis exécuter ce fichier avec l'option `Run as script`.

1 Localisation des bureaux de vote parisiens

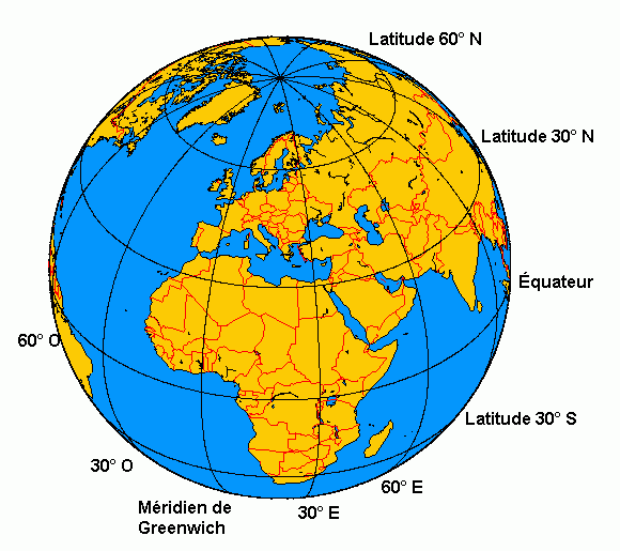
Bloc-Note 1

On assimile la sphère terrestre à une sphère de centre O.
Soit M un point de la sphère terrestre distinct des pôles N et S.
Le *méridien* de M coupe l'équateur en P et le *méridien origine* de Greenwich coupe l'équateur en A.

- ☞ La *longitude* du point M est la mesure en degrés de l'angle $\widehat{AOP}$ suivie de l'indication Ouest (O) ou Est (E).
- ☞ La *latitude* du point M est la mesure en degrés de l'angle $\widehat{POM}$ suivie de l'indication Nord (N) ou Sud (S).
- ☞ La *latitude* et la *longitude* d'un point constituent les *coordonnées géographiques* de ce point.



Source : Wikimedia domaine public



Source : Wikimedia domaine public

Exercice 2

Le jeu de données avec les coordonnées géographiques des 896 bureaux de vote parisiens est accessible depuis l'URL :
https://opendata.paris.fr/explore/dataset/bureaux-de-votes/
Pour chaque bureau de vote, les données sont structurées au format `JSON`. Depuis l'onglet information, on peut déplier le modèle de données. On veut récupérer pour chaque bureau l'identifiant unique `'id_bv'` constitué du numéro d'arrondissement et du numéro de bureau ainsi que les coordonnées géographiques du bureau.

```
{'datasetid': 'bureaux-de-votes',
 'fields': {'adresse': '6 RUE MOLIERE',
  'cp': '75001',
```

```
 'geo_point_2d': [48.864652867508056, 2.335488359834315],
 'geo_shape': {'coordinates': [2.335488359834315, 48.864652867508056],
  'type': 'Point'},
 'id_bv': '1-7',
 'lib': "COLLEGE D'ENSEIGNEMENT SECONDAIRE POQUELIN",
 'num_bv': '7',
 'objectid': 7},
'geometry': {'coordinates': [2.335488359834315, 48.864652867508056],
 'type': 'Point'},
'record_timestamp': '2017-05-13T18:21:47+02:00',
'recordid': 'cd49d9d74951ef47b590ba804e4b4b8b33e9efd0'}
```

1. Dans l'exemple ci-dessus, quel est l'identifiant unique du bureau ? Quelle est son adresse ? sa latitude ? sa longitude ?

2. Pour récupérer le jeu de données sous une forme exploitable par Python, on va utiliser la bibliothèque / module `requests` qui permet de gérer les requêtes et réponses `HTTP`.

   a. Quelle est la signification de l'acronyme `HTTP` ? Où le retrouve-t-on fréquemment ? Pourquoi ?

   b. Voici une suite d'instructions permettant de récupérer le jeu de données au format `JSON`, sous la forme d'une liste de dictionnaires `Python`. Chaque élément de la liste est un dictionnaire stockant les informations d'un bureau de vote, ainsi le dictionnaire `data[0]` est structuré comme dans l'exemple ci-dessus.

   Le lien de téléchargement s'obtient sélectionnant l'export au format `JSON` depuis la page https://opendata.paris.fr/explore/dataset/bureaux-de-votes/export/.

```
In [14]: import requests

In [14]: url = 'https://opendata.paris.fr/explore/dataset/bureaux-de-votes/
     download/?format=json&timezone=Europe/Berlin'

In [15]: reponse = requests.get(url)

In [16]: data = reponse.json()

In [17]: data[0]
```

3. On souhaite écrire une fonction `coordBureau(urlsource)` qui prend comme paramètre l'URL du jeu de données au format `JSON` avec les coordonnées des bureaux de votes, et qui retourne un dictionnaire `Python` dont les clefs sont les couples d'entiers *(n° arrondissement, n° bureau)* et les valeurs les couples de coordonnées géographiques *(longitude, latitude)* de chaque bureau.

   Par exemple, avec dans `url` le lien de téléchargement précédent, on obtient pour le deuxième bureau du premier arrondissement :

```
In [21]: lesbureaux = coordBureau(url)

In [22]: lesbureaux[(1,2)]
Out[22]: (2.342029948146027, 48.85990426535164)
```

   Recopier et compléter le code de la fonction `coordBureau(urlsource)`.

```
def coordBureau(urlsource):
    reponse = requests.get(urlsource)
    data = reponse.json()
```

```
    lesbureaux = dict()
    for bureau in data:
        #à  compléter
    return lesbureaux
```

2 Fichiers KML

Exercice 3 *Structure d'un fichier KML*

1. Lancer le logiciel `GoogleEarth`, puis dans la barre de recherche, saisir *Lycée du Parc, Lyon, France*. Le logiciel affiche une vue satellite du quartier autour du lycée et marque le lycée avec une icône.

   - Faire un clic droit sur le lieu, puis enregistrer le lieu, sous le nom `lyceeparc` dans le dossier `ressources`. Le système ajoute automatiquement l'extension `.kmz`.
   - Il s'agit d'un fichier compressé, faire alors un clic droit sur le fichier pour extraire son contenu.
   - On obtient un fichier `doc.kml` qu'on ouvre avec un éditeur de textes comme `Notepad`.

   Ce fichier au format `KML` est un calque où `GoogleEarth` peut lire les informations de localisation (coordonnées géographiques) et des méta-informations (adresse, téléphone), du lycée qui a été marqué comme `Placemark`.

2. Lire le paragraphe `Placemarks` de la page https://developers.google.com/kml/documentation/kml_tut#placemarks puis repérer dans le fichier `lyceeparc.kml` créé ci-dessus :

   - l'en-tête `XML` ;
   - la déclaration de format `KML` ;
   - la balise `<Placemark>` et ses balises filles `<name>` (le nom affiché), `<description>` (contenu de l'infobulle) et `<Point>` (avec les coordonnées géographiques) : ce sont les balises essentielles pour marquer un lieu.

3. Expliquer la définition donnée par `Wikipedia` du format de fichier `KML`, développer en particulier les acronymes.

   KML (Keyhole Markup Language) que l'on peut traduire par « langage à base de balises géolocales », est un langage fondé sur le formalisme XML et destiné à la gestion de l'affichage de données géospatiales dans les logiciels de SIG.

4. Récupérer le fichier simplifié `leParc.kml` dans `ressources`. Ouvrir ce fichier dans `GoogleEarth`, normalement le lycée devrait être marqué avec une icône différente de celle par défaut.

   - On a rajouté une balise `<Style>` avec une balise fille `<IconStyle>` pour définir un style d'icône : la couleur dans `<color>` et l'URL d'accès à l'image dans `<Icon> <href> URL de l'image </href></Icon>`.
   - Le codage de la couleur est de la forme `aabbggrr` : dans l'ordre les composantes Alpha, Blue, Green, Red dont les valeurs varient de 0 à 255 et sont notées sur deux chiffres en hexadécimal.
   - Le style d'icône a pour identifiant `"monlycee"`, valeur de l'attribut `id` de la balise `<Style>`. Il suffit d'associer ce style en insérant une balise `<styleUrl> #monlycee </styleUrl>` dans la balise `<Placemark>` marquant le lieu.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>

<Style id="monlycee">
 <IconStyle>
    <color>ff00ff00</color>
        <Icon>
          <href>http://www.gstatic.com/mapspro/images/stock/503-wht-blank_maps.
               png</href>
        </Icon>
    </IconStyle>
</Style>

<Placemark>
    <name> Lycée du Parc  </name>
    <styleUrl> #monlycee  </styleUrl>
    <description>1 Boulevard Anatole France, 69458 Lyon, France,
     Tel: +33 4 37 51 15 51 </description>
    <Point> <coordinates>4.856898,45.771707,0</coordinates> </Point>
</Placemark>

</Document>
</kml>
```

a. A quoi sert la composante alpha dans le codage d'une couleur ?

b. Quelle est la couleur codée par `ff00ff00` ? Quelle couleur doit-on obtenir en remplaçant le code de couleur par `6f0000ff` ?

5. La fonction `rgb2kml(couleur)` prend en argument un triplet d'entiers entre 0 et 255 codant une couleur en (R,G,B) et doit retourner une chaîne de caractères codant la couleur dans le format KML, sous la forme `'aabbggrr'` avec une composante alpha maximale à `'ff'`.

```
In [28]: rgb2kml((255, 18, 10))
Out[28]: 'ff0a12ff'
```

Recopier et compléter le code de cette fonction :

```
def rgb2kml(couleur):
    code = 'ff'
    for k in range(3):
        #à  compléter
    return code
```

Bloc-Note 2 *Notation hexadécimale*

☞ Les couleurs sont souvent codées sur trois composantes (Red, Green, Blue), chacune étant codée sur un octet. Les $2^8 = 256$ valeurs d'un octet varient de 0 à 255 et il est compact de les noter en notation *hexadécimale* sur deux chiffres.

☞ La notation *hexadécimale* est l'écriture en base 16 dont les chiffres sont 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, `a`, `b`, `c`, `d`, `e` et `f`. Par exemple, en hexadécimale, $165 = 10 \times 16 + 5$ s'écrit `a5` et $255 = 15 \times 16 + \times 15$ s'écrit `ff`.

☞ La fonction `hex` permet de convertir un entier en hexadécimale, on obtient une chaîne de caractères préfixée de `0x`. La conversion réciproque s'effectue avec `int` en précisant la base.

```
In [30]: hex(165)
Out[30]: '0xa5'

In [31]: int('0xa5', 16)
Out[31]: 165
```

☞ Pour coder un octet en hexadécimal, deux chiffres suffisent car $16^2 = 256$. Parfois le nombre s'écrit sur un seul chiffre mais le formatage en impose deux, sans le préfixe `'0b'`. Les méthodes de chaînes de caractères `lstrip` et `zfill` sont alors utiles.

```
Out[35]: '0xb'

In [36]: hex(11).lstrip('0x')   #suppression du préfixe
Out[36]: 'b'

In [37]: hex(11).lstrip('0x').zfill(2) #remplissage à gauche sur 2 chiffres
Out[37]: '0b'
```

Exercice 4 *Génération d'un fichier KML*

On souhaite réaliser une carte des bureaux de votes parisiens à partir du dictionnaire des coordonnées géographiques des bureaux de votes retourné par la fonction `coordBureau` de l'exercice 2.
Un premier calque avec la même icône pour chaque bureau peut être généré par la fonction `bureauKML(fichier, source)` qui se trouve dans `cadeau.py`
La génération automatique de fichiers textes repose sur le principe du *template* (ou *pattern*) dont on complète certains emplacements par des valeurs variables. En Python, il existe des modules dédiés comme `Jinja` mais on peut se contenter d'un formatage de chaînes de caractères. Ici, on utilise un seul `template`, `placemark` qu'on formate selon les caractéristiques des différents bureaux de votes lues dans `source`.

```
def bureauKML(fichier, source):
    header = """<?xml version="1.0" encoding="UTF-8"?>
    <kml xmlns="http://www.opengis.net/kml/2.2">
    <Document>
    """
    footer = """</Document></kml>"""
    iconstyle = """<Style id="bv">
      <IconStyle>
      <color>ffff0000</color>
      <Icon><href>http://www.gstatic.com/mapspro/images/stock/503-wht-blank_maps.png
           </href></Icon>
      </IconStyle>
    </Style>
    """
    placemark = """<Placemark>
    <styleUrl>#bv</styleUrl>
    <Point><coordinates>{},{}</coordinates></Point>
```

```
    </Placemark>
    """
    f = open(fichier, 'w')
    f.write(header)
    f.write(iconstyle)
    for idbur, coord in source.items():
        long, lat = coord
        f.write(placemark.format(long, lat))
    f.write(footer)
    f.close()
```

1. Tester la fonction `bureauKML`.

```
In [25]: lesbureaux = coordBureau('https://opendata.paris.fr/explore/dataset/
     bureaux-de-votes/download/?format=json&timezone=Europe/Berlin')

In [26]: bureauKML('bureauvote.kml', lesbureaux)
```

2. Modifier `bureauKML` en une fonction `bureauKML2` qui génère un styles d'icône pour chaque arrondissement et associe à chaque bureau de vote le style de son arrondissement.

   Avec la fonction `randint` du module `random` et la fonction `rgb2kml` de l'exercice précédent, voici une façon de générer une liste de 20 codes de couleur `KML` :

```
In [40]: from random import randint

In [41]: palette = [rgb2kml((randint(0,255),randint(0,255),randint(0,255))) for
      k in range(20)]
```

3 Vote parisien au premier tour de l'élection présidentielle 2017

Exercice 5

Écrire une fonction `premierKml(lesbureaux, lespremiers, fichier)` qui génère un fichier `KML` permettant de visualiser dans `GoogleEarth` la carte des bureaux de vote parisiens avec pour chaque bureau une icône dont la couleur identifie le premier sortant et la taille est proportionnelle à son score. On peut aussi intensifier la couleur en fonction du score.

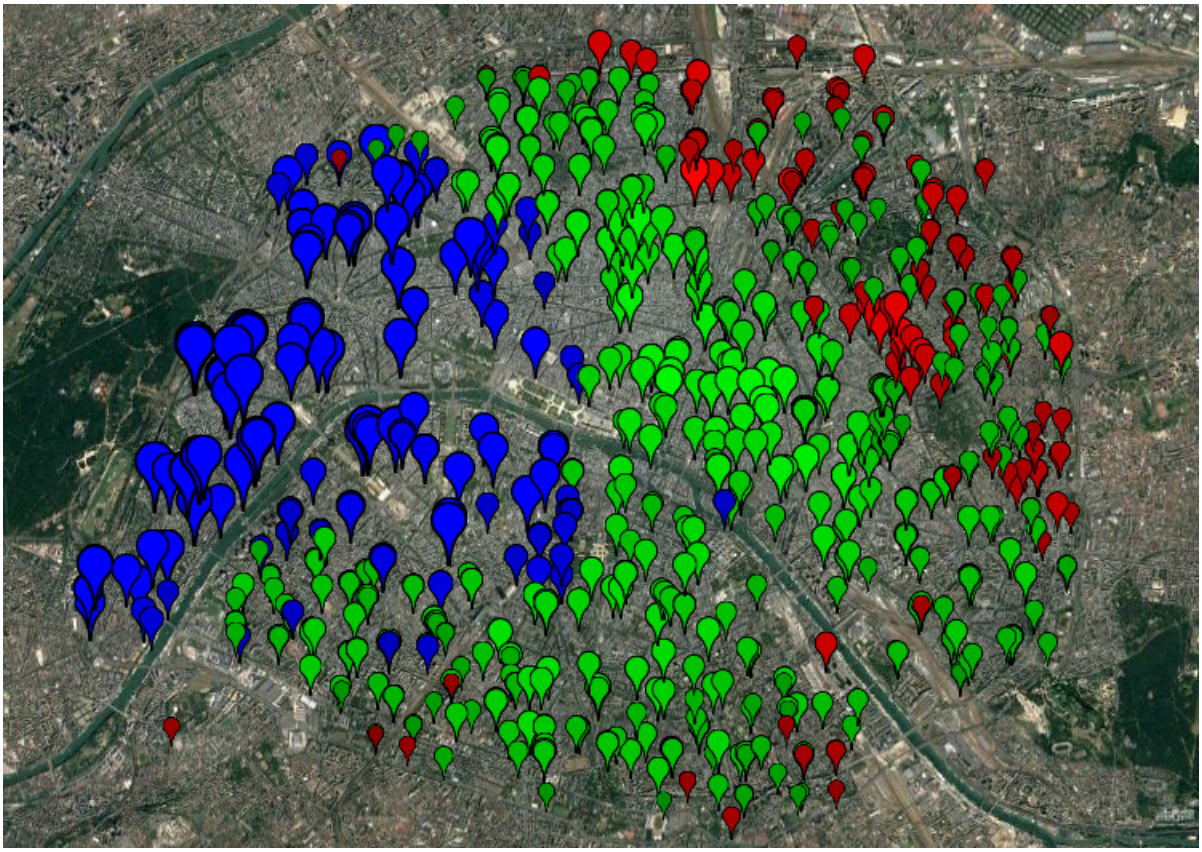
- On admet que trois candidats se sont partagés les premières places sur les bureaux parisiens : M.Fillon, M.Macron et M.Mélenchon.
- On utilisera les jeux de données triés dans les activités 1 et 2 portant sur les scores par bureau de vote et les coordonnées géographiques des bureaux.

```
In [20]: lesbureaux = coordBureau('https://opendata.paris.fr/explore/dataset/
     bureaux-de-votes/download/?format=json&timezone=Europe/Berlin')

In [21]: t = filtrage1('resultats_electoraux.csv',';','Présidentielle 2017 - 1
     er tour')

In [22]: data = filtrage2(t)
```

```
In [23]: lespremiers = premierParBureau(data)

In [24]: premierKml(lesbureaux, lespremiers, 'presidentielle_premier_tour_paris
     .kml')
```

- On créera un style et donc un identifiant pour chaque bureau de vote.
- Dans la balise `<description>` de chaque bureau de vote, on précisera l'arrondissement, le numéro du bureau, le nom du premier et son score.

```
<Placemark>
<description>Arrondissement 1, Bureau : 1, Premier : MACRON, Score : 38.07 %<
     description>
<styleUrl>#3-(1, 1)</styleUrl>
<Point><coordinates>2.341064476164004,48.860083526572126</coordinates></Point>
</Placemark>
```

- La taille des icônes, peut être réglée dans la balise `<Icon>` avec une balise fille `<scale>`.

```
<Style id="3-(1, 1)">
     <IconStyle>
        <color>ff00c200</color>
        <scale>0.7614879649890591</scale>
         <Icon>
         <href>http://maps.google.com/mapfiles/kml/paddle/wht-blank.png</href>
       </Icon>
     </IconStyle>
   </Style>
```