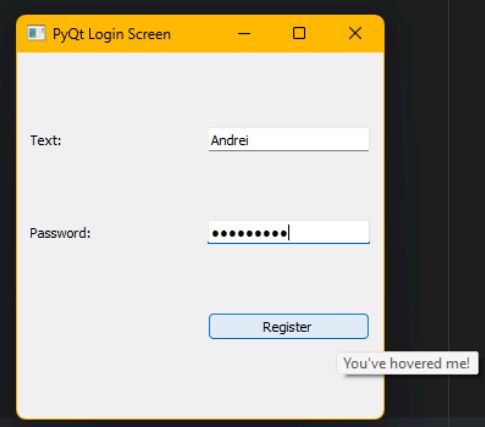
Activity 6	
Laboratory Activity 6 - GUI Design: Layout and Styling	
Course Code: CPE009B	Program: Computer Engineering
Course Title: Object Oriented Programming 2	Date Performed: 10/28/24
Section: CPE21S4	Date Submitted: 10/30/24
Name(s): Santos, Andrei R.	Instructor: Professor Maria Rizette Sayo

5. Procedure

Basic Grid Layout

gui_grid1.py × gui_grid2.py gui_simplenotepad.py

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


1 usage
class App(QWidget):

    def __init__(self):
        super().__init__()
        self.title = "PyQt Login Screen"
        self.x = 200 # or left
        self.y = 200 # or top
        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("pythonico.ico"))

        self.createGridLayout()
        self.setLayout(self.layout)
```

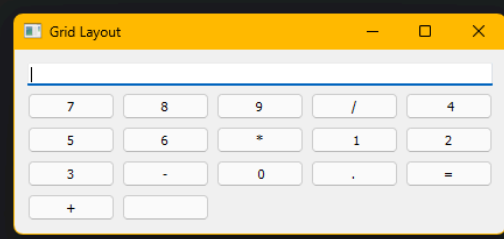


Grid Layout using Loops

 # if the (check the range (1, 6), *given code.*
positions = [(i, j) for i in range(1,7) for j in **range(1, 6)**]

gui_grid1.py gui_grid2.py gui_simplenotepad.py

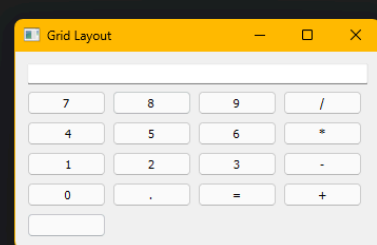
```python
#Grid Layout
import sys
from PyQt5.QtWidgets import QGridLayout, QLineEdit, QPushButton, \
    QHBoxLayout, QVBoxLayout, QWidget, QApplication
from PyQt5.QtGui import QIcon
from PyQt5.QtCore import pyqtSlot

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, 1, 1, 5)
```



 # if the (check the range (1,5),
positions = [(i, j) for i in range(1,7) for j in **range(1, 5)**]

gui_grid1.py gui_grid2.py gui_simplenotepad.py

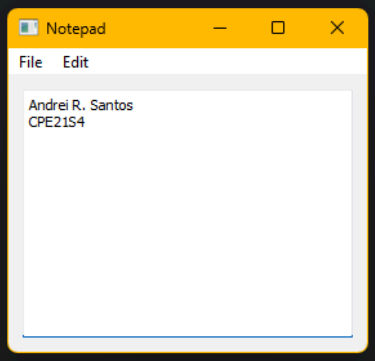
```python
#Grid Layout
import sys
from PyQt5.QtWidgets import QGridLayout, QLineEdit, QPushButton, \
    QHBoxLayout, QVBoxLayout, QWidget, QApplication
from PyQt5.QtGui import QIcon
from PyQt5.QtCore import pyqtSlot

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, 1, 1, 5)
```



Vbox and Hbox layout managers (Simple Notepad)

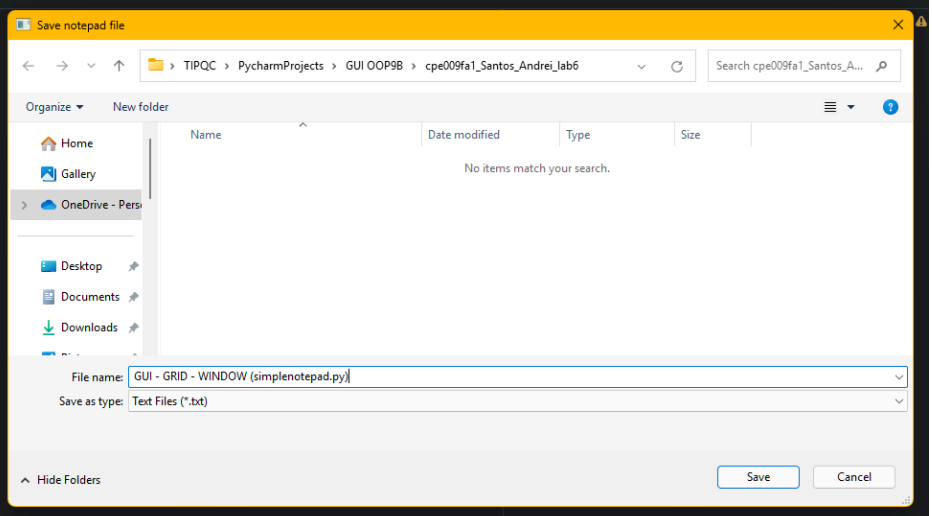
gui_grid1.py gui_grid2.py gui_simplenotepad.py

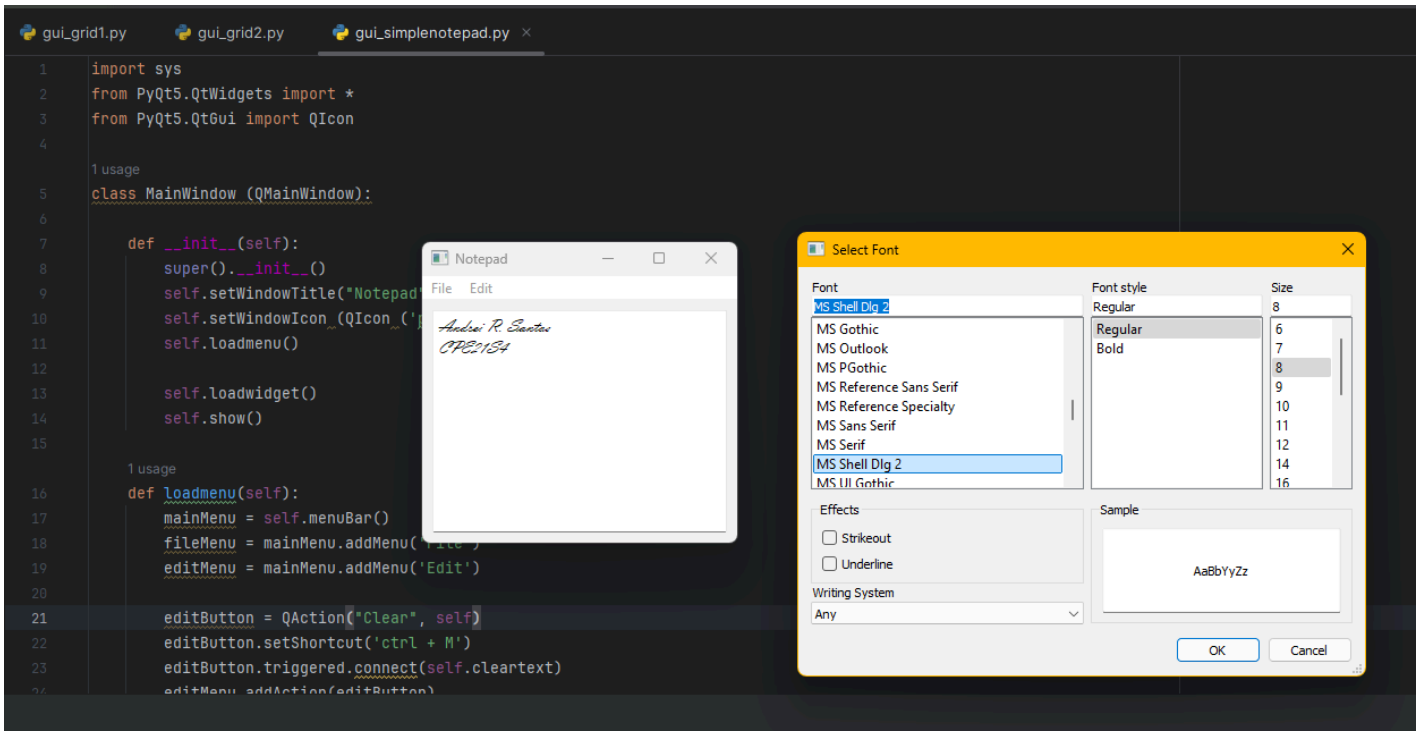
```python
import sys
from PyQt5.QtWidgets import *
from PyQt5.QtGui import QIcon

1 usage
class MainWindow (QMainWindow):

    def __init__(self):
        super().__init__()
        self.setWindowTitle("Notepad")
        self.setWindowIcon (QIcon ('pythonico.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)
```



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 ('pythonico.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)
```





 # The program, after being interpreted, can be saved, edited, and can exit the program.
 # The program, after being interpreted, can clear the content of the file and also edit the font.

6. Supplementary Activity

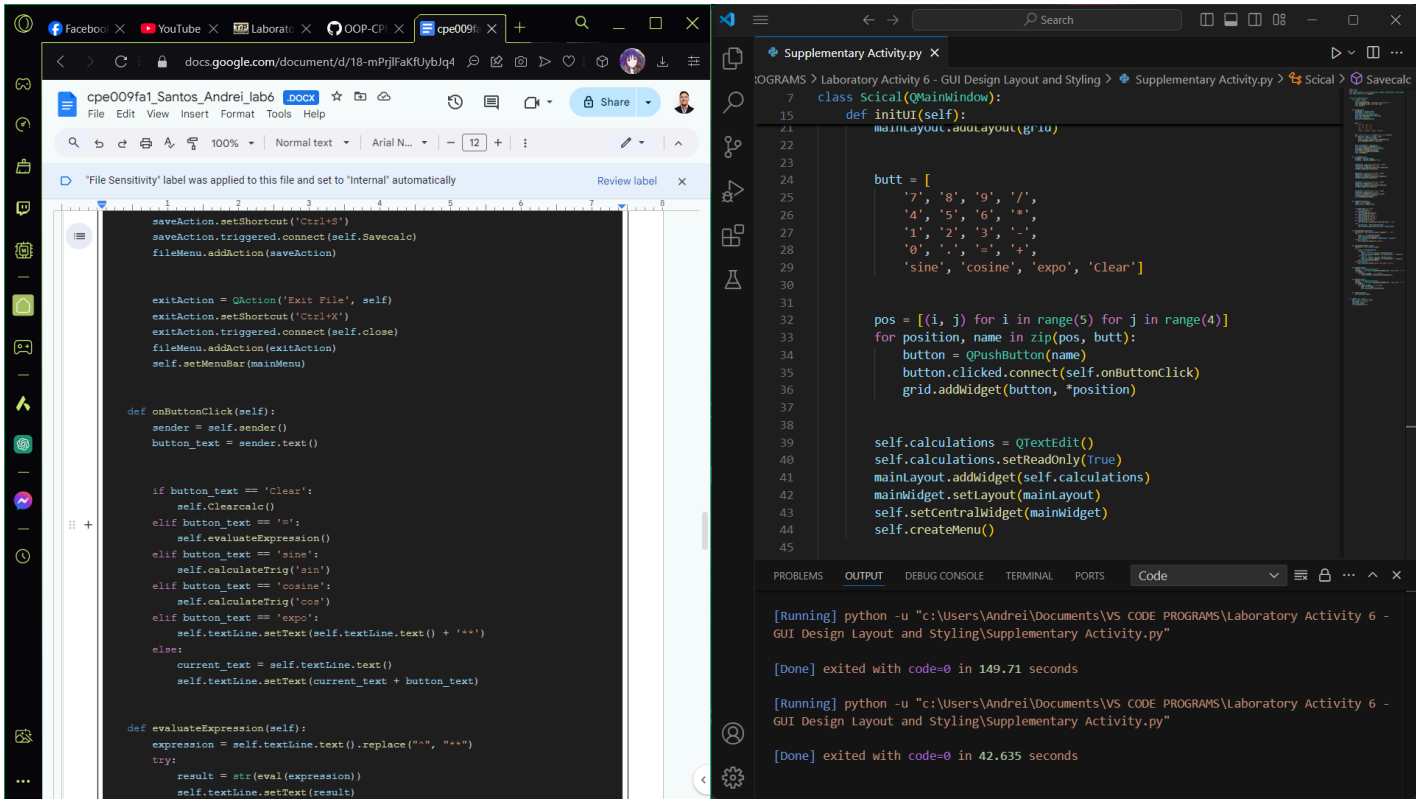
```
import sys
import math
from PyQt5.QtWidgets import QMainWindow, QWidget, QVBoxLayout, QGridLayout,
QLineEdit,QPushButton, QAction, QFileDialog, QApplication, QMenuBar, QTextEdit
from PyQt5.QtCore import pyqtSlot


class Scical(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('~ Scientific Calculator ~')
        self.setGeometry(300, 300, 400, 400)
        self.initUI()


    def initUI(self):
        mainWidget = QWidget(self)
        mainLayout = QVBoxLayout()
        self.textLine = QLineEdit(self)
        mainLayout.addWidget(self.textLine)
```

```
        grid = QGridLayout()
        mainLayout.addLayout(grid)


        butt = [
            '7', '8', '9', '/',
            '4', '5', '6', '*',
            '1', '2', '3', '-',
            '0', '.', '=', '+',
            'sine', 'cosine', 'expo', 'Clear']


        pos = [(i, j) for i in range(5) for j in range(4)]
        for position, name in zip(pos, butt):
            button = QPushButton(name)
            button.clicked.connect(self.onButtonClick)
            grid.addWidget(button, *position)


        self.calculations = QTextEdit()
        self.calculations.setReadOnly(True)
        mainLayout.addWidget(self.calculations)
        mainWidget.setLayout(mainLayout)
        self.setCentralWidget(mainWidget)
        self.createMenu()


    def createMenu(self):
        mainMenu = QMenuBar(self)
        fileMenu = mainMenu.addMenu('File')


        clearAction = QAction('Clear File', self)
        clearAction.setShortcut('Ctrl+C')
        clearAction.triggered.connect(self.Clearcalc)
        fileMenu.addAction(clearAction)


        openAction = QAction('Load File', self)
        openAction.setShortcut('Ctrl+O')
        openAction.triggered.connect(self.Opencalc)
        fileMenu.addAction(openAction)
```

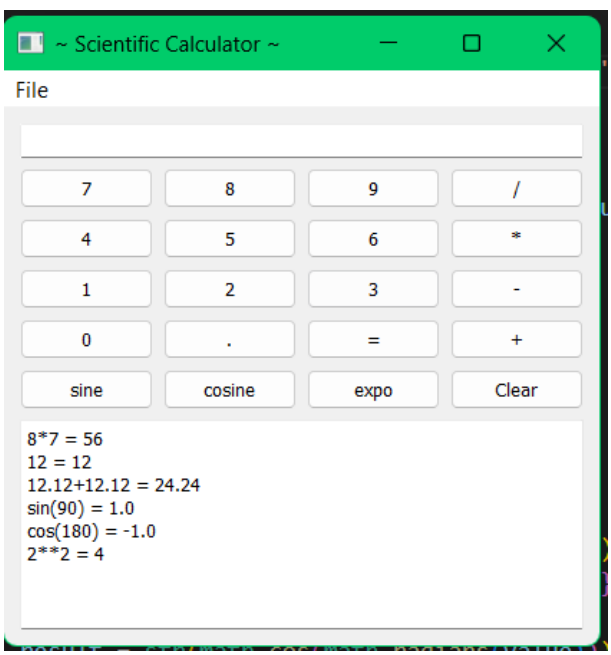
```
        saveAction = QAction('Save File', self)
        saveAction.setShortcut('Ctrl+S')
        saveAction.triggered.connect(self.Savecalc)
        fileMenu.addAction(saveAction)


        exitAction = QAction('Exit File', self)
        exitAction.setShortcut('Ctrl+X')
        exitAction.triggered.connect(self.close)
        fileMenu.addAction(exitAction)
        self.setMenuBar(mainMenu)


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


        if button_text == 'Clear':
            self.Clearcalc()
        elif button_text == '=':
            self.evaluateExpression()
        elif button_text == 'sine':
            self.calculateTrig('sin')
        elif button_text == 'cosine':
            self.calculateTrig('cos')
        elif button_text == 'expo':
            self.textLine.setText(self.textLine.text() + '**')
        else:
            current_text = self.textLine.text()
            self.textLine.setText(current_text + button_text)


    def evaluateExpression(self):
        expression = self.textLine.text().replace("^", "**")
        try:
            result = str(eval(expression))
            self.textLine.setText(result)
            self.calculations.append(f"{expression} = {result}")
        except Exception:
            self.textLine.setText("No input!")
```

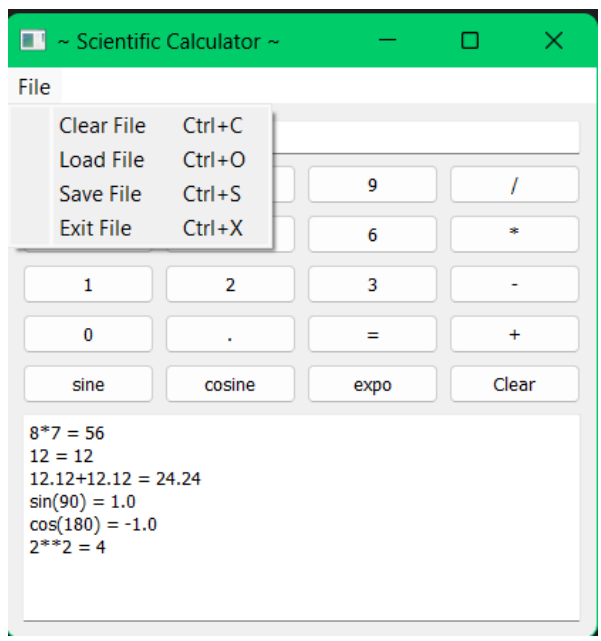
```
    def calculateTrig(self, func):
        expression = self.textLine.text()
        try:
            value = float(expression)
            if func == 'sin':
                result = str(math.sin(math.radians(value)))
                self.calculations.append(f"sin({expression}) = {result}")
            elif func == 'cos':
                result = str(math.cos(math.radians(value)))
                self.calculations.append(f"cos({expression}) = {result}")
            self.textLine.setText(result)
        except ValueError:
            self.textLine.setText("Enter the number first!")


    def Savecalc(self):
        options = QFileDialog.Options()
        fileName, _ = QFileDialog.getSaveFileName(self, "Save File", "", "Text Files
(*.txt)", options=options)
        if fileName:
            with open(fileName, 'w') as file:
                file.write(self.calculations.toPlainText())


    def Opencalc(self):
        options = QFileDialog.Options()
        fileName, _ = QFileDialog.getOpenFileName(self, "Load File", "", "Text Files
(*.txt)", options=options)
        if fileName:
            with open(fileName, 'r') as file:
                data = file.read()
                self.calculations.setText(data)


    def Clearcalc(self):
        self.textLine.clear()


if __name__ == '__main__':
    app = QApplication(sys.argv)
    calculator = Scical()
    calculator.show()
```

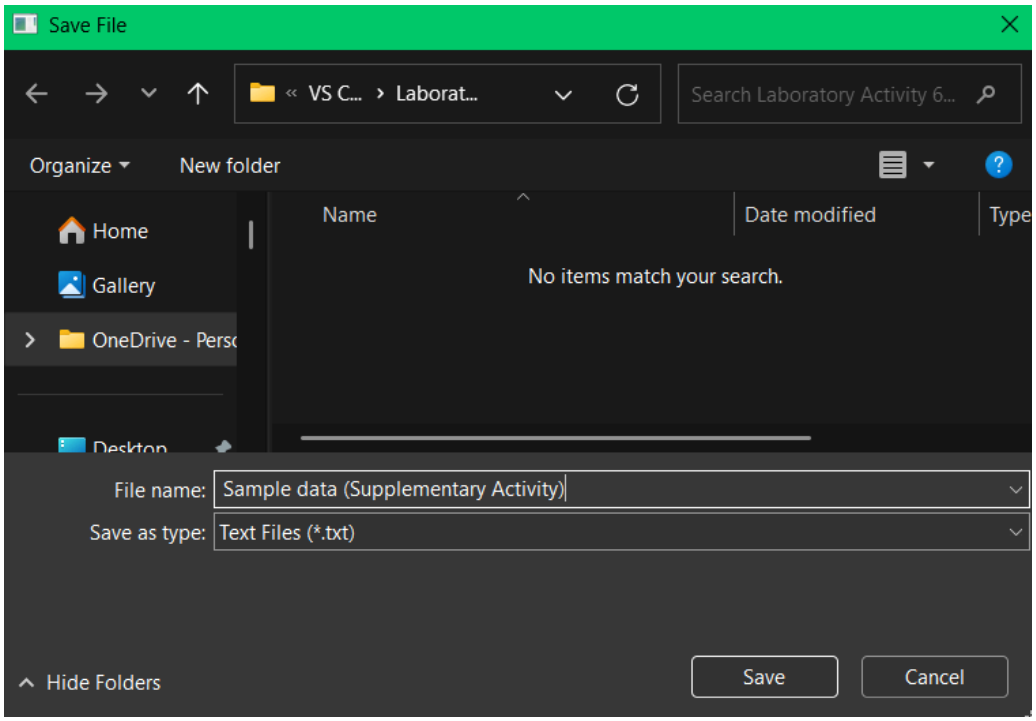
```
    sys.exit(app.exec_())
```

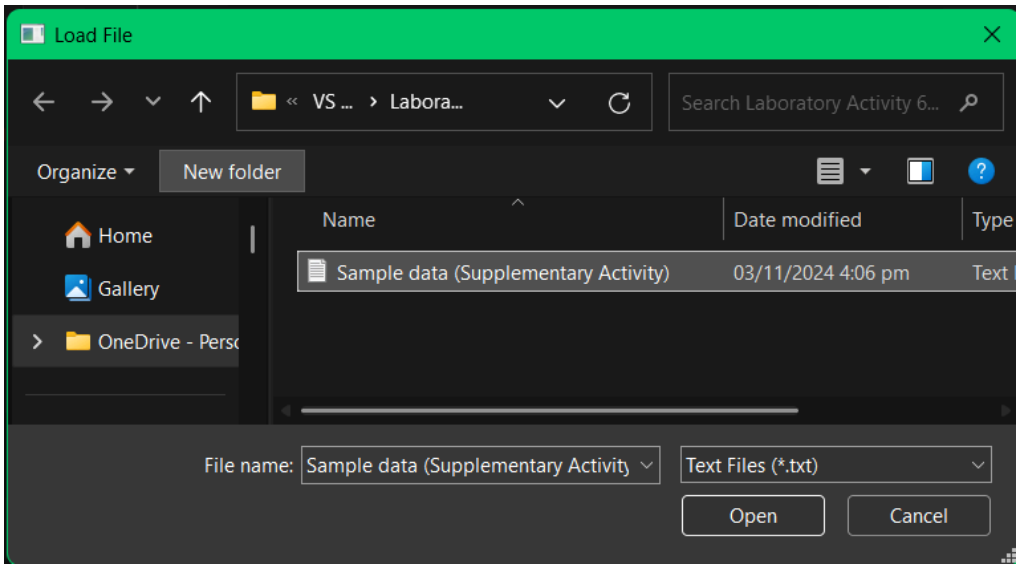


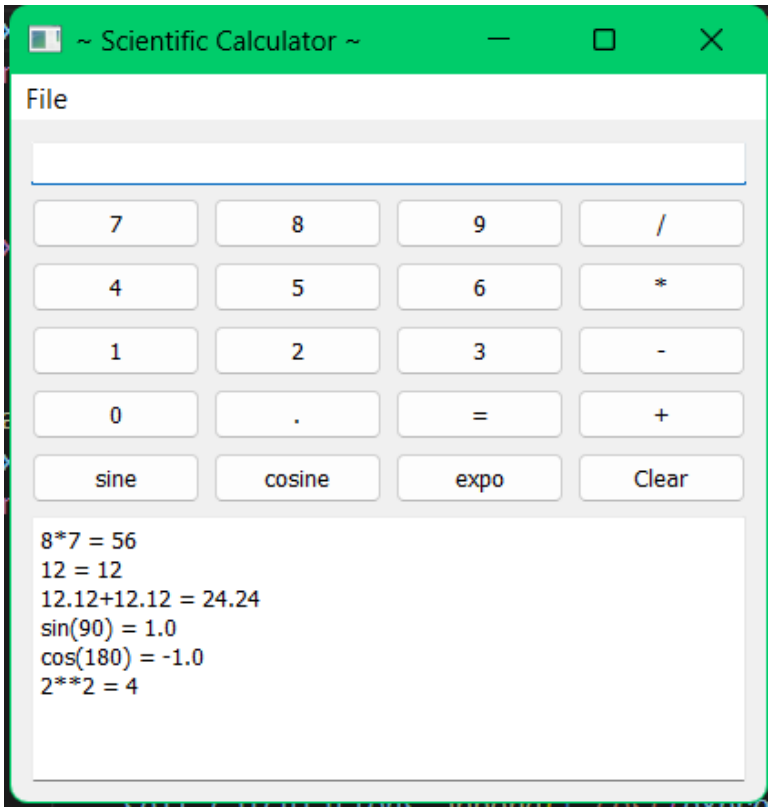
OUTPUT :






Save File
« VS C... › Laborat...
Search Laboratory Activity 6...
Organize
New folder
Home
Gallery
OneDrive - Perso
Desktop
Name
Date modified
Type
No items match your search.
File name:
Sample data (Supplementary Activity)
Save as type:
Text Files (*.txt)
Hide Folders
Save
Cancel


Load File
« VS ... › Labora...
Search Laboratory Activity 6...
Organize
New folder
Home
Gallery
OneDrive - Perso
Name
Date modified
Type
Sample data (Supplementary Activity)
03/11/2024 4:06 pm
Text
File name:
Sample data (Supplementary Activity
Text Files (*.txt)
Open
Cancel



 # Executable code that works like a calculator with added trigonometric functions.

7. Conclusion

In this activity, I learned how to implement various layouts and styling in a PyQt5 GUI application, including creating grid layouts, handling events, and integrating a menu bar. I successfully built functional applications like a calculator and a notepad, which reinforced my understanding of these concepts. Overall, I did well, but there's room for improvement in error handling and code optimization. I'll keep practicing to enhance my skills further when it comes to this kind of environment of learning.

8. Assessment Rubric