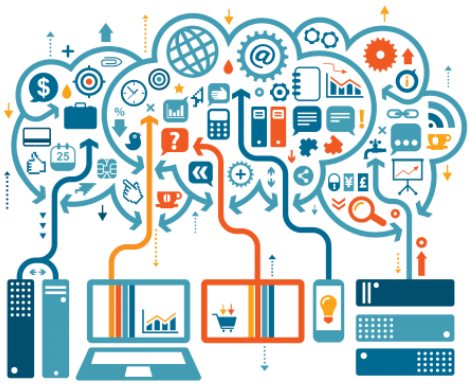
TP1 : CSV, XML, JSON

Table des matières :

1. Les données dans notre société



Les données sont devenues un enjeu pour notre société. Elles touchent tous les domaines : la santé, l'éducation, l'industrie, la sécurité, le commerce,… De nouveaux termes sont apparus : ***Big Data***, ***Open Data***, … de nouveaux métiers sont créés : *Architecte Big Data*, *Data Scientist*, … de nouvelles disciplines sont enseignées : *global data analytics*, … de nouvelles technologies sont développées : le ***Cloud Computing***, les bases de données ***NoSQL***, … et de nouveaux algorithmes sont appliqués : ***MapReduce***, ***Spark***, …

L'utilisation et la maîtrise du big data suscite beaucoup d'enthousiasme, mais également des inquiétudes, en particulier sur la protection des données à caractère personnel.

Travail n°1 :

A partir des liens ci-dessous, répondre aux questions suivantes :

https://www.lebigdata.fr/definition-big-data
https://www.lebigdata.fr/top-metiers-du-big-data-cloud
https://fr.blog.businessdecision.com/bigdata/2016/05/blockchain-big-data-enjeux-strategiques/
https://blockchainfrance.net/decouvrir-la-blockchain/c-est-quoi-la-blockchain/
https://www.lebigdata.fr/open-data-definition

1) Etablir une définition du big data.
2) Définir la règle des 3v qui définit les caractéristiques des outils big data.
3) Expliquer le métier d'ingénieur big data.
4) Indiquer en quoi la solution blockchain permet de protéger les données.
5) Lister les 3 critères fondateurs de l'opendata.

2. Mise en forme des données

Les données sont principalement représentées sous la forme de tableaux. On parle de données tabulaires.

Exemple :

Nom	Prenom	Age
Perrin	Léo	18
Petit	Loïc	32
Leroux	Pierre	27

Il existe trois formats pour représenter un tableau de données : les formats CSV, XML et JSON.
Ces trois formats sont des fichiers composés d'une suite de caractères où l'on distingue deux types d'information :

- Les données.
- Les caractères permettant de structurer ces données.

a) Le format CSV

Le format *Comma Separated Values (CSV)* structure les données sous la forme de valeurs séparées par des virgules. Ce format est très facile à générer et à manipuler.
Chaque ligne du fichier CSV correspond à une ligne du tableau et chaque valeur séparée par une virgule correspond à une colonne du tableau.

Exemple du tableau précédent au format CSV :

```
1  Nom,Prenom,Age
2  Perrin,Léo,18
3  Petit,Loic,32
4  Leroux,Pierre,27
```

La première ligne du fichier contient l'entête de la table, à savoir le nom de chacune des colonnes. Les lignes suivantes contiennent les données du tableau, en respectant l'ordre des colonnes. Le séparateur n'est pas forcément une virgule, on peut par exemple utiliser le point-virgule.

b) Le format XML

Le format *eXtensible Markup Language* (XML) est un format basé sur l'utilisation de balises pour structurer les données. Les balises sont utilisées pour encadrer un contenu : il y a une balise ouvrante et une balise fermante.

Exemple du tableau au format XML :

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <liste>
3      <personne>
4          <NOM>Perrin</NOM>
5          <PRENOM>Léo</PRENOM>
6          <AGE>18</AGE>
7      </personne>
8      <personne>
9          <NOM>Petit</NOM>
10         <PRENOM>Loic</PRENOM>
11         <AGE>32</AGE>
12     </personne>
13     <personne>
14         <NOM>Leroux</NOM>
15         <PRENOM>Pierre</PRENOM>
16         <AGE>27</AGE>
17     </personne>
18 </liste>
```

c) Le format JSON

Le format *JavaScript Object Notation* (JSON) est un format plus récent utilisé pour représenter des objets qui dérive de la notation des objets du langage JavaScript. Un document JSON est essentiellement un ensemble de *paires* constituées d'une étiquette et

d'une valeur ou d'une liste de valeurs. Les paires sont placées entres accolades et séparées par des virgules. Les valeurs des listes sont placées entre crochets et séparées par des virgules.

Exemple du tableau au format JSON :

```
{
  "liste": {
    "personne": [
      {
        "NOM": "Perrin",
        "PRENOM": "Léo",
        "AGE": 18
      },
      {
        "NOM": "Petit",
        "PRENOM": "Loic",
        "AGE": 32
      },
      {
        "NOM": "Leroux",
        "PRENOM": "Pierre",
        "AGE": 27
      }
    ]
  }
}
```

Le premier ensemble possède une paire dont la clé est ***liste*** et dont la valeur est une liste de trois éléments. Chacun de ces trois éléments est un ensemble avec trois paires représentant respectivement le nom, le prénom et l'âge de chaque personne. Remarquez que les nombres ne sont pas placés entre guillemets, contrairement aux mots et aux clés qui doivent être entre guillemets.

Travail n°2 :
A partir du tableau suivant créer trois fichiers au format CSV, XML et JSON.

Nom	Prix	code
Banane	5,99	77
Pomme	2,99	99
Poire	7,99	170

3. Traitement des données avec Processing

Processing possède de nombreuses instructions dédiées au traitement des données. L'objectif de l'activité sera de réaliser un programme pour chaque format de données : CSV, XML et JSON.
Les fichiers de données, que nous allons utiliser, sont accessibles sur l'open data de Nantes à l'adresse suivante : https://data.nantes.fr/donnees/detail/alertes-pollens-a-nantes/

Ces données représentent le niveau de pollens à Nantes en fonction des essences d'arbres ou de plantes. Ces données sont actualisées tous les jours.

Table : Alertes_pollens_nantes

Nom de l'attribut	Code de l'attribut	Type	Description	Valeurs possibles
Date	Date	DATE	Date	2018-02-14T00:00:00.000Z
Nom	Nom	STRING	Nom de l'espèce allergisante	Flouve, Houlque, Noisetier, Ray grass, Graminée, Vulpin, Chêne, Frêne, Bouleau, Dactyle, Armoise, Plantain, Fromental, Fléole, Saule, Aulne
Type	Type	STRING	Type de l'espèce	Herbacée, Arbre
Sous-type	Sous-type	STRING	Sous-type de l'espèce	Graminée, null
Etat	Etat	INTEGER	Etat de l'alerte (1 pour "pas d'émission", 2 pour 'émission en cours", 3 pour "plus d'émission" et 4 pour "non observable")	1, 2, 4

Travail n°3 :
Télécharger puis renommer les fichiers CSV, XML, JSON sous la forme Alertes_pollens_nantes.csv

a) Traitement d'un fichier CSV

Depuis la version 2 de Processing, la classe ***Table*** est recommandée pour traiter les données tabulaires issues des fichiers CSV. La classe ***Table*** est équipée de nombreuses méthodes facilitant la manipulation des données CSV : voir https://processing.org/reference/Table.html

Exemple :

```
Table tablePollens;

void setup()
{
  // --- initialisation fenêtre de base ---
  size(400, 400); // ouvre une fenêtre xpixels  x ypixels
  background(0,0,0); // couleur fond fenetre
  tablePollens = loadTable("Alertes_pollens_nantes.csv", "header");
  TableRow ligne_0=tablePollens.getRow(0);
  String nom_0=ligne_0.getString("Nom");
  println(nom_0);
}
```

- La méthode ***loadTable()*** extrait les données contenues dans le fichier "Alertes_pollens_nantes.csv". Le paramètre "header", indique que la première ligne extraite du fichier "Alertes_pollens_nantes.csv" correspond à un entête, c'est-à-dire aux noms des différentes colonnes (les attributs : Date, Nom, Type, ….).
- La méthode ***getRow()*** renvoie une ligne de la table (ici la ligne 0). Cette ligne est affectée à la variable ***ligne_0*** de type ***TableRow***.
- La méthode ***getString(« Nom »)*** renvoie la chaîne de caractères liée à l'attribut ***Nom***. Nous utilisons cette méthode ici car l'attribut ***Nom*** est de type ***String*** (si l'attribut est de type ***int***, il faudra utiliser la méthode ***getInt()***...)

Travail n°4 :
Réalisez un programme permettant d'afficher dans la console processing le nom de toutes les espèces allergisantes.
Vous utiliserez obligatoirement un algorithme avec une structure itérative (boucle for).
Pour vous aider : la méthode ***getRowCount()*** renvoie le nombre de lignes d'une table (utilisation : ***tablePollens.getRowCount()***)

b) Traitement d'un fichier XML

Le fichier XML que nous souhaitons analyser est de la forme :

```
<?xml version="1.0" encoding="UTF-8"?>
- <document>
    <version>1</version>
    <nb_results>16</nb_results>
  - <data>
    - <element>
        <Sous-type>null</Sous-type>
        <Nom>Armoise</Nom>
        <Etat>1</Etat>
        <Date>Wed Feb 14 01:00:00 CET 2018</Date>
        <Type>Herbacée</Type>
      </element>
    - <element>
        <Sous-type>null</Sous-type>
        <Nom>Plantain</Nom>
        <Etat>1</Etat>
        <Date>Wed Feb 14 01:00:00 CET 2018</Date>
        <Type>Herbacée</Type>
      </element>
    - <element>
        <Sous-type>null</Sous-type>
        <Nom>Graminée</Nom>
        <Etat>1</Etat>
        <Date>Wed Feb 14 01:00:00 CET 2018</Date>
        <Type>Herbacée</Type>
      </element>
    - <element>
        <Sous-type>null</Sous-type>
        <Nom>Frêne</Nom>
        <Etat>4</Etat>
        <Date>Wed Feb 14 01:00:00 CET 2018</Date>
        <Type>Arbre</Type>
      </element>
    - <element>
        <Sous-type>null</Sous-type>
        <Nom>Chêne</Nom>
        <Etat>4</Etat>
        <Date>Wed Feb 14 01:00:00 CET 2018</Date>
        <Type>Arbre</Type>
      </element>
    - <element>
        <Sous-type>null</Sous-type>
        <Nom>Saule</Nom>
        <Etat>2</Etat>
        <Date>Wed Feb 14 01:00:00 CET 2018</Date>
        <Type>Arbre</Type>
      </element>
  </data>
</document>
```

Pour traiter des données XML, Processing utilise la classe ***XML***.

Exemple :

```
XML xmlPollens ;
void setup()
{
  xmlPollens=loadXML("Alertes_pollens_nantes.xml");
  XML[] elements=xmlPollens.getChildren("data/element");
  for(int i=0; i<elements.length; i++)
  {
    XML espece=elements[i].getChild("Nom");
    String nom=especе.getContent();
    println("espece: "+nom);
  }
}
```

Travail n°5 :
Recopiez le code ci-dessous dans un sketch Processing. Testez et analysez le code à l'aide de la documentation Processing (https://processing.org/reference/XML.html)
Réalisez ensuite un programme permettant d'afficher dans la console processing le nom de toutes les espèces allergisantes ainsi que leur type (Herbacée ou Arbre).

Le XML est un format standard du net, notamment pour les flux RSS. De nombreux fils d'actualité (météo, bourse, informations,…) sont disponibles en ligne au format RSS. La méthode ***loadXML()*** peut lire un fichier XML en local ou un fichier XML en ligne en passant un paramètre son URL.

c) Traitement d'un fichier JSON

Le format JSON est aussi un standard du net et son utilisation est devenue prépondérante par rapport au format XML.

Le fichier XML que nous souhaitons analyser est de la forme :

```
{
    "version":"1",
    "nb_results":16,
    "data":[
    {
        "Sous-type" :  null  ,
        "Nom" : "Armoise" ,
        "Etat" : 1 ,
        "Date" : { "$date" : "2018-02-14T00:00:00.000Z"} ,
        "Type" : "Herbacée"
    },
    {
        "Sous-type" :  null  ,
        "Nom" : "Plantain" ,
        "Etat" : 1 ,
        "Date" : { "$date" : "2018-02-14T00:00:00.000Z"} ,
        "Type" : "Herbacée"
    },
```

Structure de base du JSON est une paire clef-valeur (key-value) : `"Nom": "Armoise"`

On distingue les valeurs atomiques et les valeurs complexes (construites)

- Valeurs atomiques :
  - chaînes de caractères (entourées par les classiques guillemets) `"version": "1"`
  - nombres (entiers, flottants) `"nb_results": 16`
  - valeurs booléennes (true ou false)
- Valeur complexes :
  - Tableau `"data" : [{…},{…},{…},]`
  - Objet `"date" : {"$date":"2018-02-14T00 :00 :00.000Z"}`

Pour travailler avec le format JSON, Processing va utiliser les objets de type ***JSONObject*** ou ***JSONArray.***

Exemple :

```
JSONObject jsonPollens;

void setup()
{
  jsonPollens=loadJSONObject("Alertes_pollens_nantes.json");
  //println(jsonPollens);
  JSONArray donnees=jsonPollens.getJSONArray("data");
  JSONObject element_0=donnees.getJSONObject(0);
  String espece=element_0.getString("Nom");
  String t=element_0.getString("Type");
  println(espece+"   "+t);
}
```

Travail n°6 :

Recopiez le code ci-dessous dans un sketch Processing. Testez et analysez le code à l'aide de la documentation Processing (https://processing.org/reference/JSONObject.html)

Réalisez ensuite un programme permettant d'afficher dans la console processing le nom de toutes les espèces allergisantes ainsi que leur type (Herbacée ou Arbre).

Vous utiliserez obligatoirement un algorithme avec une structure itérative (boucle for).

Pour vous aider : la méthode ***size()*** renvoie le nombre d'éléments d'un JSONArray (utilisation : https://processing.org/reference/JSONArray_size_.html)

La méthode loadJSONObject() accepte aussi une URL afin de récupérer les données en ligne. Ici nous pouvons utiliser l'URL suivante :
https://data.nantes.fr/api/publication/24440040400129_APL_APL_00254/Alertes_pollens_nantes_STBL/content

Travail n°6 bis :

Reprendre le programme précédent en utilisant l'URL ci-dessus comme source de données JSON.