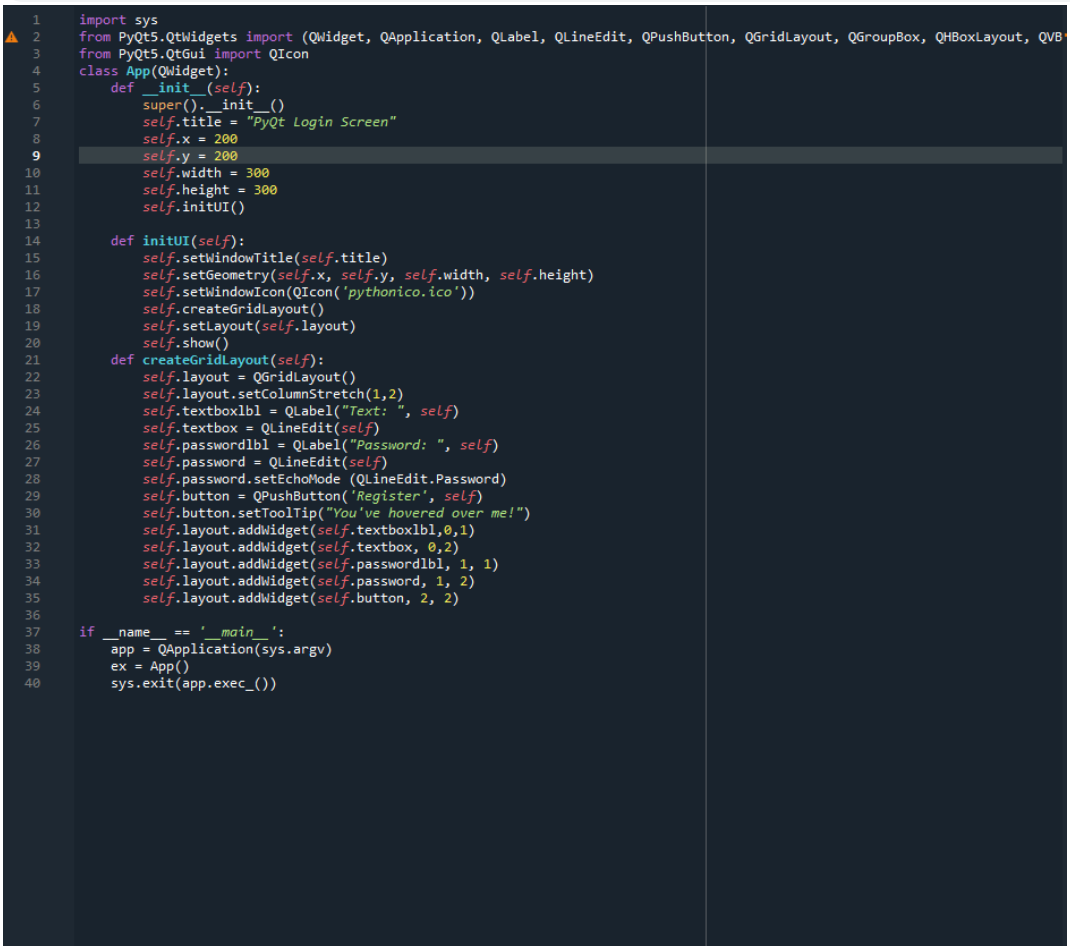
Activity Name #6 - CPE 009 Lab Activity 6 - GUI Design_ Layout and Styling	
Solis, Paul Vincent M.	10/28/24
CPE009 - CPE21S4	Prof Sayo

GUIgrid1.py

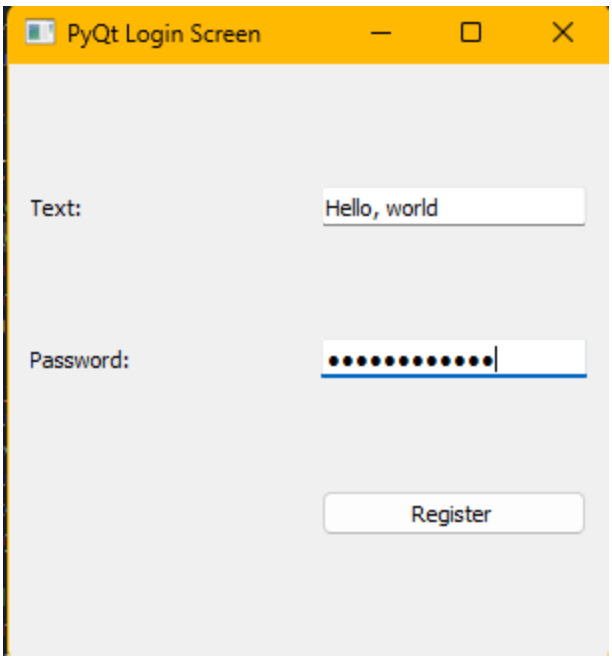
Python

```python
import sys
from PyQt5.QtWidgets import (QWidget, QApplication, QLabel, QLineEdit,
QPushButton, QGridLayout, QGroupBox, QHBoxLayout, QVBoxLayout,QTextEdit)
from PyQt5.QtGui import QIcon
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.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_())
```



```
import sys
from PyQt5.QtWidgets import (QWidget, QApplication, QLabel, QLineEdit, QPushButton, QGridLayout, QGroupBox, QHBoxLayout, QVB
from PyQt5.QtGui import QIcon
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.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_())
```



GUIGRID2.PY

Python

```
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)



        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_())
```

```python
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)



        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_())
```

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()
```

Python

```
    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)
```

Python

```
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)
```

Python

```
 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())
    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)
```

Python

```
  def cleartext(self):
        self.notepad.text.clear()
    def loadwidget(self):
        self.notepad = Notepad()
        self.setCentralWidget(self.notepad)
```

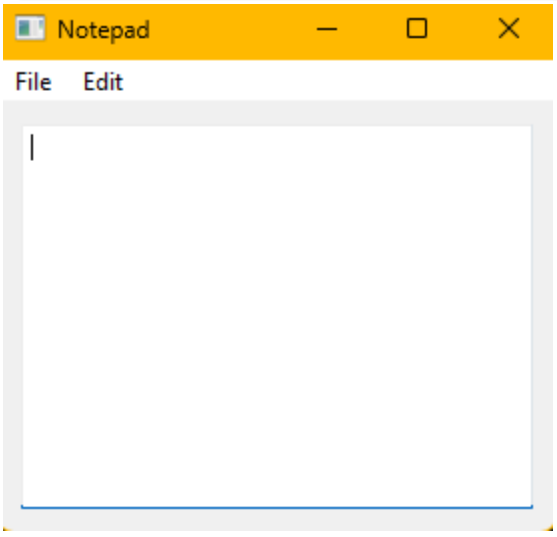
Python

```
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()
```

Python

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

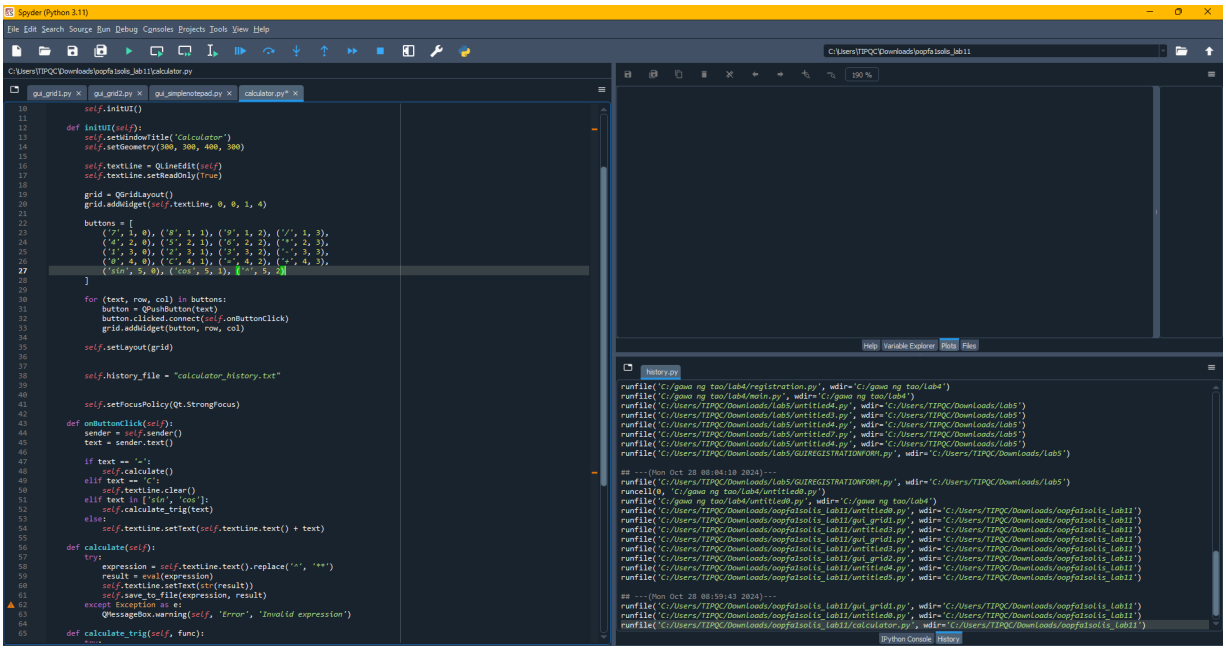


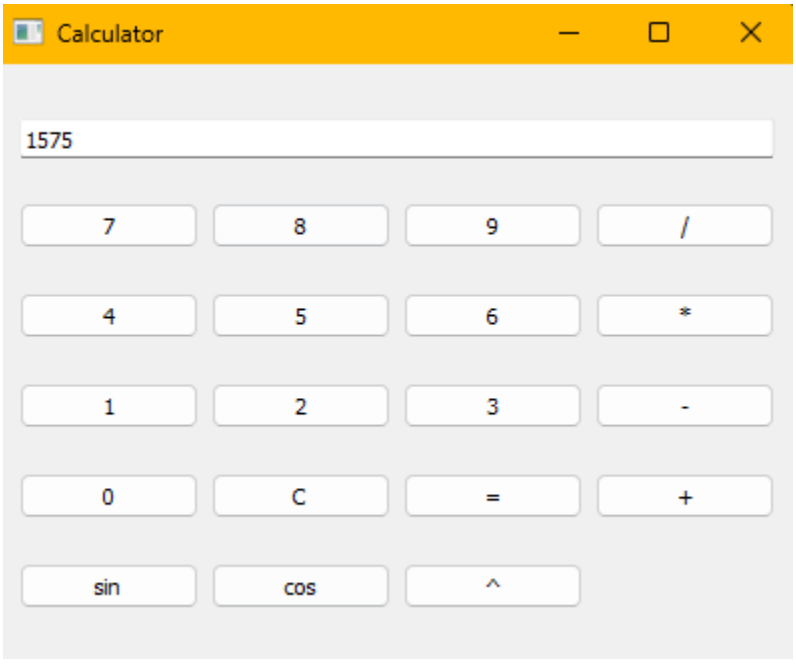
SUPPLEMENTARY ACTIVITY

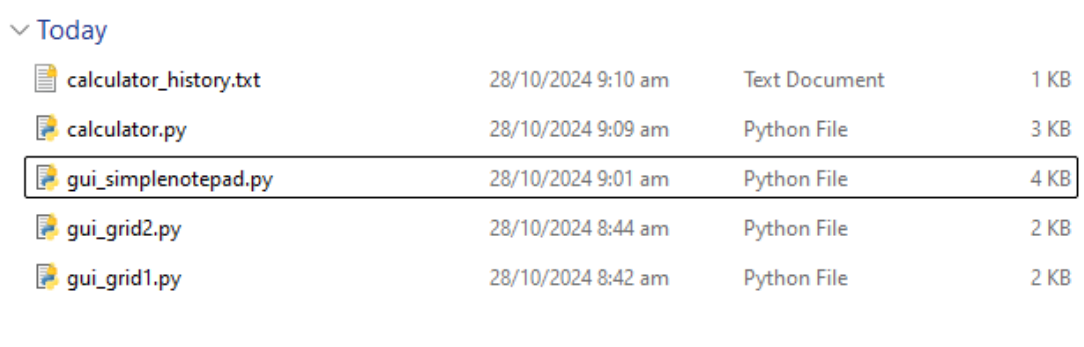
Python

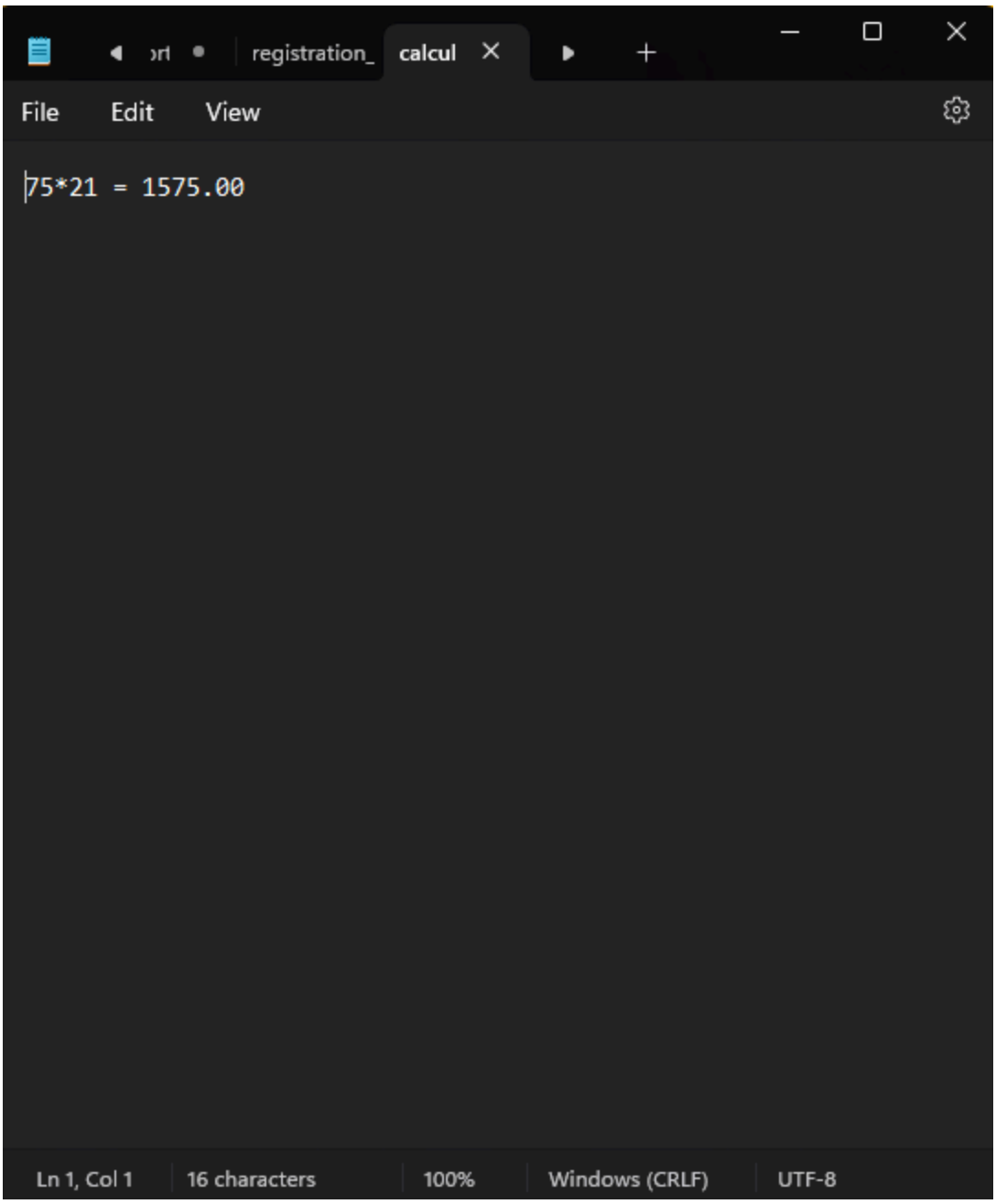
```python
import sys
import math
from PyQt5.QtWidgets import (QGridLayout, QLineEdit, QPushButton,
                             QVBoxLayout, QWidget, QApplication, QMessageBox)
from PyQt5.QtCore import Qt

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

    def initUI(self):
        self.setWindowTitle('Calculator')
        self.setGeometry(300, 300, 400, 300)

        self.textLine = QLineEdit(self)
        self.textLine.setReadOnly(True)

        grid = QGridLayout()
        grid.addWidget(self.textLine, 0, 0, 1, 4)

        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), ('C', 4, 1), ('=', 4, 2), ('+', 4, 3),
            ('sin', 5, 0), ('cos', 5, 1), ('^', 5, 2)
        ]

        for (text, row, col) in buttons:
            button = QPushButton(text)
            button.clicked.connect(self.onButtonClick)
            grid.addWidget(button, row, col)

        self.setLayout(grid)


        self.history_file = "calculator_history.txt"


        self.setFocusPolicy(Qt.StrongFocus)

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

        if text == '=':
            self.calculate()
```

```
        elif text == 'C':
            self.textLine.clear()
        elif text in ['sin', 'cos']:
            self.calculate_trig(text)
        else:
            self.textLine.setText(self.textLine.text() + text)

    def calculate(self):
        try:
            expression = self.textLine.text().replace('^', '**')
            result = eval(expression)
            self.textLine.setText(str(result))
            self.save_to_file(expression, result)
        except Exception as e:
            QMessageBox.warning(self, 'Error', 'Invalid expression')

    def calculate_trig(self, func):
        try:
            angle = float(self.textLine.text())
            if func == 'sin':
                result = math.sin(math.radians(angle))
            elif func == 'cos':
                result = math.cos(math.radians(angle))
            self.textLine.setText(str(result))
            self.save_to_file(f"{func}({angle})", result)
        except ValueError:
            QMessageBox.warning(self, 'Error', 'Invalid input for
trigonometric function')

    def save_to_file(self, operation, result):
        with open(self.history_file, 'a') as file:
            file.write(f"{operation} = {result:.2f}\n")

    def keyPressEvent(self, event):
        if event.key() == Qt.Key_Q and event.modifiers() ==
Qt.ControlModifier:
            self.close()

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


Spyder (Python 3.11)
File Edit Search Source Run Debug Consoles Projects Tools View Help
gui_grid1.py
gui_grid2.py
gui_simplenotepad.py
calculator.py*
self.initUI()
def initUI(self):
self.setWindowTitle('Calculator')
self.setGeometry(300, 300, 400, 300)
self.textLine = QLineEdit(self)
self.textLine.setReadOnly(True)
grid = QGridLayout()
grid.addWidget(self.textLine, 0, 0, 1, 4)
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), ('C', 4, 1), ('=', 4, 2), ('+', 4, 3),
('sin', 5, 0), ('cos', 5, 1), ('^', 5, 2)
]
for (text, row, col) in buttons:
button = QPushButton(text)
button.clicked.connect(self.onButtonClick)
grid.addWidget(button, row, col)
self.setLayout(grid)
self.history_file = "calculator_history.txt"
self.setFocusPolicy(Qt.StrongFocus)
def onButtonClick(self):
sender = self.sender()
text = sender.text()
if text == "=":
self.calculate()
elif text == 'C':
self.textLine.clear()
elif text in ['sin', 'cos']:
self.calculate_trig(text)
else:
self.textLine.setText(self.textLine.text() + text)
def calculate(self):
try:
expression = self.textLine.text().replace('^', '**')
result = eval(expression)
self.textLine.setText(str(result))
self.save_to_file(expression, result)
except Exception as e:
QMessageBox.warning(self, 'Error', 'Invalid expression')
def calculate_trig(self, func):
Help Variable Explorer Plots Files
history.py
runfile('C:/gawa ng tao/lab4/registration.py', wdir='C:/gawa ng tao/lab4')
runfile('C:/gawa ng tao/lab4/main.py', wdir='C:/gawa ng tao/lab4')
runfile('C:/Users/TIPQC/Downloads/lab5/untitled4.py', wdir='C:/Users/TIPQC/Downloads/lab5')
runfile('C:/Users/TIPQC/Downloads/lab5/untitled3.py', wdir='C:/Users/TIPQC/Downloads/lab5')
runfile('C:/Users/TIPQC/Downloads/lab5/untitled4.py', wdir='C:/Users/TIPQC/Downloads/lab5')
runfile('C:/Users/TIPQC/Downloads/lab5/untitled7.py', wdir='C:/Users/TIPQC/Downloads/lab5')
runfile('C:/Users/TIPQC/Downloads/lab5/untitled4.py', wdir='C:/Users/TIPQC/Downloads/lab5')
runfile('C:/Users/TIPQC/Downloads/lab5/GUIREGISTRATIONFORM.py', wdir='C:/Users/TIPQC/Downloads/lab5')
---(Mon Oct 28 08:04:10 2024)---
runfile('C:/Users/TIPQC/Downloads/lab5/GUIREGISTRATIONFORM.py', wdir='C:/Users/TIPQC/Downloads/lab5')
runcell(0, 'C:/gawa ng tao/lab4/untitled0.py')
runfile('C:/gawa ng tao/lab4/untitled0.py', wdir='C:/gawa ng tao/lab4')
runfile('C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11/untitled0.py', wdir='C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11')
runfile('C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11/gui_grid1.py', wdir='C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11')
runfile('C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11/untitled3.py', wdir='C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11')
runfile('C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11/gui_grid1.py', wdir='C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11')
runfile('C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11/untitled3.py', wdir='C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11')
runfile('C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11/gui_grid2.py', wdir='C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11')
runfile('C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11/untitled4.py', wdir='C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11')
runfile('C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11/untitled5.py', wdir='C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11')
---(Mon Oct 28 08:59:43 2024)---
runfile('C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11/gui_grid1.py', wdir='C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11')
runfile('C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11/untitled0.py', wdir='C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11')
runfile('C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11/calculator.py', wdir='C:/Users/TIPQC/Downloads/oopfa1soli1s_lab11')
IPython Console History


Calculator
1575
7
8
9
/
4
5
6
*
1
2
3
-
0
C
=
+
sin
cos
^


Today
calculator_history.txt 28/10/2024 9:10 am Text Document 1 KB
calculator.py 28/10/2024 9:09 am Python File 3 KB
gui_simplenotepad.py 28/10/2024 9:01 am Python File 4 KB
gui_grid2.py 28/10/2024 8:44 am Python File 2 KB
gui_grid1.py 28/10/2024 8:42 am Python File 2 KB



CONCLUSION:
There are multiple PyQt5 programs included in this code, each with a distinct function. The first is a straightforward login screen with text fields for the password and username. The second is a simple calculator with trigonometric and arithmetic functions. The third is a notepad program that lets you make, save, and access text documents. Every application employs buttons for interaction and has an easy-to-use interface. Additionally, they gracefully handle mistakes by displaying notifications when something goes wrong. All things considered, these examples show how to use PyQt5 to create basic

GUI applications.