📘 FICHE DE RÉVISION – Programmation Événementielle (Python / PyQt5)

🟦 1. Structure minimale d’une application PyQt5

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget

app = QApplication(sys.argv)   # Initialise l'application
window = QWidget()             # Crée une fenêtre
window.show()                  # Affiche la fenêtre
sys.exit(app.exec_())          # Boucle d'événement
```

À connaître :

- QApplication gère la boucle des événements.
- show() affiche la fenêtre.
- exec_() lance l’application (boucle infinie d’événements).

🟩 2. Widgets essentiels

Widget	Description
QPushButton	Bouton
QLabel	Texte ou image
QLineEdit	Champ de saisie
QTextEdit	Zone de texte multilignes
QCheckBox	Case à cocher
QRadioButton	Bouton radio
QComboBox	Menu déroulant
QSlider	Curseur
QListWidget	Liste simple

Widget	Description
QTableWidget	Tableau / grille

🟧 3. Layouts (disposition des widgets)

📌 **Pour éviter les positions absolues**

- QHBoxLayout() → disposition **horizontale**
- QVBoxLayout() → disposition **verticale**
- QGridLayout() → en **grille**

Exemple :

```
layout = QVBoxLayout()
layout.addWidget(self.label)
layout.addWidget(self.button)
self.setLayout(layout)
```

🟥 4. Signaux et Slots (CORE de la programmation événementielle)

📌 **Principe :** un signal → déclenche → un slot (fonction)

Exemples standards :

```
button.clicked.connect(self.action)
lineEdit.textChanged.connect(self.modif)
slider.valueChanged.connect(self.move)
```

Widget	Signal important
QPushButton	clicked
QLineEdit	textChanged
QCheckBox	stateChanged
QSlider	valueChanged
QComboBox	currentIndexChanged

Widget	Signal important
QTimer	timeout

Slot → fonction appelée automatiquement

```
def action(self):
    print("Action !")
```

🟦 5. Gestion des événements (Event Handling)

Tu peux redéfinir des méthodes spéciales :

Événement	Méthode à redéfinir
Clic souris	mousePressEvent(event)
Souris déplacée	mouseMoveEvent(event)
Touche clavier	keyPressEvent(event)
Fermeture fenêtre	closeEvent(event)
Redimensionnement	resizeEvent(event)

Exemple :

```
def mousePressEvent(self, event):
    print("Clic détecté")
```

🟨 6. Timers (QTimer)

Très fréquent en QCM :

```
from PyQt5.QtCore import QTimer


self.timer = QTimer()
self.timer.timeout.connect(self.update_time)
self.timer.start(1000)  # toutes les 1 secondes
```

🟫 7. Dialogues (boîtes de dialogue)

MessageBox

from PyQt5.QtWidgets import QMessageBox

QMessageBox.information(self, "Titre", "Message")

Ouvrir un fichier

from PyQt5.QtWidgets import QFileDialog

file, _ = QFileDialog.getOpenFileName(self, "Choisir un fichier")

🟩 8. Images dans PyQt5

Afficher une image dans un QLabel :

from PyQt5.QtGui import QPixmap

pix = QPixmap("image.png")

self.label.setPixmap(pix)

🟦 9. Fenêtres personnalisées (héritage)

Structure classique d'une classe PyQt5 :

```
class MaFenetre(QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        self.label = QLabel("Texte")
        self.button = QPushButton("OK")
        self.button.clicked.connect(self.action)
```

🔵 10. Programmation événementielle : principes théoriques

📌 Événement = action utilisateur

- clic
- déplacement souris
- frappe au clavier
- changement d'état d'un widget
- expiration d'un timer
- message du système

📌 **Propagation :**

- PyQt envoie l'événement au widget concerné (ex : bouton → fenêtre → application)

📌 **Boucle d'événement :**

- app.exec_() écoute et distribue les événements.

🔵 **11. QCM / Code à trous → choses très probables**

✓ Compléter :

button.clicked._____(self.fonction)

réponse → connect

✓ Trouver le signal d'un slider → valueChanged

✓ Trouver la méthode d'un clic souris → mousePressEvent

✓ Instancier un timer → QTimer()

✓ Charger une image → QPixmap

✓ Ajouter un widget dans un layout → addWidget()

✓ Choisir le layout vertical → QVBoxLayout

✓ Savoir que exec_() lance la boucle d'événements

🧠 **12. Mini-formules / rappel rapide**

Raccourcis

- self.setWindowTitle("Titre")
- self.resize(400, 300)

- widget.setEnabled(False) pour désactiver
- widget.setText("...")
- lineEdit.text() pour récupérer du texte

Récupérer la valeur d'un slider

val = slider.value()