Projet n°18 : aider un enseignant à partir en week-end

Binôme : Azzakhnini Zakaria & Martin Diego

Enseignant : M.Nectoux

Sommaire :

1) Problématique :

Il arrive qu'un enseignant n'ait pas de cours durant une journée complète, occasionnant parfois des week-end de 3 jours. L'objectif de ce projet est d'afficher, pour un enseignant donné et à partir d'une date d'analyse donnée, la date de ses prochains week-end de trois jours (ou plus).

Cahier des charges :

• la date d'analyse doit être paramétrable ;
• le nom de l'enseignant doit être paramétrable ;
• le résultat sera affiché :
    - sous forme d'un tableau, au format csv ou xlsx ou pdf ;
    - ET sous forme d'une frise chronologique (format libre, par exemple .png).

2) Conversion de calendrier :

On possède de base un fichier ADECal.ics qui contient l'intégralité des cours organisés l'année dernière dans le département RT de l'IUT. Pour pouvoir traiter les données de ce fichier, on doit le convertir sous csv.

```
if os.path.isfile("Calendar.csv")==False:
    from csv_ical import Convert
    convert=Convert()
    convert.CSV_FILE_LOCATION='Calendar.csv'
    convert.SAVE_LOCATION='ADECal.ics'
    convert.read_ical(convert.SAVE_LOCATION)
    convert.make_csv()
    convert.save_csv(convert.CSV_FILE_LOCATION)
```

Ce script nous permet de convertir le fichier ics en csv.

3) Saisie utilisateur :

L'utilisateur est invité à entrer le nom de l'enseignant souhaité et une date à partir de laquelle les analyses commencent.

```
nomEnseignant=input("Entrez le nom de l'enseignant pour lequel effectuer les recherches (sous la forme NOM PRENOM) : ")
dateInput=input("Entrez la date à partir de laquelle vous voulez effectuer les recherches sous la forme (JJ MM AAAA) : ")
```

4) Traitement des données du calendrier

Ce script lit le fichier csv et extrait les éléments correspondant à l'enseignant indiqué et à la date donnée.

```
with open("Calendar.csv", newline='\n', encoding='utf-8') as Calendar:
    reader=csv.reader(Calendar, delimiter=',')
    for row in reader:
        if nomEnseignant in row[3]:
            date_debut=datetime.strptime(row[1], '%Y-%m-%d %H:%M:%S+00:00')
            date_fin=datetime.strptime(row[2], '%Y-%m-%d %H:%M:%S+00:00')
            if date_debut>userDate:
                calendrier.append([row[0], row[3], date_debut, date_fin, row[4]])
```

Lorsque les éléments correspondent, on les ajoute dans la liste calendrier.

5) Identification des weekends de 3 jours ou plus

```
weekendTroisJours=[]
#On parcourt les jours du calendrier pour trouver les week-ends de 3 jours ou plus qu'on ajoutera dans la liste weekendTroisJours
for jour in range(1, len(calendrier)):
    duree=calendrier[jour][2]-calendrier[jour-1][3]  #Entre le début du prochain cours [2] et la fin du précédent [3]
    if duree.days>=3:  #Si la durée est supérieure ou égale à trois jours
        debutWeekend=calendrier[jour-1][3]
        finWeekend=calendrier[jour][2]
        dureeWeekendJours=duree.days
        dureeEnHeure=duree.total_seconds()
        dureeWeekendHeure=str(int(dureeEnHeure//3600)) + ":" + str(int((dureeEnHeure%3600)/60))
        weekendTroisJours.append([debutWeekend, finWeekend, dureeWeekendJours, dureeWeekendHeure])
```

Lorsqu'on identifie un weekend de 3 jours ou plus on l'ajoute dans la liste weekendtroisJours.

6) Affichage des résultats

```
#Affiche les weekend de trois trouvés
for element in weekendTroisJours:
    print(f"Week-end de {element[2]} jours : Du {element[0].strftime('%d/%m/%Y %H:%M')} au {element[1].strftime('%d/%m/%Y %H:%M')}")

#On rentre les résultats dans un fichier .csv et affiche les résultats sous forme d'une frise chronologique
fig, ax=plt.subplots()
fig.set_size_inches(18, 10)
plt.get_current_fig_manager().full_screen_toggle()
#Ecrit les résultats dans un fichier CSV
with open(nomEnseignant+'_Week-end.csv', 'w', newline='', encoding='utf-8') as fichiercsv:
    writer=csv.writer(fichiercsv)
    writer.writerow(['Du', 'Au', 'Durée (jours)', 'Durée (heures)'])
    #Affiche le résultat sous forme de frise chronologique
    for element in weekendTroisJours:
        dateDebutFormat=element[0].strftime("%d/%m/%y %H:%M")
        dateFinFormat=element[1].strftime("%d/%m/%y %H:%M")
        writer.writerow([dateDebutFormat, dateFinFormat, element[2], element[3]])
        ax.plot([element[0], element[1]], [0, 0], color="blue", linewidth=10)
        ax.annotate(element[0].strftime("%d"), (element[0], 0), textcoords="offset points", xytext=(0, 10), ha='center', fontsize=8)
        ax.annotate(element[1].strftime("%d"), (element[1], 0), textcoords="offset points", xytext=(0, 10), ha='center', fontsize=8)
fichiercsv.close()
ax.set_xlim(calendrier[0][3], calendrier[-1][2])
ax.xaxis_date()
ax.xaxis.set_major_formatter(DateFormatter("%B %Y"))
ax.grid(which='both', axis='x', linestyle='--', color='gray', linewidth=1)
ax.set_title(f"Frise chronologique des week-end de 3 jours ou plus de {nomEnseignant}")
ax.set_xlabel("Temps")
ax.set_yticks([])
plt.savefig(nomEnseignant+"_Week-end.png")
plt.show()
```

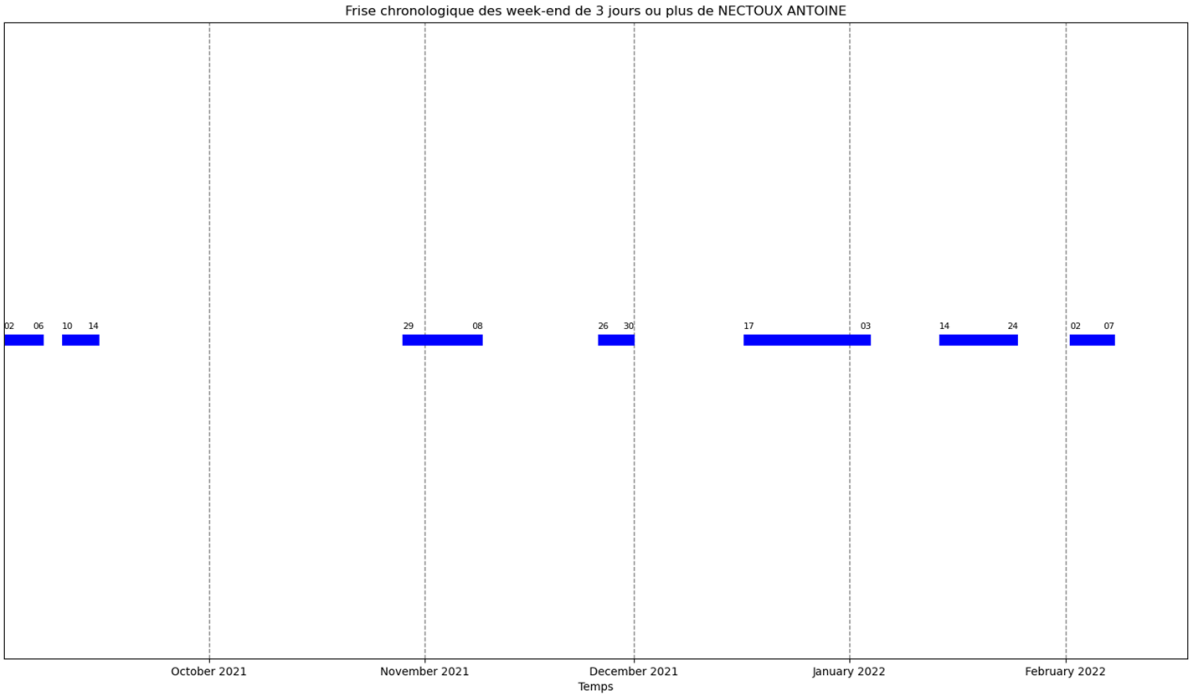
On affiche d'abord le résultat dans le terminal pour avoir une première impression, puis on créer on fichier csv pour afficher le résultat sous forme de tableau et on créer un graphique sous forme d'une frise chronologique pour mieux visualiser le résultat.

7) Exécution

Voici un exemple d’une exécution du programme :

```
Entrez le nom de l'enseignant pour lequel effectuer les recherches (sous la forme NOM PRENOM) : NECTOUX ANTOINE
Entrez la date à partir de laquelle vous voulez effectuer les recherches sous la forme (JJ MM AAAA) : 01 01 2021
Week-end de 4 jours : Du 02/09/2021 08:15 au 06/09/2021 12:00
Week-end de 3 jours : Du 10/09/2021 16:00 au 14/09/2021 12:00
Week-end de 9 jours : Du 29/10/2021 15:30 au 08/11/2021 13:00
Week-end de 3 jours : Du 26/11/2021 15:30 au 30/11/2021 07:15
Week-end de 16 jours : Du 17/12/2021 14:30 au 03/01/2022 07:15
Week-end de 9 jours : Du 14/01/2022 15:30 au 24/01/2022 09:15
Week-end de 4 jours : Du 02/02/2022 11:15 au 07/02/2022 07:15
```

Les résultats sont d’abord affichés dans le terminal.



Une frise chronologique est créée pour visualiser les résultats.

	A	B	C	D
1	Du	Au	Durée (jours)	Durée (heures)
2	02/09/21 08:15	06/09/21 12:00	4	99:45
3	10/09/21 16:00	14/09/21 12:00	3	92:0
4	29/10/21 15:30	08/11/21 13:00	9	237:30
5	26/11/21 15:30	30/11/21 07:15	3	87:45
6	17/12/21 14:30	03/01/22 07:15	16	400:45
7	14/01/22 15:30	24/01/22 09:15	9	233:45
8	02/02/22 11:15	07/02/22 07:15	4	116:0

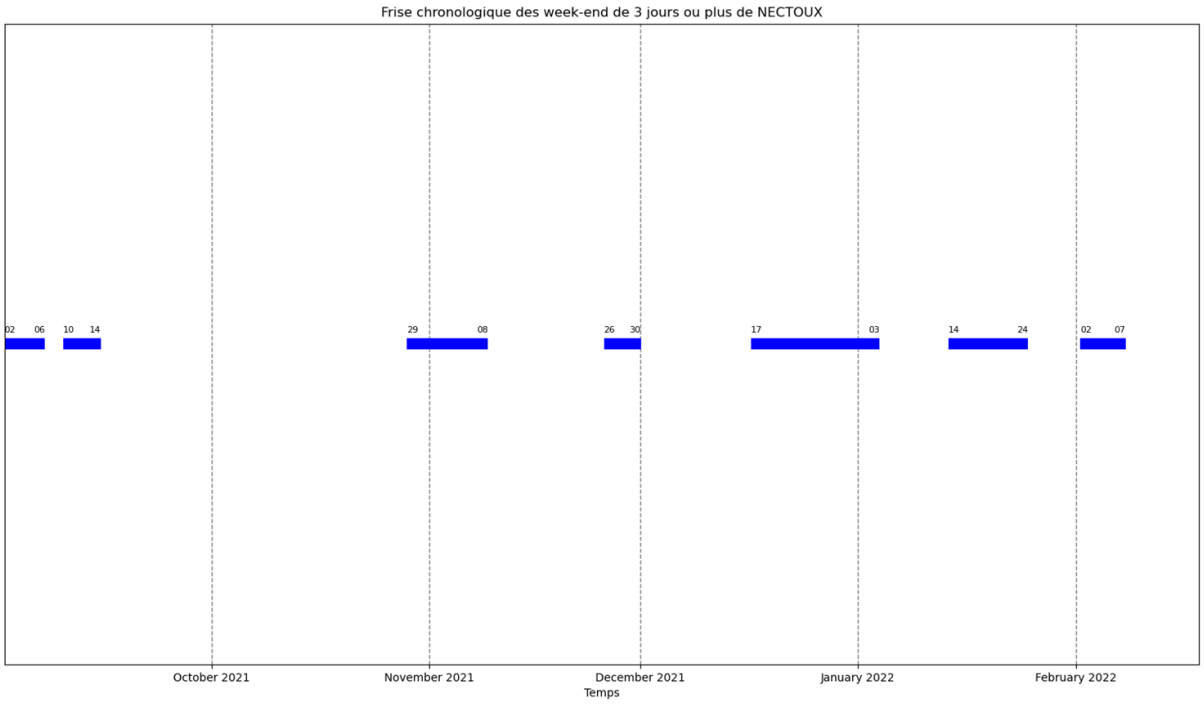
Un fichier csv est également créer pour donner le résultat sous forme de tableau.

```
Entrez le nom de l'enseignant pour lequel effectuer les recherches (sous la forme NOM PRENOM) : nectou
Entrez la date à partir de laquelle vous voulez effectuer les recherches sous la forme (JJ MM AAAA) : 01 01 2021
Nom de l'enseignant incorrect ou date en dehors du champs d'analyse
dm609764@aux-r2inf2-008:~/sae/Projet$ python3 main.py
Entrez le nom de l'enseignant pour lequel effectuer les recherches (sous la forme NOM PRENOM) : NECTOUX ANTOINE
Entrez la date à partir de laquelle vous voulez effectuer les recherches sous la forme (JJ MM AAAA) : 01 01 2025
Nom de l'enseignant incorrect ou date en dehors du champs d'analyse
```

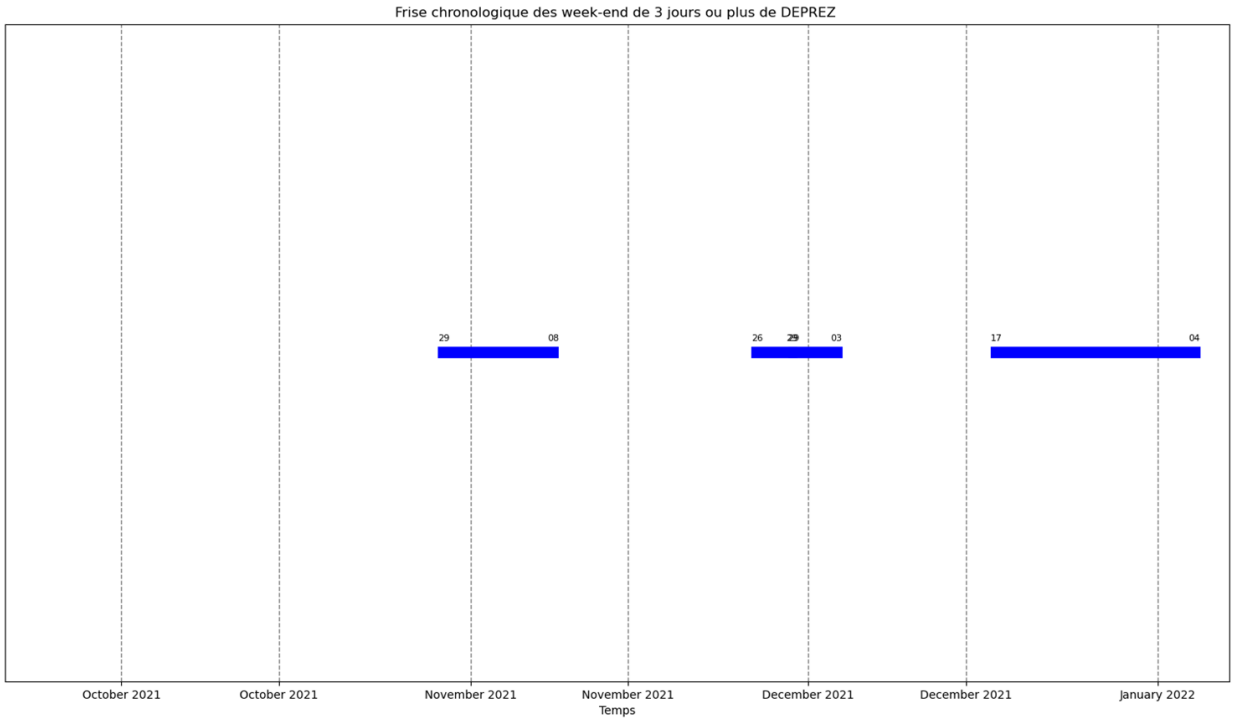
On remarque si le nom de l’enseignant saisie est incorrect, ou que la date n’est pas dans le fichier csv, cela affiche un message d’erreur.

Voici quelques exemples des weekend de 3 jours ou plus à partir du 01/01/21 :

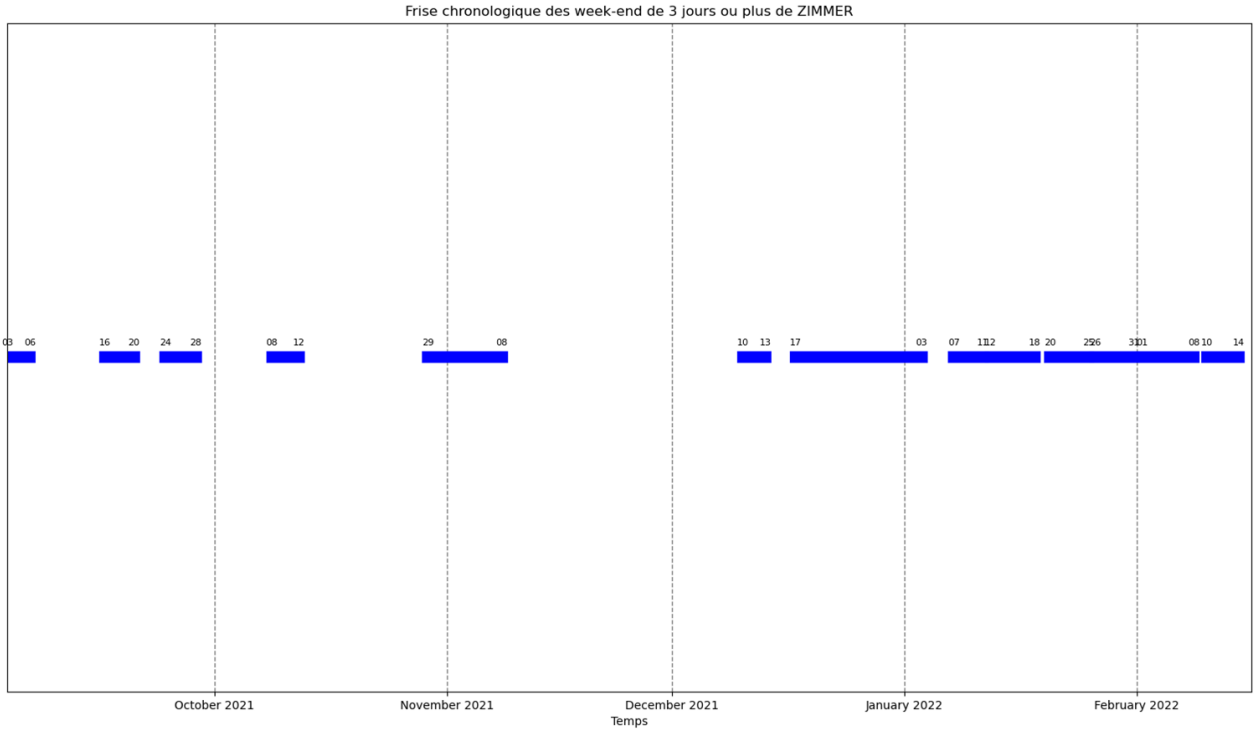
NECTOUX :



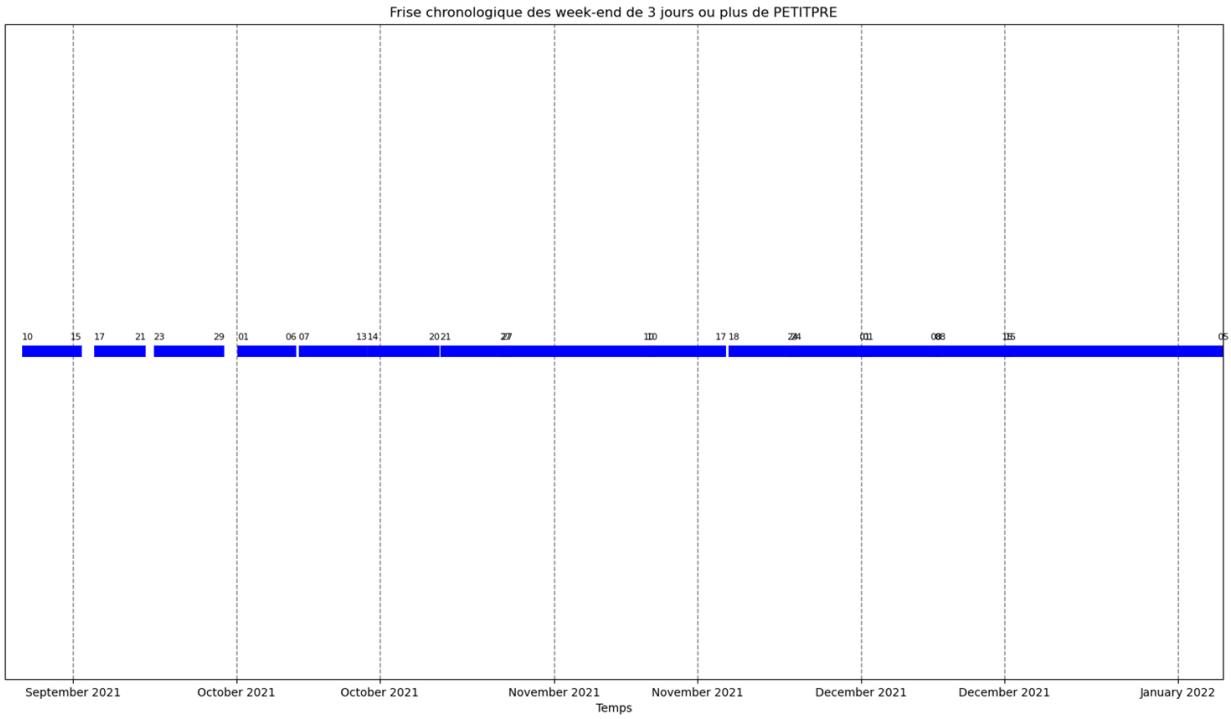
DEPREZ :



ZIMMER :



PETITPRE :


Frise chronologique des week-end de 3 jours ou plus de PETITPRE
10
15
17
21
23
29
01
06 07
13 14
20 21
17 18
05
September 2021
October 2021
October 2021
November 2021
November 2021
December 2021
December 2021
January 2022
Temps

8) Synthèse

Programme complet :

```python
#AZZAKHNINI Zakaria & MARTIN Diego - SAE15 Traitement des données, Projet 18 : aider un enseignant à partir en week-end

#Importation des modules nécessaires
import os, sys
import csv
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
from matplotlib.dates import DateFormatter

#Vérifie si le fichier csv existe, sinon ca converti le fichier cis en csv
if os.path.isfile("Calendar.csv")==False:
    from csv_ical import Convert
    convert=Convert()
    convert.CSV_FILE_LOCATION='Calendar.csv'
    convert.SAVE_LOCATION='ADECal.ics'
    convert.read_ical(convert.SAVE_LOCATION)
    convert.make_csv()
    convert.save_csv(convert.CSV_FILE_LOCATION)

#Programme principal

#On créer une liste calendrier vide dans lequel seront les données du fichier csv à analyser
calendrier=[]
#Demande à l'utilisateur d'entrée le nom de l'enseignant et la date à partir de laquelle faire les recherches
nomEnseignant=input("Entrez le nom de l'enseignant pour lequel effectuer les recherches (sous la forme NOM (PRENOM)) : ")
dateInput=input("Entrez la date à partir de laquelle vous voulez effectuer les recherches sous la forme (JJ MM AAAA) : ")

#Converti la date d'entrée en objet datetime
try:
    userDate=datetime.strptime(dateInput, "%d %m %Y")
except ValueError:
    print("Format de date incorrect.")
    sys.exit()

#On ajoute maitenant dans le calendrier uniquement les cours de l'enseignant après la date renseignée
with open("Calendar.csv", newline='\n', encoding='utf-8') as Calendar:
    reader=csv.reader(Calendar, delimiter=',')
    for row in reader:
        if nomEnseignant in row[3]:
            date_debut=datetime.strptime(row[1], '%Y-%m-%d %H:%M:%S+00:00')
            date_fin=datetime.strptime(row[2], '%Y-%m-%d %H:%M:%S+00:00')
            if date_debut>userDate:
                calendrier.append([row[0], row[3], date_debut, date_fin, row[4]])

if calendrier==[]:
    print("Nom de l'enseignant incorrect ou date en dehors du champs d'analyse")
    sys.exit()

#On trie les jours de la semaine du plus ancien au plus récent
calendrier.sort(key=lambda x: x[2]) #indice 2=date_debut
```

```
#Création d'une liste vide weekendTroisJours
weekendTroisJours=[]
#On parcourt les jours du calendrier pour trouver les week-ends de 3 jours ou plus qu'on ajoutera dans la liste weekendTroisJours
for jour in range(1, len(calendrier)):
    duree=calendrier[jour][2]-calendrier[jour-1][3]  #Entre le début du prochain cours [2] et la fin du précédent [3]
    if duree.days>=3:  #Si la durée est supérieure ou égale à trois jours
        debutWeekend=calendrier[jour-1][3]
        finWeekend=calendrier[jour][2]
        dureeWeekendJours=duree.days
        dureeEnHeure=duree.total_seconds()
        dureeWeekendHeure=str(int(dureeEnHeure//3600)) + ":" + str(int((dureeEnHeure%3600)/60))
        weekendTroisJours.append([debutWeekend, finWeekend, dureeWeekendJours, dureeWeekendHeure])

#Affiche les weekend de trois trouvés
for element in weekendTroisJours:
    print(f"Week-end de {element[2]} jours : Du {element[0].strftime('%d/%m/%Y %H:%M')} au {element[1].strftime('%d/%m/%Y %H:%M')}")

#On rentre les résultats dans un fichier .csv et affiche les résultats sous forme d'une frise chronologique
fig, ax=plt.subplots()
fig.set_size_inches(18, 10)
plt.get_current_fig_manager().full_screen_toggle()
#Ecrit les résultats dans un fichier CSV
with open(nomEnseignant+'_Week-end.csv', 'w', newline='', encoding='utf-8') as fichiercsv:
    writer=csv.writer(fichiercsv)
    writer.writerow(['Du', 'Au', 'Durée (jours)', 'Durée (heures)'])
    #Affiche le résultat sous forme de frise chronologique
    for element in weekendTroisJours:
        dateDebutFormat=element[0].strftime("%d/%m/%y %H:%M")
        dateFinFormat=element[1].strftime("%d/%m/%y %H:%M")
        writer.writerow([dateDebutFormat, dateFinFormat, element[2], element[3]])
        ax.plot([element[0], element[1]], [0, 0], color="blue", linewidth=10)
        ax.annotate(element[0].strftime("%d"), (element[0], 0), textcoords="offset points", xytext=(0, 10), ha='center', fontsize=8)
        ax.annotate(element[1].strftime("%d"), (element[1], 0), textcoords="offset points", xytext=(0, 10), ha='center', fontsize=8)
fichiercsv.close()
ax.set_xlim(calendrier[0][3], calendrier[-1][2])
ax.xaxis_date()
ax.xaxis.set_major_formatter(DateFormatter("%B %Y"))
ax.grid(which='both', axis='x', linestyle='--', color='gray', linewidth=1)
ax.set_title(f"Frise chronologique des week-end de 3 jours ou plus de {nomEnseignant}")
ax.set_xlabel("Temps")
ax.set_yticks([])
plt.savefig(nomEnseignant+"_Week-end.png")
plt.show()
```

Ce programme offre donc une solution pratique pour les enseignants souhaitant connaître facilement leur long weekends. Les résultats sous forme de frise chronologique et tableau permettent une bonne visualisation de leur weekends de 3 jours au plus. Il faut cependant remarquer que les vacances scolaires sont prises en comptes.