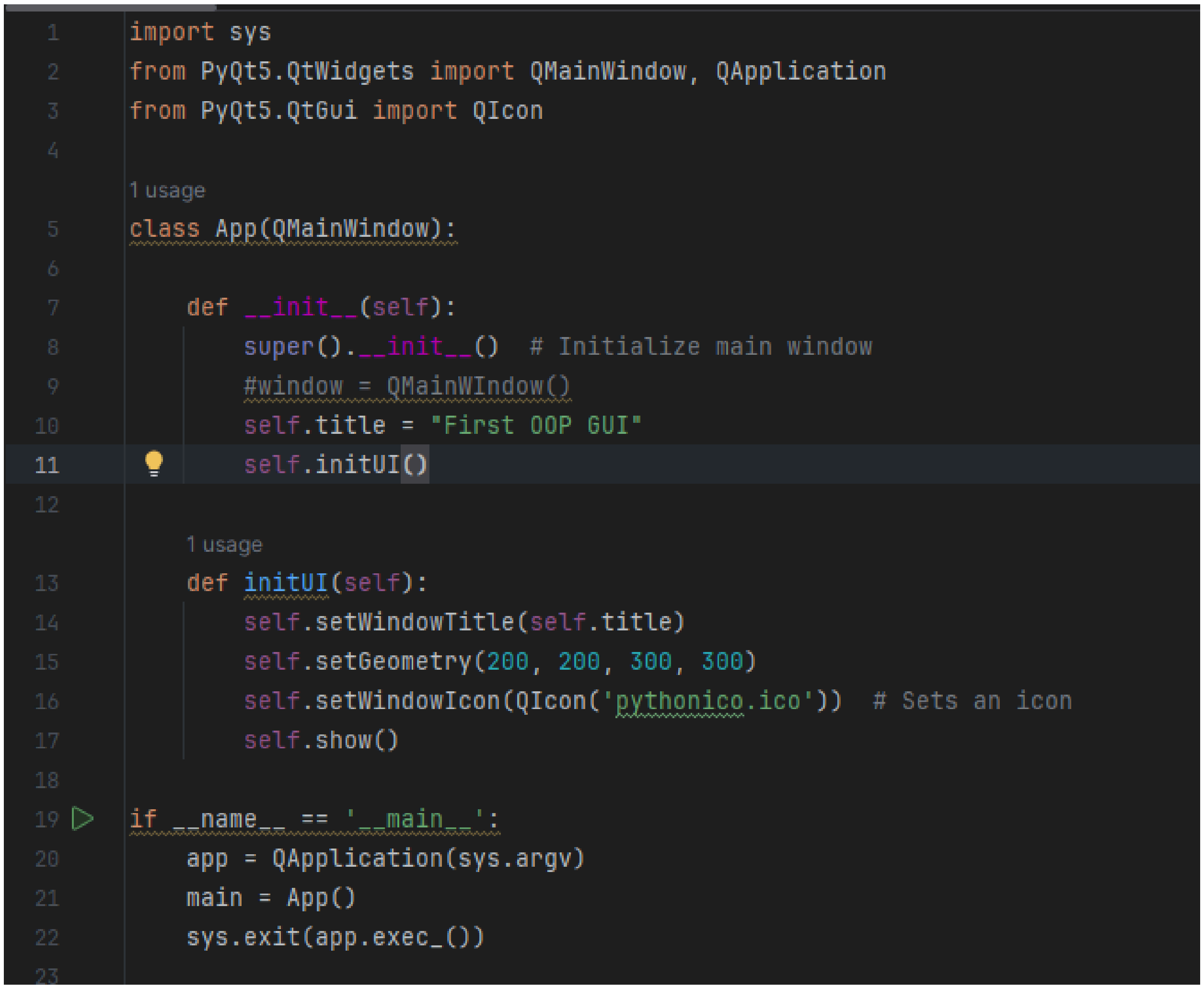
Laboratory Activity 4	
Rio, Aries, C.	10/14/24
CPE 009B - CPE21S4	Ma'am Rizette Sayo

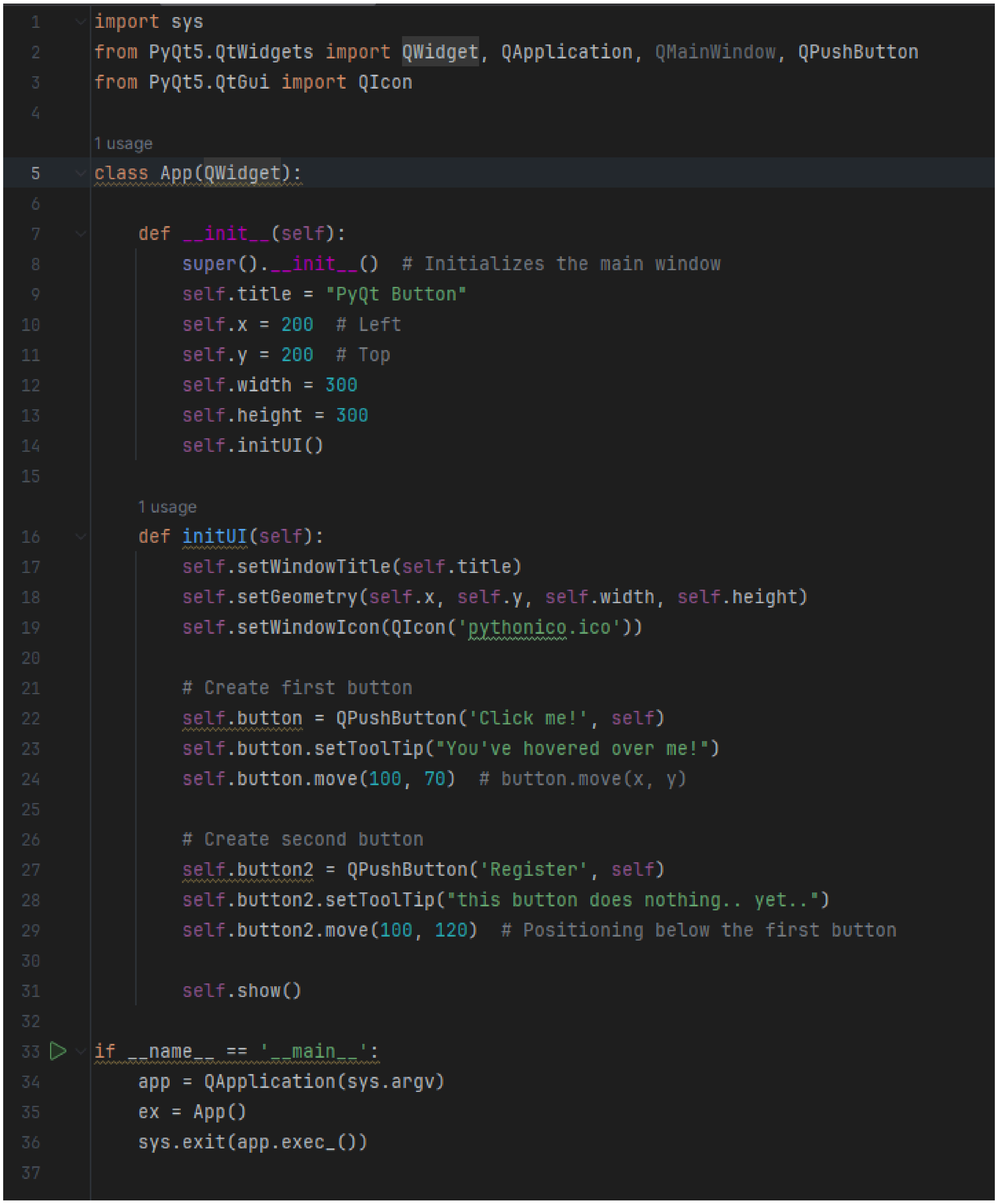
Procedure:

Adding an icon:



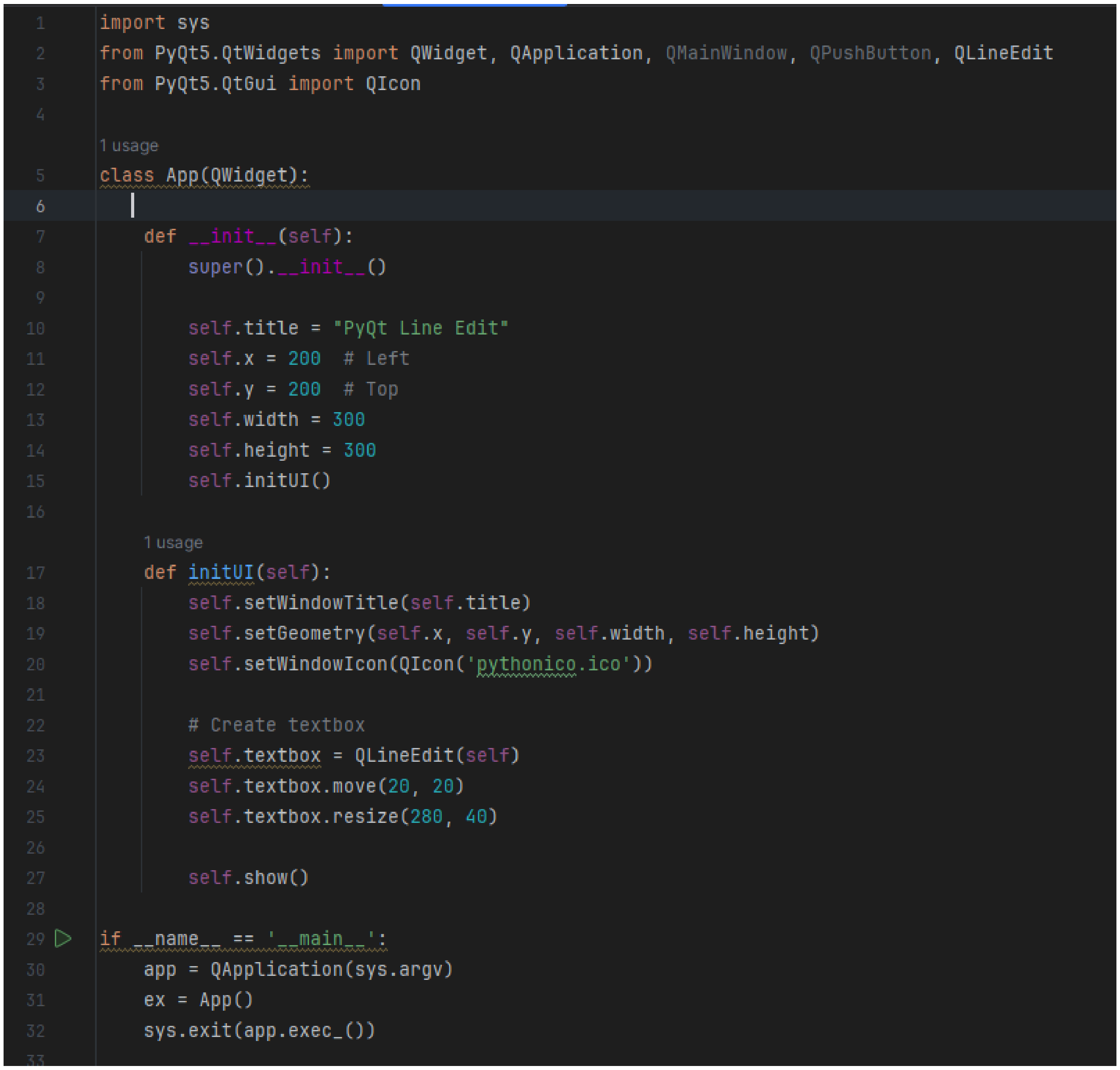
```python
import sys
from PyQt5.QtWidgets import QMainWindow, QApplication
from PyQt5.QtGui import QIcon

1 usage
class App(QMainWindow):

    def __init__(self):
        super().__init__()  # Initialize main window
        #window = QMainWIndow()
        self.title = "First OOP GUI"
        self.initUI()

    1 usage
    def initUI(self):
        self.setWindowTitle(self.title)
        self.setGeometry(200, 200, 300, 300)
        self.setWindowIcon(QIcon('pythonico.ico'))  # Sets an icon
        self.show()

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

Creating Buttons:



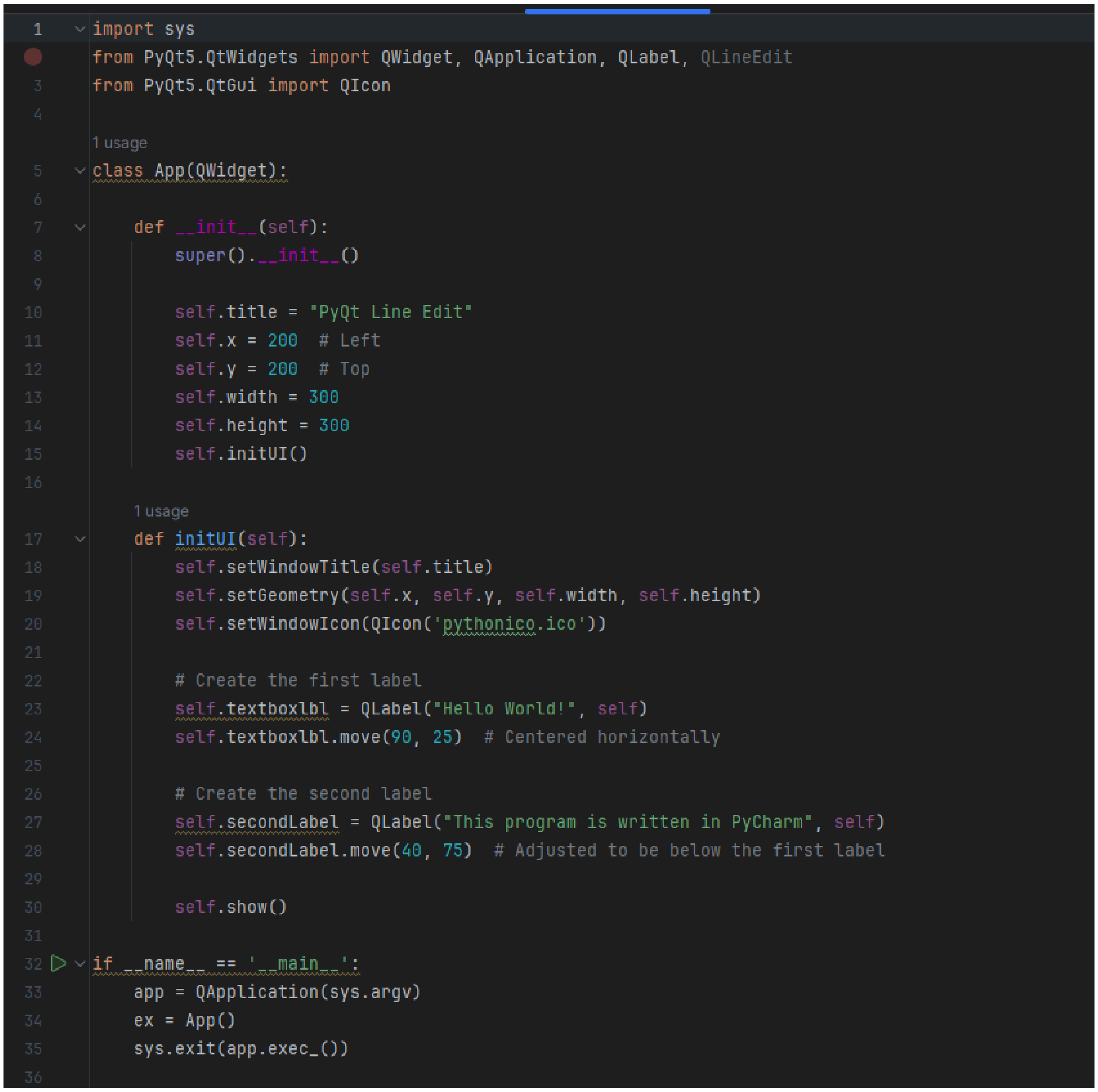
```
import sys
from PyQt5.QtWidgets import QWidget, QApplication, QMainWindow, QPushButton
from PyQt5.QtGui import QIcon

1 usage
class App(QWidget):

    def __init__(self):
        super().__init__()  # Initializes the main window
        self.title = "PyQt Button"
        self.x = 200  # Left
        self.y = 200  # 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 first button
        self.button = QPushButton('Click me!', self)
        self.button.setToolTip("You've hovered over me!")
        self.button.move(100, 70)  # button.move(x, y)

        # Create second button
        self.button2 = QPushButton('Register', self)
        self.button2.setToolTip("this button does nothing.. yet..")
        self.button2.move(100, 120)  # Positioning below the first button

        self.show()

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

Creating Text Fields:



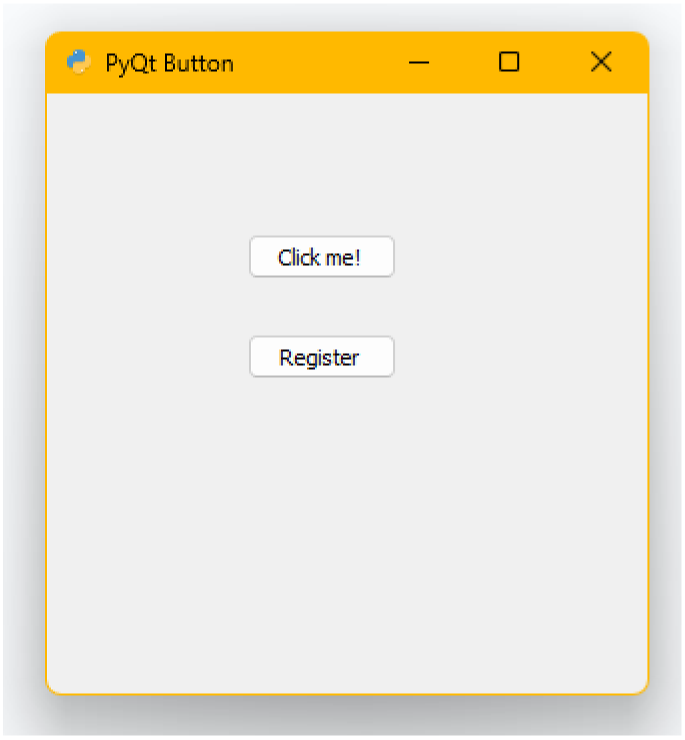
```python
import sys
from PyQt5.QtWidgets import QWidget, QApplication, QMainWindow, QPushButton, QLineEdit
from PyQt5.QtGui import QIcon

1 usage
class App(QWidget):

    def __init__(self):
        super().__init__()

        self.title = "PyQt Line Edit"
        self.x = 200  # Left
        self.y = 200  # 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.textbox = QLineEdit(self)
        self.textbox.move(20, 20)
        self.textbox.resize(280, 40)

        self.show()

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

Creating Labels:



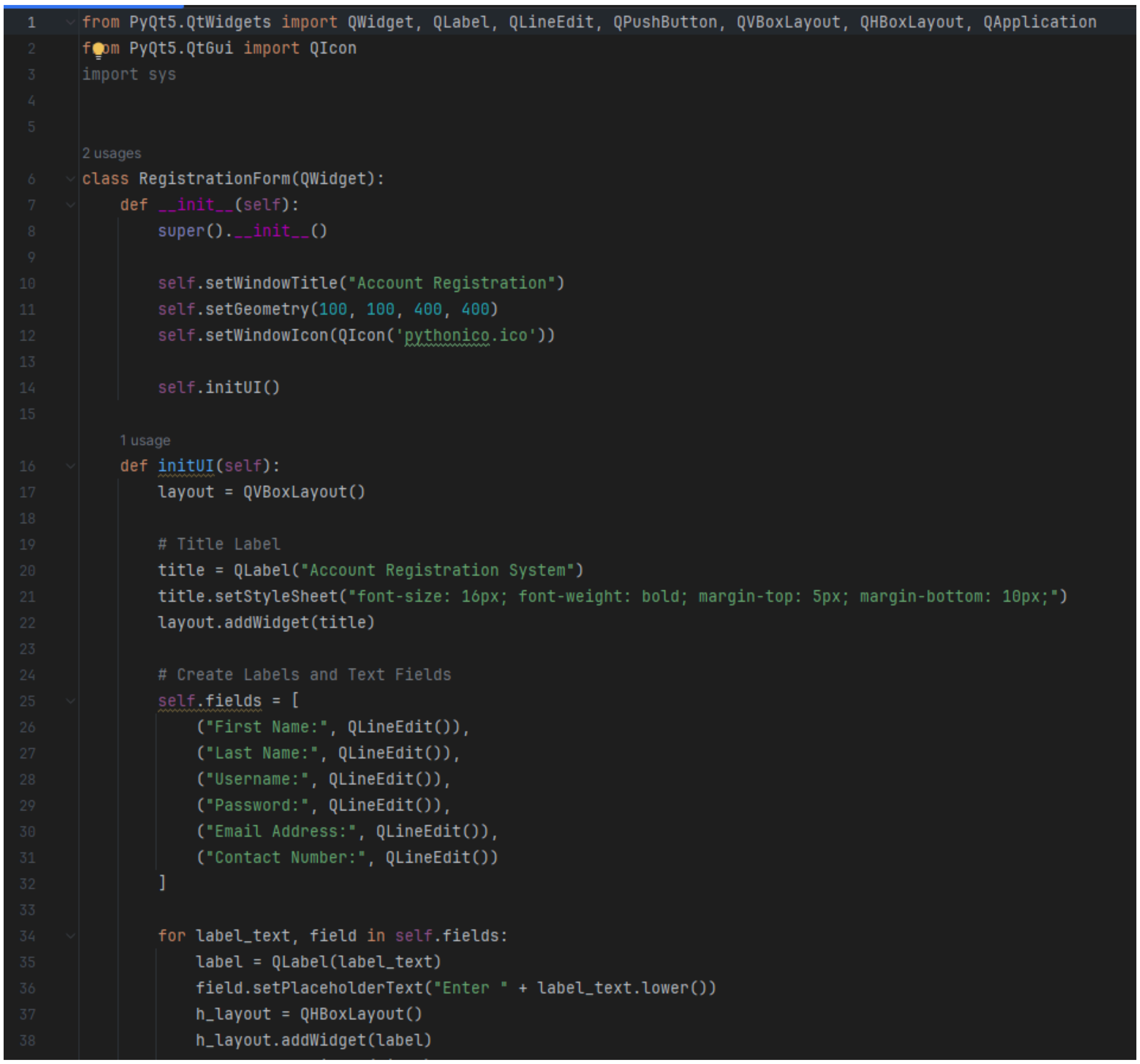
```python
import sys
from PyQt5.QtWidgets import QWidget, QApplication, QLabel, QLineEdit
from PyQt5.QtGui import QIcon

1 usage
class App(QWidget):

    def __init__(self):
        super().__init__()

        self.title = "PyQt Line Edit"
        self.x = 200  # Left
        self.y = 200  # 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 the first label
        self.textboxlbl = QLabel("Hello World!", self)
        self.textboxlbl.move(90, 25)  # Centered horizontally

        # Create the second label
        self.secondLabel = QLabel("This program is written in PyCharm", self)
        self.secondLabel.move(40, 75)  # Adjusted to be below the first label

        self.show()

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

Output:

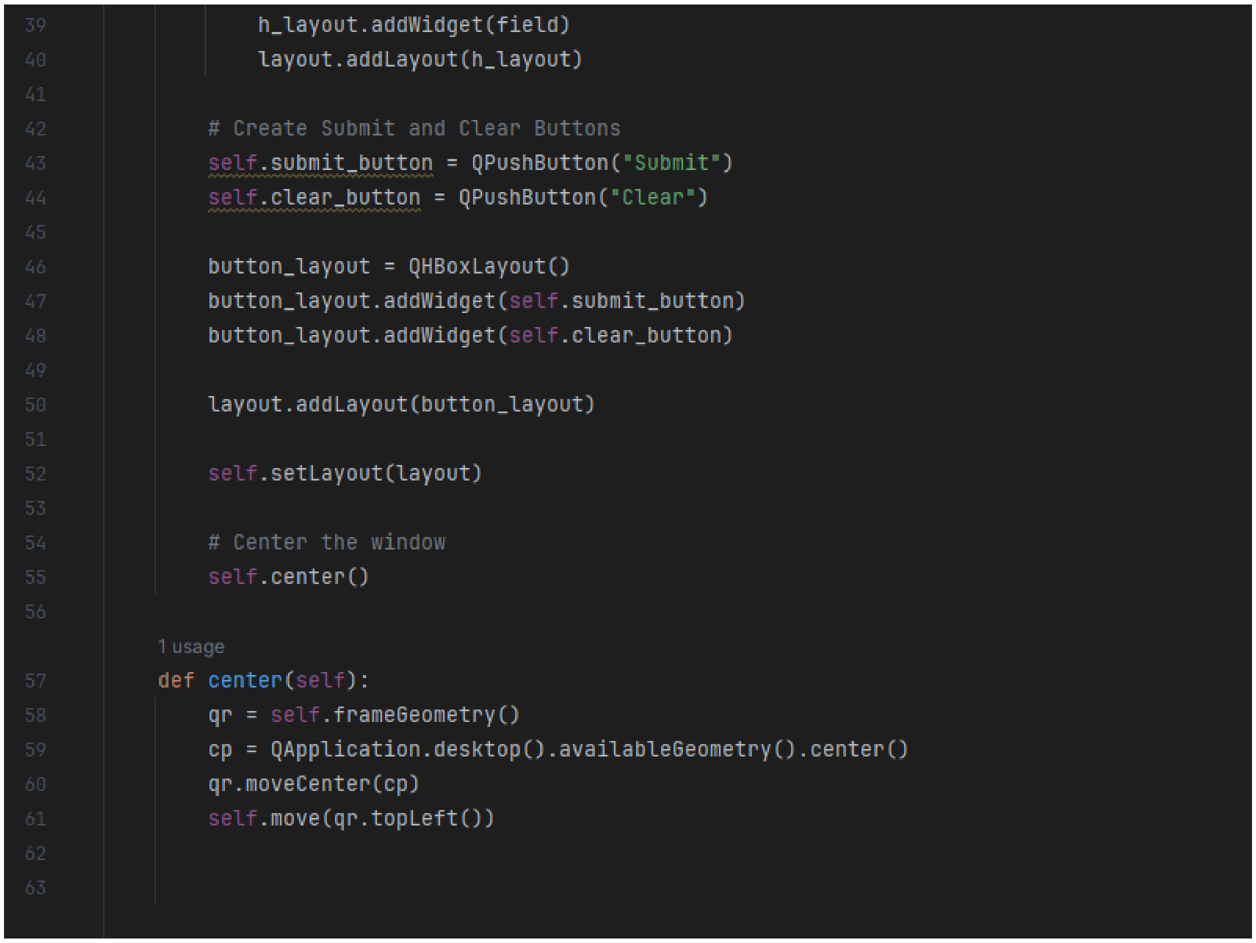


Supplementary Act:

Registration.py:

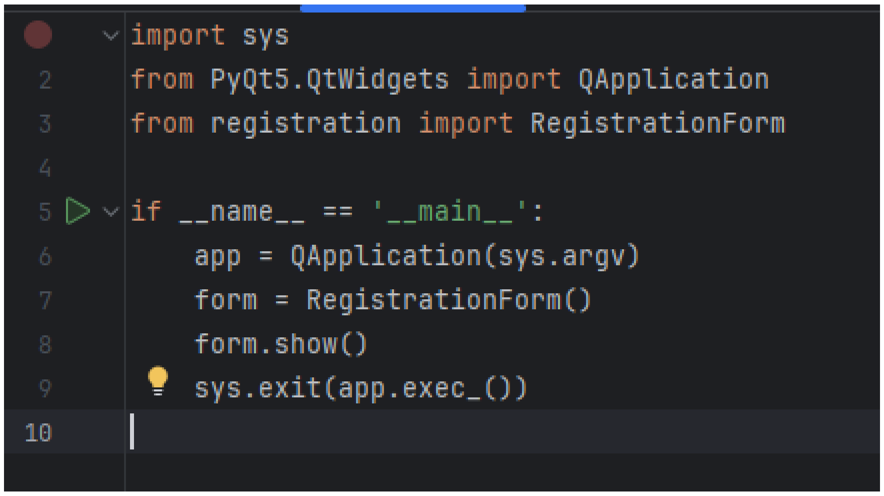


```python
from PyQt5.QtWidgets import QWidget, QLabel, QLineEdit, QPushButton, QVBoxLayout, QHBoxLayout, QApplication
from PyQt5.QtGui import QIcon
import sys


2 usages
class RegistrationForm(QWidget):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Account Registration")
        self.setGeometry(100, 100, 400, 400)
        self.setWindowIcon(QIcon('pythonico.ico'))

        self.initUI()

    1 usage
    def initUI(self):
        layout = QVBoxLayout()

        # Title Label
        title = QLabel("Account Registration System")
        title.setStyleSheet("font-size: 16px; font-weight: bold; margin-top: 5px; margin-bottom: 10px;")
        layout.addWidget(title)

        # Create Labels and Text Fields
        self.fields = [
            ("First Name:", QLineEdit()),
            ("Last Name:", QLineEdit()),
            ("Username:", QLineEdit()),
            ("Password:", QLineEdit()),
            ("Email Address:", QLineEdit()),
            ("Contact Number:", QLineEdit())
        ]

        for label_text, field in self.fields:
            label = QLabel(label_text)
            field.setPlaceholderText("Enter " + label_text.lower())
            h_layout = QHBoxLayout()
            h_layout.addWidget(label)
```

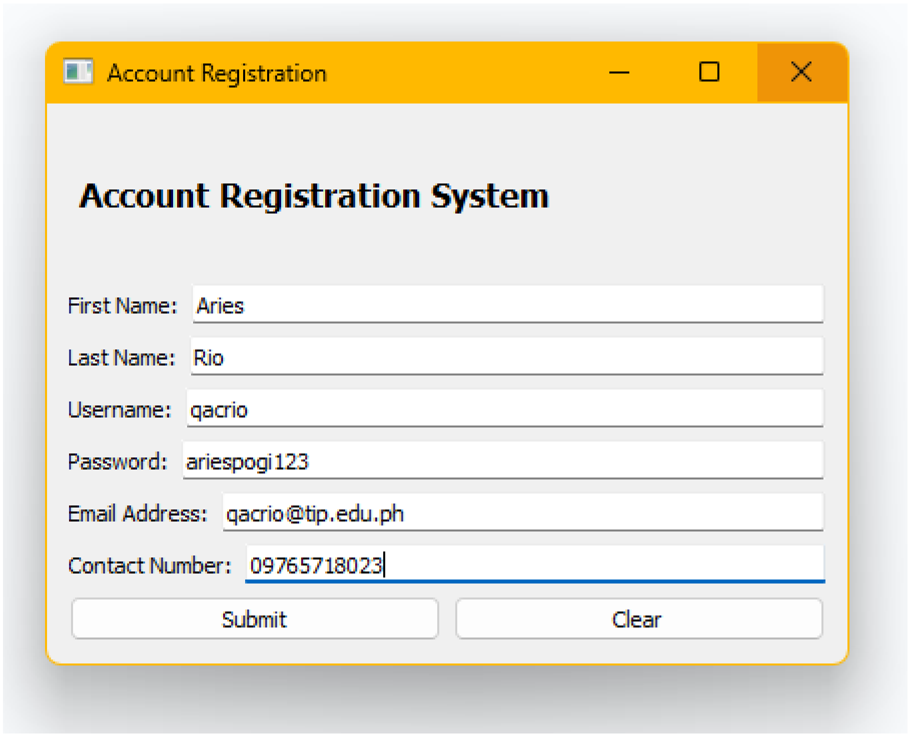


```python
            h_layout.addWidget(field)
            layout.addLayout(h_layout)

        # Create Submit and Clear Buttons
        self.submit_button = QPushButton("Submit")
        self.clear_button = QPushButton("Clear")

        button_layout = QHBoxLayout()
        button_layout.addWidget(self.submit_button)
        button_layout.addWidget(self.clear_button)

        layout.addLayout(button_layout)

        self.setLayout(layout)

        # Center the window
        self.center()

    1 usage
    def center(self):
        qr = self.frameGeometry()
        cp = QApplication.desktop().availableGeometry().center()
        qr.moveCenter(cp)
        self.move(qr.topLeft())


```

Main.py:



```python
import sys
from PyQt5.QtWidgets import QApplication
from registration import RegistrationForm

if __name__ == '__main__':
    app = QApplication(sys.argv)
    form = RegistrationForm()
    form.show()
    sys.exit(app.exec_())

```



Account Registration System

First Name:	Aries
Last Name:	Rio
Username:	qacrio
Password:	ariespogi123
Email Address:	qacrio@tip.edu.ph
Contact Number:	09765718023

Questions:

1. What are the common GUI Applications that general end-users such as home users, students, and office employees use? (give at least 3 and describe each)

Common GUI applications used by home users, students, and office employees include word processors, spreadsheets, and web browsers. **Word processors**, like Microsoft Word, allow users to create and edit text documents with formatting options, making them essential for writing essays, reports, and letters. **Spreadsheets**, such as Microsoft Excel, enable users to organize, analyze, and visualize data through tables and charts, which is crucial for budgeting, data analysis, and project management. **Web browsers**, like Google Chrome or Firefox, provide a user-friendly interface for accessing the internet, allowing users to browse websites, stream videos, and perform online research efficiently.

2. Based on your answer in question 1, why do you think home users, students, and office employees use those GUI programs?

Home users, students, and office employees use these GUI programs primarily for their ease of use and functionality. These applications are designed with intuitive interfaces that simplify complex tasks, making it easier for users to complete their work without requiring extensive technical knowledge. For example, students rely on word processors for writing assignments and spreadsheets for data organization, while office employees use these tools to enhance productivity and collaboration in their daily tasks.

3. How does Pycharm help developers in making GUI applications, what would be the difference if developers made GUI programs without GUI Frameworks such as Pycharm or Tkinter?

PyCharm helps developers by providing a robust Integrated Development Environment (IDE) with features like code completion, debugging tools, and integrated version control, which streamline the development process. If developers were to create GUI programs without frameworks like PyCharm or libraries like Tkinter, they would have to handle lower-level coding tasks and GUI rendering manually, making the development more complex and time-consuming. Frameworks abstract much of the boilerplate code, allowing developers to focus on application logic and design.

4. What are the different platforms a GUI program may be created and deployed on? (Three is required then state why might a program be created on that specific platform)

GUI applications can be created and deployed on various platforms, including Windows, macOS, and Linux. **Windows** is popular for commercial applications due to its large user base and extensive support for software development. **macOS** is favored for design and multimedia applications, leveraging its advanced graphics capabilities and strong integration with hardware. **Linux** is often used for open-source applications, appealing to developers and users who prefer customizable and community-driven software solutions.

5. What is the purpose of app = QApplication(sys.argv), ex = App(), and sys.exit(app.exec_())?

The line app = QApplication(sys.argv) initializes the application and processes command-line arguments, allowing the GUI to handle user events. ex = App() creates an instance of the main application window, which sets up the user interface. Finally, sys.exit(app.exec_()) starts the application’s event loop, which waits for user interaction and updates the interface accordingly; it ensures that the program exits cleanly when the window is closed.

Conclusion:

In conclusion, GUI applications are essential for home users, students, and office employees, with tools like word processors, spreadsheets, and web browsers being widely used for their simplicity and functionality. PyCharm and frameworks like Tkinter make it easier for developers to create these applications by providing useful tools and reducing complexity. Different platforms like Windows, macOS, and Linux offer diverse options for users based on their needs. Overall, GUI development is crucial in making technology accessible and effective for everyday tasks.