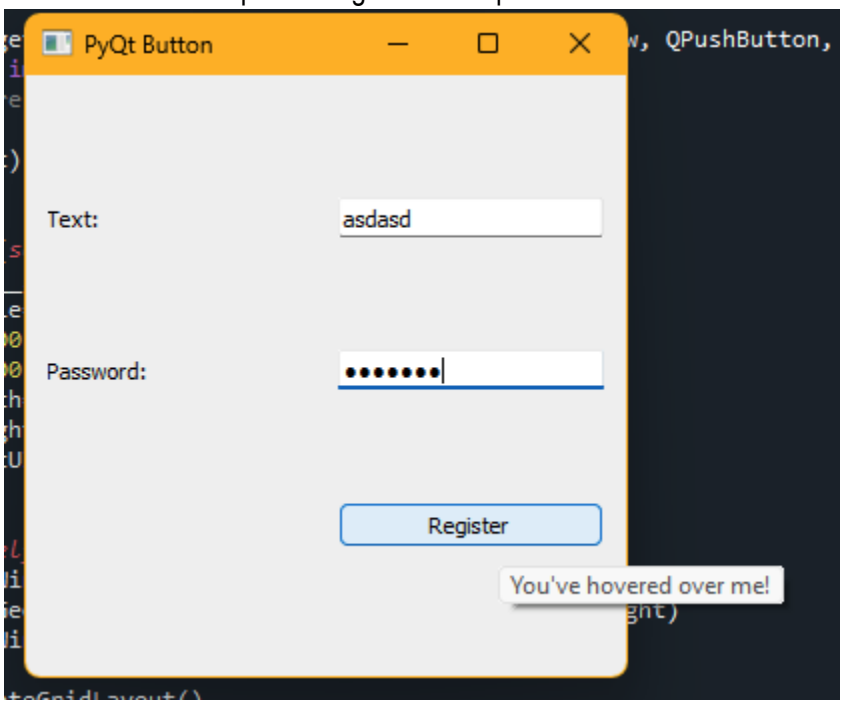
Laboratory Activity 6 - GUI Design: Layout and Styling

Planta, Calvin Earl L.	10/28/2024
CPE 009B / CPE21S4	Prof. Sayo

5. Procedure:

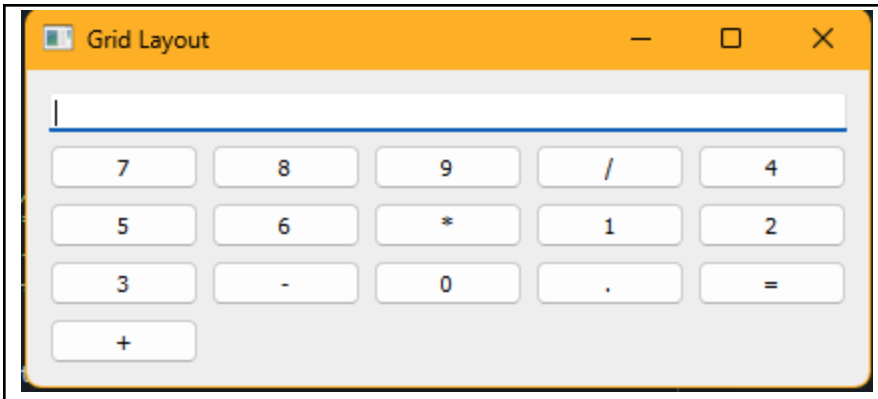
Basic Grid Layout

1. Create a folder named oopfa1<lastname>_lab11
2. Open your Anaconda Navigator and select Visual Studio Code or Spyder IDE.
3. Open that folder in your editor and create a file named gui_grid1.py then copy the code as shown:
4. Run the program and observe the positioning of the components.

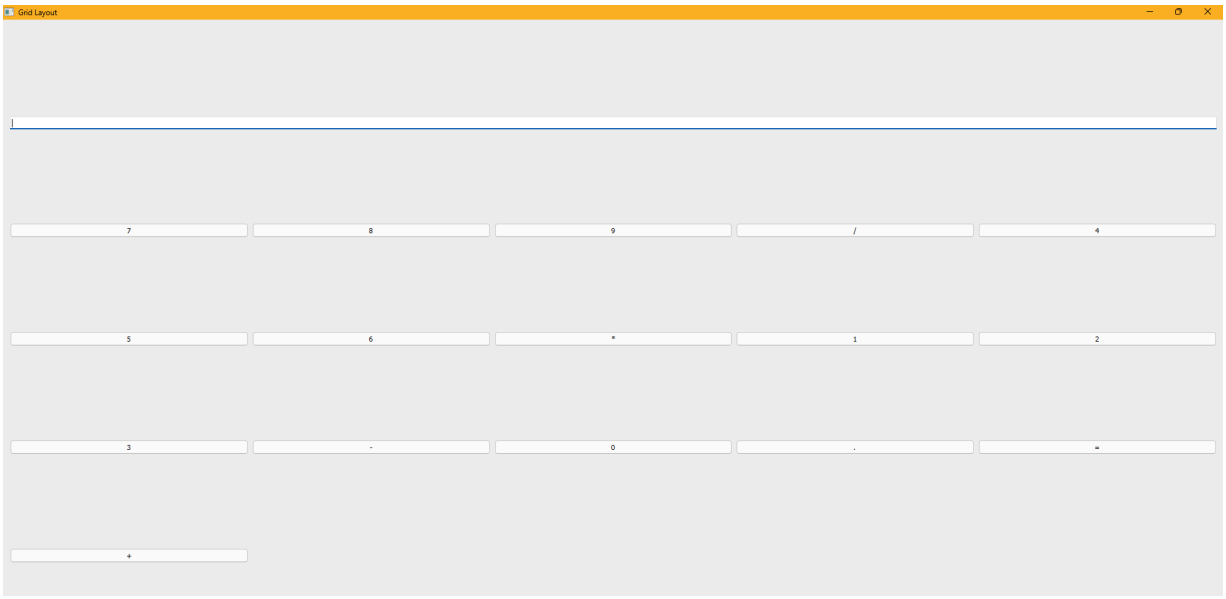


Grid Layout using Loops

1. Create a new file named gui_grid2.py and copy and run the following code:
2. Run the program and observe the output.



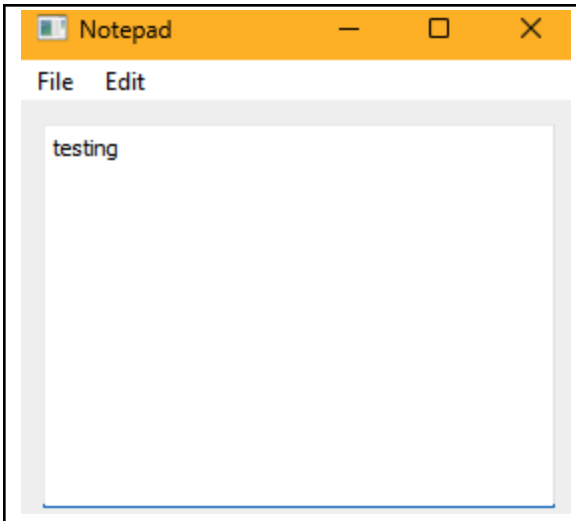
3. Try stretching the window, show the appearance and note your observations.



Upon stretching the window, the line edit field, along with the buttons gets stretched out as well.

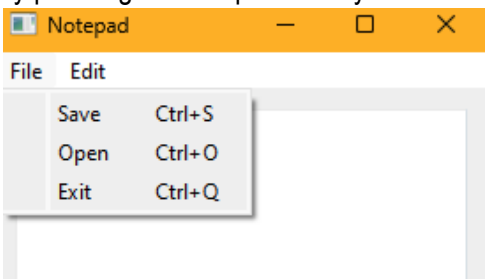
Vbox and Hbox layout managers (Simple Notepad)

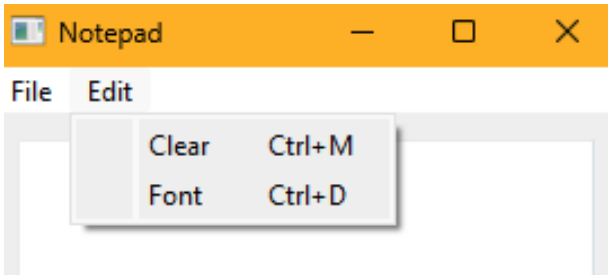
1. Create a new file named gui_simplenotepad.py and copy the following code below:
2. Run the program and observe the output GUI.



3. Try to stretch the window and take note of the response of the GUI.

Upon stretching the window, it basically zooms the window out, allowing for more texts to fit into the screen. The output of the program works like the built-in notepad app in a computer. You can type words and even change its font and clear all of it. You can also save a file as either text or python file, open a file in which you have previously saved, and exit the program. All of the said actions can be done as well by pressing their respective keybinds





6. Supplementary Activity:

Task

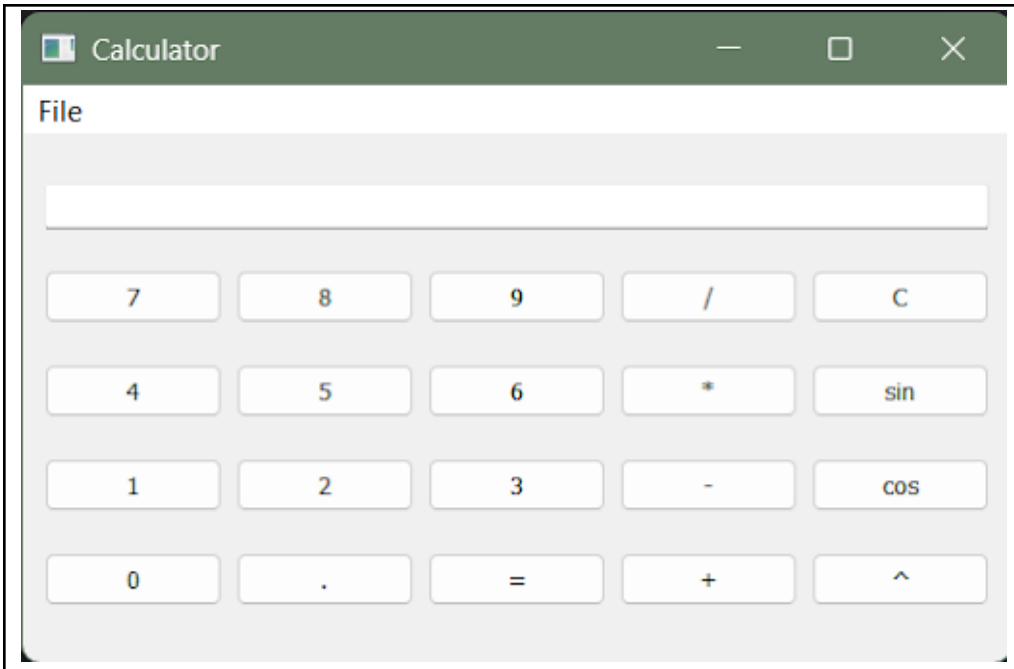
Make a calculator program that can compute perform the Arithmetic operations as well as exponential operation, sin, cosine math functions as well clearing using the C button and/or clear from a menu bar. The calculator must be able to store and retrieve the operations and result in a text file. A file menu should be available and have the option Exit which should also be triggered when ctrl+Q is pressed on the keyboard. You may refer to your calculator program in the Desktop.

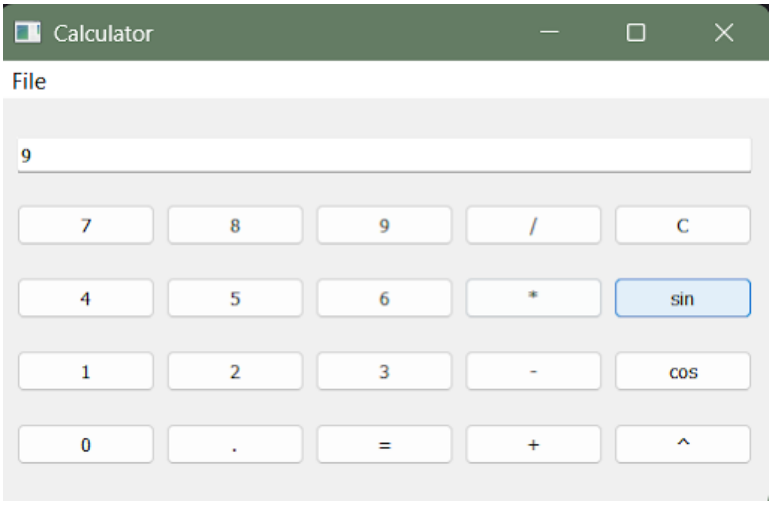
Python

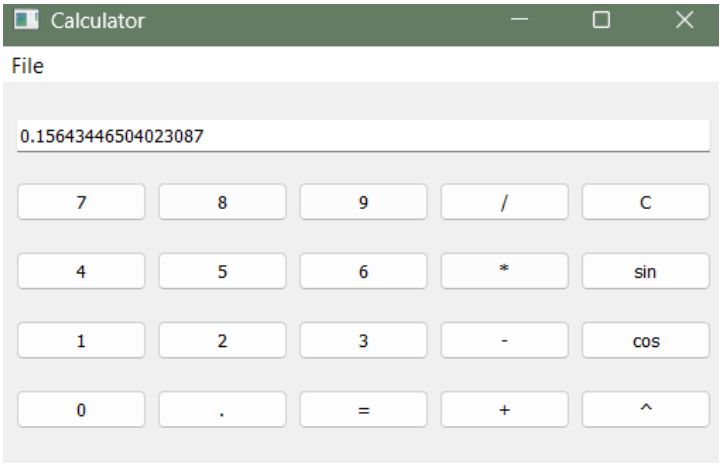
```
import sys
import math
from PyQt5.QtWidgets import (
    QGridLayout, QLineEdit, QPushButton, QVBoxLayout, QWidget, QApplication,
    QMainWindow, QMenuBar, QFileDialog, QAction, QMessageBox
)

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

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

        central_widget = QWidget(self)
        self.setCentralWidget(central_widget)

        grid = QGridLayout()
        central_widget.setLayout(grid)

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

        names = [
            '7', '8', '9', '/', 'C',
            '4', '5', '6', '*', 'sin',
            '1', '2', '3', '-', 'cos',
            '0', '.', '=', '+', '^'
        ]

        positions = [(i, j) for i in range(1, 6) for j in range(5)]
        for position, name in zip(positions, names):
            if name == '':
                continue
            button = QPushButton(name)
            button.clicked.connect(self.on_button_click)
```

```
            grid.addWidget(button, *position)

        self.create_menu()

    def create_menu(self):
        menubar = self.menuBar()
        file_menu = menubar.addMenu('File')

        save_action = QAction('Save', self)
        save_action.triggered.connect(self.save_to_file)
        file_menu.addAction(save_action)

        load_action = QAction('Load', self)
        load_action.triggered.connect(self.load_from_file)
        file_menu.addAction(load_action)


        exit_action = QAction('Exit', self)
        exit_action.setShortcut('Ctrl+Q')
        exit_action.triggered.connect(self.close)
        file_menu.addAction(exit_action)

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

        if button_text == 'C':
            self.textLine.clear()
        elif button_text == '=':
            self.calculate_result()
        elif button_text in ('sin', 'cos', '^'):
            self.handle_math_function(button_text)
        else:
            self.textLine.setText(self.textLine.text() + button_text)

    def calculate_result(self):
        try:
            expression = self.textLine.text().replace('^', '**')
            result = eval(expression)
            self.textLine.setText(str(result))
            self.save_operation(expression, result)
        except Exception as e:
            QMessageBox.warning(self, 'Error', f"Invalid Input: {str(e)}")
    def handle_math_function(self, function):
        try:
            value = float(self.textLine.text())
            if function == 'sin':
                result = math.sin(math.radians(value))
            elif function == 'cos':
                result = math.cos(math.radians(value))
```

```
            elif function == '^':
                result = None
            self.textLine.setText(str(result))
            self.save_operation(f"{function}({value})", result)
        except ValueError:
            QMessageBox.warning(self, 'Error', 'Invalid input for sine or
cosine.')

    def save_operation(self, operation, result):
        with open('operations.txt', 'a') as f:
            f.write(f"{operation} = {result}\n")

    def save_to_file(self):
        options = QFileDialog.Options()
        file_name, _ = QFileDialog.getSaveFileName(self, "Save File", "",
"Text Files (.txt);;All Files ()", options=options)
        if file_name:
            with open(file_name, 'w') as f:
                f.write(self.textLine.text())

    def load_from_file(self):
        options = QFileDialog.Options()
        file_name, _ = QFileDialog.getOpenFileName(self, "Open File", "",
"Text Files (.txt);;All Files ()", options=options)
        if file_name:
            with open(file_name, 'r') as f:
                content = f.read()
                self.textLine.setText(content)

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


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


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


Calculator
File
0.15643446504023087
7
8
9
/
C
4
5
6
*
sin
1
2
3
-
cos
0
.
=
+
^

7. Conclusion:

The lab activity taught me how to utilize Grid Layout to build a simple login page and how to use loops to define button placements in the grid. I discovered how to create a basic Notepad with a menu bar that has buttons for File and Edit. This is where you can change the typefaces, open and load files, save text to a txt or py file, leave the GUI, and clear the text in the notepad. In the supplemental activity, I discovered how to use the desktop calculator as a guide to build a simple working calculator that can carry out basic calculations like sine, cosine, and square root. I now have a solid basis for using PyQt5 to construct GUIs thanks to the knowledge I learned in this lab, which will also help me with programming projects in the future.