

SAE15 Traitement des données

TP Projet

Etape 2 : pré-traitement

voir [readme](#)

Etape 3 : traitement des données

Nous importons plusieurs modules qui peuvent être utilisés pour lire et écrire des fichiers CSV, effectuer des opérations avec des expressions régulières, et créer des graphiques à l'aide de la bibliothèque Matplotlib. Le reste du code qui suit ces importations déterminera l'utilisation spécifique de ces modules.

```
import csv
import re
import matplotlib.pyplot as plt
```

Ce morceau de code lit un fichier CSV situé à l'emplacement spécifié et stocke son contenu dans une liste appelée table.

Voici une explication ligne par ligne :

```
table = []
with
open('/home/Etudiants/RT/BUT-RT-1/lg409538/SAE105_Projet/ADECal/ADECal
.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    for ligne in reader:
        table.append(ligne)
```

On peut voir ci-dessous le résultat de la lecture du fichier.

```
FRANK\n(Exported :05/01/2022 11:04)', 'RT-Salle-TD4']], ['Transmission large bande', '2022-01-06 08:15:00+00:00', '2022-01-06 11:15:00+00:00', 'RT2Dijkstra\nAZZOUNI SOUMAYA\n(Exported :05/01/2022 11:04)', 'RT-Labo Informatique 1'], ['TP Maintien de la disponibilité', '2022-01-07 14:30:00+00:00', '2022-01-07 16:30:00+00:00', 'LP-CyberSécurité\nEMERY FRANK\n(Exported :05/01/2022 11:04)', 'RT-Salle-TD4']], ['Principes et architecture des réseaux', '2021-11-09 13:00:00+00:00', '2021-11-09 16:00:00+00:00', 'RT1Turing\nDEPREZ JEAN-LUC\n(Exported :05/01/2022 11:04)', 'RT-Labo réseaux 2'], ['SAE12: Réseaux', '2021-12-14 07:15:00+00:00', '2021-12-14 11:15:00+00:00', 'RT1Shannon\nRT1Shannon2\n(Exported :05/01/2022 11:04)', 'RT-Labo Informatique 3'], ['Source s de données', '2022-02-14 09:15:00+00:00', '2022-02-14 11:15:00+00:00', 'RT1App\nVIOIX JEAN-BAPTISTE\n(Exported :05/01/2022 11:04)', 'RT-Salle Labo Visio'], ['Réseaux opérateurs', '2021-09-22 12:00:00+00:00', '2021-09-22 14:00:00+00:00', 'RT2App\nRT2Hamming\nRT2Dijkstra\nDEJUSSIEU JEAN LOUIS\n(Exported :05/01/2022 11:04)', 'RT-Amphi,RT-Salle-TD4']]
```

Ce morceau de code est destiné à supprimer les caractères de saut de ligne \$(\n)\$ de la quatrième colonne (index 3) de chaque ligne dans la liste table.

Plus précisément, il remplace chaque occurrence de \$(\n)\$ par une virgule (,).

```
for chaine in table:
    chaine[3] = chaine[3].replace('\n', ',')
```

On peut voir que les `\n` ne sont plus la maintenant il y a des ','.

```
RT-Salle-TD1,RT-Labo Informatique 3'], ['TP Maintien de la disponibilité', '2022-01-07 12:30:00+00:00', '2022-01-07 14:30:00+00:00', 'LP-CyberSécurité,EMERY FRANCK,(Exported :05/01/2022 11:04)', 'RT-Salle-TD4'], ['Transmission large bande', '2022-01-06 08:15:00+00:00', '2022-01-06 11:15:00+00:00', 'RT2Dijkstra,AZZOUNI SOUMAYA,(Exported :05/01/2022 11:04)', 'RT-Labo Informatique 1'], ['TP Maintien de la disponibilité', '2022-01-07 14:30:00+00:00', '2022-01-07 16:30:00+00:00', 'LP-CyberSécurité,EMERY FRANCK,(Exported :05/01/2022 11:04)', 'RT-Salle-TD4'], ['Principes et architecture des réseaux', '2021-11-09 13:00:00+00:00', '2021-11-09 16:00:00+00:00', 'RTiTuring,DEPREZ JEAN-LUC,(Exported :05/01/2022 11:04)', 'RT-Labo réseaux 2'], ['SAE12: Réseaux', '2021-12-14 07:15:00+00:00', '2021-12-14 11:15:00+00:00', 'RTIshannon1,RTIshannon2,(Exported :05/01/2022 11:04)', 'RT-Labo Informatique 3'], ['Sources de données', '2022-02-14 09:15:00+00:00', '2022-02-14 11:15:00+00:00', 'RTIApp,VIOIX JEAN-BAPTISTE,(Exported :05/01/2022 11:04)', 'RT-Salle Labo Visio'], ['Réseaux opérateurs', '2021-09-22 12:00:00+00:00', '2021-09-22 14:00:00+00:00', 'RT2App,RT2Hamming,RT2Dijkstra,DEJUSSIÉ JEAN LOUIS,(Exported :05/01/2022 11:04)', 'RT-Amphi,RT-Salle-TD4']]
```

Ce morceau de code prend une entrée utilisateur à l'aide de la fonction `input`, demande le nom d'un.e enseignant.e, puis convertit cette entrée en majuscules en utilisant la méthode `upper()`. Nous allons prendre comme exemple l'enseignante **Madame ZIMMER**

```
prof = input("Entrez le nom d'un enseignant : ")
prof = prof.upper()
```

Première Fonction :

```
def calcule_duree_prof(t: list) -> float:
    """
    Calcule la durée en heures d'un cours à partir des données d'une
    ligne du fichier CSV.

    Parameters:
        -Liste contenant les données d'une ligne du fichier CSV.

    Returns:
        -Durée du cours en heures.
    """
    date_debut = t[1]
    date_fin = t[2]
    heures_debut, minutes_debut = map(int, date_debut.split()
[1].split(':')[2])
    heures_fin, minutes_fin = map(int, date_fin.split()[1].split(':')[2])
    difference_heures = heures_fin - heures_debut
    difference_minutes = minutes_fin - minutes_debut
    duree_en_heures = difference_heures + difference_minutes / 60
    return duree_en_heures
```

```
help(calcule_duree_prof)
```

```
Help on function calcule_duree_prof in module __main__:
```

```
calcule_duree_prof(t: list) -> float
```

Calcule la durée en heures d'un cours à partir des données d'une ligne du fichier CSV.

Parameters:

-Liste contenant les données d'une ligne du fichier CSV.

Returns:

-Durée du cours en heures.

Deuxième Fonction :

```
def calculer_heures_module(table: list, prof: str) -> tuple:
    """
    Calcule le total d'heures par type de cours (CM, TD, TP) et génère
    un dictionnaire contenant les heures par module.

    Parameters:
        -Liste contenant les données du fichier CSV.
        -Nom de l'enseignant.

    Returns:
        -Un tuple contenant le dictionnaire des heures par module,
        -le total d'heures de CM, le total d'heures de TD, et le total
        d'heures de TP.
    """
    module_heures = {}
    heures_CM = 0
    heures_TD = 0
    heures_TP = 0

    for element in table:
        if prof in element[3]:
            module_heures[element[0]] = module_heures.get(element[0], 0)
            + calcule_duree_prof(element)
            if re.search("Amphi", element[4]) or
re.search("DS", element[0]):
                heures_CM += calcule_duree_prof(element)
            elif re.search("Labo", element[4]):
                heures_TP += calcule_duree_prof(element)
            elif re.search("TD", element[4]) or
re.search("CA0", element[4]):
                heures_TD += calcule_duree_prof(element)

    return module_heures, heures_CM, heures_TD, heures_TP

help(calculer_heures_module)

Help on function calculer_heures_module in module __main__:

calculer_heures_module(table: list, prof: str) -> tuple
    Calcule le total d'heures par type de cours (CM, TD, TP) et génère
```

un dictionnaire contenant les heures par module.

Parameters:

- Liste contenant les données du fichier CSV.
- Nom de l'enseignant.

Returns:

- Un tuple contenant le dictionnaire des heures par module,
- le total d'heures de CM, le total d'heures de TD, et le total d'heures de TP.

Ici nous appelons la fonction `calculer_heures_module` qui prend deux arguments, `table` et `prof`. On stocke le résultat dans une variable appelée `resultats`.

```
resultats = calculer_heures_module(table, prof)
```

Ce qui nous afficheras un dict comme montrer ci-dessous:

```
Entrez le nom d'un enseignant : zimmer
{'Analyse et traitement de données structurées': 20.0, 'Transmissions guidées': 63.0, 'Fondamentaux de la programmation': 60.0, 'Electronique': 83.0, 'Réunion stage': 2.0, 'DS Réseaux cellulaires': 2.0, 'Fibres optiques': 28.0, 'Réseaux cellulaires': 14.0, 'DS TP Electronique': 12.0, 'DS Electronique': 4.0, 'Fondamentaux de la programmation DS TP': 9.0, 'Rentrée RTI APP': 1.0, 'DS TP Blanc Electronique': 9.0, 'DS TP blanc Electronique': 3.0, 'DS Fibres optiques': 2.0, 'DS Transmissions guidées': 2.0}
```

Dans ce morceau de code, nous créons plusieurs structures de données pour organiser et stocker les résultats obtenus à partir de la fonction `calculer_heures_module`

```
heures_ = {'Heures Total de CM': resultats[1], 'Heures Total de TD': resultats[2], 'Heures Total de TP': resultats[3]}

Different_Heures = list(heures_.keys())
Nombre_heures = list(heures_.values())

modules = list(resultats[0].keys())
heures = list(resultats[0].values())
```

Different_Heures :

Cette ligne crée une liste appelée `Different_Heures` en extrayant les clés du dictionnaire `heures_`. Les clés représentent les types d'enseignement, comme "Heures Total de CM", "Heures Total de TD", et "Heures Total de TP".

Nombre_heures :

Cette ligne crée une liste appelée `Nombre_heures` en extrayant les valeurs du dictionnaire `heures_`.

Les valeurs correspondent aux heures totales associées aux types d'enseignement mentionnés ci-dessus.

modules :

Cette ligne crée une liste appelée modules en extrayant les clés du premier dictionnaire contenu dans la liste resultats.

Les clés représentent les noms des modules d'études, tels que "Analyse et traitement de données structurées", "Transmissions guidées", etc.

heures :

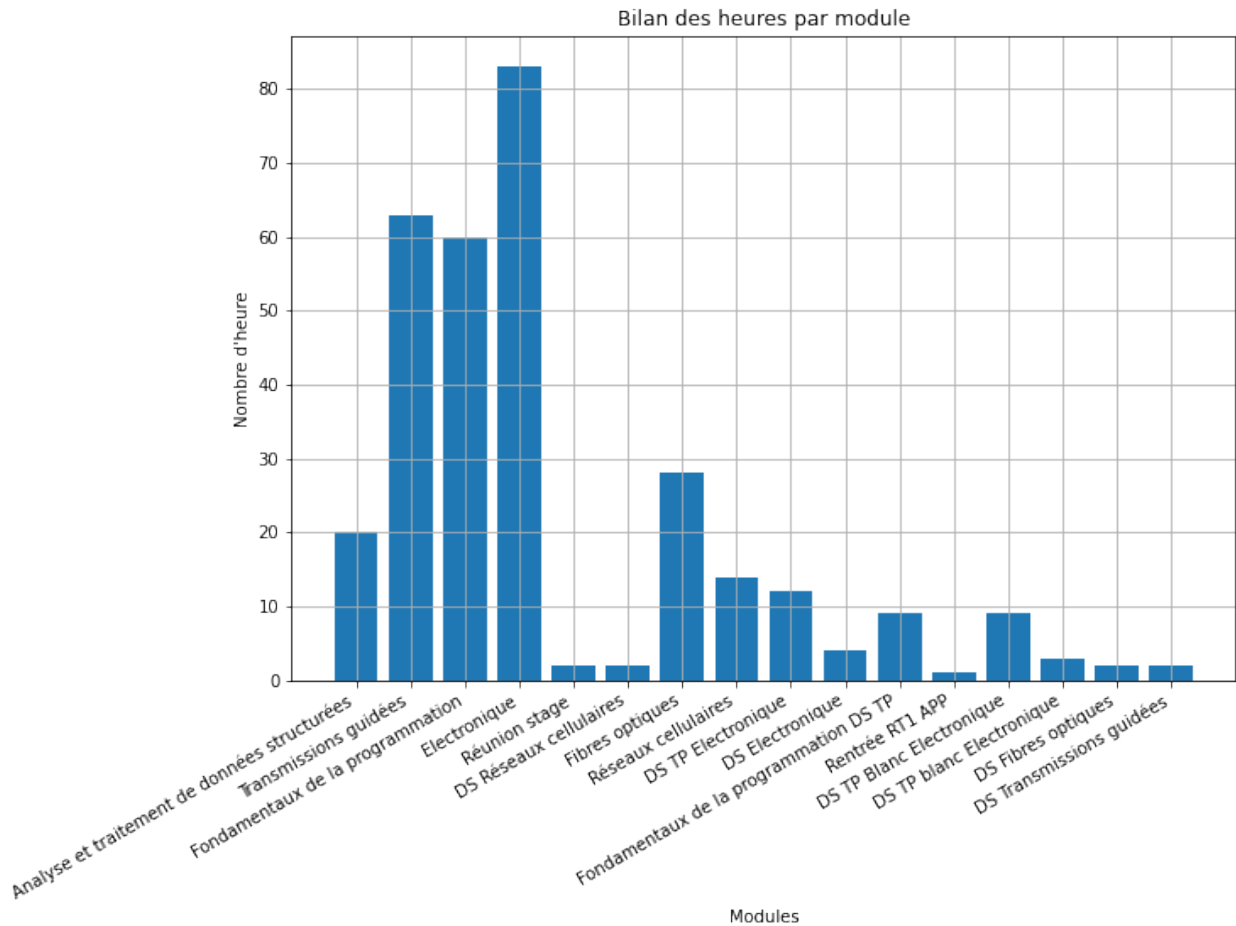
Cette ligne crée une liste appelée heures en extrayant les valeurs du premier dictionnaire contenu dans la liste resultats. Les valeurs correspondent aux nombres d'heures associés à chaque module d'études.

La variable heures_ nous renvoie un dict qui est :

```
{ 'Heures Total de CM': 47.0, 'Heures Total de TD': 114.0, 'Heures Total de TP': 153.0 }
```

Ce morceau de code utilise le module matplotlib.pyplot pour créer un diagramme à barres (bar plot) à partir des données fournies dans les listes modules et heures.

```
plt.figure(figsize=(10, 7))
plt.bar(modules, heures)
plt.title("Bilan des heures par module")
plt.xlabel("Modules")
plt.ylabel("Nombre d\'heure")
plt.xticks(rotation=30, ha='right')
plt.grid()
plt.show()
```



Ce morceau de code crée un fichier CSV appelé "bilan_heures_combined.csv" et écrit des données dans ce fichier. Dans la première boucle nous écrivons les données du premier dictionnaire, puis la deuxième boucle nous écrivons les données du deuxième dictionnaire. Tous cela avec un saut de ligne : `pythonwriter.writerow()`

Voici une explication ligne par ligne :

```
with open('bilan_heures_combined.csv', 'w', newline='') as csvfile:
    fieldnames = ['Module', 'Heures']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    writer.writeheader()

    for module, heures in resultats[0].items():
        writer.writerow({'Module': module, 'Heures': heures})

    writer.writerow({})

    for module, heures in heures_.items():
        writer.writerow({'Module': module, 'Heures': heures})
```

Module	Heures
Analyse et traitement de données structurées	20.0
Transmissions guidées	63.0
Fondamentaux de la programmation	60.0
Electronique	83.0
Réunion stage	2.0
DS Réseaux cellulaires	2.0
Fibres optiques	28.0
Réseaux cellulaires	14.0
DS TP Electronique	12.0
DS Electronique	4.0
Fondamentaux de la programmation DS TP	9.0
Rentrée RT1 APP	1.0
DS TP Blanc Electronique	9.0
DS TP blanc Electronique	3.0
DS Fibres optiques	2.0
DS Transmissions guidées	2.0
Heures Total de CM	47.0
Heures Total de TD	114.0
Heures Total de TP	153.0

Contributeurs

- [TSANGUE](#)
- [GRANVISIR-CLERC](#)