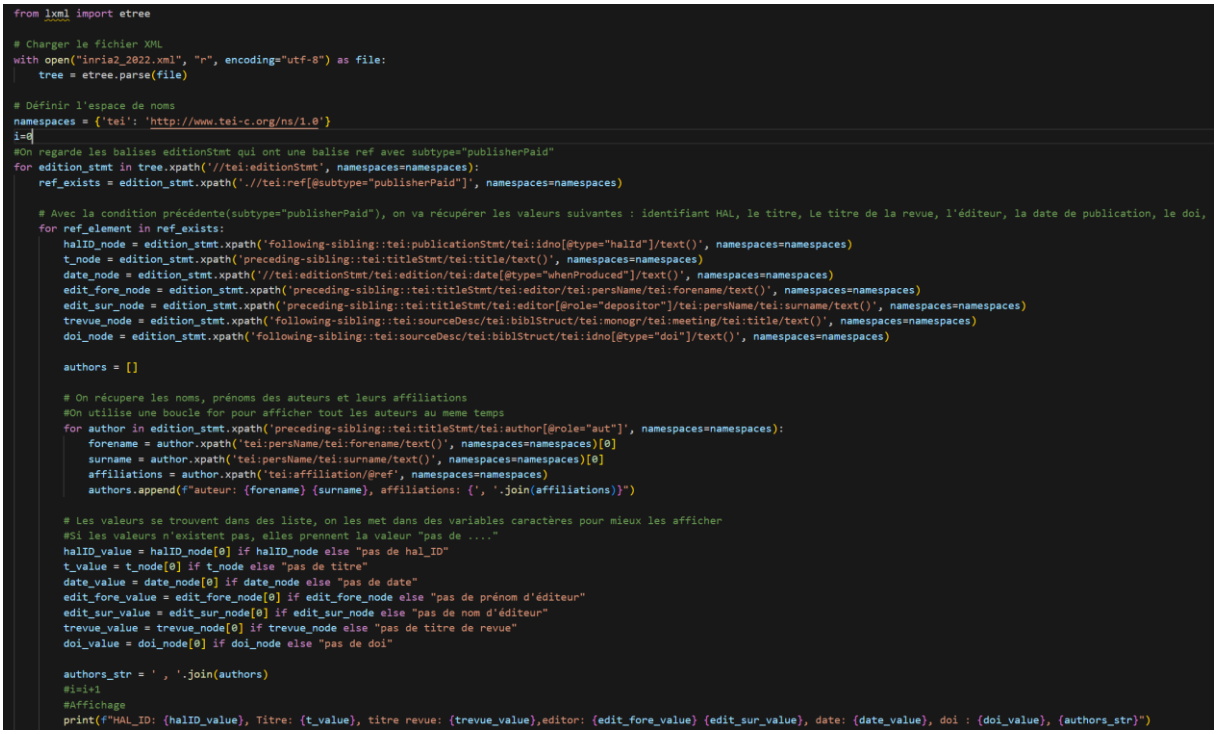
# Une présentation d’une réalisation

Dans le cadre professionnel, en octobre, je découvert deux nouvelles manières de faire de la fouille de données qui sont : les requêtes API et l’extraction de données dans des fichiers XML. Le pôle Acquisition numérique avait besoin de savoir si les publications qu’ils avaient achetées se trouvaient bien sur notre plateforme et avoir des informations supplémentaires pour négocier de futurs contrats avec des éditeurs.

L’objectif était de récupérer des informations sur les publications qui contiennent la mention " : L’institution a financé les frais de publication pour que cet article soit en libre accès". Cette information ne peut pas se trouver avec une requête API, je dois donc faire un script python pour automatiser cette tâche. Je recherche donc toutes les informations importantes à récupérer selon la structure du fichier XML et je les formate pour les écrire dans la console python pour une lecture simplifiée.

Dans ce projet, j’ai approfondi mes connaissances en XML et sa structure. J’ai également appris à respecter les contraintes que mes collègues pouvaient me donner pour le rendu final.



```python
from lxml import etree

# Charger le fichier XML
with open("inria2_2022.xml", "r", encoding="utf-8") as file:
    tree = etree.parse(file)

# Définir l'espace de noms
namespaces = {'tei': 'http://www.tei-c.org/ns/1.0'}
i=0
#On regarde les balises editionStmt qui ont une balise ref avec subtype="publisherPaid"
for edition_stmt in tree.xpath('//tei:editionStmt', namespaces=namespaces):
    ref_exists = edition_stmt.xpath('.//tei:ref[@subtype="publisherPaid"]', namespaces=namespaces)

    # Avec la condition précédente(subtype="publisherPaid"), on va récupérer les valeurs suivantes : identifiant HAL, le titre, Le titre de la revue, l'éditeur, la date de publication, le doi,
    for ref_element in ref_exists:
        halID_node = edition_stmt.xpath('following-sibling::tei:publicationStmt/tei:idno[@type="halId"]/text()', namespaces=namespaces)
        t_node = edition_stmt.xpath('preceding-sibling::tei:titleStmt/tei:title/text()', namespaces=namespaces)
        date_node = edition_stmt.xpath('//tei:editionStmt/tei:edition/tei:date[@type="whenProduced"]/text()', namespaces=namespaces)
        edit_fore_node = edition_stmt.xpath('preceding-sibling::tei:titleStmt/tei:editor/tei:persName/tei:forename/text()', namespaces=namespaces)
        edit_sur_node = edition_stmt.xpath('preceding-sibling::tei:titleStmt/tei:editor[@role="depositor"]/tei:persName/tei:surname/text()', namespaces=namespaces)
        trevue_node = edition_stmt.xpath('following-sibling::tei:sourceDesc/tei:biblStruct/tei:monogr/tei:meeting/tei:title/text()', namespaces=namespaces)
        doi_node = edition_stmt.xpath('following-sibling::tei:sourceDesc/tei:biblStruct/tei:idno[@type="doi"]/text()', namespaces=namespaces)

        authors = []

        # On récupere les noms, prénoms des auteurs et leurs affiliations
        #On utilise une boucle for pour afficher tout les auteurs au meme temps
        for author in edition_stmt.xpath('preceding-sibling::tei:titleStmt/tei:author[@role="aut"]', namespaces=namespaces):
            forename = author.xpath('tei:persName/tei:forename/text()', namespaces=namespaces)[0]
            surname = author.xpath('tei:persName/tei:surname/text()', namespaces=namespaces)[0]
            affiliations = author.xpath('tei:affiliation/@ref', namespaces=namespaces)
            authors.append(f"auteur: {forename} {surname}, affiliations: {', '.join(affiliations)}")

        # Les valeurs se trouvent dans des liste, on les met dans des variables caractères pour mieux les afficher
        #Si les valeurs n'existent pas, elles prennent la valeur "pas de ...."
        halID_value = halID_node[0] if halID_node else "pas de hal_ID"
        t_value = t_node[0] if t_node else "pas de titre"
        date_value = date_node[0] if date_node else "pas de date"
        edit_fore_value = edit_fore_node[0] if edit_fore_node else "pas de prénom d'éditeur"
        edit_sur_value = edit_sur_node[0] if edit_sur_node else "pas de nom d'éditeur"
        trevue_value = trevue_node[0] if trevue_node else "pas de titre de revue"
        doi_value = doi_node[0] if doi_node else "pas de doi"

        authors_str = ' , '.join(authors)
        #i=i+1
        #Affichage
        print(f"HAL_ID: {halID_value}, Titre: {t_value}, titre revue: {trevue_value},editor: {edit_fore_value} {edit_sur_value}, date: {date_value}, doi : {doi_value}, {authors_str}")
```