TP3. Interface graphique et tests unitaires

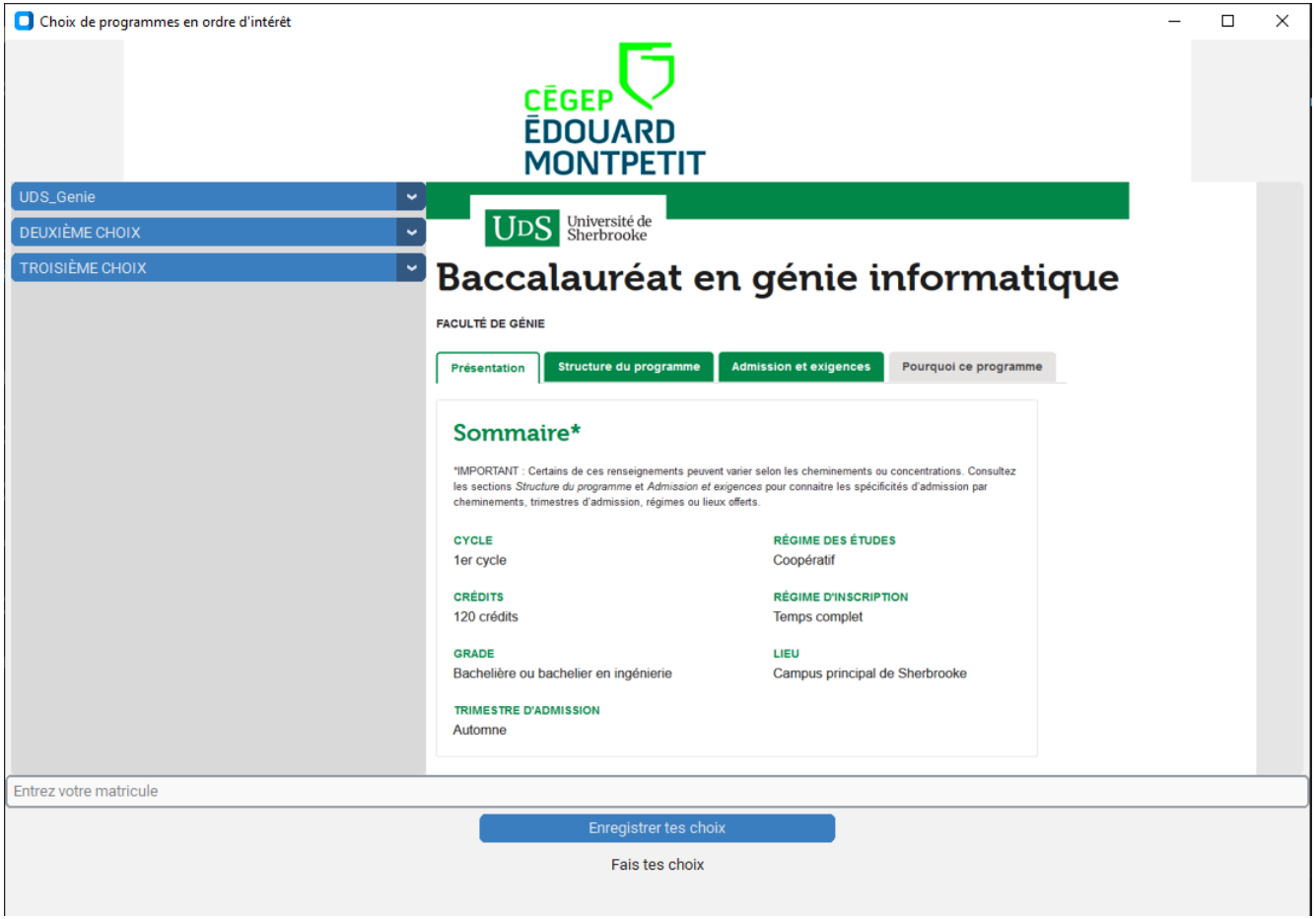
Outils nécessaires :

https://customtkinter.tomschimansky.com/documentation/

Pour effectuer ce travail, vous aurez besoin d'utiliser les librairies customtkinter et pillow. Ces deux librairies ne font pas partie de la librairie standard et doivent être installées avec : **pip install customtkiner**
et **pip install pillow**

Mise en contexte :

Afin de faciliter le passage des études collégiales vers les études universitaires, on désire faire une application qui complètera nos demandes d'inscriptions à notre place.

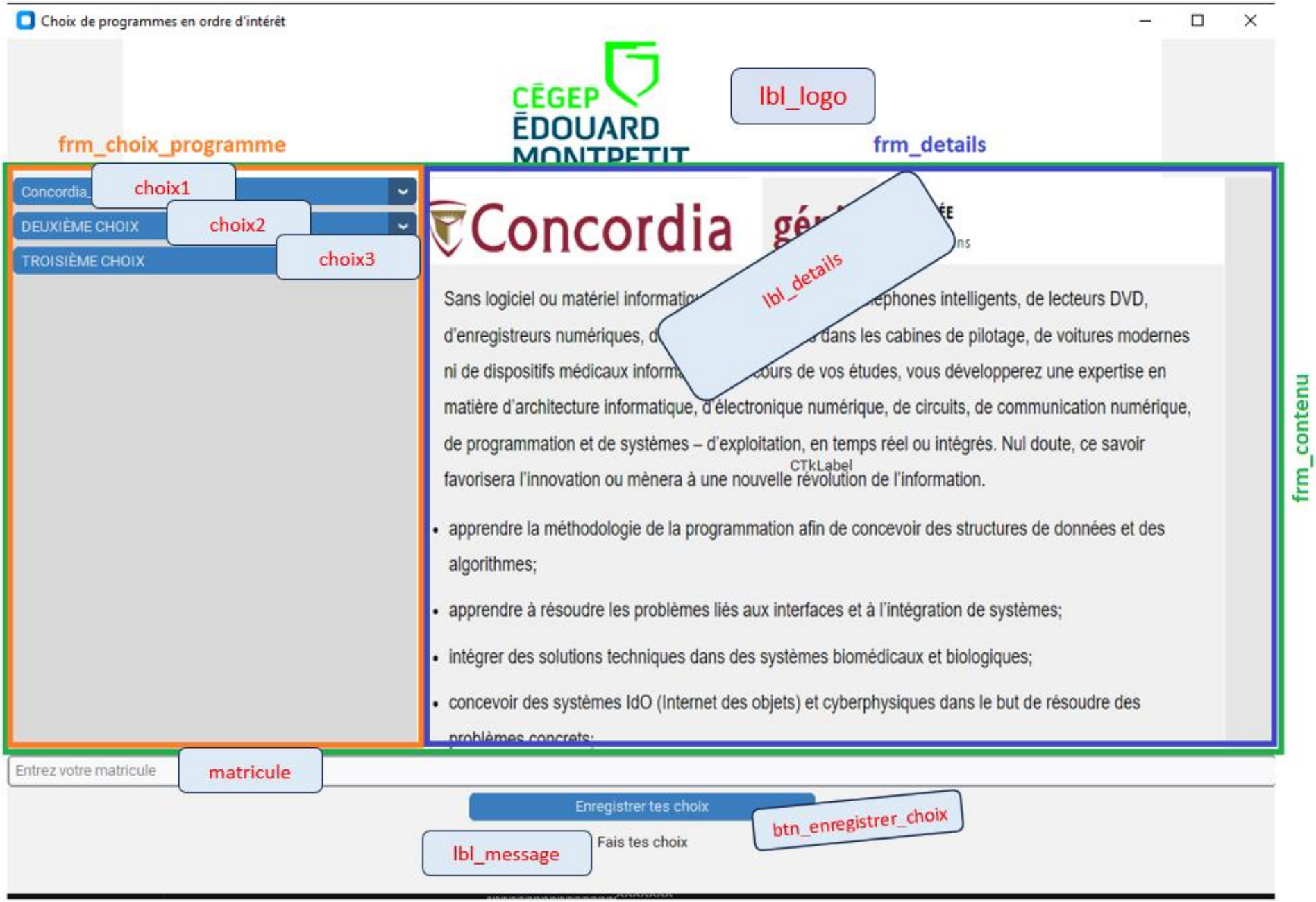


Pour utiliser cette application :

- Vous faites 3 choix.
- Vous entrez votre matricule.
- Vous appuyez sur le bouton pour enregistrer vos choix dans un fichier csv.

Vous devrez faire du code pour l'interface graphique et ensuite les tests unitaires demandés.

Les éléments de l'application graphique sont répartis dans 3 frames et portent les noms suivants :



Partie A - modification à l'interface graphique :

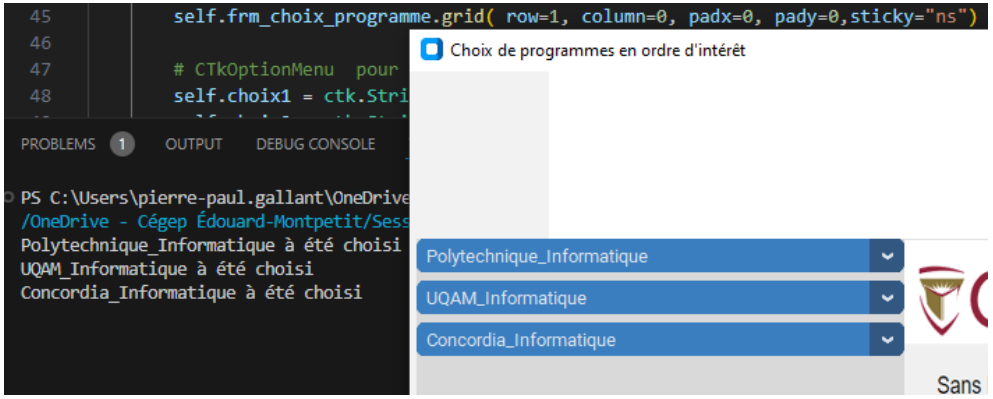
1) Vous devez associer à chaque option de menu de type Ctk.CTkOptionMenu la méthode **`voir_details`()**. Cette méthode prend une valeur autre que le self en paramètre. Une fois lié à une méthode, les objet de type CTkOptionMenu y passent automatiquement le str qu'ils on comme valeur lors de l'appel.
Les menus à option ont déjà été créer mais ne sont pas lié à la méthode.

```
    #CTkOptionMenu pour le choix de programmes
self.choix_1 = ctk.CTkOptionMenu(master=self.frm_choix_programme,
                                 values=['PREMIER CHOIX']+list(self.dict_images.keys()),
                                 command=self.voir_details,
                                 width=350, height=25)

self.choix_2 = ctk.CTkOptionMenu(master=self.frm_choix_programme,
                                 values=['DEUXIÈME CHOIX']+list(self.dict_images.keys()),
                                 width=350, height=25)

self.choix_3 = ctk.CTkOptionMenu(master=self.frm_choix_programme,
                                 values=['TROISIÈME CHOIX']+list(self.dict_images.keys()),
                                 width=350, height=25)
```

NOTE : Regarder la documentation en ligne pour des exemples de code.

En ce moment, la méthode **`voir_details`()** ne fait qu'imprimer le paramètre reçu lors de l'appel. Vous pouvez utiliser cette fonctionnalité pour vérifier que les options de menu et la méthode ont bien été liées.

Quand vous avez terminé ce premier point, changer les valeurs dans les menus à options par l'interface graphique donne le résultat suivant :



2) Vous devez aussi coder la méthode **`voir_details`()**.

La méthode **`voir_details`()** fait référence au self et prend en paramètre la variable associée au menu à option qui l'a appelé. Elle va changer l'image dans la partie droite de l'application pour qu'elle corresponde avec la valeur choisie dans la boite de contrôle.

- Le texte du choix correspond à une clef dans le dictionnaire **dict_images** qui est défini au début du code.
- Ce dictionnaire comprend une paire clef-valeur pour chaque programme d'étude.
  - La clef est un **str** correspondant au nom du programme d'étude
  - La valeur associée est un objet de type **CTkImage**, qui correspond à l'image contenant la description du programme.

```
Paires clefs-valeurs de dict_images :
        Concordia_Genie :       <customtkinter.windows.widgets.image.ctk_image.CTkImage object at 0x00000183AB261A50>
        Concordia_Informatique :        <customtkinter.windows.widgets.image.ctk_image.CTkImage object at 0x00000183AB261ED0>
        Polytechnique_Informatique :    <customtkinter.windows.widgets.image.ctk_image.CTkImage object at 0x00000183AB262210>
        UDS_Genie :     <customtkinter.windows.widgets.image.ctk_image.CTkImage object at 0x00000183AB262610>
        UDS_Informatique :      <customtkinter.windows.widgets.image.ctk_image.CTkImage object at 0x00000183AB262BD0>
        UQAM_Informatique :     <customtkinter.windows.widgets.image.ctk_image.CTkImage object at 0x00000183AB263050>
        ÉTS_Genie :     <customtkinter.windows.widgets.image.ctk_image.CTkImage object at 0x00000183AB2630D0>
```

- Vous devez changer l'image contenue dans l'objet **lbl_détails**.
  - Regardez dans la documentation en ligne comment changer un paramètre d'un objet de ype CTkLabel
  - Regardez comment la valeur initiale d'image est donnée à **lbl_détails.**

3) Vous devez associer au bouton **btn_enregistrer_choix** le code de votre méthode **`enregistrer_choix`()** que vous allez coder.

4) Vous devez coder la méthode **`enregistrer_choix`().** Cette méthode va aller chercher les données dans le formulaire, vérifier que toutes les données nécessaires ont été entrées, puis va enregistrer ces valeurs dans un document texte.

Cette méthode vas appeler deux sous-méthodes : **`_valider_données_entré`()** et **`_écrire_fichier_choix`()**

a. : **`_valider_données_entré`()** va regarder les valeurs dans les options à menus ainsi que la boite d'entrée de matricule.
   Elle écrit ensuite un message approprié dans le **lbl_message** selon la situation :
   i. Si un ou plusieurs choix de programme sont manquant => "Vous devez faire trois choix de programmes"
   ii. S'il y a de la répétition dans les choix => "Vous devez choisir trois programmes différents"
   iii. Si aucun matricule n'a été entrer => "Vous devez entrer votre matricule"
   iv. Si toutes les informations sont valides => "Choix enregistrer"

b. **`_écrire_fichier_choix`()** va prendre un nom de fichier en paramètre et ajouter une nouvelle ligne à la fin de ce fichier. Cette nouvelle ligne contient le matricule de l'étudiant ainsi que ces 3 choix en ordre.

Partie B - Tests unitaires :

Vous devez compléter les tests unitaires qui se trouvent dans le fichier R28_TP3_UnitTest.

Vous avez un point de départ incluant 3 méthodes permettant de facilité les tests sur les des éléments graphiques.

setUpClass() est appelé une fois au tout début de l'ensemble de tests dans la classe.
tearDownClass() est appelé une fois après tous les tests.
setUP() est appelé avant chaque test et remet les valeurs dans le formulaire à leurs valeurs initiales.

```python
import R28_TP3_Choix_programme_Solution as Cs
import unittest

class Test_enregistrer_choix(unittest.TestCase ):
    @classmethod
    def setUpClass(cls):
        cls.choix_programme = Cs.appChoix()
    @classmethod
    def tearDownClass(cls):
        cls.choix_programme.destroy()
    def setUp(self):
        self.choix_programme.choix_1.set("Concordia_Genie")
        self.choix_programme.choix_2.set("DEUXIÈME CHOIX")
        self.choix_programme.choix_3.set("TROISIÈME CHOIX")
        self.choix_programme.matricule.delete(0,'end')
        self.choix_programme.lbl_message.configure(text = "Fais tes choix")
```

En suivant les instructions données par les commentaires de chaque méthode, vous devez faire les tests unitaires demandés. Vous utiliserez la variable de classe **_choix_programme** pour faire les tests. Cette variable correspond à une interface graphique.

Vous avez 7 tests unitaires à faire. Ces tests portent sur la validation des données avant l'enregistrement. Ils assurent que le fonctionnement de l'interface graphique est allant chercher la valeur dans **lbl_message** après une série de manipulations.

Extra A + 1%: (Il s'agit de 1% additionnel à la note finale du cours (max 100%))

Effectuer le test unitaire **`test_ecriture_des_choix`()**. Ce test vérifie qu'**UNE** nouvelle ligne est écrite dans le fichier **ET** que cette ligne correspond à nos attentes.

Extra B +1%:

Les tests **`test_enregistrer_choix_message_OK`** et **`test_ecriture_des_choix`** enregistrent tous deux une nouvelle ligne dans le fichier choix_programmes_étudiants.csv.

Cela peut causer des problèmes, car de nouvelles valeurs sont ajoutées à notre fichier d'enregistrement.

Vous devez ajouter ou modifier les méthodes setUp et tearDown afin de pouvoir effectuer n'importe quel test sans modifier les données dans le fichier csv.

Grille de correction :

Partie A 1)	Lié méthode **`voir_details`** et variables choix aux menus à option	0.5
2)	**`voir_details`** utilise bien la valeur passée en paramètre	0.5
	`voir_details` change l'image affichée dans l'app	0.5
3)	Lié **btn_enregistrer_choix** et **`enregistrer_choix`()**	0.25
4)	Méthode **`enregistrer_choix`()** appelle **`_valider_données_entré`()** et **`_écrire_fichier_choix`()**	0.25
	Méthode **`enregistrer_choix`()** utilise le retour de **`_valider_données_entré`()**	0.25
	`_valider_données_entré`()	
	Va chercher toutes les valeurs du formulaire	0.25
	Fais les trois tests	0.3
	Bon message et bon retour	0.5
	`_écrire_fichier_choix`()	
	Écrit dans le fichier csv	0.3
	N'écrase pas les données existantes	0.3
Partie B	Tests unitaires 1 à 6 fonctionnent	0.5
	Vérifient bien les valeurs dans les éléments graphiques	0.2
	Test unitaire 7 fonctionne	0.2
	Compare bien la ligne ajoutée dans le csv	0.2
Extra A	Faire **`test_ecriture_des_choix`()**	
Extra B	Modifier setUp & tearDown	