**Lab Exercise 2 – Creating Basic Windows and Layouts in PySide6**

**Objective**

This lab will teach you how to create basic windows using QMainWindow and QWidget, and how to use different layouts (QVBoxLayout, QHBoxLayout, QGridLayout) in PySide6.

**Prerequisites**

* Python 3.7+
* PySide6 installed (pip install PySide6)

**Part 1: Creating a Basic Window**

**Step 1: Create a Simple Window using QWidget**

1. Create a new Python file basic\_window.py.
2. Write the following code:

from PySide6.QtWidgets import QApplication, QWidget

import sys

app = QApplication(sys.argv)

window = QWidget()

window.setWindowTitle("Basic Window Example")

window.resize(400, 300)

window.show()

sys.exit(app.exec())

**Explanation**

* QApplication initializes the application.
* QWidget creates a simple window.
* setWindowTitle() sets the title.
* resize() defines the window size.
* show() displays the window.
* app.exec() starts the event loop.

**Run the Code**

python basic\_window.py

**Part 2: Creating a Window with QMainWindow**

1. Create a file main\_window.py.
2. Write the following code:

from PySide6.QtWidgets import QApplication, QMainWindow

import sys

class MainWindow(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle("QMainWindow Example")

self.resize(500, 400)

app = QApplication(sys.argv)

window = MainWindow()

window.show()

sys.exit(app.exec())

**Explanation**

* QMainWindow is used for more advanced windows with menus, toolbars, and status bars.
* This code sets up a basic QMainWindow.

**Run the Code**

python main\_window.py

**Part 3: Using Layouts**

**Step 1: Vertical Layout (QVBoxLayout)**

1. Create a file vbox\_layout.py.
2. Write the following code:

from PySide6.QtWidgets import QApplication, QWidget, QPushButton, QVBoxLayout

import sys

class Window(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle("Vertical Layout Example")

layout = QVBoxLayout()

layout.addWidget(QPushButton("Button 1"))

layout.addWidget(QPushButton("Button 2"))

layout.addWidget(QPushButton("Button 3"))

self.setLayout(layout)

app = QApplication(sys.argv)

window = Window()

window.show()

sys.exit(app.exec())

**Explanation**

* QVBoxLayout() arranges widgets vertically (top to bottom).
* addWidget() adds buttons inside the layout.
* setLayout(layout) applies the layout to the window.

**Step 2: Grid Layout (QGridLayout)**

1. Create a file grid\_layout.py.
2. Write the following code:

from PySide6.QtWidgets import QApplication, QWidget, QPushButton, QGridLayout

import sys

class Window(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle("Grid Layout Example")

layout = QGridLayout()

layout.addWidget(QPushButton("Button 1"), 0, 0)

layout.addWidget(QPushButton("Button 2"), 0, 1)

layout.addWidget(QPushButton("Button 3"), 1, 0)

layout.addWidget(QPushButton("Button 4"), 1, 1)

self.setLayout(layout)

app = QApplication(sys.argv)

window = Window()

window.show()

sys.exit(app.exec())

**Explanation**

* QGridLayout() organizes widgets in a grid-like format.
* addWidget(widget, row, column) places widgets in specific positions.

**Part 4: Lab Tasks**

1. Modify the QMainWindow example to add a **menu bar** with a "File" menu containing an "Exit" action.
2. Experiment with different layouts (QHBoxLayout, QFormLayout).
3. Add a **status bar** to display messages in QMainWindow.

**Conclusion**

This lab covers:

* Creating basic windows using QWidget and QMainWindow.
* Applying different layouts (QVBoxLayout, QGridLayout).
* Structuring GUI applications efficiently in PySide6.