

Projet NSI

Tableur (suite)

Louis Guerrero – T^{le} 6

What : un logiciel permettant de faire des tableaux (comme Excel ou Google Sheets). Le logiciel aura les fonctionnalités principales que l'on peut trouver dans un tableur (formules, graphiques...). Ce projet est la suite d'un de mes deux projets de l'année dernière.

For whom : pour toutes les personnes qui ont besoin d'un tableur gratuit et simple.

Why : Parce que j'ai toujours aimé les tableurs. Je trouve ces logiciels très pratiques pour organiser, stocker & traiter des informations.

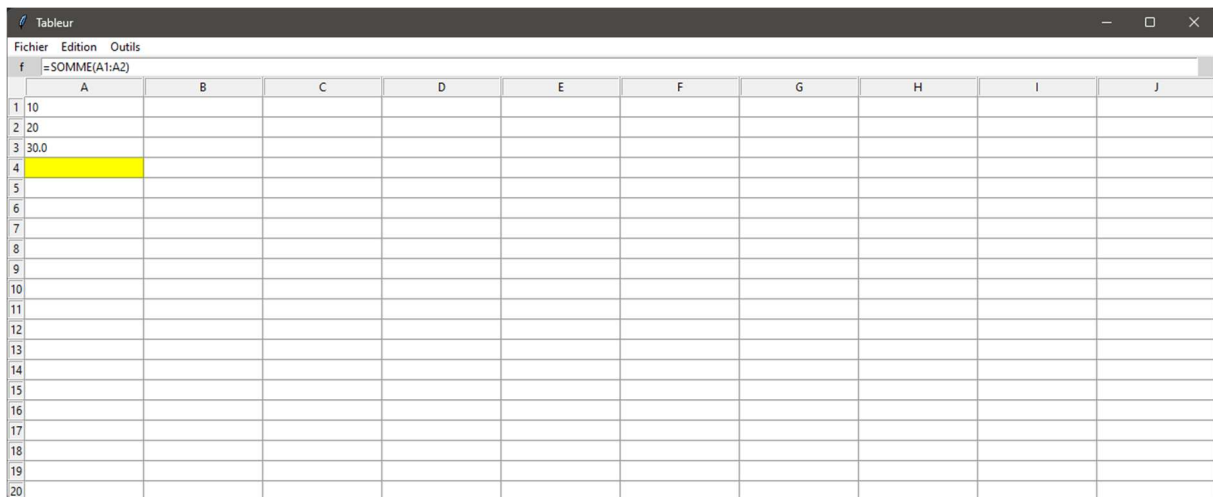
How : Pour continuer voire terminer ce projet, j'ai décidé d'abandonner l'idée d'utiliser SQL pour éviter d'avoir à gérer les données lignes par lignes, limitant donc les fonctionnalités. Je vais donc utiliser le langage Python avec les modules *tkinter* pour l'interface graphique, *pandas* qui permet de sauvegarder et ouvrir des fichiers, et autres. Je vais donc devoir refaire l'entièrement, ou en tout cas une importante partie du programme pour le réadapter à Python uniquement.

J'ai commencé mon projet l'année dernière, d'abord sur Capytale (en classe) mais ai rapidement basculé sur VSC pour des raisons de comptabilité avec les librairies graphique que je souhaitais utiliser.

When : L'année dernière, en classe, j'ai d'abord commencé par « apprendre » le langage SQL sur Capytale. Le reste du programme a été réalisé chez moi. J'ai tout d'abord commencé par faire un affichage du tableau (création du tableau & affichage de celui-ci, sans possibilité d'insérer des valeurs). J'ai ensuite implanté la possibilité d'entrer des valeurs et d'en supprimer. Enfin, j'ai ajouté la possibilité d'enregistrer les tableaux, les ouvrir, et même de les exporter vers Excel. J'ai terminé par faire les fonctions primaires d'un tableur (somme, moyenne), même si pour le moment seule la fonction de somme est utilisable mais les autres sont déjà codés, il faut juste les implanter dans le programme).

Aujourd'hui, je souhaite abandonner l'idée d'utiliser SQL pour rendre le projet plus intéressant et plus simple d'utilisation (avec SQL, il fallait gérer les données lignes par lignes, limitant grandement les possibilités et fonctionnalités). Il va donc falloir réécrire tout (ou en tout cas une grosse partie) du code pour retravailler le programme dans son entièreté et le rendre donc semblable à un tableur que l'on pourrait utiliser tel que Excel. Une fois le code réécrit et réadapter à cette nouvelle idée, nous pourrions y ajouter de nouvelles fonctionnalités, telles que la mise en forme des cellules, des nouvelles formules, la possibilité de trier et filtrer les cellules et autres.

Capture d'écran du projet :



	A	B	C	D	E	F	G	H	I	J
1	10									
2	20									
3	30.0									
4										
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										
15										
16										
17										
18										
19										
20										

Partie de code commenté : formule de somme

```
242     def evaluer_formule(self, valeur: str):
243         """
244         Cette fonction évalue des formules simples comme SOMME(A1:A3), MOYENNE(A1:B2), etc.
245         """
246         logMessage(f"[evaluer_formule] Valeur entrée : {valeur}")
247         try:
248             if re.match(self.formats["SOMME"], valeur):
249                 arguments = valeur[6:-1]
250                 nombres = self.parse_formula_arguments(arguments)
251                 nb = sum(nombres)
252                 logMessage(f"[evaluer_formule] Somme de {nombres} calculée : {nb}", indent=True)
253                 return nb
```

Ligne 242 : on définit une fonction « evaluer_formule », qui prend en paramètre une chaîne de caractère nommée *valeur*.

Ligne 246 : On envoie un message de log.

Ligne 247 : Ouverture d'un bloc Try except.

Ligne 248 : On fait un test dans lequel on vérifie que le format du texte de la cellule correspond à celui d'une somme grâce à une chaîne REGEX.

```
r"^SOMME\\(((\\[A-Z\\][0-9]+|\\[A-Z\\][0-9]+|\\[A-Z\\][0-9]+|\\d+)(\\[;\\]|\\?))*\\[A-Z\\][0-9]+|\\[A-Z\\][0-9]+|\\[A-Z\\][0-9]+|\\d+)\\)$",
```

Ligne 249 : Si le format est correct, alors la valeur de la cellule est reconnue comme une formule de somme. On définit alors les données à traiter en prenant uniquement le texte entre parenthèse dans la formule. On obtient alors une chaîne de caractère comme ceci : « A1:A3 » par exemple.

Ligne 250 : On appelle la méthode *parse_formula_arguments*, qui permet de renvoyer les données correspondant à la plage A1:A3 sous forme de tuple.

Ligne 251 : On calcule la somme des données via la fonction *sum*, présente dans Python.

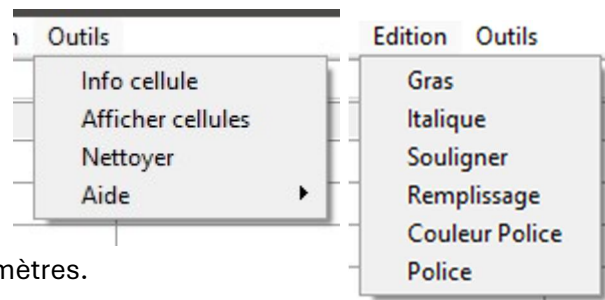
Ligne 252 : On envoie un message de log.

Ligne 253 : On renvoie la somme demandée.

Parcours :

J'ai commencé par recréer l'interface du logiciel, ainsi que les méthodes basiques pour créer et gérer la table. J'ai ensuite repris les méthodes d'enregistrement du premier tableur et les aies adaptés au nouveau fonctionnement de celui-ci. J'ai ensuite travaillé sur les différentes formules utilisables dans le logiciel. Pour commencer, les formules étaient très limitées : on ne pouvait pas faire de formule combinées ($SOMME(A1:A3) * SOMME(B1:B3)$ par exemple) et on ne pouvait mettre que des plages en paramètres ($A1:A3$ par exemple). J'ai donc commencé par rendre possible l'utilisation de constantes et de valeurs simple dans les paramètres des fonctions (on peut maintenant faire $SOMME(A1 :A3 ;A6 ;10)$ par exemple) Par la suite, j'ai décidé de traiter le contenu des cellules de deux manières différentes en fonction de leur contenu : j'ai donc fait une méthode qui permet de définir si la valeur de la cellule contient une formule simple ou bien une formule combinée. Pour les formules combinées, on utilise la fonction `eval()` présente dans Python pour rechercher les fonctions présentes dans la chaine de caractère.

Enfin, j'ai retravaillé l'interface, et ai rajouter des paramètres de mise en forme (souligner, italique, gras, remplissage de la cellule, couleur de la police et changement de la police), des outils (informations d'une cellule (comme ses coordonnées ou encore sa formule brute), aide, nettoyage de la table...) et des paramètres.



Ce qu'il reste à faire :

Il reste quelques petites choses à faire : rendre les paramètres modifiables directement dans le logiciel, régler quelques problèmes avec la barre de formule et certes options de mise en forme (souligner et changer la police notamment), régler des problèmes avec la formule `SI()`. J'aimerais également embellir l'interface avec de vrais boutons à la place d'un menu, notamment pour la mise en forme des cellules.

Conclusion & ce que cela m'a apporté :

Je pense que c'est un projet ambitieux et complexe, qui a besoin de beaucoup d'investissement et qui m'a permis de grandement développer mes compétences en Python (et en SQL grâce à ce que j'ai fait l'année dernière).