

Etude de l'utilisation des parkings de la ville de Montpellier

A noter: J'ai finalement réalisé cette SAE seul, puisque Raid Neghouche a été absent à plusieurs séances, c'est donc pour cela que j'ai préféré travailler seul pour pas me pénaliser.

Je tiens à dire que les images des courbes et du résultat du programme "fonctions maths" à la fin ne s'affiche pas dans le compte rendu en PDF, il faut voir celui en markdown ou alors voir directement les images dans le dossier captures sur mon dépôt Github .

Lien du dépôt Github:

<https://github.com/adrienvalles/SAE15>

Préparation du mini projet

Pour ce faire, j'ai utilisé les données extraites du site **open data de Montpellier** qui m'ont permis d'analyser les données .

Avant de pouvoir réaliser le mini projet, j'ai eu une séance TP afin de me préparer au mieux au traitement des données du site opendata. Pour pouvoir récupérer ces données je me suis appuyer sur les premières questions du TP qui m'a beaucoup aidé, en demandant de tester le programme.

En effet ce programme va grâce à la fonction requests, envoyer une demande http vers le site en question afin d'extraire les données pour les afficher dans un fichier texte contenant toutes les informations pour tous les parkings voiture de la Ville de Montpellier

```
PAR=[ 'FR_MTP_ANTI', 'FR_MTP_COME', 'FR_MTP_CORU', 'FR_MTP_EURO', 'FR_MTP_FOCH', 'FR_MTP_GAM
B', 'FR_MTP_GARE', 'FR_MTP_TRIA', 'FR_MTP_ARCT',
'FR_MTP_PITO', 'FR_MTP_CIRC', 'FR_MTP_SABI', 'FR_MTP_GARC', 'FR_MTP_SABL', 'FR_MTP_MOSS
', 'FR_STJ_SJLC', 'FR_MTP_MEDC', 'FR_MTP_OCCI', 'FR_CAS_VICA', 'FR_MTP_GA109', 'FR_MTP_G
A250', 'FR_CAS_CDGA', 'FR_MTP_ARCE', 'FR_MTP_POLY' ]

for i in PAR:

x=requests.get(f'https://data.montpellier3m.fr/sites/default/files/ressources/{i}.
xml')
    print(x.text)
```

Analyse de données

Ceci est un programme qui permet le suivi de l'occupation de tous les parkings de Montpellier et en affichant à la fin le pourcentage de places libres et de places occupées de toute la ville.

```
import requests
from lxml import etree
PAR=
['FR_MTP_ANTI', 'FR_MTP_COME', 'FR_MTP_CORU', 'FR_MTP_EURO', 'FR_MTP_FOCH', 'FR_MTP_GAM
B', 'FR_MTP_GARE', 'FR_MTP_TRIA', 'FR_MTP_ARCT',
'FR_MTP_PITO', 'FR_MTP_CIRC', 'FR_MTP_SABI', 'FR_MTP_GARC', 'FR_MTP_SABL', 'FR_MTP_MOSS
', 'FR_STJ_SJLC', 'FR_MTP_MEDC', 'FR_MTP_OCCI', 'FR_CAS_VICA', 'FR_MTP_GA109', 'FR_MTP_G
A250', 'FR_CAS_CDGA', 'FR_MTP_ARCE', 'FR_MTP_POLY']

y=0
w=0
p=0
for i in PAR:

x=requests.get(f'https://data.montpellier3m.fr/sites/default/files/ressources/{i}.
xml')
f1=open(f'{i}.txt','w', encoding='utf8')
f1.write(x.text)
f1.close()
tree = etree.parse(f"{i}.txt") #retirer les donnees
for user in tree.xpath("Name"): #extraire les données de name
    print('Nom du parking :',user.text)
for user1 in tree.xpath("Total"):
    print('Places totales :',user1.text)
for user2 in tree.xpath("Free"):
    print('Nombre de places libres :',user2.text)
    x=int((user1.text))-int((user2.text))
    print('Nombres de places occupées:' ,x)
y=int((user1.text))+ y
w=int((user2.text))+w
p=int((x))+ p
#print('Nombres de places occupées dans toute la ville:' ,x)
print('Nombre de Places totales de toute la ville:' ,y)
print('Nombres de places libres de toute la ville:' ,w)
print('Nombre de place occupées dans toute la ville:' ,p)
pour=int((w))*100/int(y)
opour=int((p))*100/int(y)
print('Pourcentage de places libres de toute la ville' ,round(pour,2) ,'%')
print('Pourcentage de places occupées de toute la ville' ,round(opour,2) ,'%')
```

J'ai par la suite pris l'exemple de 4 parkings afin de connaître leur pourcentages de places libres ainsi que leur taux d'occupation sur une journée pour voir l'évolution de chaque parkings et pour vérifier si ils ont la même fréquentation.

Voici le programme qui a calculé le nombres de places libres et par la suite en ajoutant une fonction , calculer un pourcentage et en déduire le taux d'occupation de chacun des 4 parkings.

```
import requests #librairie HTTP
import time
from lxml import etree

parkings=["FR_MTP_ANTI","FR_MTP_COME","FR_MTP_CORU","FR_MTP_OCCI"]
#Parkings de Montpellier

url="https://data.montpellier3m.fr/dataset/disponibilite-des-places-dans-les-
parkings-de-montpellier-mediterranee-metropole"
#lien du site open data montpellier

num=17

for j in range(num):#Nombre de fois où la boucle va se répéter
    for i in parkings:

response=requests.get("https://data.montpellier3m.fr/sites/default/files/ressource
s/"+i+".xml")
    #Récupérer tout les liens de chaque parking de montpellier
    f1=open(i,"w", encoding='utf8') #Ouvre un fichier dans lequel toutes
données seront sauvegardées et écrasées à chaque mise à jour des données
    f1.write(response.text)
    f1.close()
    f2=open("resultats.text","a",encoding='utf8') #Ouvre un fichier dans
lequel les données seront sauvegardées, et non écrasées à chaque
    tree = etree.parse(i)
    for user in tree.xpath("Date_time"):
        time=user.text
        f2.write('\n') #('\n') : sert à revenir à la ligne
        f2.write('Date:')
        f2.write(user.text) # Ecriture de la date dans le fichier
        f2.write('\n')

    for user in tree.xpath("Name"):
        print('Nom du parking :',user.text) #Ici J'ai utilisé un print qui me
permet de voir directement dans la console si le code marche, au lieu d'aller
vérifier les fichiers à chaque fois que le programme est en marche
        f2.write('Parking :')
        f2.write(user.text) #Afficher le nom du parking sur le fichier
        f2.write('\n')
        #Afficher le nom du parking

    for user in tree.xpath("Total"):
        f2.write('Nombre total de places:')
        f2.write(user.text) #Afficher le nombre de total de places du parking
        f2.write('\n')
```

```

        total=int(user.text) #Valeur nombre de places totales afin de calculer
le pourcentage

        for user in tree.xpath("Free"):
            f2.write('Nombre de places libres :')
            f2.write(user.text) #Afficher le nombre de places libres sur le
fichier
            f2.write('\n')
            free=int(user.text) #Valeur places libres afin de calculer le
pourcentage
            pourcentage_l=round((free*100)/total,2)
            f2.write('Le pourcentage de places libres est de :')
            f2.write(str(pourcentage_l)) #Affichage du pourcentage de places lbres
            f2.write("%")
            f2.write('\n')P
            f2.write('\n')
            pourcentage_o=100-(pourcentage_l*100)/total
            f2.write('Le pourcentage de places occupées est de :')
            f2.write(str(pourcentage_o)) #Affichage du taux d'occupation
            f2.write("%")
            f2.write('\n')
            f2.write('\n')

        f2.write('\n')
        f2.write('\n') #Sauter des lignes afin de rendre le fichier lisible
        f2.write('\n')
        f2.write(' Programme en pause') # Me permet de me repérer sur le fichier
puisque'il ya beaucoup de données qui ont été récupérées
        f2.write('\n')
        f2.write('\n')
        f2.write('\n')

        time.sleep(3600)# focntion qui permet d'endormir le programme selon une durée
déterminée, ici chaque 1 heure le programme s'exécute

```

J'ai voulu ensuite analyser toutes les données pour tous les parkings de Montpellier en calculant le taux d'occupation total de la ville Pour cela j'ai ajouter dans une liste tous les noms de parkings puis j'ai fais une fonction qui affiche le taux d'occupation des parkings de Montepellier.

Voici mon programme avec les différentes fonctions commentées et leur signification

```

import requests
from lxml import etree
import time

```

```

parkings=
['FR_MTP_ANTI', 'FR_MTP_COME', 'FR_MTP_CORU', 'FR_MTP_EURO', 'FR_MTP_FOCH', 'FR_MTP_GAM
B', 'FR_MTP_GARE',

'FR_MTP_TRIA', 'FR_MTP_ARCT', 'FR_MTP_PITO', 'FR_MTP_CIRC', 'FR_MTP_SABI', 'FR_MTP_GARC
', 'FR_MTP_SABL',

'FR_MTP_MOSS', 'FR_STJ_SJLC', 'FR_MTP_MEDC', 'FR_MTP_OCCI', 'FR_CAS_VICA', 'FR_MTP_GA10
9', 'FR_MTP_GA250',
    'FR_CAS_CDGA', 'FR_MTP_ARCE', 'FR_MTP_POLY']

Toville=0 #Le nombre total des parkings voituresde toute la ville
FrVille=0 #Le nombre total des parkings voitures libres de toute la ville
Nom=input("Nom du fichier:")# indiquer le nom du fichier
periode=int(input("Période(min):"))
periode=periode*60 # cela indique la durée de l'acquisition du programme en
minutes . Par exemple si je tape période= 2 , la durée sera de 120min c'est à dire
2 heures car ici on multiplie par 60
duree=int(input("durée(sec):"))
t=60 # t c'est le nombre de fois que le programme va se répéter .

for p in range(t):
    for i in parkings: #la liste "parkings" contient les noms des fichiers de
chaque zone. On fait un boucle pour récupérer des données de chaque zone.
#Récupérer les données et les mettre dans un nouveau fichier

data=requests.get(f"https://data.montpellier3m.fr/sites/default/files/ressources/{
i}.xml")
    f1=open(f"{i}.txt", "w", encoding='utf8') #Cette fonction va ouvrir un
fichier texte conteant les informations de tous les parkings de Montpellier
    f1.write(data.text)
    f1.close()
    #trier ces données et choisir le nombre des places libres et des places
totales.
    tree=tree.parse(f"{i}.txt")
    a=0
    b=0
    for user in tree.xpath("Total"):
        total=int(user.text)
        a=a+total
    for user in tree.xpath("Free"):
        libre=int(user.text)
        b=b+libre
    # ajouter des nombre dans la valeur du "nombre total". C'est pour obtenir
la somme de tous les parkings et celle de tous les parkings libres
    Toville=Toville+a
    FrVille=FrVille+b
    PVoiture=FrVille/Toville # la valeur : Parkings libres/ Parkings total
    PVoiture=1-PVoiture # pour obtenir le taux d'occupation des voitures
    temps=time.time()

```

```

temps=time.ctime(temps)
#fonction qui va permettre d'afficher la date a côté de chaque données

# Ici ça va permettre d'ouvrir un fichier avec le champ "nom" que l'on
souhaite donné et que l'on a spécifié au début avec la fonction f2.write qui va
permettre d'écrire dans
# ce fichier qui stockera les données avec la date et le taux d'occupation
f2=open(f"{Nom}.txt","a",encoding='utf8')
f2.write(f"{temps} Le Taux d'occupation des parkings de la ville de
Montpellier est de: {round(PVoiture*100, 2)}%")
f2.write('\n')
f2.close()
time.sleep(duree) #fonction qui suspend l'exécution en fonction du nombre de
secondes que j'ai attribué à ma fonction "duree"

```

- Pour la partie vélos j'ai réussi à extraire les données des différentes stations en récupérant grâce à la fonction "requests" le lien des données contenant les informations pour chaque stations ainsi que le lien contenant le statut de ces stations. J'ai ensuite stockées les données dans 2 fichiers texte différents mais je n'ai pas pu réussir à les analyser en calculant leur taux d'occupation ou encore leur moyenne car à la base cela devrait être mon binôme qui s'en occupé, j'ai donc assuré aussi cette partie en faisant ce que je pouvais faire le plus possible seul mais j'ai eu quelques problèmes qui m'ont empêché d'avoir les résultats des occupations.

Voici donc mon programme qui récupère les données json des stations vélos de Montpellier avec une fonction qui devait calculer le taux d'occupation

```

import json
import requests
import time

# Les liens vers les données JSON des stations
status_url = "https://montpellier-fr-
smooove.klervi.net/gbfs/en/station_status.json"
info_url = "https://montpellier-fr-
smooove.klervi.net/gbfs/en/station_information.json"

period=int(input("Période(min):"))
period=period*60
duration=int(input("duration(sec):"))

# Récupère les données JSON des stations
response = requests.get(status_url)
status_data = response.json()

```

```
response = requests.get(info_url)
info_data = response.json()

# Stocke les données dans des fichiers différents
with open("status_data.txt", "w") as outfile:
    json.dump(status_data, outfile)

with open("info_data.txt", "w") as outfile:
    json.dump(info_data, outfile)

# Calcule le taux d'occupation des stations
total_stations = len(status_data)
occupied_stations = 0
for station in status_data:
    bikes_available = station["num_bikes_available"]
    station_id = station["station_id"]
    capacity = info_data[station_id]["capacity"]
    if bikes_available < capacity:
        occupied_stations += 1
occupancy_rate = occupied_stations / total_stations

# Affiche le taux d'occupation
print("Taux d'occupation des stations: {:.2f}%".format(occupancy_rate * 100))

time.sleep(duration)
```

Traitement des données

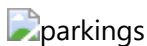
Pour traiter les différentes données j'ai choisi d'utiliser "Gnuplot" qui est un programme de ligne de commande et d'interface graphique qui peut générer des tracés en deux et trois dimensions de fonctions, de données et d'ajustements de données. J'ai trouvé cette interface pratique et très simple d'utilisation.

Voici ci-dessous les différentes lignes de commandes qui m'ont permis de tracer une courbe de données qui prends en paramètre le taux d'occupation de 4 parkings voitures au cours du temps.

```
set terminal png size 800,600
set output 'image.png'
set key inside bottom right
set autoscale
set xlabel 'temps (en heure)'
set xdata time
set timefmt "%H:%M:%S"
set format x "%H:%M":"%S"
set ylabel 'Taux d occupation des parkings voitures'
set title 'Evolution du taux d occupation des parkings de Montpellier'
plot "datas_parks.txt" using 1:2 title "ANTI" with linepoints, using 1:3 title "COME" with linepoints, using 1:4 title "OCCI"
```

La ligne timefmt permet de définir le format de la date, ici par exemple cela sera en Heure, Minutes, Secondes

le "using 1:2" extrait les données de la 1ère et 2 ème colonne du fichier en question .





On peut constater qu'ils n'ont pas la même fréquentation car cela dépend de leur emplacement. Par exemple, pour le parking "Occitanie" j'ai constaté qu'il se situe à coté d'un IUT et j'en ai déduit une corrélation puisque si l'on remarque, ce parking se remplit fortement entre 9h et 12h qui est souvent l'heure du début des cours.

Puis vers 12h et jusqu'à 13 heures il y a une légère diminution du taux d'occupation puisque c'est souvent l'heure du repas . Et puis enfin on peut constater que de 14h jusqu'à 15 heures l'occupation reste constante et à partir de 16 heures et ce, jusqu'à 9 heures il subit une plus forte diminution de l'occupation comparé aux autres parkings. On peut analyser cela par le fait que vers 16 heures certaines personnes finissent les cours et le travail.

Voici maintenant les différentes courbes concernant le taux d'occupation total pour tous les parkings de la ville de Montpellier à des jours différents

Pour le samedi 21/01/2023  Samedi

Pour le dimanche 22/01/2023  Dimanche

Pour le jeudi 26/01/2023  Jeudi

Je constate tout d'abord que le taux d'occupation n'est pas du tout le même selon les jours , en effet le samedi 21/01 l'évolution du taux d'occupation ne cesse de varier.

De 14h à 16h le taux d'occupation augmente fortement de 6 à 7% . J'imagine que le Samedi la plupart des gens ne travaillent pas, pratiquement toutes les écoles sont fermés, hors mis pour certaines universités ou les étudiants ont cours.

Le samedi en général les parkings sont pas mal occupés puisque les gens vont faire les courses sortir au cinéma, au restaurant avec des amis ou en famille ou encore tout simplement sortir en ville pour juste la visiter. Un autre facteur aussi est que le samedi certains parkings peuvent être gratuits.

Le dimanche je constate que le taux d'occupation a diminué comparé à Samedi. De 14h à 23h la courbe ne cesse de diminué à part de 19h à 20h ou il y a une légère augmentation, cela peut s'expliquer peut-être par le fait qu'un dimanche certaines personnes vont dans un restaurant ou chez la famille manger. Puis le reste de la soirée le taux diminue, les gens doivent rentrer chez eux.

Le jeudi, je constate que le taux d'occupation des parkings voitures de Montpellier est beaucoup plus grand que le week-end.

En effet je pense que la semaine la plupart des gens travaillent, les écoles sont ouvertes et tous les magasins sont ouverts. Je constate que le pic d'occupation est à 15h puis cela diminue fortement et constamment tout au long de la journée jusqu'à 21h le soir quand tous les étudiants, employés... rentrent chez eux.

Calcul mathématiques des données

Pour avoir plus d'informations concernant l'analyse de ces données j'ai effectué un programme qui permet de calculer la moyenne, l'écart type et la variance des différentes valeurs du taux d'occupation des parkings de Montpellier pour le Jeudi, le Samedi et le Dimanche.

Pour cela j'ai d'abord importé toutes les librairies de "statistiques" et de "math" nécessaires pour calculer les fonctions.

Voici ci-dessous le programme qui effectue les différents calculs sur les données

```

from statistics import *
from math import *

def moyenne(liste):
    return (1/len(liste)) * sum(liste)

def ecart_type(liste):
    moy = moyenne(liste)
    N = len(liste)
    var = 0
    for nb in liste:
        var += (nb - moy)**2
    return sqrt((1/N) * var)

occup=[58.79, 61.46, 62.78, 64.82, 65.41, 65.28, 63.93, 62.43, 60.17, 57.59,
55.68, 51.70, 49.91, 48.12, 46.09, 43.29 ,39.94]

print(moyenne(occup))
print(ecart_type(occup))
print(pvariance(occup))

```

J'ai créer une fonction "moyenne" qui caclule la moyenne à partir de la longueur de la liste ("len") puis une focntion "ecart-type" qui calcule la variance à partir des valeurs et de la moyenne de L'opérateur += effectue une addition puis affecte le résultat à la même variable Puis le calcul de l'écart type s'effectue en divisant 1 par la longueur de la liste multiplié par le résultat de la variance

Voici les résultats:

 résultats1

**Pour la journée de Jeudi, la moyenne du taux d'occupation est d'environ egale à 56, l'écart type est de 8 et la variance de 64

J'en conclue que la semaine le taux d'occupation des parkings voitures de Montpellier est plus important que le week-end pour prendre l'exemple de mes résultats

Pour retrouver toutes les données vous pouvez aller consulter ma page web disponible sur mon dépôt Github.