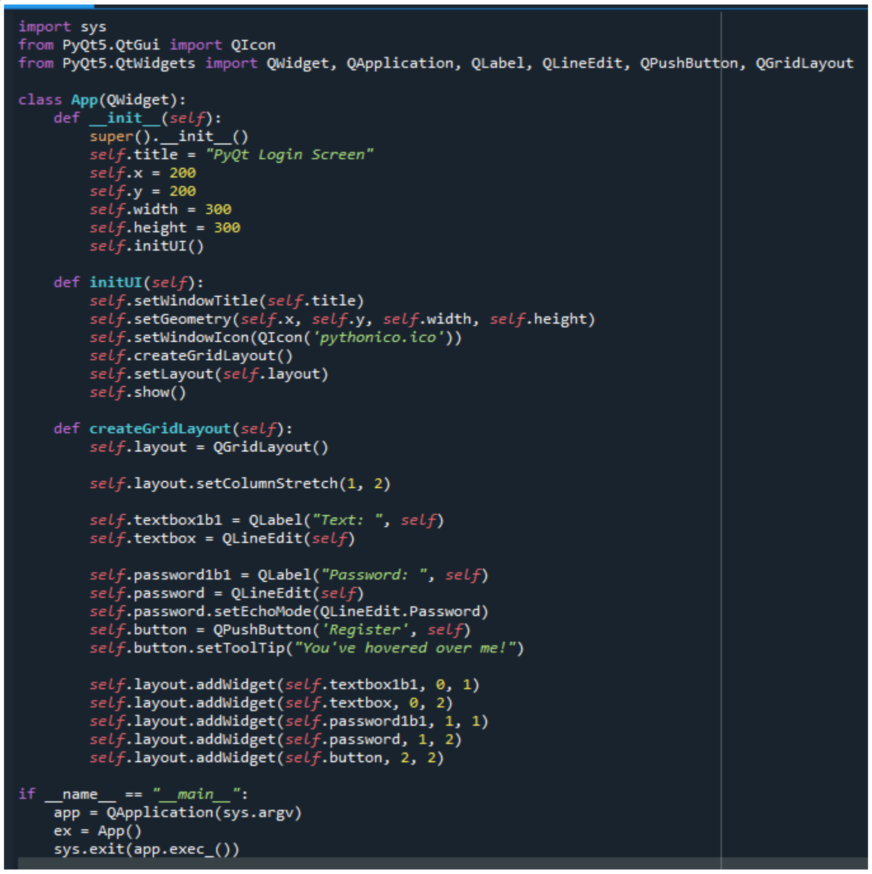
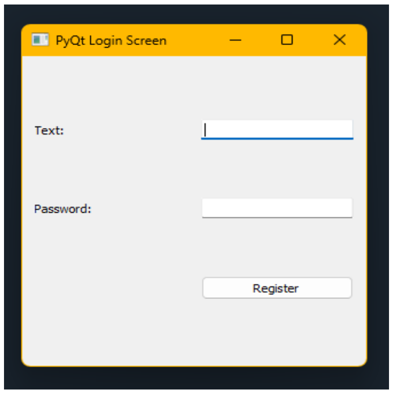
Activity Name #7 - Laboratory Activity 6 - GUI Design: Layout and Styling	
Fernandez, Don Eleazar T.	10/28/2024
CPE009/CPE21S4	Maria Rizette Sayo

Procedure:

Basic Grid Layout

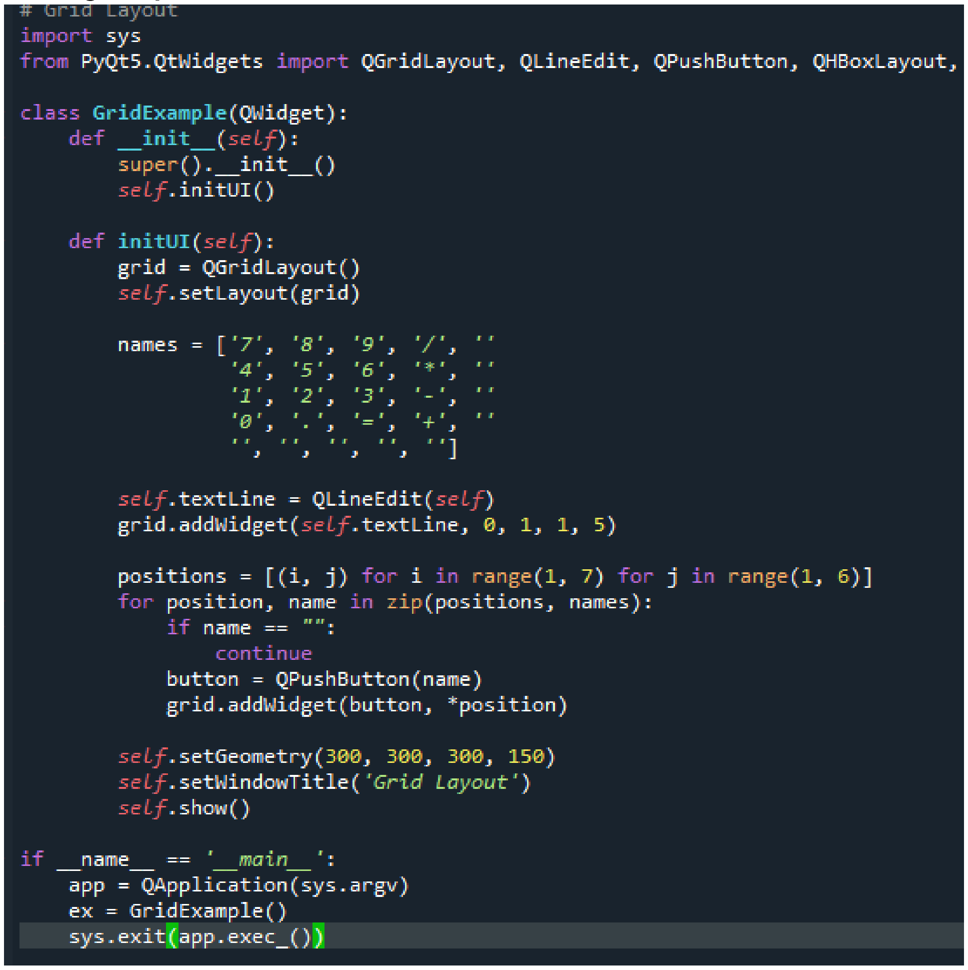


```
import sys
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import QWidget, QApplication, QLabel, QLineEdit, QPushButton, QGridLayout

class App(QWidget):
    def __init__(self):
        super().__init__()
        self.title = "PyQt Login Screen"
        self.x = 200
        self.y = 200
        self.width = 300
        self.height = 300
        self.initUI()

    def initUI(self):
        self.setWindowTitle(self.title)
        self.setGeometry(self.x, self.y, self.width, self.height)
        self.setWindowIcon(QIcon('pythonico.ico'))
        self.createGridLayout()
        self.setLayout(self.layout)
        self.show()

    def createGridLayout(self):
        self.layout = QGridLayout()

        self.layout.setColumnStretch(1, 2)

        self.textbox1b1 = QLabel("Text: ", self)
        self.textbox = QLineEdit(self)

        self.password1b1 = 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.textbox1b1, 0, 1)
        self.layout.addWidget(self.textbox, 0, 2)
        self.layout.addWidget(self.password1b1, 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_())
```

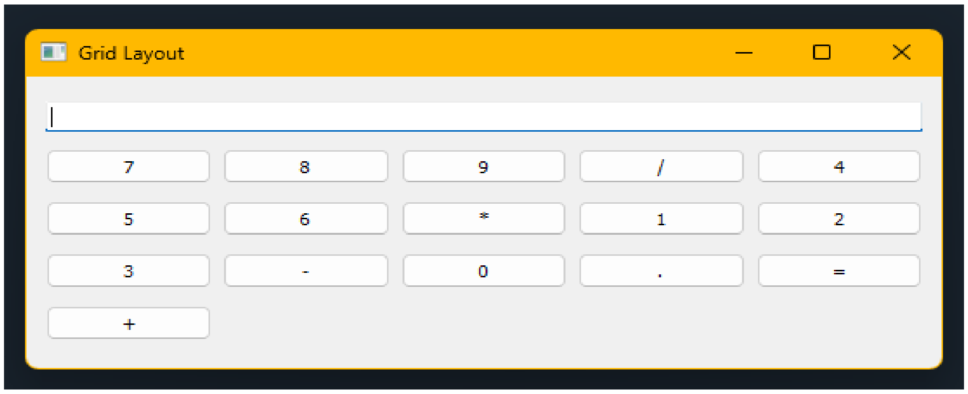


Observation: The position of the different components of the program are all aligned to one another.

Grid Layout using Loops

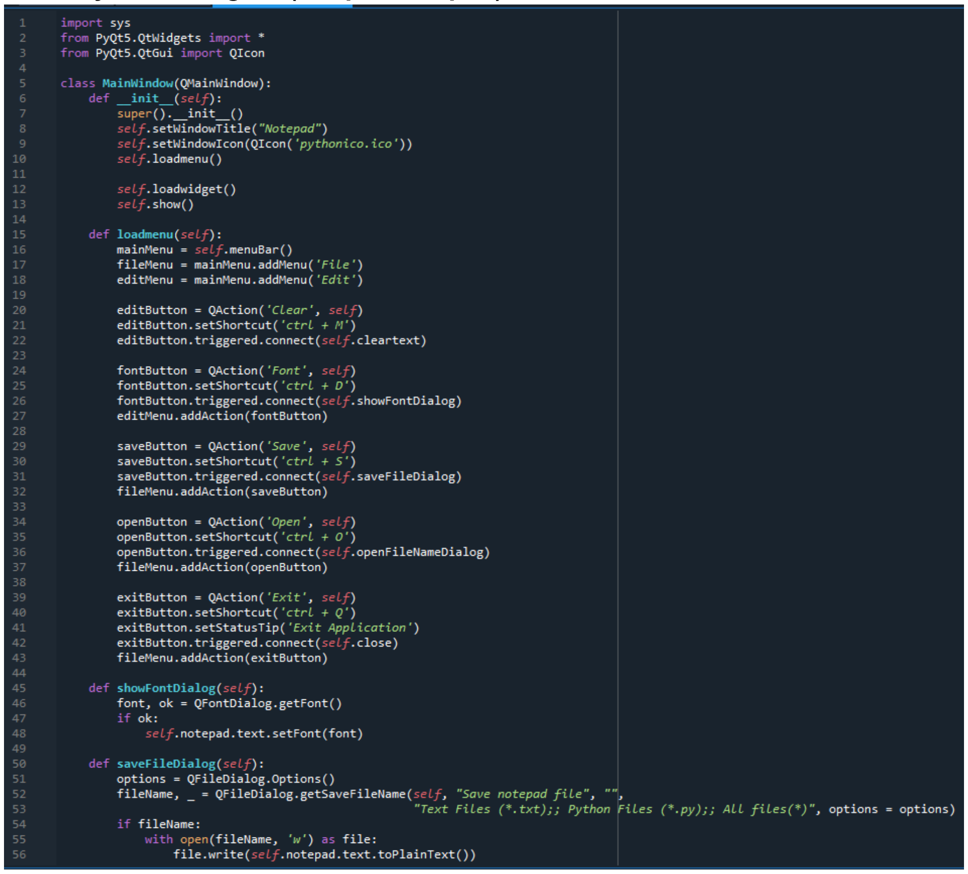


```python
# Grid Layout
import sys
from PyQt5.QtWidgets import QGridLayout, QLineEdit, QPushButton, QHBoxLayout,

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)

        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_())
```

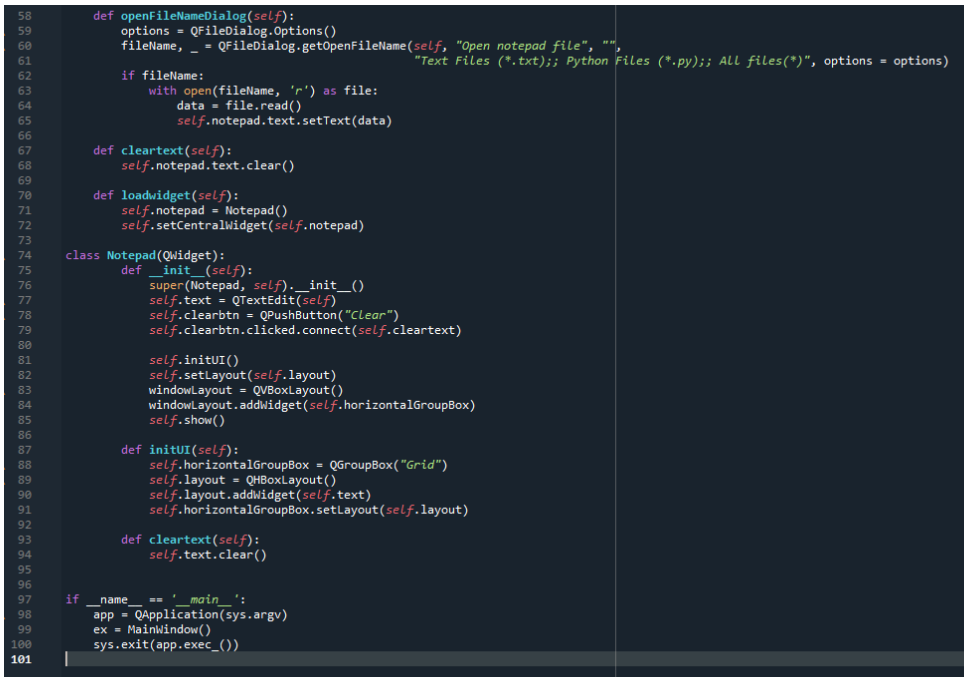


Observation: In the image of the output that I have provided, it can be seen that as the program window is stretched out, the different elements are also stretched, which stay proportionally to it.

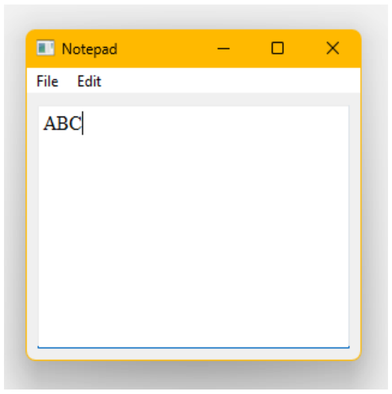
Vbox and Hbox layout managers (Simple Notepad)

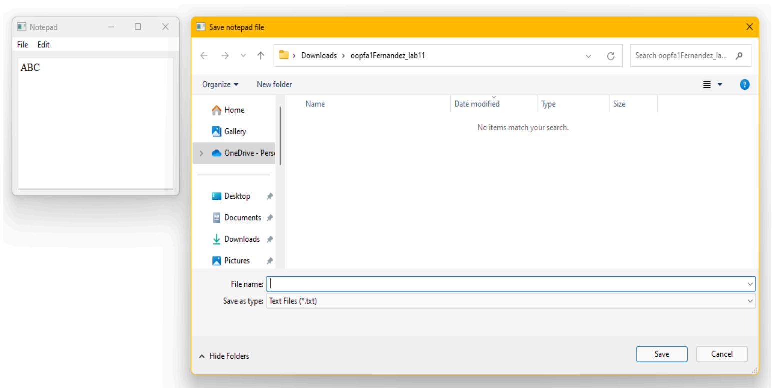


```python
import sys
from PyQt5.QtWidgets import *
from PyQt5.QtGui import QIcon

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)

        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 + O')
        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()
        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())
```



```python
    def openFileNameDialog(self):
        options = QFileDialog.Options()
        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.show()

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

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


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

```

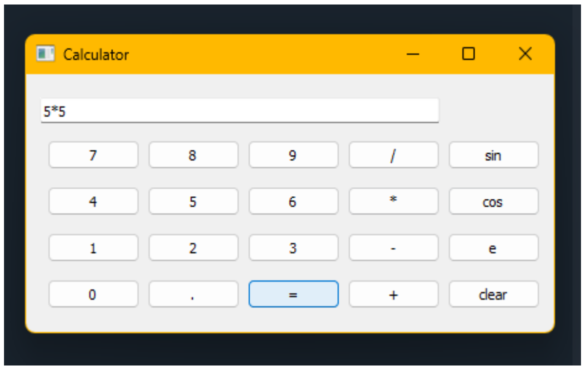




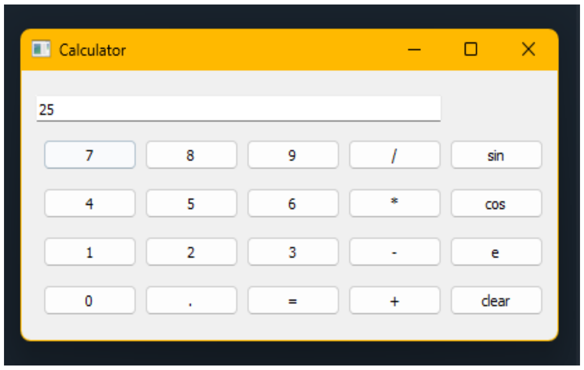
Observation: The notepad program does function similar to the default program of the computer. It can also be saved to the drive of the computer.

Supplementary Activity:

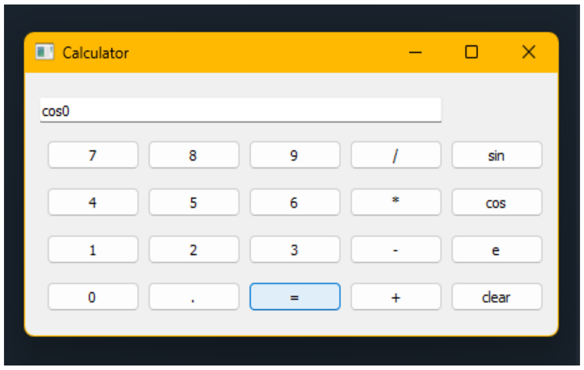
```
import sys
import math
from PyQt5.QtWidgets import (
    QGridLayout, QLineEdit, QPushButton, QWidget, QApplication, QMessageBox
)

class Calculator(QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        grid = QGridLayout()
        self.setLayout(grid)
        names = [
            '7', '8', '9', '/', 'sin',
            '4', '5', '6', '*', 'cos',
            '1', '2', '3', '-', 'e',
            '0', '.', '=', '+', 'clear']

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

        positions = [(i, j) for i in range(1, 6) for j in range(1, 6)]
        for position, name in zip(positions, names):
            button = QPushButton(name)
            button.clicked.connect(self.Button)
            grid.addWidget(button, *position)

        self.setGeometry(300, 300, 300, 200)
        self.setWindowTitle('Calculator')
        self.show()

    def Button(self):
        sender = self.sender()
        b_text = sender.text()

        if b_text == 'clear':
            self.textLine.clear()
        elif b_text == '=':
            self.Result()
        else:
            c_text = self.textLine.text()
            self.textLine.setText(c_text + b_text)

    def Result(self):
        try:
            expression = self.textLine.text()
            if 'sin' in expression:
                angle = self.getAngle(expression, 'sin')
                if angle is not None:
                    result = math.sin(math.radians(angle))
                    self.textLine.setText(str(result))
                    return

            if 'cos' in expression:
                angle = self.getAngle(expression, 'cos')
                if angle is not None:
                    result = math.cos(math.radians(angle))
                    self.textLine.setText(str(result))
                    return

            expression = expression.replace('e', str(math.e))
            result = eval(expression)
            self.textLine.setText(str(result))
        except Exception as e:
            QMessageBox.warning(self, "Error", f"Invalid input: {e}")

    def getAngle(self, expression, function):
        try:
            start = expression.index(function) + len(function)
            Angle = ''
            while start < len(expression) and (expression[start].isdigit() or expression[start] == '.'):
                Angle += expression[start]
                start += 1

            if Angle:
                return float(Angle)
            else:
                return None
        except ValueError:
            return None

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



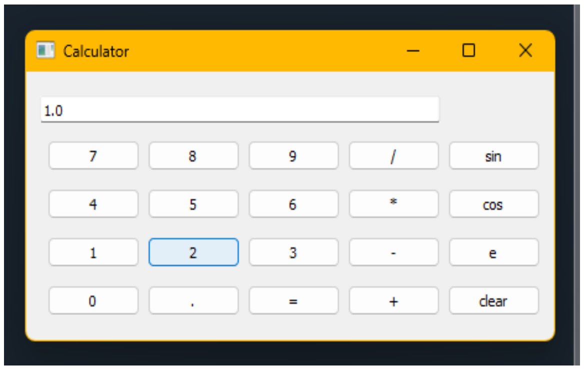
5*5



25



cos0



1.0

Conclusion:
To conclude, this laboratory activity taught me how to align various elements with the utilization of QGridLayout (from the PyQt5 library) and for loops. Also, I learned to create functional elements for a simple notepad, such as the “file” and “edit” section. In the supplementary activity, the calculator combined all the concepts from the previous exercises, aligning and laying out responsive elements to create a fully functional application similar to a standard calculator on a computer or mobile phone. To decide the order of elements from left to right on the interface of the application was a challenge for me.