```
# -*- coding: utf-8 -*-
"""
Created on Thu Nov 21 16:13:33 2024

@author: gsagot01
"""

import json
import pandas as pd
import html
from datetime import datetime
import openpyxl
from openpyxl.utils import get_column_letter
from openpyxl.styles import Alignment, Font, PatternFill, Border, Side

# Fonction pour formater les dates et heures
def format_date_time(date_time_str):
    try:
        date_time = datetime.strptime(date_time_str, "%Y-%m-%dT%H:%M:%S%z")
        return date_time.strftime("%d/%m/%Y"), date_time.strftime("%H:%M")
    except Exception:
        return "", ""

# Charger le fichier JSON
try:
    with open('que-faire-a-paris.json', 'r', encoding='utf-8') as file:
        data = json.load(file)
except FileNotFoundError:
```

```
    print("Erreur : Le fichier 'que-faire-a-paris.json' est introuvable.")
    data = []
except json.JSONDecodeError:
    print("Erreur : Le fichier JSON est mal formaté.")
    data = []

# Colonnes pour le fichier Excel
columns = [
    "ID", "URL de l'événement", "Titre de l'événement", "Résumé", "Description complète",
    "Mots clés", "Période (dates et heures)", "Nom du lieu", "Adresse", "Code Postal", "Ville",
    "Coordonnées géographiques", "Accès PMR", "Accès mal voyant", "Accès mal entendant",
    "Transports disponibles", "Nom du contact", "Téléphone du contact", "Email du contact",
    "URL du contact", "Type d'accès", "Détail du prix", "Image de couverture (URL)"
]

# Préparer les données pour le fichier Excel
rows = []

if data:  # Vérifier qu'il y a des données à traiter
    for record in data:
        # Vérifiez si chaque élément est un dictionnaire
        if not isinstance(record, dict):
            print("Élément inattendu dans la liste :", record)
            continue

        # Extraire les champs directement
        date_start, time_start = format_date_time(record.get('date_start', ''))
```

```
date_end, time_end = format_date_time(record.get('date_end', ''))

# Préparer les valeurs en tenant compte des valeurs nulles
description = record.get('description', '')
if description:
    if len(description) > 200:
        description = description[:200] + "..."
    description = html.unescape(description.strip())
else:
    description = ""

transport = record.get('transport', '')
if transport:
    transport = transport.replace('\n', ' ').replace('<br>', '\n').strip()
else:
    transport = ""

row = {
    "ID": record.get('id', ''),
    "URL de l'événement": record.get('url', ''),
    "Titre de l'événement": record.get('title', ''),
    "Résumé": record.get('lead_text', ''),
    "Description complète": description,
    "Mots clés": ", ".join(record.get('tags', []) or []),
    "Période (dates et heures)": f"{date_start} {time_start} - {date_end} {time_end}",
    "Nom du lieu": record.get('address_name', ''),
    "Adresse": record.get('address_street', ''),
    "Code Postal": record.get('address_zipcode', ''),
```

```
        "Ville": record.get('address_city', ''),
        "Coordonnées géographiques": ", ".join(map(str, record.get('lat_lon', []) or [])),
        "Accès PMR": "Oui" if record.get('pmr') else "Non",
        "Accès mal voyant": "Oui" if record.get('blind') else "Non",
        "Accès mal entendant": "Oui" if record.get('deaf') else "Non",
        "Transports disponibles": transport,
        "Nom du contact": record.get('contact_name', ''),
        "Téléphone du contact": record.get('contact_phone', ''),
        "Email du contact": record.get('contact_mail', ''),
        "URL du contact": record.get('contact_url', ''),
        "Type d’accès": record.get('access_type', ''),
        "Détail du prix": record.get('price_type', ''),
        "Image de couverture (URL)": record.get('cover_url', '')
    }

    rows.append(row)

# Convertir les données en DataFrame et sauvegarder au format Excel
excel_file = 'que-faire-a-paris.xlsx'

if rows:
  df = pd.DataFrame(rows, columns=columns)
  df.to_excel(excel_file, index=False, engine='openpyxl')

  # Appliquer le style au fichier Excel
  wb = openpyxl.load_workbook(excel_file)
  sheet = wb.active
```

```
# Gel des en-têtes
sheet.freeze_panes = "A2"

# Style des en-têtes
header_font = Font(bold=True, color="FFFFFF", name="Calibri")
header_fill = PatternFill(start_color="4F81BD", end_color="4F81BD", fill_type="solid")
header_alignment = Alignment(horizontal="center", vertical="center")
thin_border = Border(
    left=Side(style="thin"), right=Side(style="thin"),
    top=Side(style="thin"), bottom=Side(style="thin")
)

# Appliquer un style aux en-têtes
for col_num, column_title in enumerate(columns, 1):
    cell = sheet.cell(row=1, column=col_num)
    cell.value = column_title
    cell.font = header_font
    cell.fill = header_fill
    cell.alignment = header_alignment
    cell.border = thin_border
    sheet.row_dimensions[1].height = 25  # Ajuster la hauteur des en-têtes

# Ajuster automatiquement les tailles des colonnes
for column in sheet.columns:
    max_length = 0
    column_letter = get_column_letter(column[0].column)
    for cell in column:
        if cell.value:
```

```
            max_length = max(max_length, len(str(cell.value)))
        adjusted_width = max_length + 2
        sheet.column_dimensions[column_letter].width = adjusted_width

    # Appliquer des couleurs alternées pour les lignes
    fill_gray = PatternFill(start_color="F2F2F2", end_color="F2F2F2", fill_type="solid")
    for row_num, row in enumerate(sheet.iter_rows(min_row=2), start=2):
        for cell in row:
            cell.border = thin_border
            if row_num % 2 == 0:  # Ligne paire
                cell.fill = fill_gray
            cell.alignment = Alignment(horizontal="left", vertical="center")

    wb.save(excel_file)
    print(f"Le fichier Excel '{excel_file}' a bien été généré avec succès ! ")
else:
    print("Aucune donnée valide à transformer.")
```