Application de génération de script d'insertion

Du 16 au 19 Mars

Développement en Python avec Eclipse équipé du plugin PyDev et du module d'interface graphique Tkinter

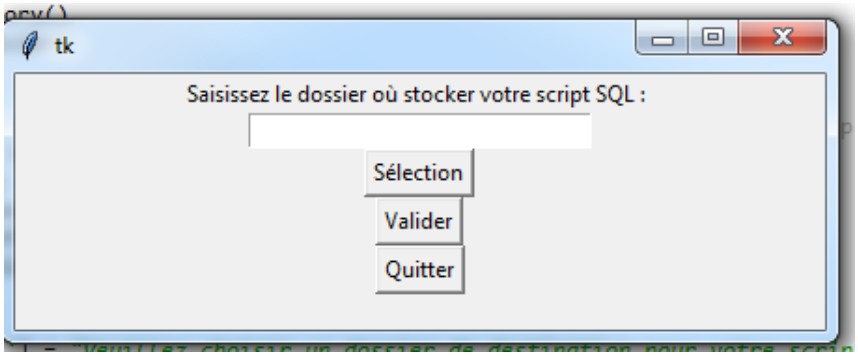
L'application devait être capable, en prenant pour paramètre un fichier xls ou xlsx, d'en récupérer les données contenues et d'en générer un fichier script SQL d'insertion. Ce fichier xls contenait la liste des tournées correspondantes à un cabinet d'expertise dans chaque ville.

De nombreux nouveaux cabinets ont rejoint le réseau Expertise & Concept. Afin d'officialiser ce regroupement, le site internet du réseau devait afficher toutes les informations relatives à ces nouveaux cabinets, dont la liste des tournées fait partie. Chaque nouveau cabinet a donc fait parvenir sous le même format les fichiers correspondants à ces données. Il a donc été incombé au département informatique du réseau de saisir dans la base de données ces informations afin de mettre à jour le site internet.

Au vu de la taille et de la quantité des éléments retournés, une saisie manuelle était inenvisageable. Il a donc rapidement été décidé de trouver une solution alternative. Le formalisme des données retournées étant différent d'un fichier à l'autre ainsi que leurs tailles conséquentes, rendaient impossible la simple conversion d'un fichier xls en fichier csv afin de l'importer directement dans la base de données. Le développement d'un script afin d'automatiser la génération de requête s'est alors imposé de lui-même. La solution devait être mise en place de manière relativement rapide, et donc utiliser le maximum de ressources déjà présentes sur la machine développeur.

Eclipse étant présent sur la machine, il a naturellement été choisi comme IDE, en particulier grâce à sa capacité à détecter les erreurs et régler les problèmes d'indentations, au contraire de Notepad, également installé. Python étant un langage simple d'utilisation et adapté à ce genre de situation où le développement doit s'effectuer rapidement, il s'est avéré être un choix potentiel, après une pratique de veille technologique sur sa syntaxe et ses capacités. Une forte communauté, la profusion de librairies utilitaires et sa capacité de manipulation de fichiers csv parachevèrent le choix de cette solution.

La première étape a consisté à développer une première fenêtre de saisie. Cette dernière demande à l'utilisateur l'emplacement du futur fichier script SQL grâce à une boîte de dialogue déclenchée par un clic sur le bouton « Sélection ». Ensuite, cette adresse est passée en paramètre d'une nouvelle fenêtre qui s'occupera par la suite de l''exploiter, une fois que l'utilisateur le décidera grâce au bouton « Valider ».

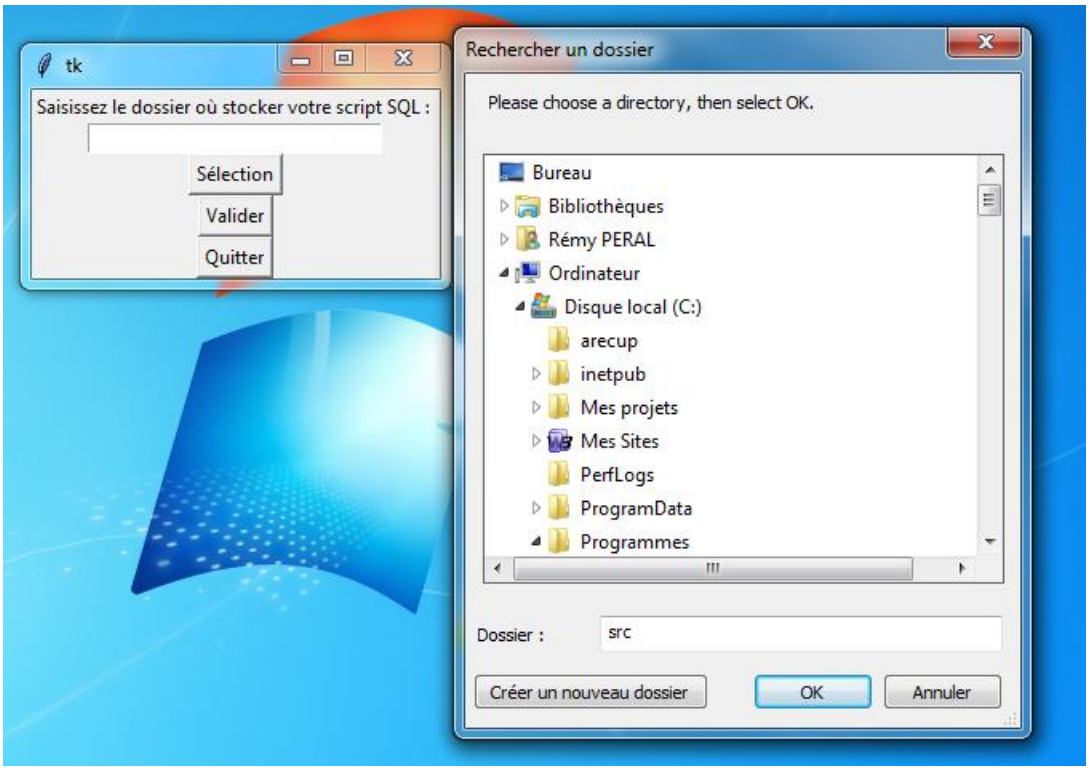


Première fenêtre de saisie

```
25 # Création de l'interface de l'utilisateur
26 fenetre = Tk()
27 champ_label = Label(fenetre, text="Saisissez le dossier où stocker votre script SQL : ")
28 champ_label.pack()
29
30 var_dossier = StringVar()
31
32 zoneDossier = Entry(fenetre, textvariable=var_dossier, width=30)
33 zoneDossier.pack()
34
35 #Permet a l'utilisateur de sélectionner un dossier de destination pour son script SQL
36 bouton_chercher = Button(fenetre, text="Sélection", command = selectionDossier)
37 bouton_chercher.pack()
38
39 bouton_valider = Button(fenetre, text = "Valider", command = setValue)
40 bouton_valider.pack()
41
42
43 bouton_quitter = Button(fenetre, text="Quitter", command=fenetre.quit)
44 bouton_quitter.pack()
45
46
47 fenetre.mainloop()
48
```

Cette partie du code permet la mise en place de l'interface graphique. Les propriétés de chaque widget est définit puis affiché grâce à la méthode pack(). La fenêtre est initialisée en début de code puis maintenue en permanence grâce à la méthode mainloop().

L'application récupère ensuite l'adresse, vérifie que l'utilisateur ai bien saisit une adresse de destination puis crée une nouvelle fenêtre. Dans le cas contraire, un message d'alerte lui indique de le faire.

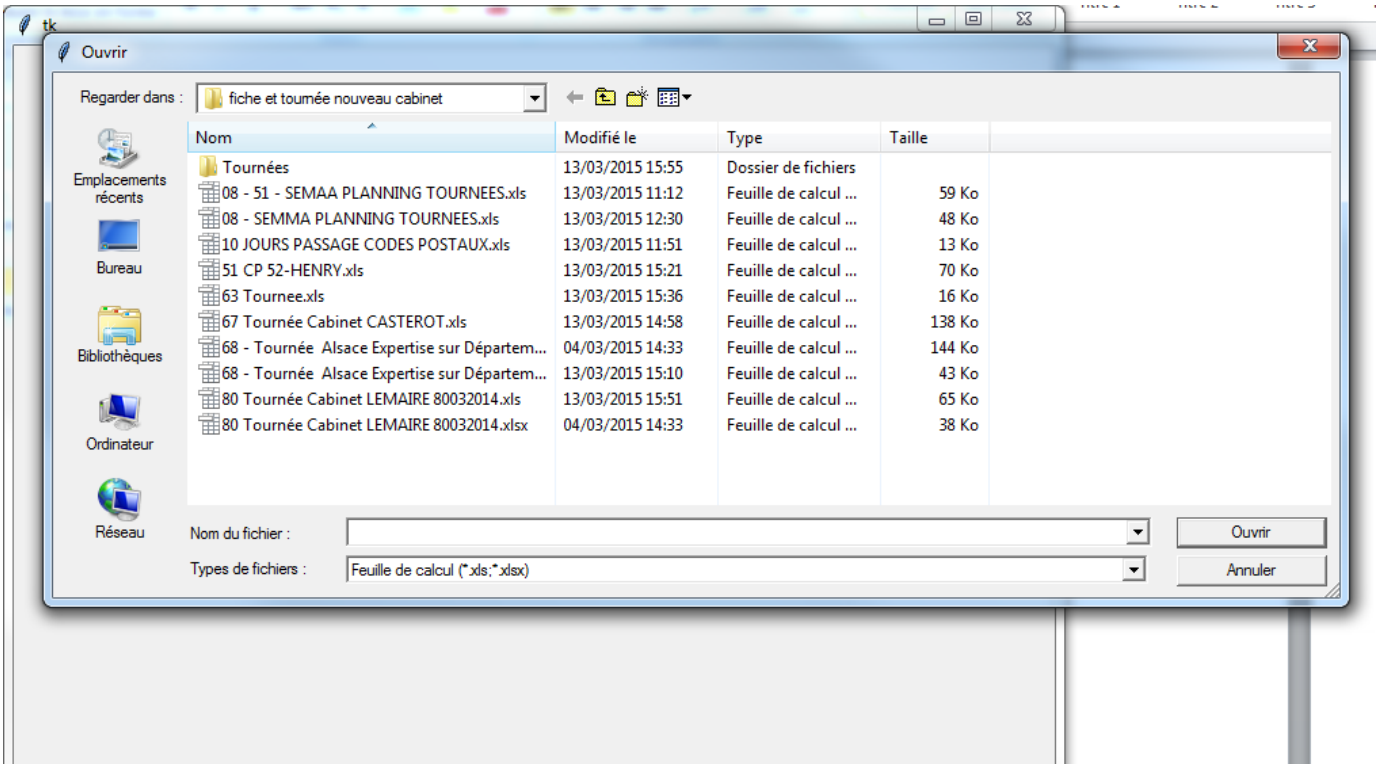


Fenêtre de saisie de l'adresse

```
def selectionDossier():

    leDossier = askdirectory()
    var_dossier.set(leDossier)

def setValue():
    #Si l'utilisateur à selectionné un dossier, la fenêtre actuelle est détruite et fait place à la suiv
    if zoneDossier.get() != "":
        fenetre.destroy()
        fenetreInformation = Tk()
        informations = Information(fenetreInformation, var_dossier.get())
        informations.mainloop()

    else:
        champ_label['text'] = "Veuillez choisir un dossier de destination pour votre script SQL"
        champ_label["fg"] = "red"
```

La librairie Tkinter, qui permets le développement d'une interface graphique, comprend un module spécifiquement dédié à la manipulation d'adresses de dossiers. Ici c'est la méthode askDirectory() qui s'en charge, après l'appel de la fonction selectionDossier, déclenché après un clic sur le bouton « selection »

Une fois la sélection validée, l'utilisateur doit sélectionner le fichier xls ou xlsx qu'il désire exploiter. Une fois cette sélection validée, il doit sélectionner le cabinet qui effectue cette tournée. Dans un souci de rapidité et de simplicité, le réseau Expertise & Concept comptant 55 cabinets environ, l'application se charge de ne proposer à l'utilisateur que le nom des cabinets situés dans le même département que celui des tournées, ces dernières étant déjà triées par département par chaque cabinet.

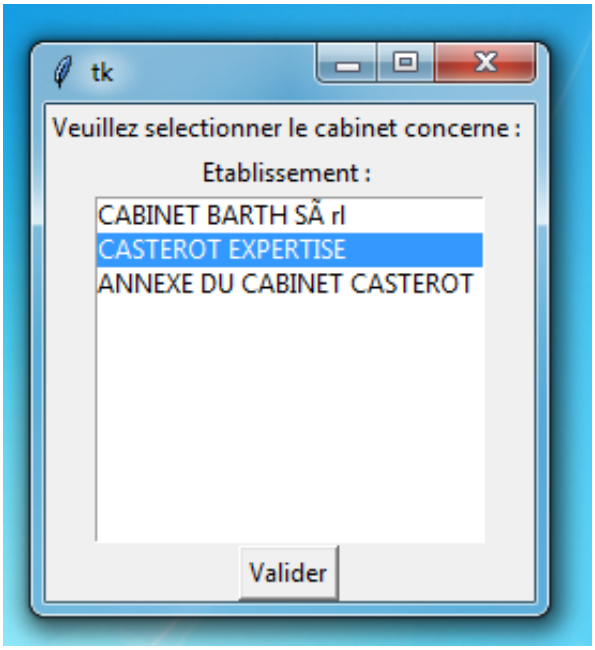


Fenêtre de sélection du fichier xls correspondant à une tournée

```
18 class Information(Frame):
19     '''
20     classdocs
21     '''
22     def __init__(self, fenetre, leDossier, **kwargs):
23
24         Frame.__init__(self, fenetre, width=768, height=576, **kwargs)
25         self.pack(fill=BOTH)
26
27         self.leDossier = leDossier
28
29         self.fname = askopenfilename(filetypes=(
30                                         ("Feuille de calcul", "*.xls;*.xlsx"),
31                                         ("All files", "*.*") ))
32         #Récupère la feuille Excel de la tournée et la convertit en CSV
33         xls = pd.ExcelFile(self.fname)
34         df = xls.parse('Feuil1', index_col= None, na_values=['NA'])
35         df.to_csv('csvTemporaire.csv', index = False, index_label= False, header= False)
36
37         self.champInformations = Label(self, text = "Veuillez selectionner le cabinet concerne : ")
38         self.champInformations.pack()
39
40         self.champEtablishment = Label(self, text="Etablissement : ")
41         self.champEtablishment.pack()
42
43         self.lEtablishment = Listbox(self, width = 30)
44
```

Cette classe se construit grâce aux informations récemment retournées par le programme. Il récupère ensuite l'adresse du fichier grâce à la méthode askopenfilename() et le convertit au format csv afin qu'il soit plus facilement manipulable. Les champs de sélection de cabinets sont ensuite affichés.

Le programme remplit ensuite une liste de cabinets à sélectionner à partir du département de la tournée et de la liste de tous les cabinets contenue dans un fichier source csv.



Fenêtre de sélection du cabinet

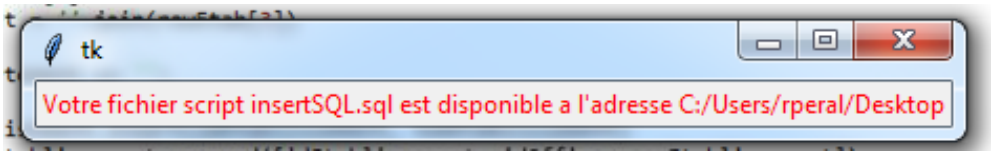
```
45          with open('csvTemporaire.csv', newline='', encoding="utf8", errors='ignore') as csvfile:
46
47              reader = csv.reader(csvfile, delimiter=',', quotechar='|')
48              self.leDepartement = 0;
49
50              #Récupère le numéro de département de l'établissement concerné par l'établissement
51              for row in reader:
52                  self.zipCode = row[1]
53
54                  if len(self.zipCode) < 5:
55                      self.leDepartement = self.zipCode[:1]
56                      self.leDepartement = '0'.join(self.leDepartement)
57                  else:
58                      self.leDepartement = self.zipCode[:2]
59
60                  self.leDepartement = int(self.leDepartement)
61                  if self.leDepartement > 20:
62                      self.leDepartement = self.leDepartement + 1
63                  self.leDepartement = str(self.leDepartement)
64
65          csvfile.close()
66
```

Le programme ouvre le fichier convertit et récupère le numéro de département de la tournée. Ce numéro de département sera ensuite conservée puis réinjecté dans la requête.

```
with open('establishment1.csv', newline= '') as csvEtablissement:

    idEtablissement = 0
    idOffice = 0
    self.listeEtablissements = []

    readerEtab = csv.reader(csvEtablissement, delimiter = ',', quotechar = '/')

    #Remplis la listbox du nom de tous les cabinets qui correspondent au numéro du département précédemment récupéré
    #Permets également de récupérer l'id du cabinet et de l'établissement concerné

    for rowEtab in readerEtab:

        idEtablissement = rowEtab[0]
        idOffice = rowEtab[1]
        nomEtablissement = ''.join(rowEtab[3])

        if self.leDepartement == "":

            self.lEtablishment.insert(idEtablissement, nomEtablissement)
            self.listeEtablissements.append([idEtablissement, idOffice, nomEtablissement])
        else:
            if rowEtab[2] == self.leDepartement:

                self.lEtablishment.insert(idEtablissement, nomEtablissement)
                self.listeEtablissements.append([idEtablissement, idOffice, nomEtablissement])

    if self.listeEtablissements == []:
        for rowEtab in readerEtab:
            idEtablissement = rowEtab[0]
            idOffice = rowEtab[1]
            nomEtablissement = ''.join(rowEtab[3])
            self.lEtablishment.insert(idEtablissement, nomEtablissement)
            self.listeEtablissements.append([idEtablissement, idOffice, nomEtablissement])

self.lEtablishment.pack()
self.bouton_envoyer = Button(self, text="Valider", command = self.envoyer)
self.bouton_envoyer.pack()
```

Il compare ensuite ce numéro à ceux présents dans la liste des établissements et remplis la liste s'ils correspondent. Cette liste est affiché, et la sélection de l'utilisateur est renvoyée grâce au « self » dans la fonction envoyer, appelée après un clic sur le bouton « Valider »

Le programme récupère ensuite le fichier csv temporaire et le manipule afin que son formalisme corresponde à celui de la base de données. En effet, la manière de présentation dont le passage des tournées s'effectue est différente d'un fichier à l'autre, tandis que la base de données comprend uniquement des 0 et des 1 pour déterminer un passage ou non à une journée donnée. Enfin, ces données sont retriées et écrites dans un fichier script en respectant la syntaxe d'Insertion SQL, puis une fenêtre de confirmation rappelant la destination du fichier apparaît.



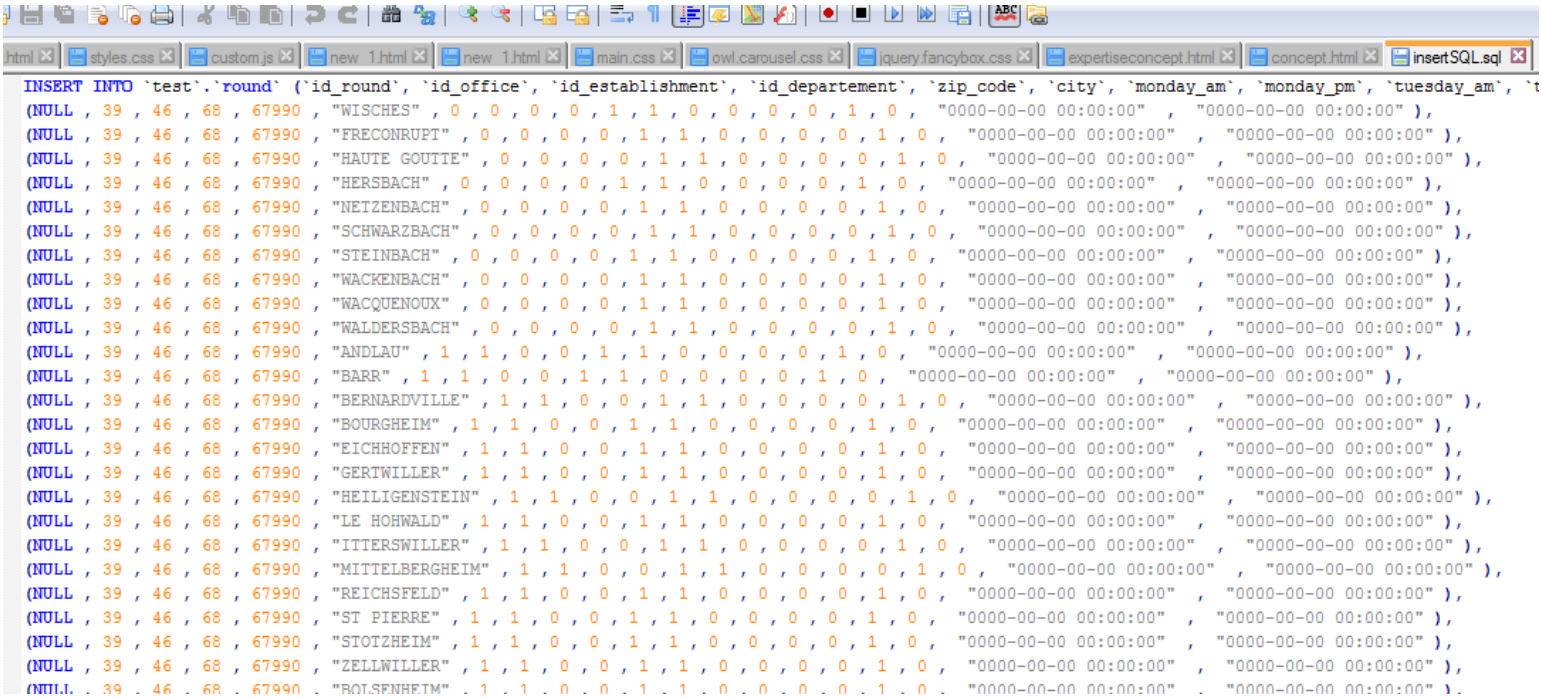
Fenêtre de confirmation

```
106    def envoyer(self):
107
108        with open('csvTemporaire.csv', newline='', encoding="utf8", errors='ignore') as csvfile:
109
110            #Ce bout de code permet de déterminer le nombre de lignes du Csv généré à partir de la liste des tournées
111            limite = 0
112            buf_size = 1024 * 1024
113            read_f = csvfile.read
114            buf = read_f(buf_size)
115            while buf:
116                limite += buf.count('\n')
117                buf = read_f(buf_size)
118            csvfile.close()
119
120            with open('csvTemporaire.csv', newline='', encoding="utf8", errors='ignore') as csvfile:
121
122                reader = csv.reader(csvfile, delimiter=',', quotechar='|')
123                writer = csv.writer(csvfile, delimiter=',', quotechar='|', quoting=csv.QUOTE_MINIMAL)
124
125                keyElement = 0
126                data =[]
127                valeurJournees = ""
128                nbElements = 0
129
130                #Récupère les informations concernant les jours de la tournée et traite les résultats de manière à ce qu'ils
131                for row in reader:
132                    keyElement = keyElement + 1;
133                    nbElements = nbElements + 1
134                    ville = row[0]
135
136                    if row[3] != "":
137                        row[3] = row[3].replace("X", '1')
138                        row[3] = row[3].replace("M", '1')
139                        row[3] = row[3].replace("A", '1')
140                        row[3] = row[3].replace("J", '1')
141                        row[3] = row[3].replace("x", '1')
142
143                    else :
144                        row[3] = row[3].replace(" ", '0')
145                        row[3] = row[3].replace("", '0')
146
147                    monday = row[3]
148
149                    if row[4] != "":
150                        row[4] = row[4].replace("X", '1')
151                        row[4] = row[4].replace("M", '1')
152                        row[4] = row[4].replace("A", '1')
153                        row[4] = row[4].replace("J", '1')
154                        row[4] = row[4].replace("x", '1')
155
```

Le programme détermine ensuite le nombre de lignes à manipuler, récupère le nom de la ville, puis remplace les données de 5 colonnes correspondantes à chaque journée pour chaque ville par les données correctes. Pour chaque ligne et chaque colonne une variable tampon correspondante au jour de la colonne récupère les données voulues.

```
    keyElement = str(keyElement)
    reqOffice = '0'
    reqEtablissement = '0'

    for rowList in self.listeEtablissements:
        if rowList[2] == self.lEtablishment.get(self.lEtablishment.curselection()):
            reqOffice = str(rowList[1])
            reqEtablissement = str(rowList[0])

    data = ["NULL", reqOffice, reqEtablissement, self.leDepartement, self.zipCode, '"'+ville+'"', monday, monday, tuesday, tuesday, wed

    keyElement = int(keyElement)

    if(nbElements == limite):
        valeurJournees += '('+(' , ' .join(data)) + ')'

    else:
        valeurJournees += '('+(' , ' .join(data)) + '),\n'


#Génération de la requête SQL en fonction des paramètres récupérés dans le programme
requeteSQL = "INSERT INTO `test`.`round` (`id_round`, `id_office`, `id_establishment`, `id_departement`, `zip_code`, `city`, `monday_a

#Création du fichier script SQL
nomFichier = "insertSQL.sql"

fichierRequete = open(self.leDossier+ "//"+ nomFichier, "w")
fichierRequete.write(requeteSQL)

self.lEtablishment.destroy()

self.champEtablishment.destroy()
self.bouton_envoyer.destroy()
self.champInformations["fg"] = "red"
self.champInformations["text"] = "Votre fichier script " + nomFichier + " est disponible a l'adresse " + self.leDossier

fichierRequete.close()
csvfile.close()
os.remove("csvTemporaire.csv")
```

Le numéro du bureau et de l'établissement concerné sont ensuite récupérés, puis placés dans la liste data accompagné des dates de passage. Cette liste correspond en fait à une ligne de la requête, ligne ensuite ajouté au tableau global. Une fois la boucle terminé, le tableau est concaténé avec la requête, puis inséré dans le fichier script



Petite partie du fichier script généré par l'application

Le développement de cette application a permis de grandement réduire le temps de saisie des données, de la rendre plus simple et plus efficace. Cette activité m'a permis d'apprendre un nouveau langage, le Python. Du fait de sa forte communauté et du nombre de contributeurs, il m'a également imposé naturellement une forte contrainte de veille tout au long du développement, au fur et à mesure des difficultés rencontrées.