



Création des interfaces graphiques avec Qt Designer et Python

⊙ C'est quoi une interface graphique d'une application ?

► Activité 1

En utilisant l'éditeur de Python disponible sur l'ordinateur, ouvrir le fichier « *Factorielle.py* » existant dans le dossier « **FormationQt** » situé sur le lecteur C, l'exécuter puis dégager son rôle.

► Activité 2

En utilisant l'éditeur de Python disponible sur l'ordinateur, ouvrir le fichier « *Factorielleinterface.py* » existant dans le dossier « **FormationQt** » situé sur le lecteur C, l'exécuter puis dégager son rôle.

► Activité 3

Comparer les exécutions des deux programmes puis dégager les constatations nécessaires.

► Activité 4

Le code python du programme « *Factorielleinterface.py* » est présenté dans le tableau ci-dessous. Remplir la colonne rôle par les rôles de la liste suivante :

/ Événement sur le bouton / Lecture d'un fichier externe avec l'extension « ui » / Bibliothèques /
/ Création de l'application graphique / Fonction écrite par l'utilisateur /
/ Affichage de l'interface graphique /

Bloc de code	Rôle
<code>from PyQt5.uic import loadUi</code>	
<code>from PyQt5.QtWidgets import QApplication</code>	
<code>def factorielle () :</code>	
<code>N = int (fenetre.Nb.text ())</code>	
<code>f = 1</code>	
<code>for i in range (2 , N+1) :</code>	
<code>f = f * i</code>	
<code>fenetre.Res.setText ("La factorielle = " + str (f))</code>	
<code>app = QApplication ([])</code>	
<code>fenetre = loadUi ("interfacefact.ui")</code>	
<code>fenetre . show ()</code>	
<code>fenetre.btcalcul.clicked.connect (factorielle)</code>	
<code>app.exec_ ()</code>	

► Activité 5

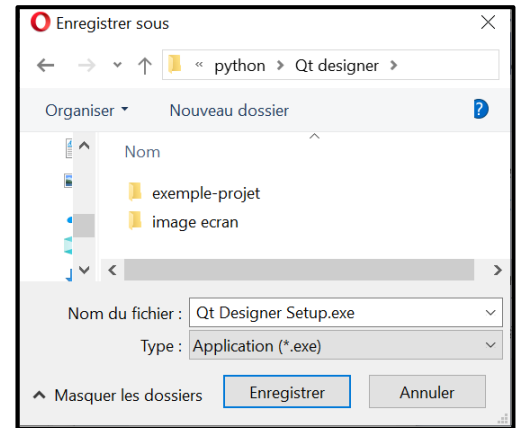
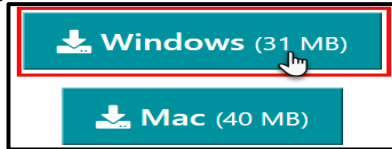
Dans le tableau ci-dessous, on présente un extrait du programme « *Factorielleinterface.py* », désactiver chaque bloc de code, puis remplir la colonne obligatoire par « **Oui** » si le bloc de code est indispensable à la création de l'interface graphique ou par « **Non** » dans le cas contraire.

Bloc de code	Obligatoire (Oui / Non)
<code>from PyQt5.uic import loadUi</code>	
<code>from PyQt5.QtWidgets import QApplication</code>	
<code>app = QApplication ([])</code>	
<code>fenetre = loadUi ("interfacefact.ui")</code>	
<code>fenetre . show ()</code>	
<code>app.exec_ ()</code>	

🕒 Téléchargement de Qt Designer

👉 Marche à suivre

- 1) Sur la barre d'adresse du navigateur, taper l'adresse URL suivante :
«<https://build-system.fman.io/qt-designer-download>»
- 2) Cliquer sur le bouton « **Windows (31 MB)** ».
- 3) Enregistrer le fichier d'installation
« **Qt Designer Setup.exe** ».
- 4) Installer le logiciel et l'exécuter.



🕒 Installation du plugin « **thonny-tunisiaschools 0.0.6** »

▶ Activité 6

Dans le logiciel « **Thonny** », installer le plugin « **thonny-tunisiaschools 0.0.6** », sachant que le fichier d'installation « **thonny_tunisiaschools-0.0.6-py3-none-any.whl** » existe dans le dossier « **FormationQt** » situé sur le lecteur C.

🔒 Important :

- Sur le lecteur C, créer un dossier qui va contenir tous les travaux effectués.
- La création et la modification des fichiers à extension « **.ui** » doit être manipulé en utilisant le logiciel « **Qt Designer** ».
- La création et la modification des fichiers à extension « **.py** » doivent être effectués avec le logiciel « **Thonny** ».
- Les méthodes des objets utilisés pour la résolution des problèmes existent dans l'annexe.

🕒 Création d'une nouvelle fenêtre de dialogue sans boutons

▶ Activité 7

- 1/ Créer un nouveau formulaire de type « **Dialog without button** » et l'enregistrer sous le nom « **Bienvenue.ui** » dans le dossier de travail.
- 2/ Modifier l'interface graphique comme indiqué ci-dessous :
 - a) Modifier le titre de votre fenêtre par « **Application N° 1** ».
 - b) Insérer une zone de texte et afficher le mot « **Bienvenue** » dans son libellé.
 - c) Modifier la police, le style de la police et la taille de la zone de texte « **Bienvenue** ».
- 3/ Visualiser le formulaire créé.

🕒 Appel d'un fichier Qt designer dans un fichier Python

▶ Activité 8

En utilisant le logiciel « **Thonny** », créer un nouveau fichier, utiliser le bouton « **Ajouter code PyQt5** » qui permet d'insérer un script Python permettant d'appeler le fichier « **Bienvenue.ui** », l'enregistrer sous le nom « **interface1.py** » puis l'exécuter.

🕒 Insertion d'un bouton « QPushButton » dans un formulaire

► Activité 9

- Insérer dans le formulaire « *Bienvenue.ui* », un bouton de type « **Push button** », modifier son texte par « **Cliquer ici** » et le nommer par le nom « **Bouton1** ».
- Utiliser le signal de l'interface pour découvrir les différents événements qu'on peut déclencher avec ce bouton puis compléter la colonne « **Nom événement** » du tableau ci-dessous par l'événement correspondant à son rôle.

Nom événement	Rôle
	Déclencher quand le bouton est cliqué.
	Déclencher quand le bouton est pressé.
	Déclencher quand le bouton est relâché
	Déclencher quand l'attribut « checkable=Oui ».

► Activité 10

En utilisant le logiciel « **Thonny** », créer un nouveau fichier, utiliser le bouton « **Ajouter code PyQt5** » qui permet d'insérer un script Python permettant d'appeler le fichier « *Bienvenue.ui* », modifier l'événement sur le bouton « **Bouton1** » qui permet de remplacer le mot « **Bienvenue** » par « **Bonne Année 2023** » suite au clic sur le bouton, l'enregistrer sous le nom « *interface2.py* » puis l'exécuter.

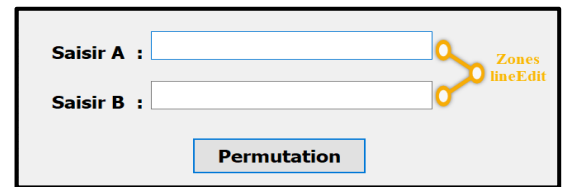
🔗 Code :

```
# Importation des bibliothèques du graphique
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
# La procédure modification qui permet la modification du label
def modification () :
    fenetre.label.setText ("Bonne année 2023")
# Création de l'application graphique
app = QApplication( [ ] )
# Lecture du fichier crée par Qt designer dans une variable interface nommée « fenetre »
fenetre = loadUi ( "bienvenue.ui" )
# Exécution de l'application
fenetre . show ( )
# Programmation du déclencheur d'évènement sur le bouton afin d'exécuter la procédure modification
fenetre.bt1.clicked.connect ( modification )
app.exec_ ( )
```

⊙ Manipulation des zones de saisies « QlineEdit »

► Activité 11

Créer le fichier « *interfacepermut.ui* », contenant le formulaire ci-contre :



Ecrire un nouveau fichier « *prog_permut.py* » qui contiendra une procédure nommée « **permuter** » qui permet lors du clic sur le bouton «**Permutation**», de permuter le contenu de la zone de saisie de A avec le contenu de la zone de saisie de B.

↳ Code :

```
#importation des bibliothèques du graphique
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
#La procédure permuter qui permet la permutation
def permuter () :
    a = fenetre.edita.text ()
    b = fenetre.editb.text ()
    fenetre.editb.setText ( a )
    fenetre.edita.setText ( b )
# Création de l'application graphique
app = QApplication ( [ ] )
# Lecture du fichier crée par Qt designer dans une variable interface nommée « fenetre »
fenetre = loadUi ( "interfacepermut.ui " )
# Affichage de la fenêtre
fenetre.show ()
# Programmation du déclencheur d'évènement sur le bouton afin d'exécuter la procédure permuter
fenetre.bt1.clicked.connect ( permuter )
app.exec_ ( )
```

⊙ Manipulation des listes déroulantes « QComboBox »

► Activité 12

On se propose de simuler une calculatrice qui effectue les opérations élémentaires de calcul + , - , * et / de deux nombres.

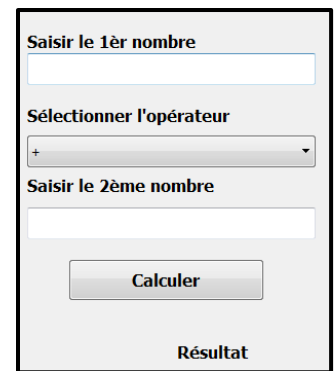
Pour cela, on se propose de concevoir l'interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**Saisir le 1er Nombre**".
- Une zone de saisie du 1^{er} nombre.
- Un label contenant le texte : "**Sélectionner l'opérateur**".
- Une liste déroulante contenant les options : + , - , * et /.
- Un label contenant le texte : "**Saisir le 2ème Nombre**".
- Une zone de saisie du 2^{ème} nombre.
- Un bouton intitulé "**Calculer**" qui permet de réaliser le calcul correspondant.
- Un label contenant le texte "**Résultat**" permettant d'afficher le résultat du calcul.

Travail demandé :

Copier le fichier « *calculatrice.ui* » situé dans « C:\ FormationQt » dans le dossier de travail.

- 1) Compléter l'interface graphique du fichier « *calculatrice.ui* » par les éléments nécessaires comme le montre la figure ci-contre.
- 2) Créer un nouveau code Python et l'enregistrer sous le nom « *prog_calcul.py* » qui permet lors du clic sur le bouton "Calculer" d'afficher dans la zone "Résultat" le résultat de l'opération de calcul effectuée sur le 1^{er} et le 2^{ème} nombre selon l'opérateur sélectionné.



N.B. :

- Le 1^{er} et le 2^{ème} nombre sont des valeurs numériques (le contrôle n'est pas demandé).
- Les traitements de calcul existent dans le fichier « *code.txt* » situé dans le dossier « FormationQt » du lecteur C.

Code :

#Importation des bibliothèques du graphique

```
from PyQt5.uic import loadUi
```

```
from PyQt5.QtWidgets import QApplication
```

#Fonction qui réalise l'addition, la soustraction, la multiplication et la division

```
def calculelementaire ( nb1 , nb2 , op ) :
```

```
    if (op == "+") :
```

```
        res = nb1 + nb2
```

```
    elif (op == "-") :
```

```
        res = nb1 - nb2
```

```
    elif (op == "/" ) :
```

```
        res = nb1 / nb2
```

```
    elif (op == "*" ) :
```

```
        res = nb1 * nb2
```

```
    return res
```

#Procédure qui permet le calcul

```
def calcul ( ) :
```

```
    nb1 = int ( fenetre.nb1.text ( ) )
```

```
    nb2 = int ( fenetre.nb2.text ( ) )
```

```
    op = fenetre.op.currentText ( )
```

```
    res = calculelementaire ( nb1 , nb2 , op )
```

```
    fenetre.Res.setText ( "Le résultat = " + str ( res ) )
```

#Création de l'application graphique

```
app = QApplication( [ ] )
```

#Lecture du fichier crée par Qt designer dans une variable interface nommée « fenetre »

```
fenetre = loadUi ( "calculatrice.ui" )
```

#Affichage de la fenêtre

```
fenetre.show ( )
```

Programmation du déclencheur d'évènement sur le bouton afin d'exécuter la procédure calcul

```
fenetre.bt1.clicked.connect ( calcul )
```

```
app.exec_ ( )
```

☉ Manipulation des boutons radio « QPushButton »

► Activité 13

On se propose de simuler une calculatrice arithmétique qui permet de calculer le **PGCD**, le **PPCM** et la **Puissance** de deux nombres.

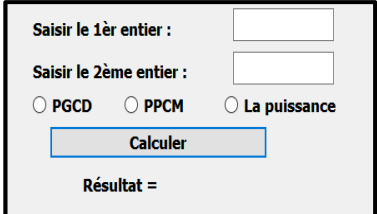
Pour cela, on se propose de concevoir l'interface graphique contenant les éléments suivants :

- Un label contenant le texte : "Saisir le 1er Nombre"
- Une zone de saisie du 1^{er} nombre.
- Un label contenant le texte : "Saisir le 2ème Nombre"
- Trois cases radios nommées respectivement "R1", "R2" et "R3" dont leurs libellés respectifs sont : "PGCD", "PPCM" et "Puissance".
- Un bouton intitulé "Calculer", qui permet de réaliser le calcul correspondant.
- Un label contenant le texte "Résultat =" permettant d'afficher le résultat du calcul.

Travail demandé :

Copier le fichier « *CalculMath.ui* » situé dans « C:\FormationQt » dans le dossier de travail.

1. Compléter l'interface graphique du fichier « *CalculMath.ui* » par les éléments nécessaires comme le montre la figure ci-contre.
2. Créer un nouveau code Python et l'enregistrer sous le nom « *arithmetique.py* », écrire la procédure « *calcul* » qui se déclenche lors du clic sur le bouton "Calculer". Cette procédure permet de calculer le **PGCD**, **PPCM** et la **Puissance** de 2 nombres selon le choix de l'utilisateur puis d'afficher le résultat dans la zone "Résultat =".



N.B. : Les traitements du **PGCD**, **PPCM** et de la **Puissance** existent dans le fichier « *code.txt* » existant dans le dossier « **FormationQt** » du lecteur C.

↳ Code :

```
#Importation des bibliothèques du graphique
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication

# Fonction Puissance
def puissance (N, x ):
    p = 1
    for i in range ( x ):
        p = p * N
    return p

# Fonction PGCD
def PGCD ( a , b ):
    while b != 0 :
        r = a %b
        a = b
        b = r
    return a

# Fonction PPCM
def PPCM ( a , b ):
    c = a
    d = b
```



```

while (a != b):
    if (a > b):
        b = b + d
    else:
        a = a + c
return a
#Procédure qui permet le calcul
def calcul():
    Nb1 = int(fenetre.Nb1.text())
    Nb2 = int(fenetre.Nb2.text())
    if (fenetre.R1.isChecked() == True):
        resultat = PGCD(Nb1, Nb2)
    elif (fenetre.R2.isChecked() == True):
        resultat = PPCM(Nb1, Nb2)
    elif (fenetre.R3.isChecked() == True):
        resultat = puissance(Nb1, Nb2)
    fenetre.Res.setText(str(resultat))
#création application graphique
app = QApplication([])
#Chargement du fichier Qt designer
fenetre = loadUi("CalculMath.ui")
#affichage de la fenêtre graphique
fenetre.show()
#programmation du déclencheur d'évènement sur le bouton afin d'exécuter la procédure calcul
fenetre.bt.clicked.connect(calcul)
app.exec_()

```

⦿ Manipulation des zones de saisie à plusieurs lignes « QText Edit »

► Activité 14

On se propose de créer un logiciel de saisie de texte qui ressemble à l'application « **Bloc-notes** » de Windows. Celui-ci offre à l'utilisateur la possibilité de remplir et de lire le contenu du fichier texte intitulé « *phrases.txt* » comportant une suite de phrases. Une ligne de ce fichier contient une seule phrase.

Pour ce faire, on se propose de concevoir l'interface graphique contenant les éléments suivants :

- Un label contenant le texte : "Saisir votre Texte"
- Une zone de saisie à plusieurs lignes permettant la saisie des phrases.
- Un bouton intitulé "Enregistrer" qui permet d'enregistrer le texte saisi dans le fichier « *phrases.txt* ».
- Un bouton intitulé "Ouvrir" qui permet de charger le contenu du texte « *phrases.txt* » dans la zone de saisie.
- Un bouton intitulé "Effacer" qui permet d'effacer le contenu de la zone de saisie.

Travail demandé :

Copier le fichier « *interbloccnote01.ui* » situé dans « C:\ FormationQt » dans le dossier de travail.

1. Compléter l'interface graphique du fichier « *interbloccnote01.ui* » par les éléments nécessaires comme le montre la figure ci-contre.

2. Créer un nouveau code Python et l'enregistrer sous le nom « *blocnote.py* » qui doit contenir :
 - La procédure « **enregistrement** » qui se déclenche lors du clic sur le bouton « **Enregistrer** » et qui permet de sauvegarder le contenu de la zone de texte dans le fichier « *phrases.txt* » situé dans le dossier de travail.
 - La procédure « **chargement** » qui se déclenche lors du clic sur le bouton "Ouvrir" et qui permet de charger dans la zone de saisie, le contenu du fichier « *phrases.txt* » situé dans le dossier de travail.
 - La procédure « **suppression** » qui se déclenche lors du clic sur le bouton "Effacer" et qui permet d'effacer le contenu de la zone de saisie.

🔗 Code :

```
#Importation des bibliothèques du graphique
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
#Procédure permettant l'enregistrement du contenu de zone de textes dans le fichier
def enregistrement () :
    ch = fenetre.textEdit.toPlainText ()
    fph = open ( "phrases.txt" , "w" )
    fph.write ( ch )
    fph.close ()
#Procédure permettant la suppression
def suppression () :
    fenetre.textEdit.clear ()
#Procédure chargement qui permet le chargement du contenu du fichier
def chargement () :
    fph = open ( "phrases.txt" , "r" )
    ch = fph.read ()
    fenetre.textEdit.setPlainText ( ch )
    fph.close ()
#Création application graphique
app = QApplication ( [] )
#Chargement du fichier Qt designer
fenetre = loadUi ( "interfaceblocnote.ui" )
#Affichage de la fenêtre graphique
fenetre.show ()
#Programmation des déclencheurs d'évènement sur les boutons
fenetre.enregistrer.clicked.connect ( enregistrement )
fenetre.ouvrir.clicked.connect ( chargement )
fenetre.effacer.clicked.connect ( suppression )
app.exec_ ()
```

🕒 Manipulation des Listes « QListWidget »

► Activité 15

On se propose de créer un logiciel de saisie de texte qui ressemble à l'application « **Bloc-notes** » de Windows. Celui-ci offre à l'utilisateur la possibilité de remplir un fichier texte intitulé « *phrases.txt* » comportant une suite de phrases. Une ligne de ce fichier contient une seule phrase. Il permet aussi d'apporter aux phrases de ce fichier les corrections suivantes :

- on enlève les espaces superflus pour n'en garder qu'un seul entre deux mots successifs,
- on enlève éventuellement le ou les espaces au début et à la fin de chaque phrase,
- on ajoute un point final à chaque phrase s'il en manque.

Pour ce faire, on se propose de concevoir l'interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**Saisir votre Texte**"
- Une zone de saisie à plusieurs lignes permettant la saisie des phrases.
- Un bouton intitulé "**Enregistrer**", qui permet d'enregistrer le texte saisi dans le fichier « *phrases.txt* ».
- Un bouton intitulé "**Corriger**" qui permet la lecture du contenu du fichier « *phrases.txt* », effectuer les corrections demandées précédemment et d'afficher les phrases corrigées à raison d'une phrase par ligne dans la zone **Liste Widget**.
- Un label contenant le texte : "**Résultat des corrections :**".
- Une **Liste Widget** pour afficher les phrases corrigées.

Travail demandé :

Copier le fichier « *interfaceblocnote02.ui* » situé dans « C:\FormationQt » dans le dossier de travail.

1. Compléter l'interface graphique du fichier « *interfaceblocnote02.ui* » par les éléments nécessaires comme le montre la figure ci-contre.

2. Créer un nouveau code Python et l'enregistrer sous le nom « *blocnote02.py* » qui doit contenir :

- La procédure « **enregistrement** » qui se déclenche lors du clic sur le bouton "**Enregistrer**" et qui permet de sauvegarder le contenu de la zone de texte dans le fichier « *phrases.txt* » situé dans le dossier de travail.
- La procédure « **Correction** » qui se déclenche lors du clic sur le bouton "**Corriger**", qui permet la lecture du contenu du fichier « *phrases.txt* » situé dans le dossier de travail, d'effectuer les corrections demandées précédemment et d'afficher les phrases corrigées à raison d'une phrase par ligne dans la zone **Liste Widget**.

N.B. : Les traitements de correction des phrases existent dans le fichier « *code.txt* » existant dans le dossier « **FormationQt** » du lecteur C.

Code :

#Importation des bibliothèques du graphique

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
```

#Procédure permettant l'enregistrement du contenu de zone de textes dans le fichier

```
def enregistrement () :
    ch = fenetre.textEdit.toPlainText ()
    fph = open ( "phrases.txt" , "w" )
    fph.write ( ch )
    fph.close ()
```

#Fonction qui efface des espaces supplémentaires et l'espace au début et à la fin

```
def nettoyage ( ch ) :  
    while ( ch.find ( " " ) != -1 ) :  
        p = ch.find ( " " )  
        ch = ch[ : p ] + ch[ p+1 : ]  
    if ( ch[ len( ch ) - 1 ] == " " ) :  
        ch = ch [ : len(ch) - 1 ]  
    if ( ch[0] == " " ) :  
        ch = ch [ 1 : ]  
    if ( ch[ len( ch ) - 1 ] != "." ) :  
        ch = ch + "."  
    return ch
```

#Procédure d'affichage des phrases corrigées à raison d'une ligne par ligne dans la zone List Widget

```
def correction ( ) :  
    fph = open ( "phrases.txt" , "r" )  
    ch = fph.readline ( )  
    while ch != "" :  
        chcor = nettoyage ( ch.rstrip ( ) )  
        fenetre.liste1.addItem ( chcor )  
        ch = fph.readline ( )  
    fph.close ( )
```

#Création application graphique

```
app = QApplication ( [ ] )
```

#Chargement du fichier Qt designer

```
fenetre = loadUi ( "interfaceblocnote02.ui" )
```

#Affichage de la fenêtre graphique

```
fenetre.show ( )
```

#programmation des déclencheurs d'évènement sur les boutons

```
fenetre.enregistrer.clicked.connect ( enregistrement )
```

```
fenetre.Corriger.clicked.connect ( correction )
```

```
app.exec_ ( )
```

☉ Manipulation « QMessageBox » et « QWidget »

► Activité 16

Pour augmenter l'audience de ses émissions, une chaîne de télévision a organisé un jeu qui consiste à demander aux téléspectateurs de rappeler le classement des quatre pays demi-finalistes de la dernière coupe de monde de football «**Qatar 2022**».

Les quatre pays demi-finalistes sont :

- L'Argentine, représentée par la lettre "A".
- La France, représentée par la lettre "F".
- La Croatie, représentée par la lettre "C".
- Le Maroc, représentée par la lettre "M".

Chaque téléspectateur est appelé à envoyer par SMS, sa proposition de classement en mettant les lettres représentant chacun des pays demi-finalistes selon leur classement à la fin de la coupe du monde Qatar 2022. Ainsi, le texte d'un SMS n'est considéré valide que s'il contient une chaîne formée exactement des quatre lettres "A", "F", "C" et "M" et que chacune d'elles n'apparaît qu'une seule fois dans cette chaîne.

N.B. : le programme ne fait de distinction entre les lettres majuscules et minuscules.

Exemples :

- Si le téléspectateur envoie la chaîne "AFFM", son SMS est considéré comme invalide car il ne contient pas la lettre "C".
- Si le téléspectateur envoie la chaîne "AFCM", son SMS est considéré comme valide car il contient les quatre lettres "A", "F", "C" et "M".

Tâche N° 1 : Manipulation des messages « QMessageBox »

On se propose de créer un programme qui permet de remplir un fichier « **SMS.dat** » par les SMS valides des téléspectateurs. Sachant que chaque SMS est formé par :

- Le numéro du téléphone du téléspectateur doit être formé de 8 chiffres dont le premier chiffre doit être « 9 » ou « 2 ».
- La chaîne du SMS doit être formée par les quatre lettres "A", "F", "C" et "M" sans répétition.

Pour cela, on se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**Saisir le numéro de téléphone :**"
- Une zone de saisie du numéro de téléphone.
- Un label contenant le texte : "**Saisir le SMS :**".
- Une zone de saisie de la chaîne du SMS.
- Un bouton intitulé "**Enregistrer**" qui permet d'enregistrer le SMS saisi dans le fichier « **SMS.dat** » si les contraintes de la saisie des champs citées ci-dessus sont respectées, ou d'afficher un message d'erreur via "**QMessageBox**" dans le cas contraire.

Travail demandé :

Copier le fichier « **SMS.ui** » situé dans « **C:\ FormationQt** » dans le dossier de travail.

Créer un nouveau code Python et l'enregistrer sous le nom « **SMS.py** » qui doit contenir la procédure « **enregistrement** » qui se déclenche lors du clic sur le bouton "**Enregistrer**" et qui permet de/d' :

- enregistrer le contenu des zones de texte dans le fichier "**SMS.dat**" si les contraintes de la saisie des champs citées ci-dessus sont respectées ou,
- afficher un message d'erreur via "**QMessageBox**" dans le cas contraire.

Tâche N° 2 : Manipulation des tableaux « QTableWidgetItem »

On se propose de modifier le programme de la tâche précédente de telle sorte qu'il affiche la liste des numéros des téléspectateurs, les chaînes des SMS et les scores.

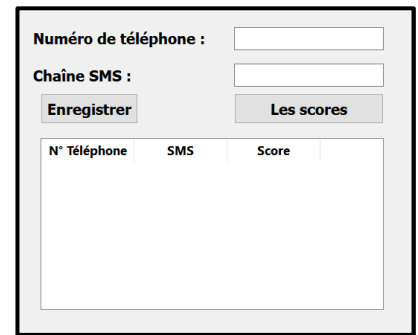
Pour cela, on se propose de concevoir une interface graphique contenant les éléments de la tâche N° 1 en lui ajoutant les éléments suivants :

- Un bouton intitulé "**Les scores**", qui permet d'afficher des informations dans une table Widget.
- Une **Table Widget** contenant les colonnes "**Numéro Téléphone**", "**SMS**" et "**Score**" pour afficher la liste des téléspectateurs (Numéro de téléphone, chaîne SMS et score).

Travail demandé :

Copier le fichier « *SMS01.ui* » situé dans « C:\ FormationQt » dans le dossier de travail.

1. Compléter l'interface graphique du fichier « *SMS01.ui* » par les éléments nécessaires comme le montre la figure ci-contre.
2. Modifier le code Python de la tâche précédente et l'enregistrer sous le nom « *SMS01.py* » afin d'ajouter la procédure « **score** », qui se déclenche lors du clic sur le bouton « **Les scores** » et qui permet d'afficher la liste des téléspectateurs (Numéro de téléphone, chaîne SMS et score) dans une table Widget, sachant que le score est calculé de la façon suivante :



- ♦ 100 points si les quatre lettres sont à la bonne position.
- ♦ 50 points si deux lettres sont à la bonne position.
- ♦ 25 points si une lettre est à la bonne position.
- ♦ 0 point si aucune lettre n'est à la bonne position.

N.B. : Le parcours du fichier, le calcul du score existent dans le fichier « *code.txt* » existant dans le dossier « **FormationQt** » du lecteur C.

🔗 Code de la tâche N° 2 :

#importation des bibliothèques du graphique

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication , QMessageBox
from pickle import dump , load
```

#Fonction permettant de vérifier la validité du numéro de téléphone

```
def verifnum ( num ) :
    if ( (len(num) != 8) or (num.isdecimal () == False) or ( (num[0] != "9") and (num[0] != "2" ) ) ) :
        return False
    else :
        return True
```

#Fonction permettant de vérifier la validité de la chaîne du SMS

```
def verifsms ( sms ) :
    ch = "AFCM"
    sms = sms.upper()
    i = 0
    erreur = False
    while ( i < len( ch ) ) and ( erreur == False ) :
        if sms.find ( ch[ i ] ) == -1 :
            erreur = True
        else :
            i = i + 1
    return ( erreur == False) and ( len( sms ) == 4 )
```

#Procédure permettant l'enregistrement du contenu des zones de saisie si la saisie est correcte ou d'afficher un message d'erreur dans le cas contraire.

```
def enregistrement ( ) :
    fsms = open ( "SMS.dat" , "ab" )
    enreg = dict ( Numtel = str() , sms = str() )
    num = fenetre.numtel.text ( )
    sms = fenetre.sms.text ( )
    if ( verifnum ( num ) == False ) or ( verifsms ( sms ) == False ) :
```

```

    QMessageBox.critical ( fenetre , "Erreur" , "Vous avez des erreurs. Vérifiez vos données" )
else :
    enreg [ "Numtel" ] = num
    enreg [ "sms" ] = sms
    dump ( enreg , fsms )
    fsms.close ( )
#Fonction permettant de calculer le score d'un SMS
def calculscore ( sms ) :
    ch = "AFCM"
    nb = 0
    for i in range ( len( ch ) ) :
        if ch[ i ] == sms[ i ] :
            nb = nb + 1
    return nb * 25
#Procédure permettant la lecture du fichier SMS.dat, le Calcul du score et l'affichage des données dans la Table Widget .
def score ( ) :
    fsms = open ( "SMS.dat" , "rb" )
    enreg = dict ( Numtel = str ( ) , sms = str ( ) )
    fin = False
    L = 0
    while ( fin == False ) :
        try :
            enreg = load ( fsms )
            fenetre.tableWidget.insertRow ( L )
            fenetre.tableWidget.setItem ( L , 0 , QTableWidgetItem ( enreg [ "Numtel" ] ) )
            fenetre.tableWidget.setItem ( L , 1 , QTableWidgetItem ( enreg [ "sms" ] ) )
            nb = calculscore ( enreg [ "sms" ] )
            fenetre.tableWidget.setItem ( L , 2 , QTableWidgetItem ( str ( nb ) ) )
            L = L + 1
        except :
            fin = True
    fsms.close ( )

#création application graphique
app = QApplication ( [ ] )
#Chargement du fichier Qt designer
fenetre = loadUi ( "SMS.ui" )
#affichage de la fenêtre graphique
fenetre.show ( )
#programmation des déclencheurs d'évènement sur les boutons
fenetre.enregistrer.clicked.connect ( enregistrement )
fenetre.scores.clicked.connect ( score )
app.exec_ ( )

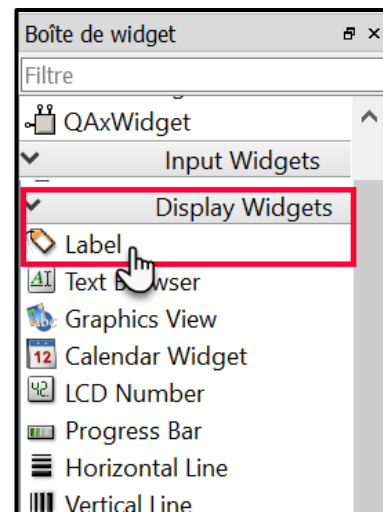
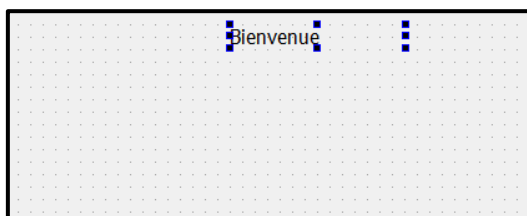
```

Annexe

1 Insertion d'une zone de texte « QLabel » d'une fenêtre de formulaire

Marche à suivre

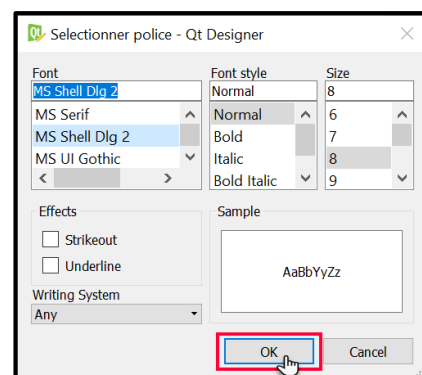
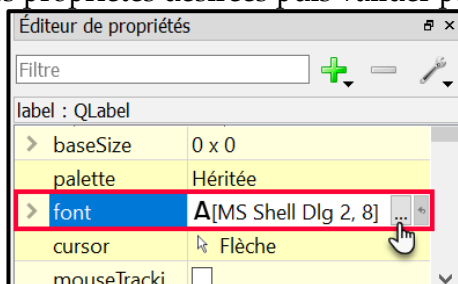
- 1) Ouvrir le groupe « Display Widgets » qui existe dans la « boîte Widget ».
- 2) Glisser le composant « Label » dans la zone de travail.
- 3) Double cliquer sur le label inséré puis taper le nouveau texte.



2 Modification des propriétés d'une zone de texte « QLabel » (Taille, police, style, etc.)

Marche à suivre

- 1) Sélectionner la zone de texte
- 2) Dans l'onglet « Editeur de propriétés », sélectionner la propriété « font »
- 3) Cliquer sur le bouton « ... », une fenêtre s'affiche
- 4) Modifier les propriétés désirées puis valider par « Ok »



Méthodes d'une zone de textes :

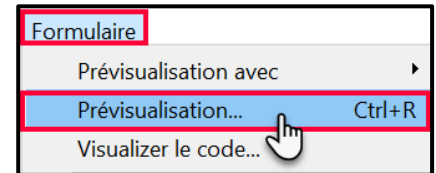
Chaque objet inséré dans un formulaire possède des méthodes. Pour l'objet QLabel on s'intéresse aux méthodes qui offre à l'utilisateur de modifier leurs valeurs et de récupérer leurs contenus.

Objets	Méthode	Explication
QLabel fenetre.label.	setText ()	une méthode qui permet d'initialiser le texte de l'objet «label» à une valeur.
	text ()	une méthode qui permet de récupérer le texte existant dans un objet «label».

3 Prévisualisation du formulaire

Marche à suivre

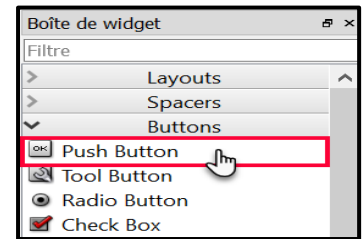
- 1) Cliquer sur le menu « **Formulaire** »,
- 2) Cliquer sur la commande « **Prévisualisation** »



4 Insertion d'un bouton « QPushButton » dans un formulaire

Marche à suivre

- 1) Ouvrir le groupe « **buttons** » qui existe dans la « **boîte Widget** ».
- 2) Glisser le composant « **Push button** » dans la zone de travail.
- 3) Double clic sur le bouton inséré puis modifier le texte du bouton.



5 Les événements d'un bouton « QPushButton »

Événements d'un bouton « QPushButton » :

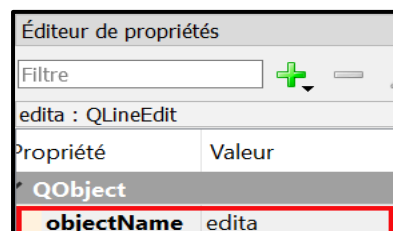
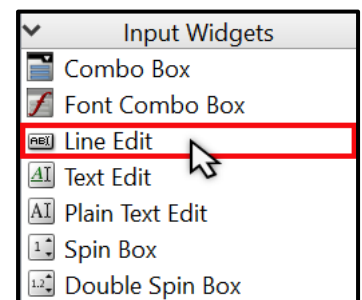
Chaque objet inséré dans un formulaire possède des événements. Pour l'objet **QPushButton** on s'intéresse aux événements suivantes :

Événements	Explication
clicked ()	Déclencher quand le bouton est cliqué.
pressed ()	Déclencher quand le bouton est pressé.
released ()	Déclencher quand le bouton est relâché
toggled ()	Déclencher quand l'attribut « checkable = Oui ».

6 Manipulation des zones de saisies « QLineEdit »

Marche à suivre

- 1) Ouvrir le groupe « **Input Widgets** » qui existe dans la « **boîte Widget** ».
- 2) Glisser le composant « **Line Edit** » dans la zone de travail.
- 3) Modifier la valeur de la propriété « **ObjectName** » qui existe dans « **l'éditeur de propriétés** » d'afin de renommer le nom de la zone « **Line Edit** ».

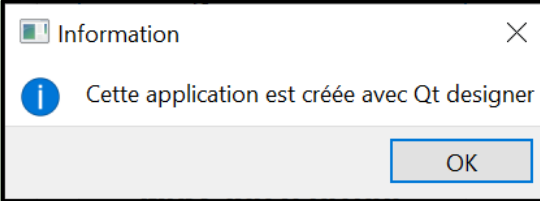
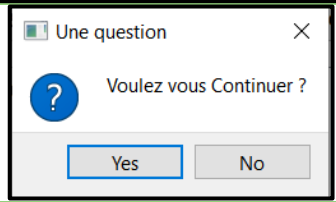
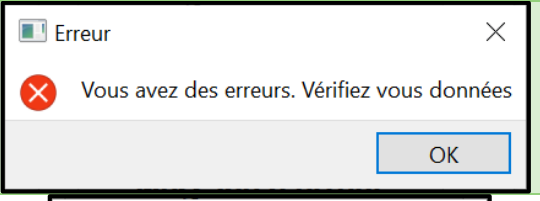
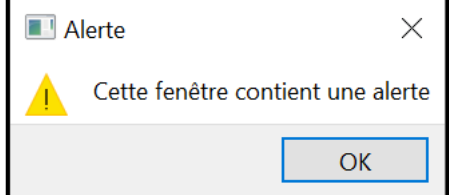


➤ Méthodes d'une zone de saisie QLineEdit:

Objets	Méthode	Explication
QlineEdit fenetre. lineEdit.	setText ()	une méthode qui permet d'initialiser le texte de l'objet «QlineEdit» à une valeur.
	text()	Une méthode qui permet de récupérer le texte existant dans un objet «QlineEdit».
	clear()	une méthode qui permet d'effacer le contenu de l'objet «QlineEdit»

7 Manipulation des messages avec « QMessageBox »

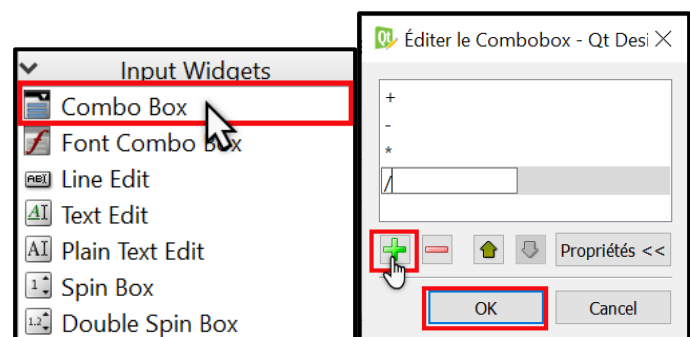
➤ Syntaxe :

Tâche	Boîte de dialogue a affiché
<pre>from PyQt5.QtWidgets import QMessageBox QMessageBox.information (fenetre , "Information" , "Cette application est créée avec Qt designer")</pre>	
<pre>from PyQt5.QtWidgets import QMessageBox QMessageBox.question (fenetre , "Une question" , "Voulez vous Continuer ? ")</pre>	
<pre>from PyQt5.QtWidgets import QMessageBox QMessageBox.critical (fenetre , "Erreur" , "Vous avez des erreurs. Vérifiez vos données")</pre>	
<pre>from PyQt5.QtWidgets import QMessageBox QMessageBox.warning (fenetre , "Alerte" , "Cette fenêtre contient une alerte")</pre>	

8 Manipulation des listes déroulantes « QComboBox »

➤ Marche à suivre

- 1) Utiliser le composant « **combo Box** » qui existe dans le groupe « **Input Widgets** »
- 2) Double cliquer sur la zone « **combo Box** » puis cliquer sur le bouton « + » pour ajouter des options.
- 3) Modifier la valeur de la propriété « **ObjectName** » des différents objets.



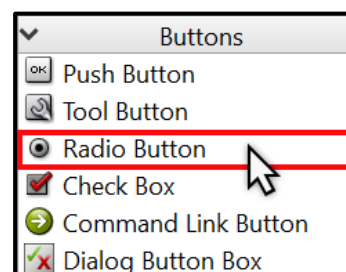
➤ Méthodes d'une zone de saisie QComboBox:

Objets	Méthode	Explication
QComboBox fenetre.Combo.	addItem ("V1")	Une méthode qui permet d'ajouter une valeur
	currentText ()	Une méthode qui permet de récupérer la valeur sélectionnée dans un objet «QComboBox».
	count ()	Une méthode qui calcul de nombre d'éléments dans une liste déroulante
	currentIndex ()	Une méthode qui retourne l'indice de l'élément sélectionné.
	itemText (ind)	Une méthode qui retourne l'élément de la liste déroulante qui existe dans l'indice «ind».

9 Manipulation des boutons radio « QRadioButton »

➤ Marche à suivre

- 1) Utiliser le composant « **Radio Button** » qui existe dans le groupe « **Buttons** »
- 2) Modifier la valeur de la propriété « **ObjectName** » de l'objet.



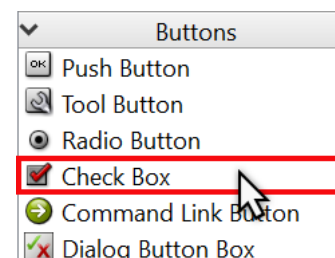
➤ Méthodes d'une zone de sélection QRadioButton:

Objets	Méthode	Explication
QRadioButton fenetre.radioButton.	isChecked ()	Une méthode qui return True si le bouton Radio est coché et False dans le cas contraire
	setChecked (True/ False)	Une méthode qui permet d'initialiser le cochage d'un bouton radio a Vrai (coché) ou Faux (non coché)
	setText ()	une méthode qui permet d'initialiser le texte de l'objet «QRadioButton» à une valeur.
	text ()	Une méthode qui permet de récupérer le texte existant dans un objet «QRadioButton».

10 Manipulation des cases à cocher « QCheck Box »

➤ Marche à suivre

- 1) Utiliser le composant « **Check Box** » qui existe dans le groupe « **Buttons** »
- 2) Modifier la valeur de la propriété « **ObjectName** » de l'objet.



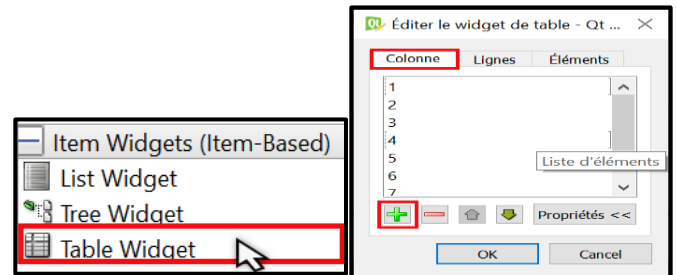
➤ Méthodes d'une zone de sélection QCheckBox :

Objets	Méthode	Explication
fenetre.CheckBox.	isChecked ()	Une méthode qui return True si la case à cocher est cochée et False dans le cas contraire
	setChecked (True/ False)	Une méthode qui permet d'initialiser le cochage la case à cocher a Vrai (coché) ou Faux (non coché)
	setText ()	une méthode qui permet d'initialiser le texte de l'objet «QCheckBox» à une valeur.
	text ()	une méthode qui permet de récupérer le texte existant dans un objet «QCheckBox».

① ① Manipulation des tableaux « QTableWidgetItem »

➤ Marche à suivre

- 1) Utiliser le composant « **Table Widget** » qui existe dans le groupe « **Item Widgets** »
- 2) Double clic sur le tableau dessiné dans le formulaire, puis Ajouter ou Modifier les colonnes et/ou les lignes.
- 3) Modifier la valeur de la propriété « **ObjectName** » de l'objet.



➤ Méthodes d'une zone de saisie QTableWidgetItem:

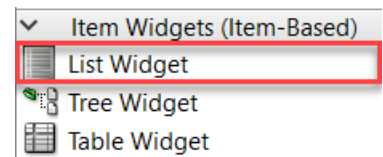
from PyQt5.QtWidgets import QTableWidgetItem, QTableWidgetItem

Objets	Méthode	Explication
QTableWidgetItem fenetre.Tableau.	insertRow (L)	Une méthode permet d'insérer une ligne N° L
	setRowCount (N)	Une méthode permet de créer un tableau de N lignes
	setItem(L , C , QTableWidgetItem(V1))	Une méthode qui permet d'initialiser le texte de la cellule qui existe dans la ligne L et la colonne C par la valeur V1
	item (L , C).text ()	Une méthode qui permet retourner le texte qui existe dans la ligne L et la colonne C
	rowCount ()	Une méthode qui retourne le nombre de lignes

1 2 Manipulation des Listes « QListWidget »

➤ Marche à suivre

- 1) Utiliser le composant « **List Widget** » qui existe dans le groupe « **Item Widgets** »
- 2) Modifier la valeur de la propriété « **ObjectName** » de l'objet.



➤ Méthodes de QListWidget :

Objets	Méthode	Explication
QListWidget fenetre. QListWidget	addItem ("V1")	Une méthode qui permet d'ajouter une valeur
	currentText ()	Une méthode qui permet de récupérer la valeur sélectionnée dans un objet « QListWidget ».
	count ()	Une méthode qui calcul de nombre d'éléments dans une liste déroulante
	currentIndex ()	Une méthode qui retourne l'indice de l'élément sélectionné.
	item(ind).text ()	Une méthode qui retourne l'élément de la liste qui existe dans l'indice « ind ».

Les classes enseignées

Widgets	4 ^{ème} scientifiques	3 ^{ème} S.I	4 ^{ème} S.I
PushButton	X	X	X
RadioButton		X	X
checkBox			
ComboBox			
Label	X	X	X
LineEdit	X	X	X
TextEdit		X	X
listWidget			X
tableWidget			X