

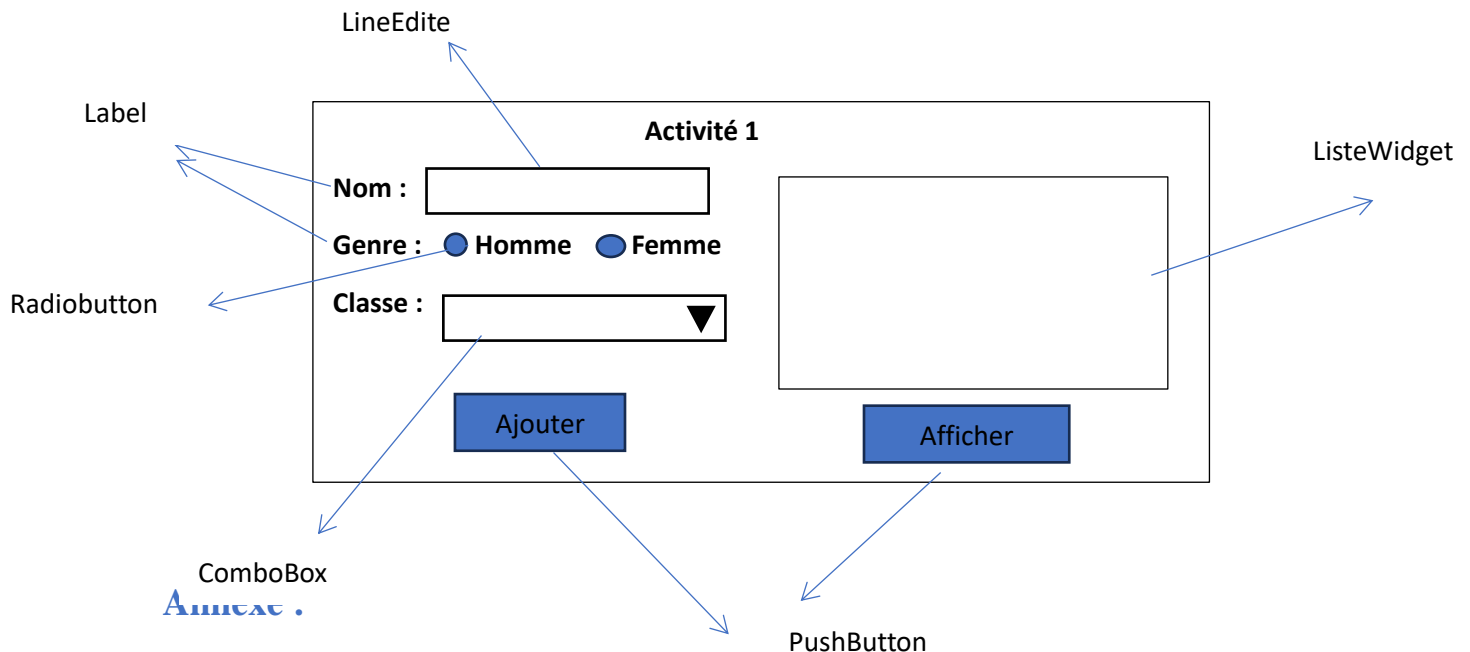
## Séance 12 : Les interfaces graphiques

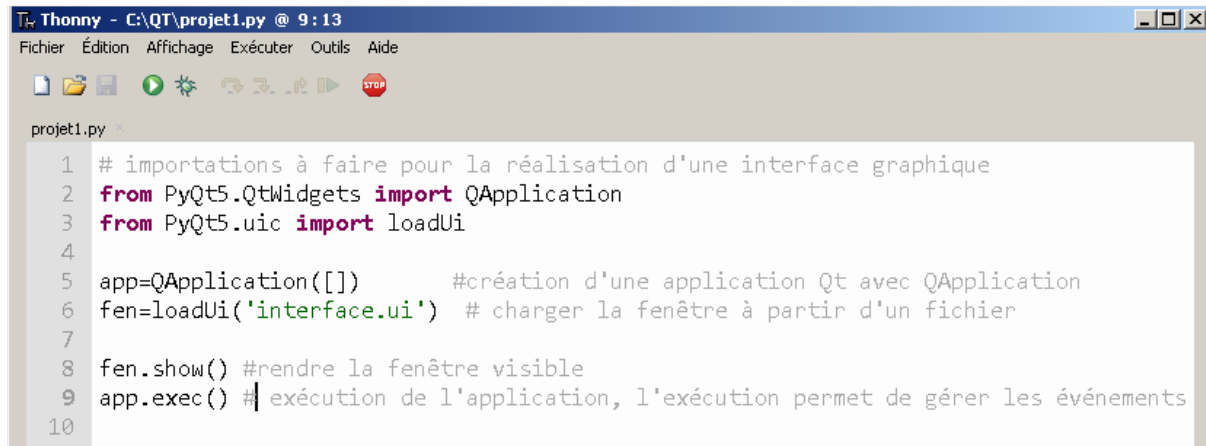
Groupe BacInfo

### Résumé :

Élément	Opération	Méthode
<b>Line Edit</b>	Récupération du texte	<code>win.Nom_Element.text()</code>
<b>Label</b>	Modification du texte	<code>win.Nom_Element.setText("texte")</code>
<b>Combo box</b>	Récupération du contenu	<code>win.Nom_Element.currentText()</code>
<b>Radio button</b>	Vérification cocher/Non cocher	<code>win.Nom_Element.isChecked()</code>
<b>Check Box</b>	Vérification cocher/Non cocher	<code>win.Nom_Element.isChecked()</code>
<b>Liste Widget</b>	Ajouter élément	<code>win.Nom_Element.addItem("contenu")</code>
	Récupération du contenu d'un Item	<code>win.Nom_Element.item(indice).text()</code>
	Initialisation du nombre des lignes	<code>win.Nom_Element.setRowCount(nombre de ligne)</code>
	Initialisation du nombre des colonnes	<code>win.Nom_Element.setColumnCount(nombre de ligne)</code>
<b>Table Widget</b>	Insertion nouvelle ligne	<code>win.Nom_Element.insertRow(indice du ligne)</code>
	Insertion nouvelle ligne	<code>win.Nom_Element.insertColumn(indice du colonne)</code>
	Modification du contenu d'un Item	<code>win.Nom_Element.setItem(indice_ligne, indice_colonne, QTableWidgetItem("contenu"))</code>
	Récupération du contenu d'un Item	<code>win.Nom_Element.item(indice_ligne, indice_colonne).text()</code>
<b>Text Edit</b>	Récupération du contenu	<code>win.Nom_Element.toPlainText()</code>
	Insertion du contenu	<code>win.Nom_Element.setPlainText("Contenu")</code>
	Ajouter une ligne	<code>win.Nom_Element.append("Contenu")</code>

### Activité :





```
Thonny - C:\QT\projet1.py @ 9:13
Fichier  Édition  Affichage  Exécuter  Outils  Aide

projet1.py
1  # importations à faire pour la réalisation d'une interface graphique
2  from PyQt5.QtWidgets import QApplication
3  from PyQt5.uic import loadUi
4
5  app=QApplication([])          #création d'une application Qt avec QApplication
6  fen=loadUi('interface.ui')    # charger la fenêtre à partir d'un fichier
7
8  fen.show() #rendre la fenêtre visible
9  app.exec() # exécution de l'application, l'exécution permet de gérer les événements
10
```

Travail demandé :

- Remplir un fichier texte nommé school.txt par les informations saisies dans l'interface.
- Afficher tout le contenu de fichier school.txt dans la list widget lw.

**Correction :**

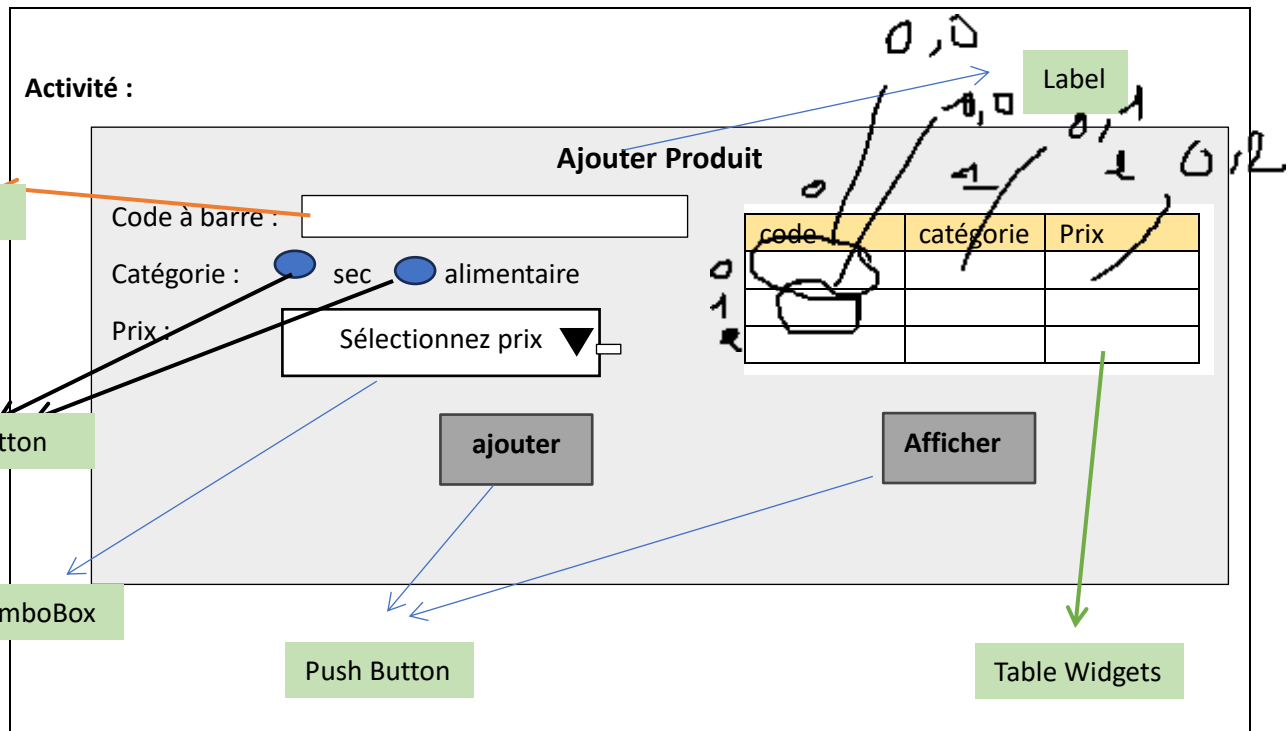
```
from PyQt5.uic import *
from PyQt5.QtWidgets import *
def ajouter():
    F=open("school.txt","a")
    n=fen.nom.text()
    if fen.h.isChecked():
        g="homme"
    elif fen.f.isChecked():
        g="femme"
    else:
        g=""

    c=fen.classe.currentText()
    if n==" " or g=="":
        QMessageBox.critical(fen,"Erreur","Veuillez introduire tout les
donners")
    else:
        F.write(n+" "+g+" "+c+"\n")
        QMessageBox.information(fen,"Success","L'élève "+n+" ajouter avec
succès")
    F.close()
def affiche():
    F=open("school.txt","r")
    ch=F.readline()
    while ch!="":
        fen.lw.addItem(ch)
        ch=F.readline()
    F.close()

app=QApplication([])
fen=loadUi('act1.ui')
fen.show()
```

```
fen.add.clicked.connect(ajouter)
fen.aff.clicked.connect(affiche)
app.exec_()
```

## Activité 2 :



Créer cette interface pour remplir un fichier binaire nommé **magasin.dat**

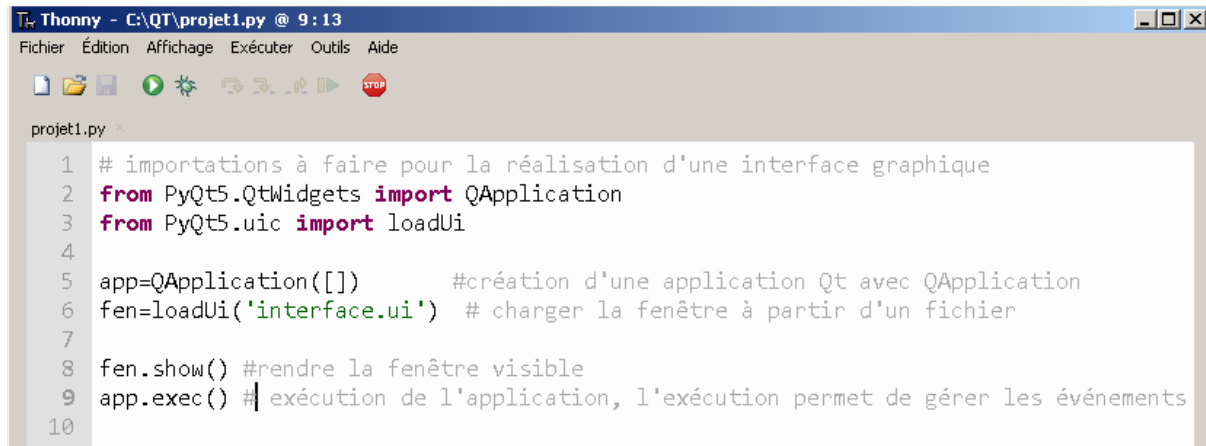
La fichier magasin contient des enregistrement nommée **produit** représenter par les champs suivants :

- **Code** : chaine (8 chiffres) (non vide)
- **Catégorie** : sec ou alimentaire(non vide)
- **Prix** : réel (non vide)

### Travail demandé :

1. Créer l'interface et nommée **store.ui (20 min)**
2. Développer le module **ajouter** pour ajouter des produits dans le fichier magasin.dat
3. Développer le module **afficher** pour afficher la liste des produits

\*\*\*\*\*Annexe\*\*\*\*\*



```
Thonny - C:\QT\projet1.py @ 9:13
Fichier  Édition  Affichage  Exécuter  Outils  Aide

projet1.py
1  # importations à faire pour la réalisation d'une interface graphique
2  from PyQt5.QtWidgets import QApplication
3  from PyQt5.uic import loadUi
4
5  app=QApplication([])          #création d'une application Qt avec QApplication
6  fen=loadUi('interface.ui')    # charger la fenêtre à partir d'un fichier
7
8  fen.show() #rendre la fenêtre visible
9  app.exec() # exécution de l'application, l'exécution permet de gérer les événements
10
```

### Interface Qt :

```
from PyQt5.uic import *
from PyQt5.QtWidgets import *
from pickle import dump,load
def ajouter() :
    F=open("magasin2.dat","ab")
    p=dict()
    p["code"]=fen.code.text()
    if fen.S.isChecked():
        p["cat"]="Sec"
    elif fen.A.isChecked():
        p["cat"]="Alimentaire"
    else:
        p["cat"]=" "
    p["prix"]=fen.pr.currentText()

    if p["code"]==" " or p["cat"]==" ":
        QMessageBox.critical(fen,"Erreur","NULL INTERDIT !!")
    elif not(len(p["code"])==8 and p["code"].isdigit()):
        QMessageBox.critical(fen,"Erreur","INVALIDE CODE !!!")
    else:
        dump(p,F)
        QMessageBox.information(fen,"SUCCES","GOOD JOB !!")
    F.close()
def afficher() :
    F=open("magasin2.dat","rb")
    i=0
    fin=False
    while fin==False:
        try:
            p=load(F)
            fen.tw.insertRow(i)
            fen.tw.setItem(i,0,QTableWidgetItem(p["code"]))
            fen.tw.setItem(i,1,QTableWidgetItem(p["cat"]))
            fen.tw.setItem(i,2,QTableWidgetItem(p["prix"]))
            i=i+1
        except:
            fin=True
    F.close()
app=QApplication([])
fen=loadUi("store.ui")
fen.show()
```

```
fen.add.clicked.connect(ajouter)
fen.aff.clicked.connect(afficher)
app.exec_()
```