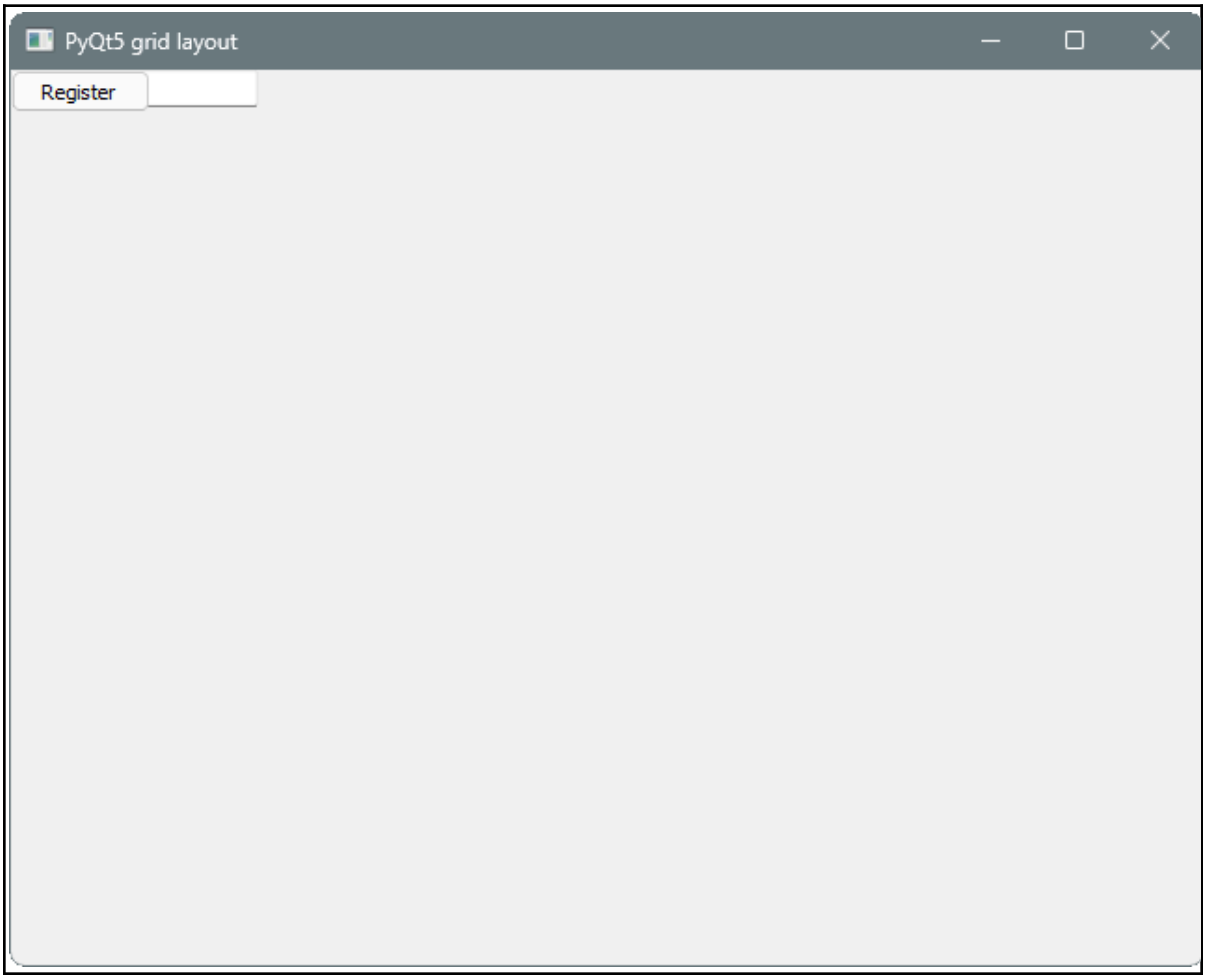
Laboratory Activity 6 - GUI Design: Layout and Styling	
Bonifacio, Redj Guillian F.	11/07/2024
CPE 009B / CPE21S1	Sayo, Maria Rizette

5. Procedure:

Basic Grid Layout

gui_grid1.py

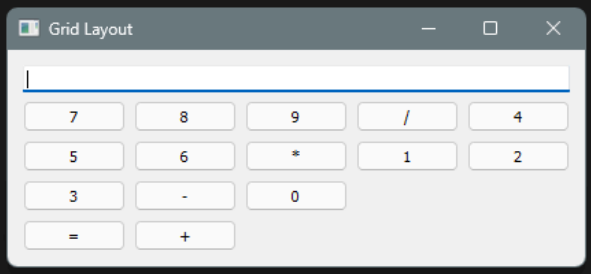
```
from PyQt5.QtWidgets import QWidget, QGridLayout, QLabel, QLineEdit, QPushButton, QApplication
import sys

class App(QWidget):
    def __init__(self):
        super().__init__()
        self.title = 'PyQt5 grid layout'
        self.left = 100
        self.top = 100
        self.width = 640
        self.height = 480
        self.initUI()

    def initUI(self):
        self.setWindowTitle(self.title)
        self.setGeometry(self.left, self.top, self.width, self.height)

        self.createGridLayout()

        self.show()

    def createGridLayout(self):
        self.layout = QGridLayout()
        self.layout.setColumnStretch(1,2)
        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)
        self.layout.addWidget(self.password, 1, 2)
        self.layout.addWidget(self.button, 2, 2)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = App()
    sys.exit(app.exec_())

```



Grid Layout using Loops

gui_grid2.py

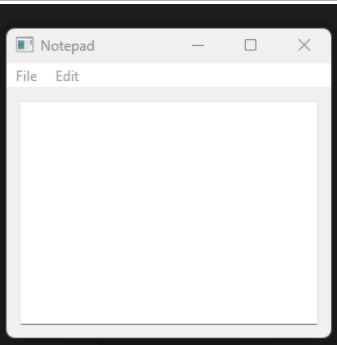
```python
#Grid Layout
import sys
from PyQt5.QtWidgets import QGridLayout, QLineEdit, QPushButton, \
QHBoxLayout, QVBoxLayout, QWidget, QApplication

class GridExample (QWidget):

    def __init__(self):
        super().__init__()
        self.initUI()

    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, 0,1,1,5)

        # using a loop to generate positions
        positions = [(i,j) for i in range(1,7) for j in range(1,6)]
        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()
    sys.exit(app.exec_())

```



Vbox and Hbox layout managers (Simple Notepad)

gui_simplenotepad.py

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

class MainWindow(QMainWindow):

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

    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)
        fileMenu.addAction(saveButton)
        openButton = QAction('Open', self)
        openButton.setShortcut('Ctrl+0')
        openButton.triggered.connect(self.openFileNameDialog)
        fileMenu.addAction (openButton)
        exitButton = QAction('Exit', self)
        exitButton.setShortcut('Ctrl+Q')
        exitButton.setStatusTip('Exit application')
        exitButton.triggered.connect(self.close)
        fileMenu.addAction(exitButton)

    def showFontDialog(self):
        font, ok = QFontDialog.getFont()
        if ok:
            self.notepad.text.setFont(font)

    def saveFileDialog(self):
        options = QFileDialog.Options()
        # options |= QFileDialog.DontUseNativeDialog
        fileName, _ = QFileDialog.getSaveFileName(self, "Save notepad file", "",
```



```
                                                    Text Files (*.txt);; Python Files (*.py);; All Files (*)", options=options)
        if fileName:
            with open(fileName, 'w') as file:
                file.write (self.notepad.text.toPlainText())

    def openFileNameDialog(self):
        options = QFileDialog.Options()
        #options = QFileDialog.DontUseNativeDialog
        fileName, _ = QFileDialog.getOpenFileName(self, "Open notepad file", "",
                                                  "Text Files (*.txt);; Python Files (*.py);; All files (*)", options=options)
        if fileName:
            with open(fileName, 'r') as file:
                data = file.read()
                self.notepad.text.setText(data)

    def cleartext(self):
        self.notepad.text.clear()

    def loadwidget(self):
        self.notepad = Notepad()
        self.setCentralWidget(self.notepad)

class Notepad(QWidget):
    def __init__(self):
        super(Notepad, self).__init__()
        self.text = QTextEdit(self)
        self.clearbtn = QPushButton("Clear")
        self.clearbtn.clicked.connect(self.cleartext)
        self.initUI()
        self.setLayout(self.layout)
        windowLayout = QVBoxLayout()
        windowLayout.addWidget(self.horizontalGroupBox)
        self.setLayout(windowLayout)
        self.show()

    def initUI(self):
        self.horizontalGroupBox = QGroupBox("Grid")
        self.layout = QHBoxLayout()
        self.layout.addWidget(self.text)
        #self.layout.addWidget(self.clearbtn)
        self.horizontalGroupBox.setLayout(self.layout)

    def cleartext(self):
        self.text.clear()

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = MainWindow()
    sys.exit(app.exec_())

```

6. Supplementary Activity:

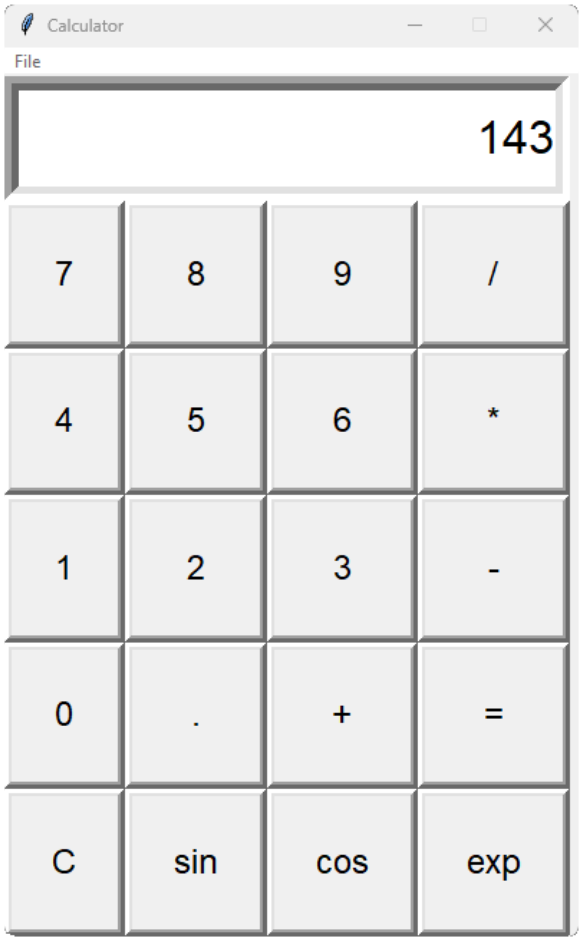
Source Code: Calculator Program

```
import tkinter as tk
from tkinter import messagebox, filedialog
import math

class Calculator(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Calculator")
```

```
        self.geometry("400x600")
        self.resizable(False, False)

        self.result_var = tk.StringVar()
        self.create_widgets()

        self.bind('<Control-q>', lambda e: self.exit_program())

    def create_widgets(self):
        entry = tk.Entry(self, textvariable=self.result_var, font=("Arial", 24), justify='right', bd=10)
        entry.grid(row=0, column=0, columnspan=4, sticky='nsew')

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

        for (text, row, col) in buttons:
            button = tk.Button(self, text=text, command=lambda t=text: self.on_button_click(t),
font=("Arial", 18), bd=5)
            button.grid(row=row, column=col, sticky='nsew', ipadx=10, ipady=10)

        for i in range(6):
            self.grid_rowconfigure(i, weight=1)
            self.grid_columnconfigure(i, weight=1)

        menu = tk.Menu(self)
        file_menu = tk.Menu(menu, tearoff=0)
        file_menu.add_command(label="Save", command=self.save_to_file)
        file_menu.add_command(label="Load", command=self.load_from_file)
        file_menu.add_separator()
        file_menu.add_command(label="Exit", command=self.exit_program)
        menu.add_cascade(label="File", menu=file_menu)
        self.config(menu=menu)

    def on_button_click(self, char):
        if char == 'C':
            self.result_var.set('')
        elif char == '=':
            try:
                expression = self.result_var.get()
                result = eval(expression)
                self.result_var.set(result)
```

```
            self.save_to_file(expression + ' = ' + str(result))
        except Exception as e:
            messagebox.showerror("Error", "Invalid expression")
            self.result_var.set('')
    elif char in ('sin', 'cos', 'exp'):
        try:
            value = float(self.result_var.get())
            if char == 'sin':
                result = math.sin(math.radians(value))
            elif char == 'cos':
                result = math.cos(math.radians(value))
            elif char == 'exp':
                result = math.exp(value)
            self.result_var.set(result)
            self.save_to_file(f"{char}({value}) = {result}")
        except ValueError:
            messagebox.showerror("Error", "Invalid input for sin/cos/exp")
            self.result_var.set('')
    else:
        current_text = self.result_var.get()
        self.result_var.set(current_text + char)

def save_to_file(self, data):
    with open("calculator_history.txt", "a") as file:
        file.write(data + '\n')

def load_from_file(self):
    try:
        with open("calculator_history.txt", "r") as file:
            history = file.read()
        messagebox.showinfo("History", history)
    except FileNotFoundError:
        messagebox.showerror("Error", "No history file found")

def exit_program(self):
    self.quit()

if __name__ == "__main__":
    app = Calculator()
    app.mainloop()
```

Output: Calculator Program



7. Conclusion:

This lab on GUI Design helped me build important skills in creating user-friendly applications. By making a calculator and notepad, I learned how to arrange buttons and text boxes neatly using layout tools, so the apps look good on different screen sizes. I also practiced adding interactive features, like button clicks and shortcuts, and made my apps look better with custom fonts and icons. This hands-on experience with PyQt5 has prepared me to design easy-to-use interfaces in future projects.