

Laboratory Activity No. 9

Introduction to GUI Development using Pycharm

Course Code: CPE103

Program: BSCPE

Course Title: Object-Oriented Programming

Date Performed: MARCH 22, 2025

Section: 1A

Date Submitted: MARCH 22, 2025

Name: CATAHAN, JOSHUA A.

Instructor: ENGR. MARIA RIZETTE SAYO

1. Objective(s):

This activity aims to familiarize students with the Pycharm framework for GUI Development

2. Intended Learning Outcomes (ILOs):

The students should be able to:

2.1 Identify the main components in a GUI Application

2.2 Create a simple GUI Application using Pycharm Widgets

3. Discussion:

A Graphical User Interface (GUI) application is a program that the user can interact with through graphics (windows, buttons, text fields, checkboxes, images, icons, etc..) such as the Desktop GUI of Windows OS by using a mouse and keyboard unlike with a Command-line program or Terminal program that support keyboard inputs only.

Pycharm is an integrated development environment used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems, and supports web development with Django.

4. Materials and Equipment:

Desktop Computer with Anaconda Python or Pycharm
Windows Operating System

5. Procedure:

```

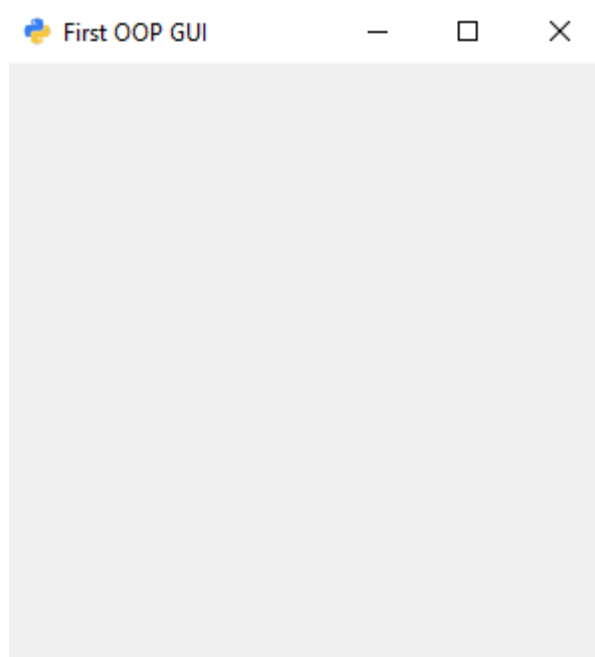
1  import sys
2  from PyQt5.QtWidgets import QMainWindow, QApplication
3  from PyQt5.QtGui import QIcon
4
5  class App(QMainWindow):
6
7      def __init__(self):
8          super().__init__() # initializes the main window like in the previous one
9          # window = QMainWindow()
10         self.title= "First OOP GUI"
11         self.initUI()
12
13     def initUI(self):
14         self.setWindowTitle(self.title)
15         self.setGeometry(200,200,300,300)
16         self.setWindowIcon(QIcon('pythonico.ico')) # sets an icon
17         self.show()
18
19 if __name__ == '__main__':
20     app = QApplication(sys.argv)
21     Main = App()
22     sys.exit(app.exec_())
23

```

2. Run the program and observe the output.

Adding an icon

3. Download any .ico picture from <https://icon-icons.com/> or any similar sites.
4. Place the icon in your folder (ex. Oopfa1<lastname>_lab8)
5. Run the program again, the program should now have an icon similar to the program below.



Creating Buttons

1. Create a new .py file named **gui_buttons.py** then copy the program as shown below:

```

1  import sys
2  from PyQt5.QtWidgets import QWidget, QApplication, QMainWindow, QPushButton
3  from PyQt5.QtGui import QIcon
4
5  class App(QWidget):
6
7      def __init__(self):
8          super().__init__() # initializes the main window like in the previous one
9          # window = QMainWindow()
10         self.title= "PyQt Button"
11         self.x=200 # or left
12         self.y=200 # or top
13         self.width=300
14         self.height=300
15         self.initUI()
16
17     def initUI(self):
18         self.setWindowTitle(self.title)
19         self.setGeometry(self.x,self.y,self.width,self.height)
20         self.setWindowIcon(QIcon('pythonico.ico'))
21
22         # In GUI Python, these buttons, textboxes, labels are called Widgets
23         self.button = QPushButton('Click me!', self)
24         self.button.setToolTip("You've hovered over me!")
25         self.button.move(100,70) # button.move(x,y)
26
27         self.show()
28
29
30 if __name__ == '__main__':
31     app = QApplication(sys.argv)
32     ex = App()
33     sys.exit(app.exec_())

```

2. Run the program and observe the output.
3. Add a new button named button2 named Register to the GUI that will display “this button does nothing.. yet..” when it is hovered.

Creating Text Fields

1. Create a new file named **gui_text.py** and copy the code shown below:

```

1  import sys
2  from PyQt5.QtWidgets import QWidget, QApplication, QMainWindow, QPushButton
3  from PyQt5.QtGui import QIcon
4
5  class App(QWidget):
6
7      def __init__(self):
8          super().__init__() # initializes the main window like in the previous one
9          # window = QMainWindow()
10         self.title= "PyQt Line Edit"
11         self.x=200 # or left
12         self.y=200 # or top
13         self.width=300
14         self.height=300
15         self.initUI()
16
17     def initUI(self):
18         self.setWindowTitle(self.title)
19         self.setGeometry(self.x,self.y,self.width,self.height)
20         self.setWindowIcon(QIcon('pythonico.ico'))
21
22         # Create textbox
23         self.textbox = QLineEdit(self)
24         self.textbox.move(20, 20)
25         self.textbox.resize(280,40)
26
27         self.show()
28
29 if __name__ == '__main__':
30     app = QApplication(sys.argv)
31     ex = App()
32     sys.exit(app.exec_())

```

2. Run the program and observe the error.
3. Add an import QLineEdit to the Pycharm.Widgets import
4. Run the program and observe the output.
5. Add the following code below self.textbox.resize()

```
self.textbox.setText("Set this text value")
```

4. Run the program again then resize the textbox so that it fits in the Window and that its height is just above the written text's height.

Creating Labels

1. Create a new file called **gui_labels.py** and copy the following code below:

```

1  import sys
2  from PyQt5.QtWidgets import QWidget, QApplication, QMainWindow, QPushButton, QLineEdit
3  from PyQt5.QtGui import QIcon
4
5  class App(QWidget):
6
7      def __init__(self):
8          super().__init__() # initializes the main window like in the previous one
9          # window = QMainWindow()
10         self.title= "PyQt Line Edit"
11         self.x=200 # or left
12         self.y=200 # or top
13         self.width=300
14         self.height=300
15         self.initUI()
16
17     def initUI(self):
18         self.setWindowTitle(self.title)
19         self.setGeometry(self.x,self.y,self.width,self.height)
20         self.setWindowIcon(QIcon('pythonico.ico'))
21
22         self.textboxlbl = QLabel("Hello World! ",self)
23         self.textboxlbl.move(30,25)
24
25         self.show()
26
27 if __name__ == '__main__':
28     app = QApplication(sys.argv)
29     ex = App()
30     sys.exit(app.exec_())
31

```

2. Run the program and observe the output.
3. Add the necessary Widget at the import line to make the program run.
4. Center the label by adjusting the parameters of .move(). This is called Absolute Positioning.
5. Create a new label called " " and place it at the center and below the Hello World!

6. Supplementary Activity:

Task

Create an Object-Oriented GUI Application for a simple Account Registration System with the following required information: first name, last name, username, password, email address, contact number.

Requirements:

- The GUI must be centered on your screen.
- The GUI Components should be organized according to the order of information required using Absolute Positioning.
- The position of the components should be automatically computed based on the top component.
- All the text fields should be accompanied with their corresponding label on the left side while the text field is on the right side.
- There should a program title other than the Window Title.
- There should be a submit button and clear button at the bottom (submit button on the left, clear button on the right).
- The program should be launched on **main.py** while the GUI Codes should be on a separate file called **registration.py**

SUPPLEMENTARY ACT PICTURE:

Catahan - Laboratory Activity #9

Account Registration System

Enter your first name:

Enter your last name:

Enter your username:

Enter your password:

Enter your own email:

Enter your contact #:

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)

Some common GUI apps people use are Microsoft Word, Google Chrome, and Microsoft Excel. Word is used for writing documents, Chrome for browsing the internet, and Excel for organizing and analyzing data. These programs are easy to use and help with everyday tasks.

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

People use these apps because they are needed for basic activities. Word helps with writing, Chrome helps with browsing, and Excel helps with organizing information. They're simple, widely used, and make tasks much easier.

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 makes it easier for developers to create GUI apps by giving them everything in one place to code, test, and fix errors. Without tools like PyCharm or Tkinter, developers would have to create everything from scratch, which would take a lot more time and be harder.

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 programs can be created for Windows, macOS, and Linux. Windows is common because so many people use it, macOS is popular for Apple users, and Linux is good for people who like open-source software and customizing their systems.

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

The lines `app = QApplication(sys.argv)`, `ex = App()`, and `sys.exit(app.exec_())` are used to start the app.

`app = QApplication(sys.argv)` begins the app, `ex = App()` creates the app, and `sys.exit(app.exec_())`

keeps it running until the user closes it

7. Conclusion:

In this lab, we focused on coding a simple "Account Registration System" using PyQt5. We learned how to set up a GUI with labels, textboxes, and buttons for user input. The lab showed how frameworks like PyQt5 simplify the development process by handling complex tasks like window design, layout, and interaction. While the questions provided context on common GUI apps and development tools like PyCharm, the main takeaway was how coding with PyQt5 allows us to quickly build functional, interactive applications. By understanding how to use the basic code structure and elements, we were able to create a user-friendly app that could be expanded and customized further.

8. Assessment Rubric: