**Lab Exercise 28 – Singleton Design Pattern in PyQT**

In PyQt, you can implement the Singleton design pattern using a metaclass. Here's an example of how to create a Singleton class in PyQt:

from PyQt5.QtCore import QObject

class Singleton(QObject):

\_instance = None

def \_\_new\_\_(cls, \*args, \*\*kwargs):

if not cls.\_instance:

cls.\_instance = super().\_\_new\_\_(cls, \*args, \*\*kwargs)

return cls.\_instance

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

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

def some\_method(self):

# Add your methods here

pass

# Example usage

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

instance1 = Singleton()

instance2 = Singleton()

print(instance1 is instance2) # Output: True

In this implementation, the Singleton class ensures that only one instance of the class is created. When you create multiple instances of the Singleton class, it returns the same instance every time, making it truly a singleton.

You can add your own methods and properties to the Singleton class as needed. This implementation is a basic example of how to create a Singleton in PyQt, ensuring that there is only one instance of a class throughout the application.