

SESSION REPORT

# Qt Parity Checker

*Object-oriented Python with PyQt5*

|             |                                                  |
|-------------|--------------------------------------------------|
| PROGRAM     | AEC Infrastructure Informatique et Cybersecurite |
| INSTITUTION | College de Bois-de-Boulogne                      |
| AREA        | Programming, object-oriented coursework          |

## 1. Objective

---

Practice object-oriented structure and GUI event handling in Python using PyQt5, loading a Qt Designer interface file and wiring it to application logic.

## 2. Design

---

A QWidget subclass, MaFenetre, loads a .ui file built in Qt Designer containing a text field (champNombre), a button (boutonVerifier), and a result label (labelResultat), then connects the button's clicked signal to a handler method.

```
CODE
class MaFenetre(QWidget):
    def __init__(self):
        super().__init__()
        uic.loadUi("interface.ui", self)
        self.boutonVerifier.clicked.connect(self.verifier_pair_ou_impair)
```

## 3. Logic

---

On click, the handler reads the field, validates that it is a non-empty digit string, and computes parity with the modulo operator. An invalid entry raises a QMessageBox error and clears the field and the result instead of crashing or showing a stale value.

```
CODE
def verifier_pair_ou_impair(self):
    texte = self.champNombre.text()
    if texte.strip().isdigit():
        nombre = int(texte)
        self.labelResultat.setText("Nombre pair" if nombre % 2 == 0 else "Nombre impair")
    else:
        QMessageBox.critical(self, "Erreur", "Erreur de saisi.")
        self.champNombre.clear()
        self.labelResultat.clear()
```

## 4. Outcome

---

A minimal, correctly validated desktop utility that demonstrates event-driven programming, separating the interface definition (.ui) from the application logic (.py), and handling bad input without crashing.