**Lab Exercise 32– TDD in PyQT**

Test-driven development (TDD) is a software development approach where tests are written before the actual code. In the context of PyQt, this methodology involves writing tests for your application's functionality before implementing the code that makes the application work. This approach helps ensure that the code is functional and meets the specified requirements.

Here's an example of how you can implement test-driven development in PyQt:

import sys

import unittest

from PyQt5.QtWidgets import QApplication, QMainWindow, QPushButton

class MyWindow(QMainWindow):

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

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

self.setGeometry(100, 100, 300, 200)

self.setWindowTitle('Test Driven Development in PyQt')

self.button = QPushButton('Click me', self)

self.button.setGeometry(100, 50, 100, 50)

self.button.clicked.connect(self.on\_click)

def on\_click(self):

self.button.setText('Clicked')

class TestMyWindow(unittest.TestCase):

def setUp(self):

self.app = QApplication(sys.argv)

self.window = MyWindow()

def tearDown(self):

self.app.quit()

def test\_initial\_button\_text(self):

self.assertEqual(self.window.button.text(), 'Click me')

def test\_button\_click(self):

self.window.button.click()

self.assertEqual(self.window.button.text(), 'Clicked')

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

In this example, we have a simple PyQt application with a button. We write unit tests for this application using the unittest module. The setUp and tearDown methods handle the application's setup and cleanup. The test methods check the initial state of the button text and validate its behavior after clicking.

You can run this script using any Python IDE or by executing it from the command line. Make sure you have PyQt5 and the unittest module installed. This approach can help you create more reliable and robust PyQt applications by ensuring that each component behaves as expected.