Le Nettoyage de la base CSV :

Manipulation sur base à partir du fichier csv

Ouverture du fichier avec un lecteur approprié : OpenOffice LibreOffice. ou Excel
Retrait extraction des colonne qui sont les suivante :

- ◯ Nouvel identifiant
- ◯ Ancien identifiant

Régime prioritaire
Régime particulier
Arrondissement
Zone Résidentielle
Tarification
Type de voie
Nom de la voie
Parité
Longueur
Longueur calculée
Signalisation horizontale
Signalisation verticale
Conformité signalisation
Plage horaire 1-Début
Plage horaire 1-Fin
Plage horaire 2-Début
Plage horaire 2-Fin

- ◯ Plage horaire 3-Début
- ◯ Plage horaire 3-Fin

Date du relevé
Dernière date édition

- ◯ Code Voie Ville de Paris
- ◯ Numéro Séquentiel
- ◯ Tronçon Voie
- ◯ Numéro ilot
- ◯ Numéro IRIS

Zone ASP
Numéro Section
Territoriale de Voirie
Zone préfecture

geo_shape

-Epurer les colonne non nécessaire

-Passer de 33 colonnes à 20 colonnes.
-Et rassembler la premier et seconde colonne en une
-Puis Enregistrer sous le forma cdv avec séparateur fabulation utf8

Imports:

```
import tkinter as tk
import googlemaps
import numpy as np
import pandas as pd
import tkinter
import PIL
import json
import scipy
import matplotlib
import matplotlib.pyplot as plt
import math

from PIL import Image, ImageTk
from tkinter import Tk, PhotoImage, Canvas, ttk
```

Question 1 :

```
# Q1 : dans quel arrondissement il y a le plus de places disponibles suivant les plages de stationnements ouverts/ non interdits ?

df_Q1_Clean = pd.read_csv('C:/Users/Briche/Documents/GRETA_2021/projet_stationnement_paris/df_Q1_Clean.xls', header = 0)
df          = pd.read_csv('C:/Users/Briche/Documents/GRETA_2021/projet_stationnement_paris/stationnements.txt', sep ='\t', header = 0)
df1         = pd.read_csv('C:/Users/Briche/Documents/GRETA_2021/projet_stationnement_paris/df_Q3_Clean.xls', header = 0)

df_Q1_Clean.head()

# Groupby par Arrondissement

g_Q1 = df_Q1_Clean.groupby('Arrondissement')
# print("groupe par arrondissement" , g_Q1)

# Somme de duree par arrondissement

df_duree_sum = g_Q1[['Duree']].agg([pd.Series.sum])

# Reset the index and the columns

df_duree_sum = df_duree_sum.reset_index()
df_duree_sum.columns = df_duree_sum.columns.droplevel(1)
df_duree_sum = df_duree_sum.rename(columns = {'Duree':'Duree_Sum'})

# Supprimer arrondissement 0
```

Q 1-a : Max

```
df_Q1_Court = df_duree_sum.sort_values(by = 'Duree_Sum')
# print("Stationnement des plus courts aux plus longs :", df_Q1_Court)
```

Q 1-b : Min

```
# le moins

df_Q1_Long = df_duree_sum.sort_values(by = 'Duree_Sum', ascending = False)
# print("Stationnement les plus longs aux plus courts :", df_Q1_Long)
```

Question 2 :

```
# Combobox creation Durée
n1 = tk.StringVar()
duree = ttk.Combobox(fenetre , width = 15 , textvariable = n1)
n2 = tk.StringVar()
# Adding combobox drop down list
duree['values'] = ('Min','Max')
duree.place(x = 250 , y = 70)
duree.current()
# créer un Combobox pour 'Regime_particulier'
n4 = tk.StringVar()
labelChoix5 = tk.Label(fenetre , text = 'Régime particulier' , bg = 'ivory')
# labelChoix5.place(x = 535 , y = 50)
regCom = ttk.Combobox(fenetre , width = 25 , textvariable = n4)
regCom['values'] =( 'Livraison_BUS',
                    'Stationnement_genant',
                    'Arret_vigipirate_non_perennise',
                    'Arret_vigipirate_perennise',
                    'Arret_genant_divers',
                    'Stationnement_simple',
                    'Arret_simple',
                    'Arret_Pompiers',
                    'rien')
```

```
# Q2 : pour chaque arrondissememnt donne le type de stationnemment le plus et moins populaire
# doit répondre à la selection dans les menus déroulants de Arrondissement et de Durees sur stationnement
# interdit puis click du boutton afficher

freq = df.groupby(['Arrondissement', 'Regime_particulier']).size()
# pour construire la matrice d'arrondissement et combien de fois on a un type de stationnement
freq = freq.reset_index()
# diviser une colonne en 3 colonnes
cd = pd.DataFrame(freq)
# en dataframe
groups = cd.groupby(by = ['Arrondissement'])
global freq2
freq2 = groups.apply(lambda g: g[g[0] == g[0].max()])
# pour afficher le maximum de chaque arrondissement et le type de stationnement à proprier
global cd2
cd2 = pd.DataFrame(freq2)
# data frame pour le max : donne l'arrondissement, le type de sttaionnement et le nombre d'ocurrence de stationnement
groups = cd.groupby(by = ['Arrondissement'])
global freq3
freq3 = groups.apply(lambda g: g[g[0] == g[0].min()])
global cd3
cd3 = pd.DataFrame(freq3)
# data frame pour le min :  donne l'arrondissement, le type de sttaionnement et le nombre d'ocurrence de stationnement
```

Question 3 : Ventilation balisage

```
# Q3 : donne le mois de la durée min et max de stationnement

pd.options.display.max_rows = 10
newFrame = df1.groupby(by = 'Mois')
newFrame = df1.groupby(['Mois'])["Duree"].agg("sum")
g = newFrame.reset_index()
keys = newFrame.keys ()
months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
 'September', 'October', 'November', 'December']
newdict = dict(zip(months, keys))


def answerq1():
      text_affichage.delete("1.0","end")
      arrond.get()
      duree.get()
      text_affichage.insert('3.0', arrond.get())

def answerq2(): # fichier cd2 et cd3
      text_affichage.delete("1.0","end")
      arr = arrond.get()
      dur = duree.get()
      text_affichage.insert('3.0', cd2)
      print(cd2)
      text_affichage.insert('3.0', arr )
      text_affichage.insert('3.0', dur )

def answerq3(): # fichier d'interêt :

      text_affichage.delete("1.0","end")
      arrond.get()
      duree.get()
      text_affichage.insert('3.0', 'Q3')
```

Question 4 :

```
# interface
fenetre = tk.Tk()
fenetre.geometry('800x560')                # Definir la taille de la fenetre principale
fenetre.title("Stationnement à Paris")     # Titre
fenetre.iconbitmap("")
fenetre.configure(bg = 'ivory')            # Définir la couleur de la fenetre titre
# Background image
# img = tk.PhotoImage(file = 'C:/Users/Briche/Documents/GRETA_2021/projet_stationnement_paris/parking.png')
im  = Image.open('C:/Users/Briche/Documents/GRETA_2021/projet_stationnement_paris/parking.png')
img = ImageTk.PhotoImage(im, master = fenetre)
C = Canvas(fenetre , width = 800 , height = 554)
Image = C.create_image(0, 0, image = img)
# C.pack(padx = 10 , pady = 50)
"""
Vous pouvez utiliser une StringVar comme variable de contrôle d'un widget Scale.
Il sera tout de même nécessaire de préciser des valeurs numériques pour les options
from_ et to, Mais les valeurs numériques du widget seront converties en une chaîne de
caractères pour être mémorisées dans la StringVar
J'ai une fenêtre qui s'affiche demandant de spécifier si le besoin est primaire ou non.
Un stringVar me permet de récupérer cette information et ainsi de créer un objet besoin
avec l'état primaire=True ou False (par défaut). Ensuite dans une seconde fenêtre je demande
de spécifier l'intitulé du besoin.

"""
```

```
# label arrondissement
labelChoix1 = tk.Label(fenetre , text = "Arrondissement" , bg = 'ivory')
labelChoix1.place(x = 55 , y = 50)
# label choix de la durée
labelChoix2 = tk.Label(fenetre , text = "Durées sur stationnement interdit" , bg = 'ivory')
labelChoix2.place(x = 220 , y = 50)
# Zone de texte pour affichage
v = tk.StringVar()
text_affichage = tk.Text(fenetre , width = 200 , height = 40 )
text_affichage.place(x = 42 , y = 140)
text_affichage.insert('3.0', 'Q1, Q2, Q3')
# Combobox creation arrondissement
n0 = tk.StringVar()
arrond = ttk.Combobox(fenetre , width = 18 , textvariable = n0)
# Adding combobox drop down list
arrond['values'] = ('Arrondissement -1','Arrondissement 1','Arrondissement 2','Arrondissement 3',
                    'Arrondissement 4','Aroondissement 5','Arrondissement 6','Arrondissement 7',
                    'Arrondissement 8','Aroondissement 9','Arrondissement 10','Arrondissement 11',
                    'Arrondissement 12','Aroondissement 13','Arrondissement 14','Arrondissement 15',
                    'Arrondissement 16','Arrondissement 17','Arrondissement 18','Arrondissement 19',
                    'Arrondissement 20','Arrondissement 21','Arrondissement 22',)

arrond.place(x = 40 , y = 70)
arrond.current()
```