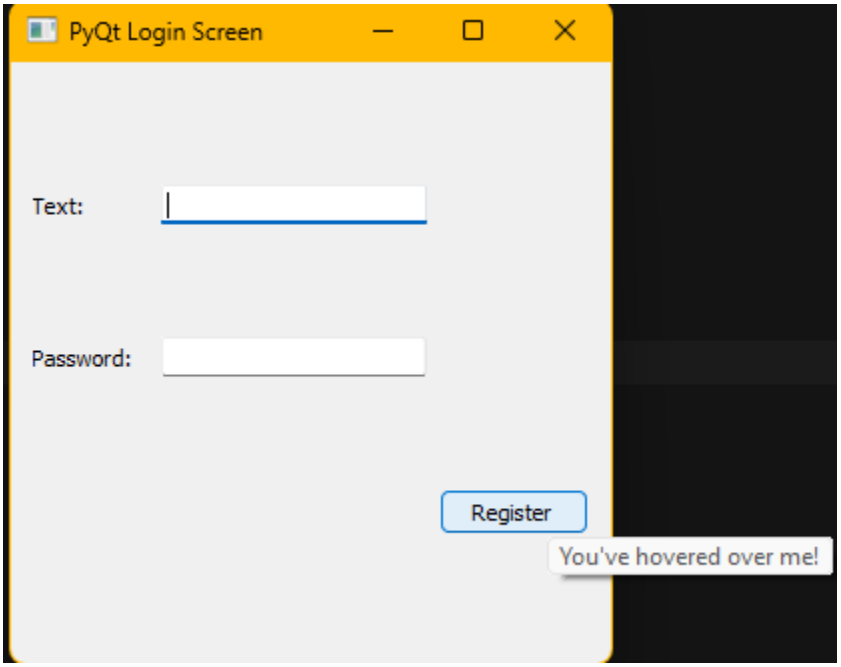
Laboratory Activity No. 6 GUI Design: Layout and Styling	
Magistrado, Aira Pauleen M.	10/28/24
BSCPE/CPE21S4	Maria Rizette Sayo

Basic Grid Layout

gui_grid1.py

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

1 usage
class App(QWidget):
    def __init__(self):
        super().__init__()
        self.title = "PyQt Login Screen"
        self.x = 200  # or left
        self.y = 200  # or right
        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('python.ico'))
        self.createGridLayout()
        self.setLayout(self.layout)
        self.show()

    1 usage
    def createGridLayout(self):
        self.layout = QGridLayout()
        self.layout.setColumnStretch( column: 1,  stretch: 2)

        self.textboxlbl = QLabel("Text: ", self)
        self.textbox = QLineEdit(self)
        self.passwordlbl = QLabel("Password: ", self)
        self.password = QLineEdit(self)
        self.password.setEchoMode(QLineEdit.Password)
        self.button = QPushButton('Register', self)
        self.button.setToolTip("You've hovered over me!")

        self.layout.addWidget(self.textboxlbl, 0, 1)
        self.layout.addWidget(self.textbox, 0, 2)
        self.layout.addWidget(self.passwordlbl, 1, 1)
```

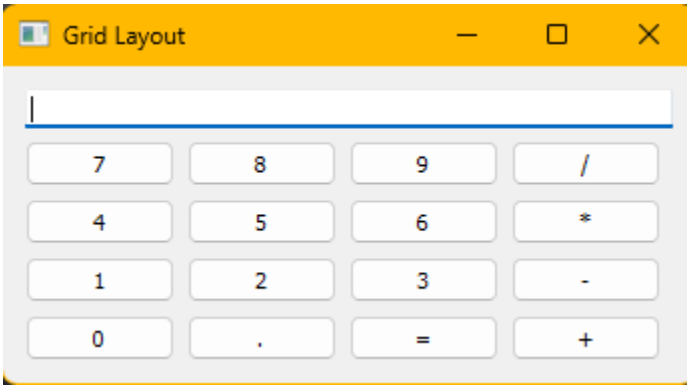
```
35          self.layout.addWidget(self.textboxlbl, 0, 1)
36          self.layout.addWidget(self.textbox, 0, 2)
37          self.layout.addWidget(self.passwordlbl, 1, 1)
38          self.layout.addWidget(self.password, 1, 2)
39          self.layout.addWidget(self.button, 2, 3)
40
41
42  if __name__ == '__main__':
43      app = QApplication(sys.argv)
44      ex = App()
45      sys.exit(app.exec_())
46
```

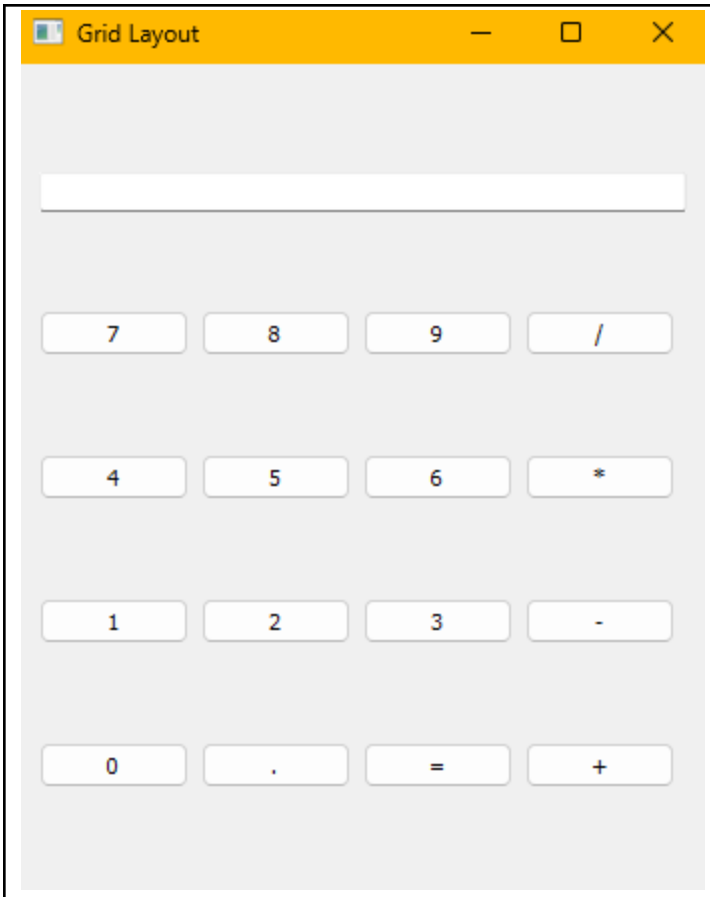


Grid Layout using Loops

gui_grid1.py gui_grid2.py

```python
# Grid Layout
import sys
from PyQt5.QtWidgets import QGridLayout, QLineEdit, QPushButton, QHBoxLayout, QVBoxLayout, QWidget, QApplication

1 usage
class GridExample(QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()

    1 usage
    def initUI(self):
        grid = QGridLayout()
        self.setLayout(grid)

        names = [
            '7', '8', '9', '/', '',
            '4', '5', '6', '*', '',
            '1', '2', '3', '-', '',
            '0', '.', '=', '+', ''
        ]

        self.textLine = QLineEdit(self)
        grid.addWidget(self.textLine, 0, 0, 1, 5)

        # Using a loop to generate positions
        positions = [(i, j) for i in range(1, 6) for j in range(5)]
        for position, name in zip(positions, names):
            if name == '':
                continue
            button = QPushButton(name)
            grid.addWidget(button, *position)

        self.setGeometry(300, 300, 300, 150)
        self.setWindowTitle('Grid Layout')
        self.show()

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = GridExample()
```





Vbox and Hbox layout managers (Simple Notepad)

gui_grid1.py gui_grid2.py gui_simplenotepad.py

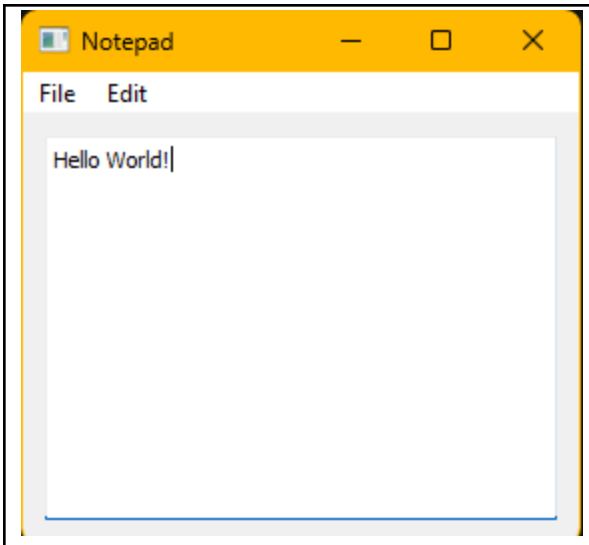
```python
import sys
from PyQt5.QtWidgets import*
from PyQt5.QtGui import QIcon

1 usage
class MainWindow(QMainWindow):

    def __init__(self):
        super().__init__()
        self.setWindowTitle("Notepad")
        self.setWindowIcon(QIcon('python.ico'))
        self.loadmenu()

        self.loadwidget()
        self.show()

    1 usage
    def loadmenu(self):
        mainMenu = self.menuBar()
        fileMenu = mainMenu.addMenu('File')
        editMenu = mainMenu.addMenu('Edit')

        editButton = QAction('Clear', self)
        editButton.setShortcut('Ctrl+M')
        editButton.triggered.connect(self.cleartext)
        editMenu.addAction(editButton)

        fontButton = QAction('Font', self)
        fontButton.setShortcut('Ctrl+D')
        fontButton.triggered.connect(self.showFontDialog)
        editMenu.addAction(fontButton)

        saveButton = QAction('Save', self)
        saveButton.setShortcut('Ctrl+S')
        saveButton.triggered.connect(self.saveFileDialog)
        editMenu.addAction(saveButton)

        openButton = QAction('Open', self)
        openButton.setShortcut('Ctrl+O')
        openButton.triggered.connect(self.openFileNameDialog)
```

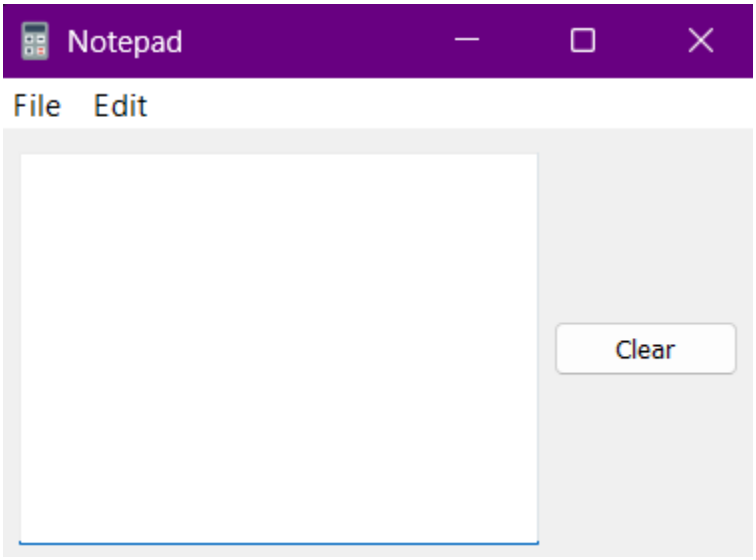
gui_grid1.py gui_grid2.py gui_simplenotepad.py ×

```
5   class MainWindow(QMainWindow):
16      def loadmenu(self):
39          fileMenu.addAction(openButton)
40
41          exitButton = QAction('Exit', self)
42          exitButton.setShortcut('Ctrl+Q')
43          exitButton.setStatusTip('Exit application')
44          exitButton.triggered.connect(self.close)
45          fileMenu.addAction(exitButton)
46
            1 usage
47      def showFontDialog(self):
48          font, ok = QFontDialog.getFont()
49          if ok:
50              self.notepad.text.setFont(font)
51
            1 usage
52      def saveFileDialog(self):
53          options = QFileDialog.Options()
54          # options l = QFileDialog.DontUseNativeDialog
55          fileName, _ = QFileDialog.getSaveFileName(self, caption: "Save notepad file", directory: "",
56                                                    filter: "Text Files(*.txt);; Python Files (*.py);; All files (*)", options=options)
57          if fileName:
58              with open(fileName, 'w') as file:
59                  file.write(self.notepad.text.toPlainText())
60
            1 usage
61      def openFileNameDialog(self):
62          options = QFileDialog.Options()
63          # options = QFileDialog.DontUseNativeDialog
64          fileName, _ = QFileDialog.getOpenFileName(self, caption: "Open notepad file", directory: "",
65                                                    filter: "Text Files (*.txt);; Python Files (*.py);; All files (*)", options=options)
66          if fileName:
67              with open(fileName, 'r') as file:
68                  data = file.read()
69                  self.notepad.text.setText(data)
70
            1 usage
71      def cleartext(self):
72          self.notepad.text.clear()
```

gui_grid1.py gui_grid2.py gui_simplenotepad.py

```
5   class MainWindow(QMainWindow):
71      def cleartext(self):
72          self.notepad.text.clear()
73
            1 usage
74      def loadwidget(self):
75          self.notepad = Notepad()
76          self.setCentralWidget(self.notepad)
77
        2 usages
78  class Notepad(QWidget):
79
80      def __init__(self):
81          super(Notepad, self).__init__()
82          self.text = QTextEdit(self)
83          self.clearbtn = QPushButton("Clear")
84          self.clearbtn.clicked.connect(self.cleartext)
85
86          self.initUI()
87          self.setLayout(self.layout)
88          windowLayout = QVBoxLayout()
89          windowLayout.addWidget(self.horizontalGroupBox)
90          self.show()
91
            1 usage
92      def initUI(self):
93          self.horizontalGroupBox = QGroupBox("Grid")
94          self.layout = QHBoxLayout()
95          self.layout.addWidget(self.text)
96          #self.layout.addWidget(self.clearbtn)
97          self.horizontalGroupBox.setLayout(self.layout)
98
            1 usage
99      def cleartext(self):
100         self.text.clear()
101
102 if __name__ == '__main__':
103     app = QApplication(sys.argv)
104     ex = MainWindow()
105     sys.exit(app.exec_())
```





Supplementary Activity

Task

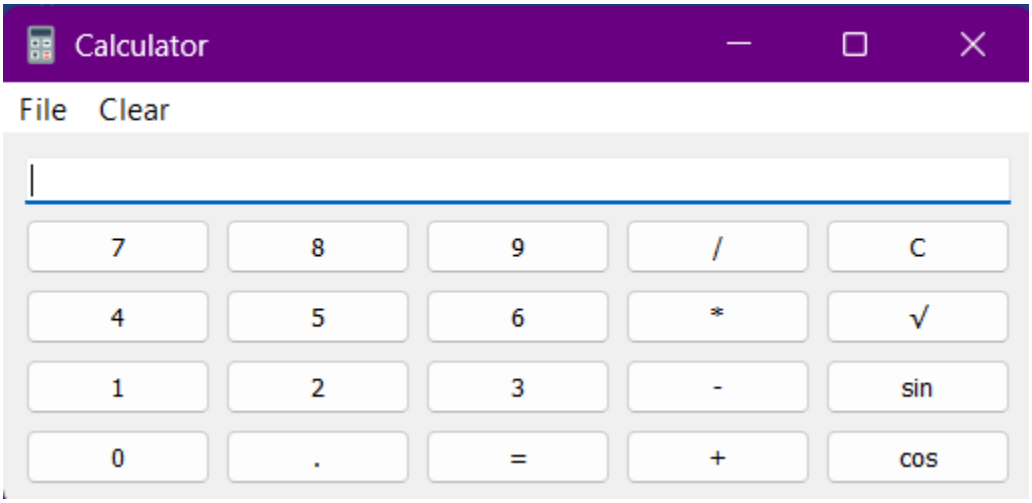
Make a calculator program that can compute perform the Arithmetic operations as well as exponential operation, sin, cosine math functions as well clearing using the C button and/or clear from a menu bar. The calculator must be able to store and retrieve the operations and results in a text file. A file menu should be available and have the option Exit which should also be triggered when ctrl+Q is pressed on the keyboard. You may refer to your calculator program in the Desktop.

```
import sys
import math
from PyQt5.QtWidgets import QMainWindow, QApplication, QWidget, QLineEdit,
QPushButton, QGridLayout, \
   QVBoxLayout, QAction, QFileDialog
from PyQt5.QtGui import QIcon


class Calculator(QMainWindow):
   def __init__(self):
```

```
        super().__init__()
        self.setWindowTitle("Calculator")
        self.setWindowIcon(QIcon('python.ico'))
        self.initUI()

    def initUI(self):
        self.centralWidget = QWidget()
        self.setCentralWidget(self.centralWidget)

        self.createMenu()
        self.createButtons()
        self.createLayout()

        self.show()

    def createMenu(self):
        menuBar = self.menuBar()

        fileMenu = menuBar.addMenu('File')

        saveAction = QAction('Save', self)
        saveAction.setShortcut('Ctrl+S')
        saveAction.triggered.connect(self.saveToFile)
        fileMenu.addAction(saveAction)

        openAction = QAction('Open', self)
        openAction.setShortcut('Ctrl+O')
        openAction.triggered.connect(self.openFileNameDialog)
        fileMenu.addAction(openAction)

        exitAction = QAction('Exit', self)
        exitAction.setShortcut('Ctrl+Q')
        exitAction.triggered.connect(self.close)
        fileMenu.addAction(exitAction)

        clearAction = QAction('Clear', self)
        clearAction.setShortcut('Ctrl+C')
        clearAction.triggered.connect(self.clear)
        menuBar.addAction(clearAction)

    def createButtons(self):
        self.display = QLineEdit(self)

        self.buttons = [
            '7', '8', '9', '/', 'C',
            '4', '5', '6', '*', '√',
            '1', '2', '3', '-', 'sin',
            '0', '.', '=', '+', 'cos'
        ]

        self.gridLayout = QGridLayout()
        self.gridLayout.addWidget(self.display, 0, 0, 1, 5)

        # Using a loop to generate positions
        positions = [(i, j) for i in range(1, 5) for j in range(5)]
        for position, button_text in zip(positions, self.buttons):
            button = QPushButton(button_text)
```

```
            button.clicked.connect(self.onButtonClick)
            self.gridLayout.addWidget(button, *position)

    def createLayout(self):
        mainLayout = QVBoxLayout()
        mainLayout.addLayout(self.gridLayout)
        self.centralWidget.setLayout(mainLayout)

    def onButtonClick(self):
        sender = self.sender().text()

        if sender == 'C':
            self.clear()
        elif sender == '=':
            self.calculate()
        elif sender in ['sin', 'cos', '√']:
            self.calculateSpecial(sender)
        else:
            self.display.setText(self.display.text() + sender)

    def calculate(self):
        try:
            result = eval(self.display.text())
            self.display.setText(str(result))
        except Exception as e:
            self.display.setText("Error")

    def calculateSpecial(self, operation):
        try:
            value = float(self.display.text())
            if operation == 'sin':
                result = math.sin(math.radians(value))
            elif operation == 'cos':
                result = math.cos(math.radians(value))
            elif operation == '√':
                result = math.sqrt(value)
            self.display.setText(str(result))
        except Exception as e:
            self.display.setText("Error")

    def clear(self):
        self.display.clear()

    def saveToFile(self):
        options = QFileDialog.Options()
        fileName, _ = QFileDialog.getSaveFileName(self, "Save Calculator
History", "",
                                                  "Text Files (*.txt);;All
Files (*)", options=options)
        if fileName:
            with open(fileName, 'w') as file:
                file.write(self.display.text())

    def openFileNameDialog(self):
        options = QFileDialog.Options()
        fileName, _ = QFileDialog.getOpenFileName(self, "Load Calculator
History", "",
```

```
                                                    "Text Files (*.txt);;All
Files (*)", options=options)
        if fileName:
            with open(fileName, 'r') as file:
                data = file.read()
                self.display.setText(data)


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



Conclusion

Through the lab exercise, I learned how to use Grid Layout to construct a basic login screen and how to create button positions in the grid using loops. I learned how to make a simple Notepad that includes a menu bar with File and Edit buttons. Where you can save text to a txt or py file, open and load files, exit the GUI, alter the fonts, and clear the notepad's text. In the supplementary activity, I learned how to create a basic functional calculator that performs fundamental computations like sine, cosine, and square root using the desktop calculator as a guide. The knowledge I gained from this lab gave me a good foundation for using PyQt5 to create GUIs and will help me with future programming projects.