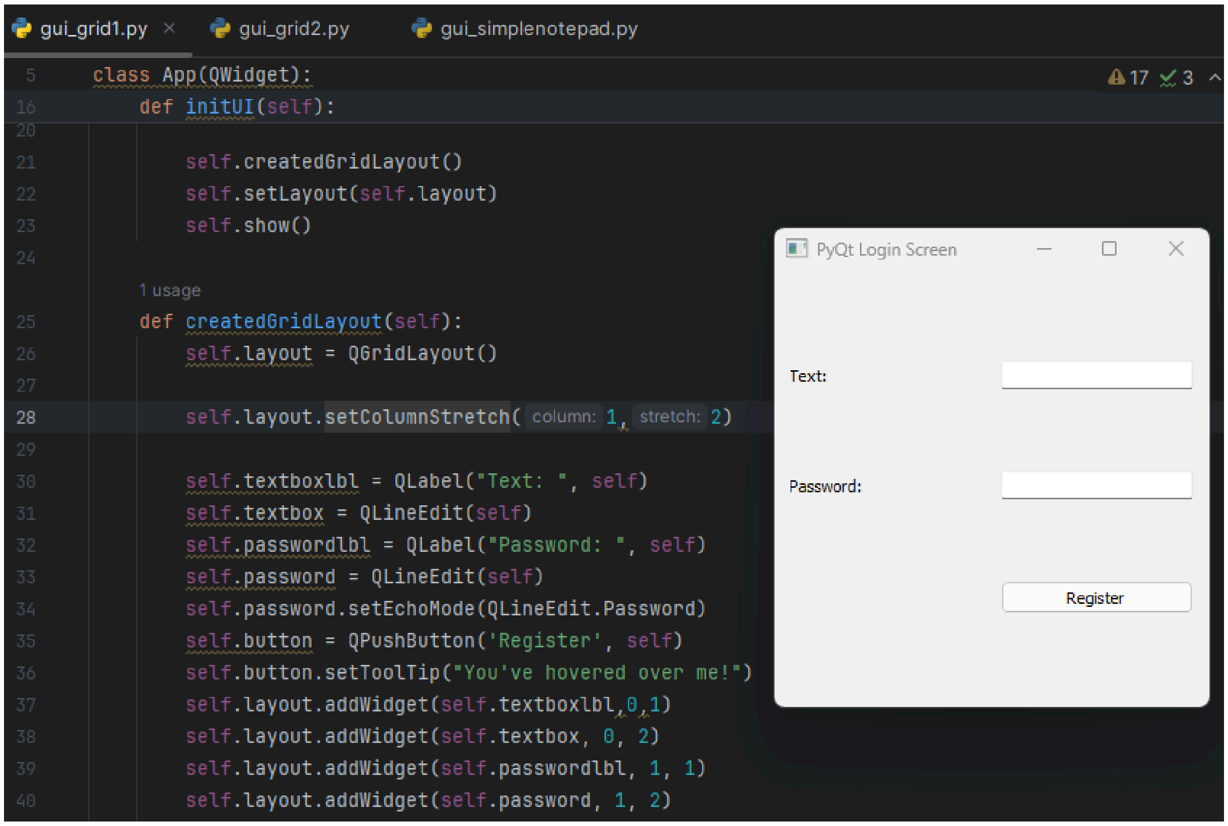
# Activity Name # 6 - GUI Design: Layout and Styling

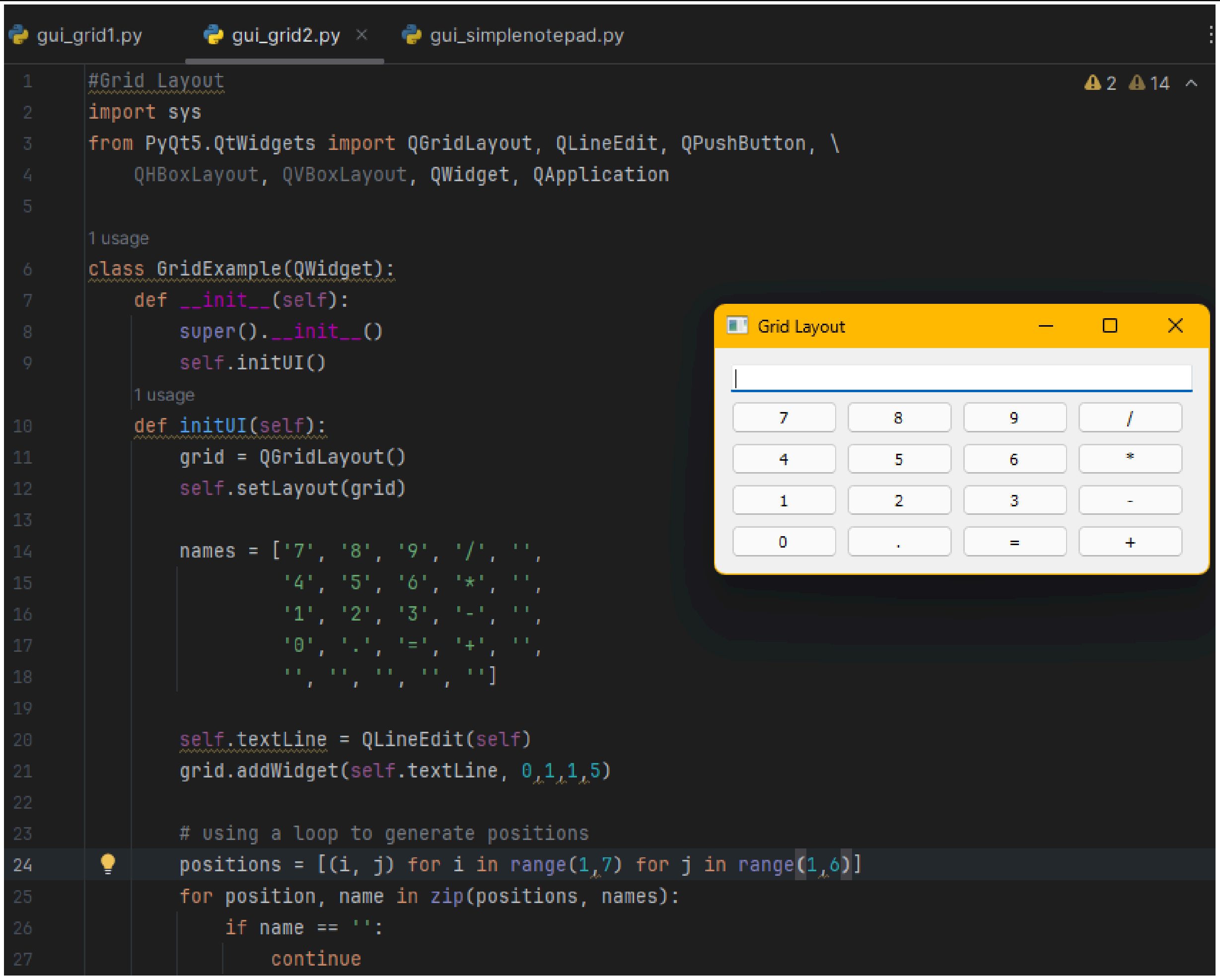
| | |
|---|---|
| Bona, Andrei Nycole So | 10/28/24 |
| CPE009B-CPE21S4 | Prof. Maria Rizette Sayo |

**gui_grid1.py**



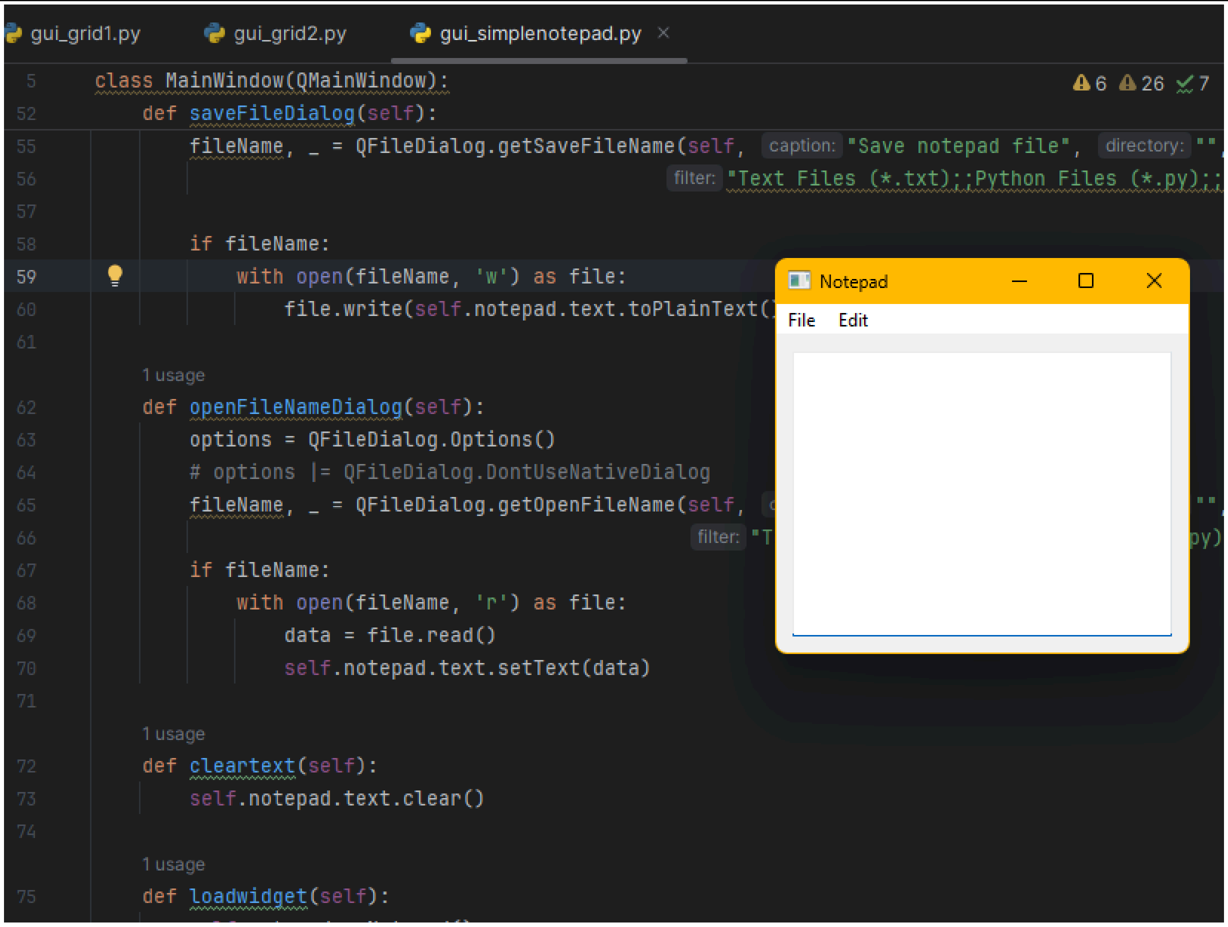
```python
class App(QWidget):
    def initUI(self):

        self.createdGridLayout()
        self.setLayout(self.layout)
        self.show()

    def createdGridLayout(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)
        self.layout.addWidget(self.password, 1, 2)
```

**gui_grid2.py**



```
#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
```

**gui_simplenotepad.py**



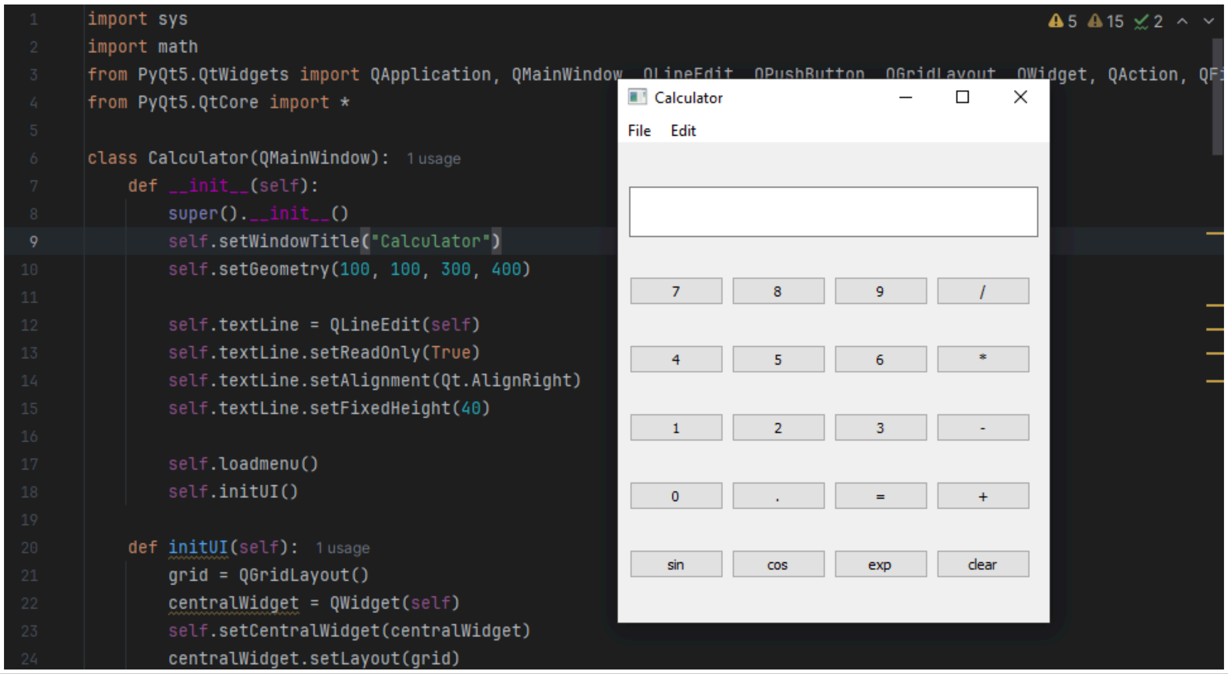
```
class MainWindow(QMainWindow):
    def saveFileDialog(self):
        fileName, _ = QFileDialog.getSaveFileName(self, caption: "Save notepad file", directory: "",
                                                  filter: "Text Files (*.txt);;Python Files (*.py);;

        if fileName:
            with open(fileName, 'w') as file:
                file.write(self.notepad.text.toPlainText(

    1 usage
    def openFileNameDialog(self):
        options = QFileDialog.Options()
        # options |= QFileDialog.DontUseNativeDialog
        fileName, _ = QFileDialog.getOpenFileName(self, c                                       "",
                                                    filter: "T                                  py)

        if fileName:
            with open(fileName, 'r') as file:
                data = file.read()
                self.notepad.text.setText(data)

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

    1 usage
    def loadwidget(self):
```

## 6. Supplementary Activity:

```
import sys
import math
from PyQt5.QtWidgets import QApplication, QMainWindow, QLineEdit,
QPushButton, QGridLayout, QWidget, QAction, QFileDialog, QMessageBox
from PyQt5.QtCore import *

class Calculator(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Calculator")
        self.setGeometry(100, 100, 300, 400)

        self.textLine = QLineEdit(self)
        self.textLine.setReadOnly(True)
        self.textLine.setAlignment(Qt.AlignRight)
        self.textLine.setFixedHeight(40)

        self.loadmenu()
        self.initUI()

    def initUI(self):
        grid = QGridLayout()
        centralWidget = QWidget(self)
```

```python
        self.setCentralWidget(centralWidget)
        centralWidget.setLayout(grid)

        # Add the display area for the calculator
        grid.addWidget(self.textLine, 0, 0, 1, 5)

        # Button names and layout
        names = ['7', '8', '9', '/', '',
                 '4', '5', '6', '*', '',
                 '1', '2', '3', '-', '',
                 '0', '.', '=', '+', '',
                 'sin', 'cos', 'exp', 'clear', '']

        # Using a loop to generate positions and button connections
        positions = [(i, j) for i in range(1, 7) for j in range(5)]
        for position, name in zip(positions, names):
            if name == '':
                continue
            button = QPushButton(name)
            button.clicked.connect(self.create_button_handler(name))
            grid.addWidget(button, *position)

    def create_button_handler(self, char):
        def handler():
            self.on_button_click(char)
        return handler

    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)

        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 cleartext(self):
        self.textLine.clear()

    def on_button_click(self, char):
        if char == '=':
            self.calculate_result()
        elif char == 'clear':
            self.cleartext()
        else:
            current_text = self.textLine.text()
            self.textLine.setText(current_text + char)

    def calculate_result(self):
        try:
            expression = self.textLine.text()
            result = self.evaluate_expression(expression)
            self.textLine.setText(str(result))
        except Exception:
            self.show_error("Invalid input")

    def evaluate_expression(self, expression):
        if expression.startswith('sin'):
            value = float(expression[3:])
            return math.sin(math.radians(value))
        elif expression.startswith('cos'):
            value = float(expression[3:])
            return math.cos(math.radians(value))
        elif expression.startswith('exp'):
            value = float(expression[3:])
            return math.exp(value)
        else:
            return eval(expression)

    def show_error(self, message):
        QMessageBox.critical(self, "Error", message)

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

    def openFileNameDialog(self):
        options = QFileDialog.Options()
        fileName, _ = QFileDialog.getOpenFileName(self, "Open notepad file",
 "",
                                                   "Text Files (*.txt)",
 options=options)
        if fileName:
            with open(fileName, 'r') as file:
                data = file.read()
                self.textLine.setText(data)

 if __name__ == "__main__":
    app = QApplication(sys.argv)
    calc = Calculator()
    calc.show()
```

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



```
import sys
import math
from PyQt5.QtWidgets import QApplication, QMainWindow, QLineEdit, QPushButton, QGridLayout, QWidget, QAction, QF
from PyQt5.QtCore import *

class Calculator(QMainWindow):  1 usage
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Calculator")
        self.setGeometry(100, 100, 300, 400)

        self.textLine = QLineEdit(self)
        self.textLine.setReadOnly(True)
        self.textLine.setAlignment(Qt.AlignRight)
        self.textLine.setFixedHeight(40)

        self.loadmenu()
        self.initUI()

    def initUI(self):  1 usage
        grid = QGridLayout()
        centralWidget = QWidget(self)
        self.setCentralWidget(centralWidget)
        centralWidget.setLayout(grid)
```

## 7. Conclusion:

This laboratory showcases the concepts involved in the creation of interactive and usable PyQt applications. The codes being illustrated are all on the same line of GUI design. The first code contains a very basic layout containing labeled screen-in and password entry fields arranged neatly using the grid layout. The second code creates several interfaces of a calculator by implementing separate drawing functions that place the numerical and operator buttons in the grid. It is an example of a good form of layout management for number-pad interfaces. Next is the example of a simple notepad application in which the basic file operations of opening, saving, or clearing a file and choosing a font are available from a menu located at the top of the window and can also be accessed by the keyboard. Finally, the last code will introduce further advanced features of the said calculator by including sine, cosine, and exponential functions together with the use of message boxes to deal with errors, which will make the application useful and easy to operate. All these code samples demonstrate how one can construct functional layouts, and easy-to-use controls, and even incorporate aspects such as file manipulation, which would, as a result, improve user experience within the GUI applications implemented with PyQt.

## 8. Assessment Rubric: