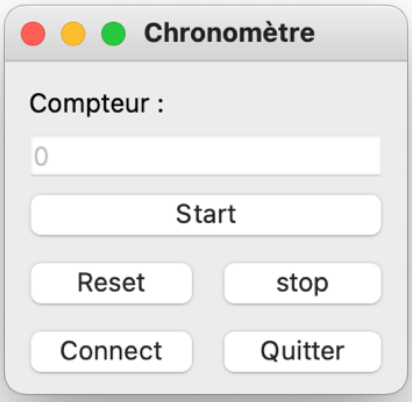
# Chronomètre

Vous avez droit à ***vos*** codes, au cours et aux exercices vus dans le cadre de R3.09. Tout autre document et toute recherche sur internet est interdit.
Votre téléphone portable devra être rangé dans votre sac et celui-ci mis sur le devant de la salle.
Le rendu se fera sur Moodle dans ce devoir. Vous devrez également mettre dans votre code un lien sur le github où vous avez mis le code l'examen.

## Sujet

L'idée est de réaliser un chronomètre selon le modèle ci-dessous :




## Partie 1 : réaliser l'interface graphique ci-dessus

Vous remarquerez que la valeur de compteur n'est pas accessible.

## Partie 2 : Boutons Start, Reset, Quitter

- ☐ Bouton Start : démarrera le comptage et affichera la valeur + 1 dans la partie texte (sans faire de boucle) → fonction start
- ☐ Bouton Reset : remet à 0 le compteur → fonction reset
- ☐ Bouton Quitter : quitte l'interface comme nous l'avons déjà vu → fonction quitter

## Partie 3 : Bouton Start – suite

- ☐ Réaliser une boucle dans la fonction start en utilisant un booléen sous la forme :

```
Tant que arret_thread est faux faire
      Compteur = compteur + 1
      # à noter qu'ici un affichage est fait dans la QLineEdit
Fait
```

## Partie 4 : Thread …

Si vous exécutez le code tel qu'il est, vous verrez que votre interface est bloquée. Ceci est tout à fait normal puisque la boucle ci-dessus va s'exécuter et ne permet pas d'interaction avec l'interface. Il faut donc … « *threader* » le comptage.
Pour cela, ajouter renommer la fonction start en __start.
Créer une nouvelle fonction start qui démarrer la *thread* avec comme cible la fonction __start.
Nous allons également un temps d'attente de 1 seconde avant chaque nouvel affichage.
Si vous testez ce code, vous devriez avoir l'affichage se dérouler et pouvoir cliquer sur les boutons quitter. Malgré tout, en cliquant sur quitter, vous restez bloquer puisque la thread continue de s'exécuter.

## Partie 3 : Bouton Stop – arrêt du compteur

- Dans la fonction stop (associée au bouton Stop), mettez le booléen arret_thread à True.
- Normalement si votre code est correct, quand vous cliquer sur le bouton Stop, votre compteur doit arrêter de défiler.
- Toujours dans la fonction stop, attendez la fin du thread avec la fonction correspondante.

## Partie 4 : Bouton Quitter – suite et fin

- Dans la fonction quitter (associée au bouton Quitter), appelez la fonction stop avant le QApplication.exit(0)

## Partie 5 : Bouton Connect

Reprenez le petit serveur proposé dans l'examen (serveur.py).
Ce serveur fonctionne sur localhost avec le port 10000.
Dans votre chronomètre ajouter les fonctionnalités suivantes :

- Bouton Connect : connexion au serveur – un message peut s'afficher si une erreur se produit
- Si une connexion est établie, pour chaque bouton, envoyer le nom du bouton cliquer au serveur.
- Si une connexion est établie, dans la fonction start, envoyer le compteur au serveur.
- Si une connexion est établie, dans le bouton quitter, envoyer le message « bye »

## Partie 6 : ah les exceptions !

Il faudrait montrer que vous savez manipuler les exceptions, deux choix possibles :

- Facile : vous allez devoir faire une addition sur un nombre dans votre compteur. Gérer le fait que vous pourriez avoir un texte à ce niveau même si ce n'est pas possible dans ce contexte.
- Normal : gérer le fait que les exceptions sur la partie client/serveur par exemple lors de la connexion si le serveur n'est pas démarré.