# Séance du 02 février 2023

Ce TD regroupe 2 étapes, et son évaluation est prévue pour la **mi-mars**. L'objectif est de mettre en oeuvre le calcul des *itemsets* par l'algorithme proposé par R Agrawal & R. Srikant, ainsi que le calcul des règles d'association qui en découle. La première étapes va être la création de deux classes (voir TP01_part1):

1. la classe `Apriori` va s'occuper du calcul des *itemsets fréquents*,
2. la classe `Arules` s'occupera de la génération des règles d'association

La seconde étape (voir TP01_part2) va être l'exploitation de ces classes sur différents fichiers de données. Même s'il s'agit de fichiers `csv` la syntaxe n'est pas uniforme, ce qui va nécessiter des opérations de pré-traitements et de post-traitements.

## Informations générales

Vous n'avez pas de liberté sur le codage des classes, ni sur les structures choisies. La programmation se fait en python 3.7+, avec une approche objets. L'évaluation est faite en fonction de votre taux de réussite aux tests fonctionnels qui sont fournis.

Il y a une fiche indépendante pour chaque étape. Pour faciliter la lisibilité la première fiche a été scindée en 2 parties, une pour chaque classe. Les deux parties sont rédigées, et hormis les corrections typographiques, elles ne devraient pas évoluer. La fiche pour la seconde étape (exploitation) est en cours de rédaction et sera publiée sur moodle d'ici la fin des vacances de février.

### structure de données

- Un article (*item*) est représenté par un entier $> 0$.
- Un ensemble d'articles (*itemset*) est représenté par un *tuple* **ordonné** de manière croissante.
- Nous allons utiliser des ensembles (`set` python), listes (`list` python) et des dictionnaires (`dict` python) lorsqu'il faudra regrouper des informations.

Pour faire "court" un *tuple* est similaire à une *liste* python mais

1. on la note entre parenthèses et non entre crochets
2. elle n'est pas modifiable (pas d'ajout ni de suppression)
3. peut servir de clef dans un dictionnaire contrairement aux listes

Pour faire "court" un *dictionnaire* permet un usage très proche de la liste *python* mais

1. à chaque information est associée une clef

2. les informations ne sont pas stockées de manière contiguë
3. les modifications se font en temps presque constant
4. ne supporte pas la notion de 'tranches' (*slices*) contrairement aux listes et tuples

**Exemple**

```
l = [1,2,3] ; t = (1,2,3) ; d = {100: 1, 23: 2, 42: 3} # une liste, un
tuple et un dictionnaire
print(l[2] == t[2]) # True
print(d[42] == t[2]) # True
l[1] = 17 # l devient [1, 17, 3]
t[1] = 13 # provoque une erreur t non modifiable
d[100] = 11 # d devient {100: 11, 23: 2, 42: 3}
l.append(1) # l devient [1, 17, 3, 1]
d['toto'] = t # d devient {100: 11, 23: 2, 42: 3, 'toto': (1, 2, 3)}
print(len(l) == len(d)) # True
print(d.keys()) # affiche dict_keys([100, 23, 42, 'toto'])
print(d.values()) # affiche dict_values([11, 2, 3, (1,2,3)])
```

# Installation conseillée

Je vous encourage à créer un nouveau répertoire que je vais dénommer `TOTO` (vous pouvez/devriez choisir un nom plus approprié). Vous allez récupérer l'archive `panier.zip` qui se trouve sur moodle. **Attention** cette archive va être modifiée pendant les semaines à venir, il faudra donc vérifier régulièrement que vous disposez de la dernière version.

Pour le codage, je vous encourage à avoir un fichier par classe, et un fichier séparé pour mener vos propres tests.

## archive `panier.zip`

Lorsque vous allez la décompresser elle va mettre dans le répertoire un fichier `main_tests.py` et va créer 3 sous-répertoires *data*, *tools* & *tests*. Il est inutile d'aller regarder dedans pour le moment. La seule chose dont vous avez besoin est le fichier `main_tests.py` qui va vous permettre de vérifier que le code que vous produisez est conforme à la demande. Le répertoire *data* contient des petits exemples (sample_x.csv) et 4 fichiers plus conséquents pour lesquels un pré-traitement sera nécessaire, nous en reparlerons dans la seconde partie du projet (voir TP01_part2)

## mise en place

Vous écrirez vos codes dans des fichiers qui seront dans le répertoire **TOTO** (vos fichiers contiendront les classes de ce TP). Après chaque méthode ajoutée, je vous encourage à lancer le fichier `main_tests.py` qui vous demandera de saisir le nom de votre fichier à

tester et qui vous demandera quelle série de tests vous souhaitez passer. Pour répondre *NON* il suffit d'appuyer sur la touche <entrée> pour répondre *OUI* vous pouvez utiliser un 'o' (la lettre, majuscule ou minuscule) ou un 'y' (lettre majuscule ou minuscule) ou un '0' (le chiffre zéro).
Si vous utilisez *spyder* je vous encourage à créer un nouveau projet (menu 'project') et à indiquer le répertoire **TOTO** comme lieu du projet.