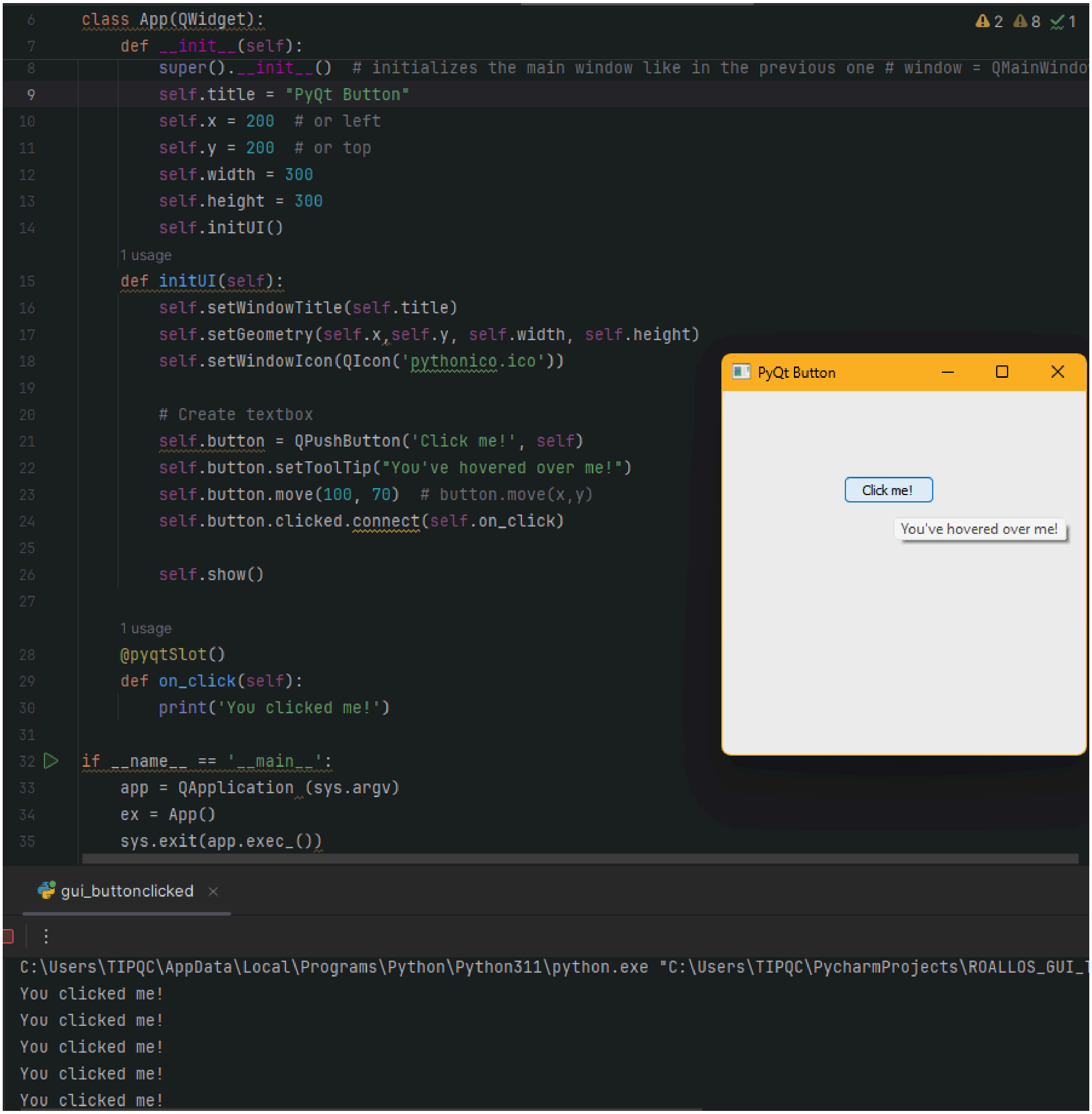
Activity No. 5

Introduction to Event Handling in GUI Development

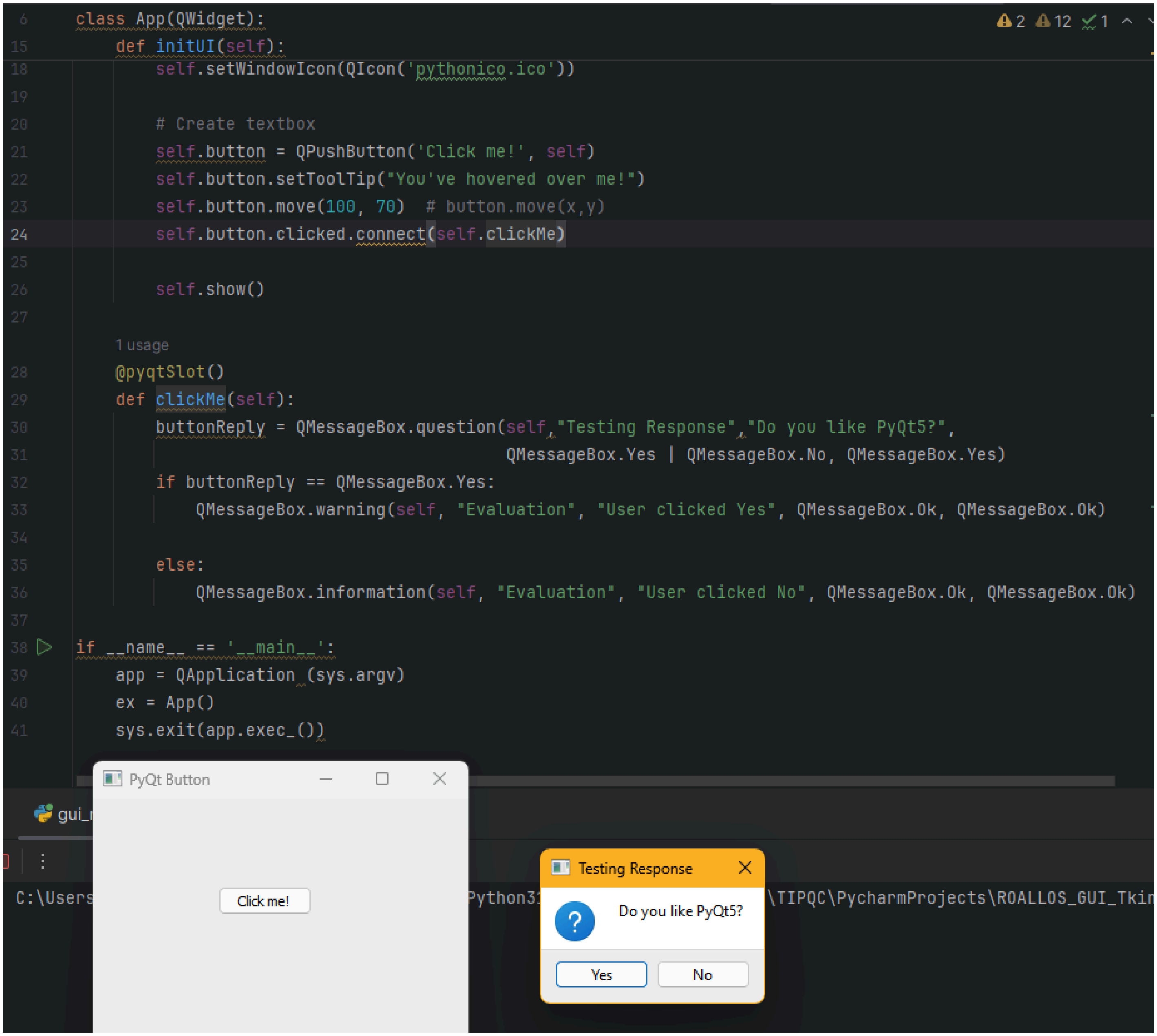
Course Code: CPE009	**Program:** BS Computer Engineering
Course Title: Object-Oriented Programming 2	**Date Performed:** 10 / 21 / 2024
Section: CPE21S4	**Date Submitted:** 10 / 21 / 2024
Name(s): ROALLOS, Jean Gabriel Vincent G.	**Instructor:** Prof. Maria Rizette Sayo

Documentation



```
class App(QWidget):
    def __init__(self):
        super().__init__()  # initializes the main window like in the previous one # window = QMainWindo
        self.title = "PyQt Button"
        self.x = 200  # or left
        self.y = 200  # or top
        self.width = 300
        self.height = 300
        self.initUI()

    def initUI(self):
        self.setWindowTitle(self.title)
        self.setGeometry(self.x,self.y, self.width, self.height)
        self.setWindowIcon(QIcon('pythonico.ico'))

        # Create textbox
        self.button = QPushButton('Click me!', self)
        self.button.setToolTip("You've hovered over me!")
        self.button.move(100, 70)  # button.move(x,y)
        self.button.clicked.connect(self.on_click)

        self.show()

    @pyqtSlot()
    def on_click(self):
        print('You clicked me!')

if __name__ == '__main__':
    app = QApplication (sys.argv)
    ex = App()
    sys.exit(app.exec_())
```

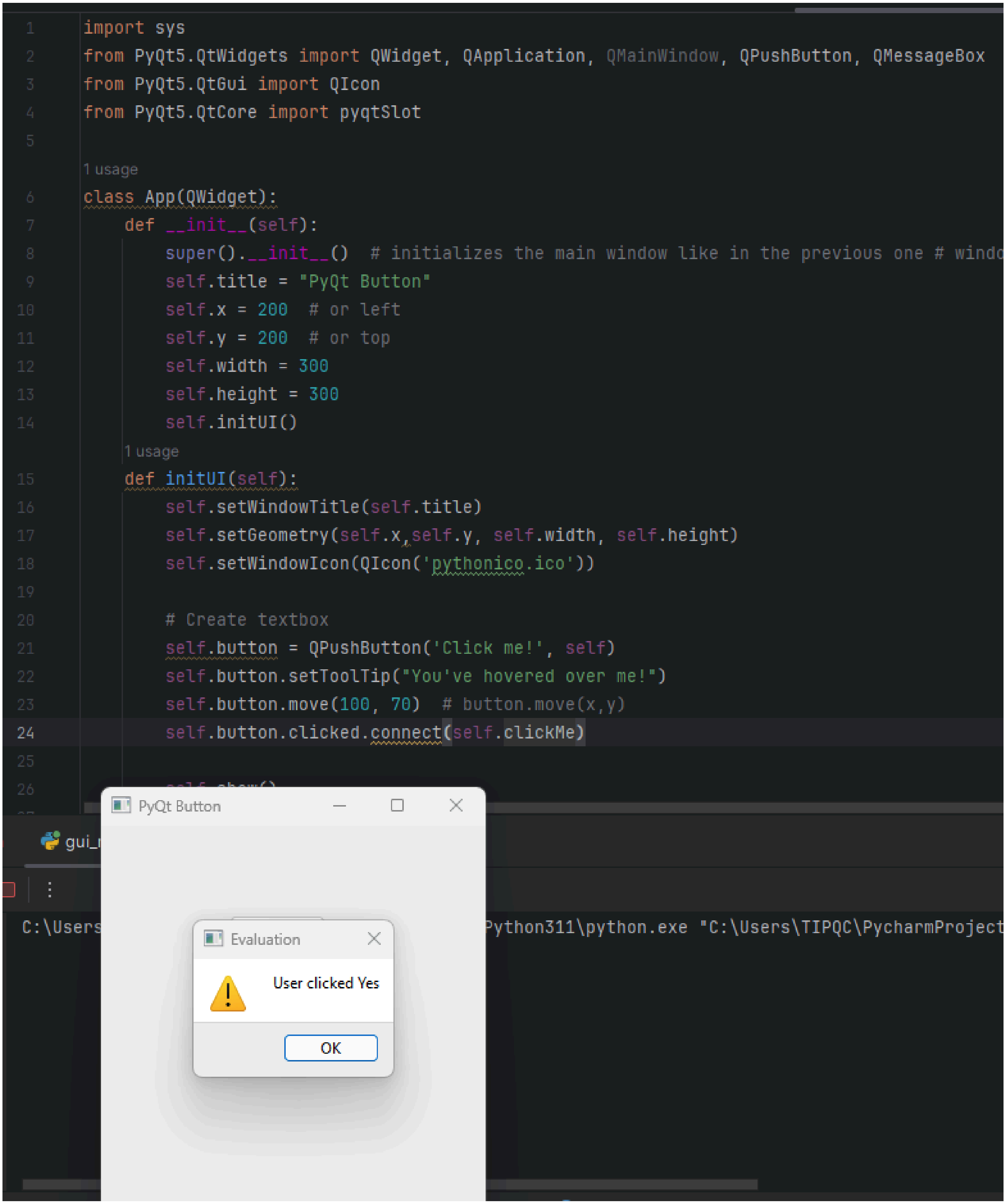
```
C:\Users\TIPQC\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\TIPQC\PycharmProjects\ROALLOS_GUI_
You clicked me!
You clicked me!
You clicked me!
You clicked me!
You clicked me!
```



```
class App(QWidget):
    def initUI(self):
        self.setWindowIcon(QIcon('pythonico.ico'))

        # Create textbox
        self.button = QPushButton('Click me!', self)
        self.button.setToolTip("You've hovered over me!")
        self.button.move(100, 70)  # button.move(x,y)
        self.button.clicked.connect(self.clickMe)

        self.show()

    1 usage
    @pyqtSlot()
    def clickMe(self):
        buttonReply = QMessageBox.question(self,"Testing Response","Do you like PyQt5?",
                                           QMessageBox.Yes | QMessageBox.No, QMessageBox.Yes)
        if buttonReply == QMessageBox.Yes:
            QMessageBox.warning(self, "Evaluation", "User clicked Yes", QMessageBox.Ok, QMessageBox.Ok)

        else:
            QMessageBox.information(self, "Evaluation", "User clicked No", QMessageBox.Ok, QMessageBox.Ok)

if __name__ == '__main__':
    app = QApplication (sys.argv)
    ex = App()
    sys.exit(app.exec_())
```

```
C:\Users                                   Python31         \TIPQC\PycharmProjects\ROALLOS_GUI_Tkin
```

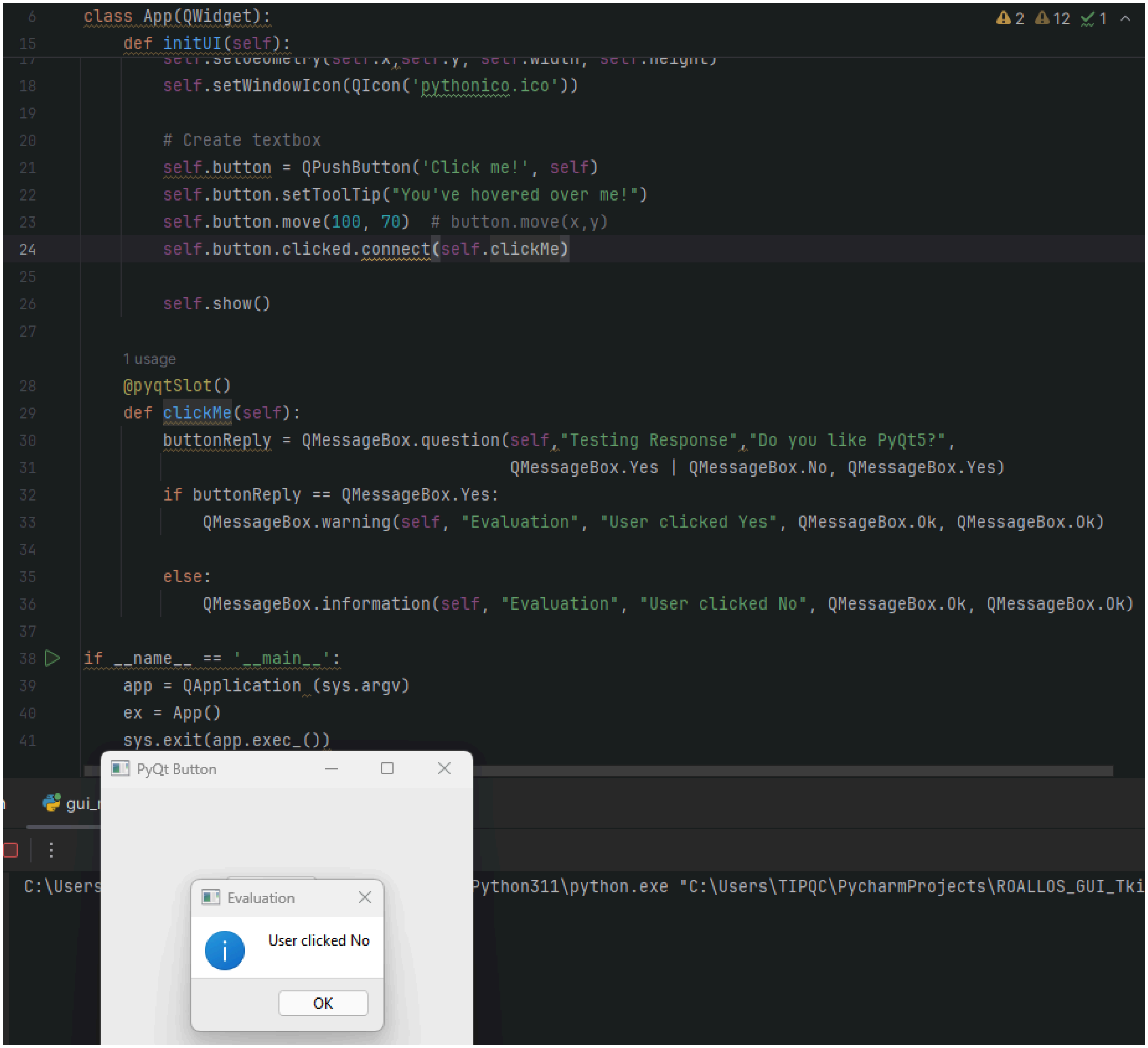
Do you like PyQt5?



```
import sys
from PyQt5.QtWidgets import QWidget, QApplication, QMainWindow, QPushButton, QMessageBox
from PyQt5.QtGui import QIcon
from PyQt5.QtCore import pyqtSlot

1 usage
class App(QWidget):
    def __init__(self):
        super().__init__()  # initializes the main window like in the previous one # windo
        self.title = "PyQt Button"
        self.x = 200  # or left
        self.y = 200  # or top
        self.width = 300
        self.height = 300
        self.initUI()
    1 usage
    def initUI(self):
        self.setWindowTitle(self.title)
        self.setGeometry(self.x,self.y, self.width, self.height)
        self.setWindowIcon(QIcon('pythonico.ico'))

        # Create textbox
        self.button = QPushButton('Click me!', self)
        self.button.setToolTip("You've hovered over me!")
        self.button.move(100, 70)  # button.move(x,y)
        self.button.clicked.connect(self.clickMe)

        self.show()
```

```
C:\Users                                    Python311\python.exe "C:\Users\TIPQC\PycharmProject
```

User clicked Yes

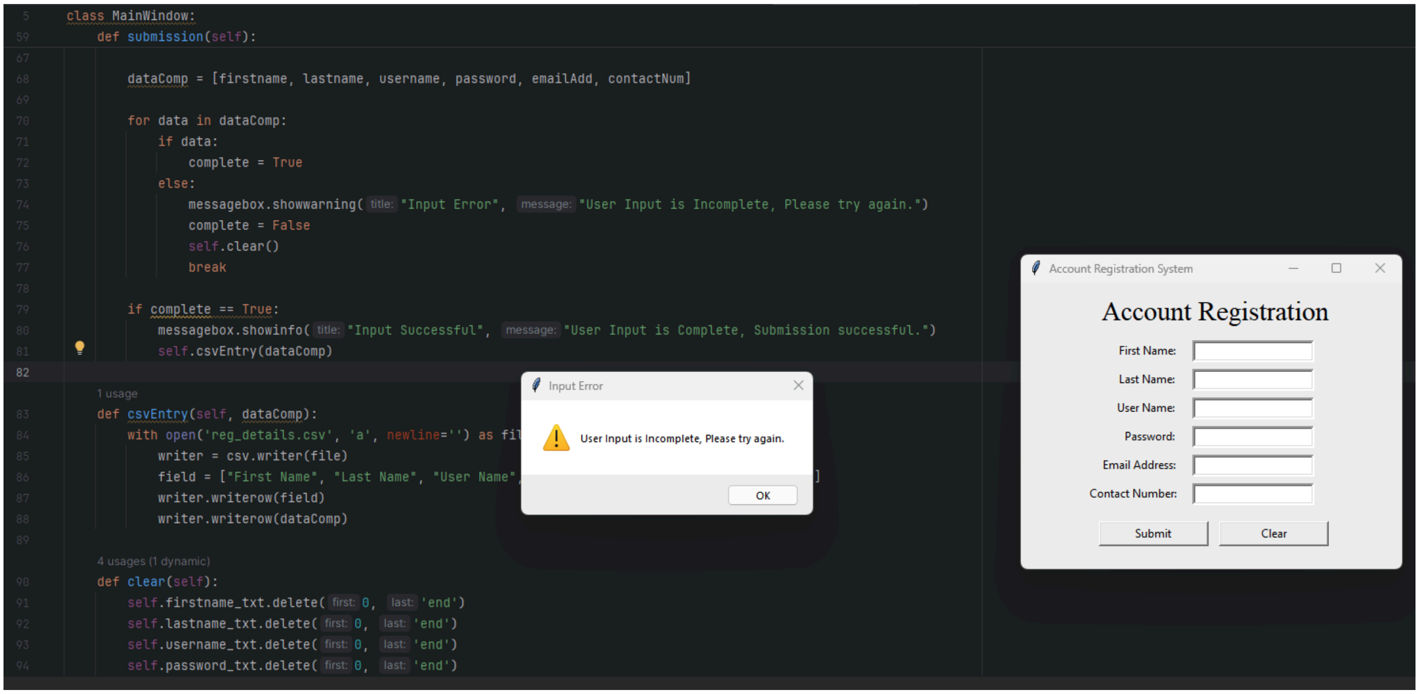


```
class App(QWidget):
    def initUI(self):
        self.setGeometry(self.x, self.y, self.width, self.height)
        self.setWindowIcon(QIcon('pythonico.ico'))

        # Create textbox
        self.button = QPushButton('Click me!', self)
        self.button.setToolTip("You've hovered over me!")
        self.button.move(100, 70)  # button.move(x,y)
        self.button.clicked.connect(self.clickMe)

        self.show()

    1 usage
    @pyqtSlot()
    def clickMe(self):
        buttonReply = QMessageBox.question(self,"Testing Response","Do you like PyQt5?",
                                           QMessageBox.Yes | QMessageBox.No, QMessageBox.Yes)
        if buttonReply == QMessageBox.Yes:
            QMessageBox.warning(self, "Evaluation", "User clicked Yes", QMessageBox.Ok, QMessageBox.Ok)

        else:
            QMessageBox.information(self, "Evaluation", "User clicked No", QMessageBox.Ok, QMessageBox.Ok)

if __name__ == '__main__':
    app = QApplication (sys.argv)
    ex = App()
    sys.exit(app.exec_())
```

```
C:\Users                 Python311\python.exe "C:\Users\TIPQC\PycharmProjects\ROALLOS_GUI_Tki
```

User clicked No

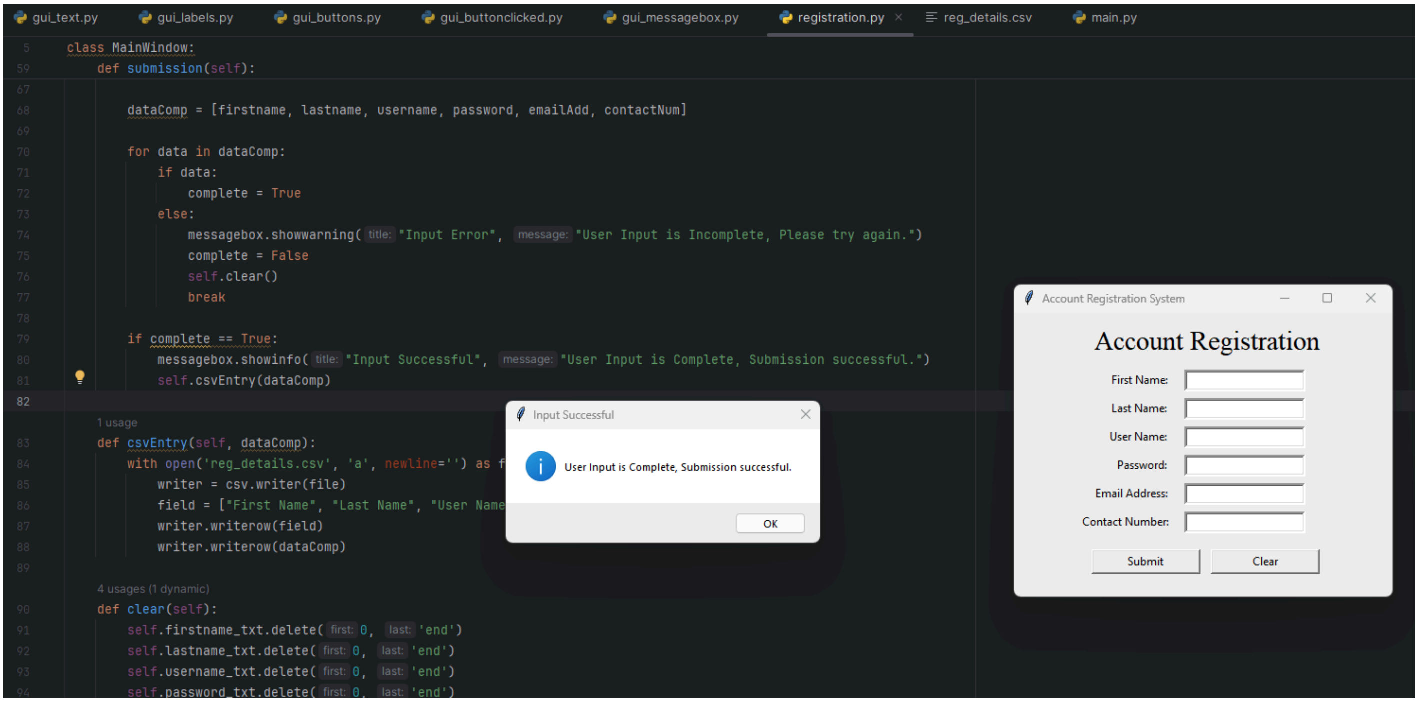
6. Supplementary Activity



```
class MainWindow:
    def submission(self):

        dataComp = [firstname, lastname, username, password, emailAdd, contactNum]

        for data in dataComp:
            if data:
                complete = True
            else:
                messagebox.showwarning( title: "Input Error",  message: "User Input is Incomplete, Please try again.")
                complete = False
                self.clear()
                break

        if complete == True:
            messagebox.showinfo( title: "Input Successful",  message: "User Input is Complete, Submission successful.")
            self.csvEntry(dataComp)

    1 usage
    def csvEntry(self, dataComp):
        with open('reg_details.csv', 'a', newline='') as fi
            writer = csv.writer(file)
            field = ["First Name", "Last Name", "User Name",                                   ]
            writer.writerow(field)
            writer.writerow(dataComp)

    4 usages (1 dynamic)
    def clear(self):
        self.firstname_txt.delete( first: 0,  last: 'end')
        self.lastname_txt.delete( first: 0,  last: 'end')
        self.username_txt.delete( first: 0,  last: 'end')
        self.password_txt.delete( first: 0,  last: 'end')
```

Input Error

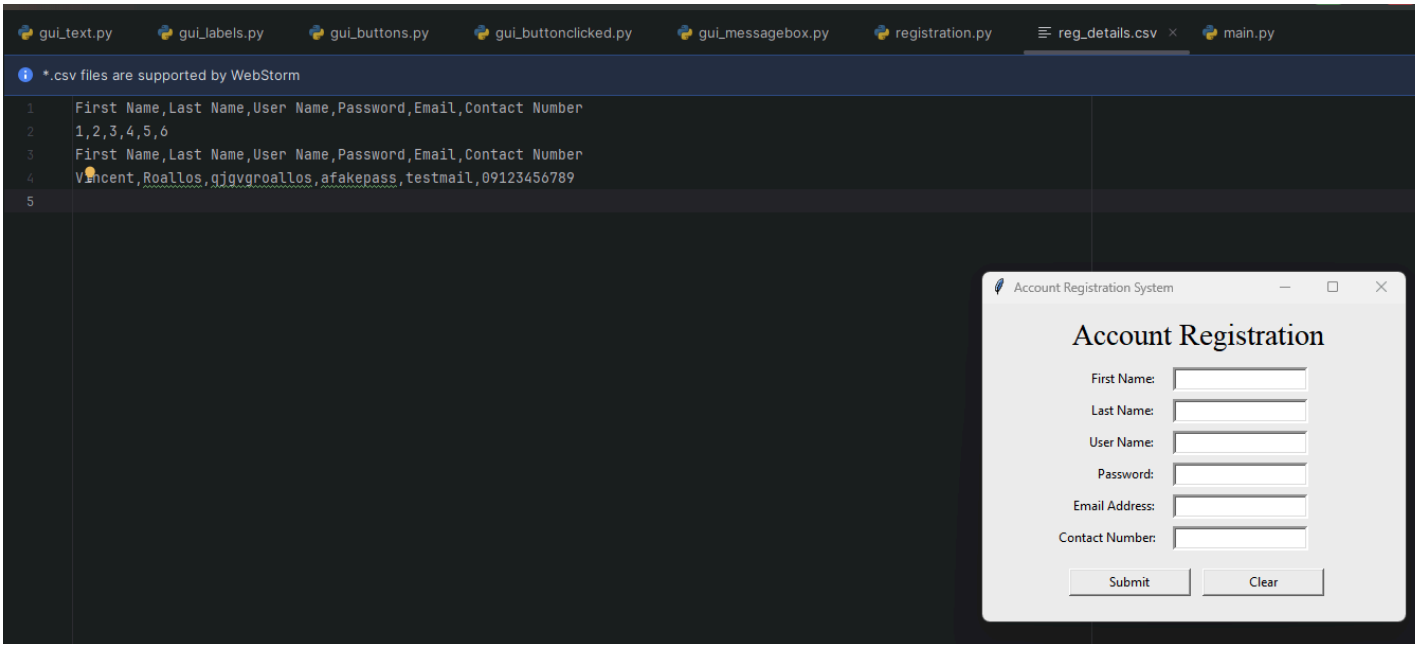
User Input is Incomplete, Please try again.



```
class MainWindow:
    def submission(self):

        dataComp = [firstname, lastname, username, password, emailAdd, contactNum]

        for data in dataComp:
            if data:
                complete = True
            else:
                messagebox.showwarning( title: "Input Error",  message: "User Input is Incomplete, Please try again.")
                complete = False
                self.clear()
                break

        if complete == True:
            messagebox.showinfo( title: "Input Successful",  message: "User Input is Complete, Submission successful.")
            self.csvEntry(dataComp)

    1 usage
    def csvEntry(self, dataComp):
        with open('reg_details.csv', 'a', newline='') as f
            writer = csv.writer(file)
            field = ["First Name", "Last Name", "User Name
            writer.writerow(field)
            writer.writerow(dataComp)

    4 usages (1 dynamic)
    def clear(self):
        self.firstname_txt.delete( first: 0,  last: 'end')
        self.lastname_txt.delete( first: 0,  last: 'end')
        self.username_txt.delete( first: 0,  last: 'end')
        self.password_txt.delete( first: 0,  last: 'end')
```

Input Successful

User Input is Complete, Submission successful.



*.csv files are supported by WebStorm

```
First Name,Last Name,User Name,Password,Email,Contact Number
1,2,3,4,5,6
First Name,Last Name,User Name,Password,Email,Contact Number
Vincent,Roallos,gjgvgroallos,afakepass,testmail,09123456789
```

Questions

1. What are the other signals available in PyQt5? (give at least 3 and describe each)

   Few signals in the PyQt5 library are move(), warning() and question(). move(), from the QWidget library, moves the item or object within the GUI application. warning() and information(), both from the QMessageBox library, pops up a message box with a predefined window title and contents. Their difference would be found in the icons included in their window. warning() includes a yellow triangular sign, while information() has a blue circular sign, meant to inform the user that something has happened.

2. Why do you think that event handling in Python is divided into signals and slots?

   This separates the cause and effect of every action of the user in the running program. Signals would be the cause, such as user interactions in the program. Slots respond to the signal that is set to trigger them

3. How can message boxes be used to provide a better User Experience or how can message boxes be used to make a GUI Application more user-friendly?

   Message boxes can be used to inform users about the several processes within the program and if it is successful or not in it. Icons within these message boxes give an idea to the user if something went wrong or not.

4. What is Error-handling and how was it applied in the task performed?

   Error handling pertains to a technique implemented to programs that have processes with data involved. It aims to address an error during the execution of a program and inform its user that something went wrong. This technique is applied in the activity by creating a condition that all fields within the Account Registration program should have contents before getting passed along and saved in a .csv file.

5. What may be the reasons behind the need to implement error handling?

   It is important for programs to consider errors within them during user interface. Error handling can let the user know that something went wrong within the program after their interaction.

7. Conclusion

The activity has helped me be familiarized with the other various features of the PyQt5 library. I have improved the user-friendliness of the Account Registration program by implementing error handling to hand out useful information to the user. Being able to communicate to the user through the program can guide the user in doing things the intended way and therefore the program would also be able to do its process correctly.

8. Assessment Rubric