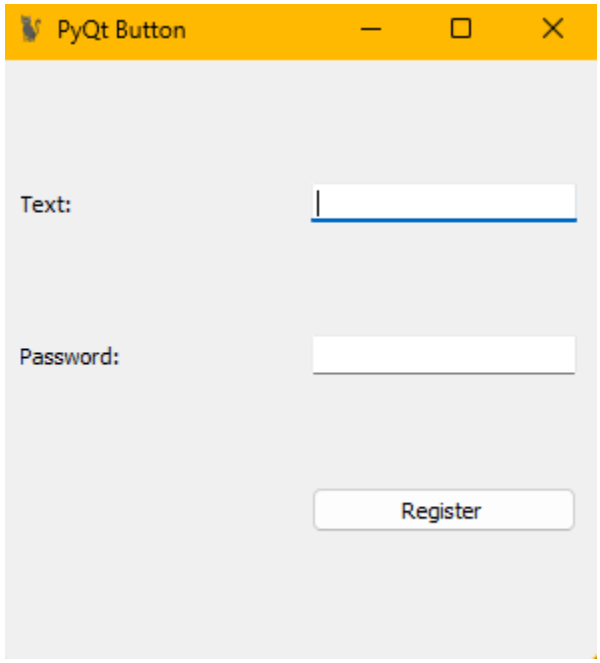
Activity Name #6 - Activity GUI Design_ Layout and Styling	
Valleser, Kenn Jie L.	28/10/2024
CPE009/CPE21S4	Engr. Ma. Rizette Sayo

Basic
Grid
Layout
(code & output)

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


class App(QWidget):

    def __init__(self):
        super().__init__()
        self.title="PyQt Button"
        self.x = 200 #or left
        self.y = 200#or top
        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())
```

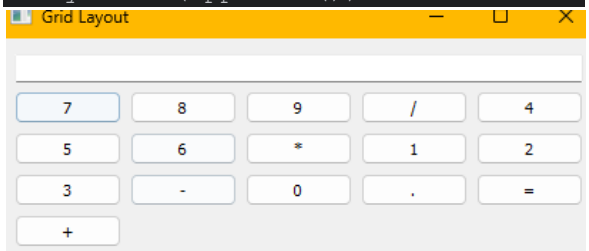


Observation	The output has a window with 2 Texboxes for text and Password and a Button (“Register”) which does nothing

Code & Output

```
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
            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	The Window has basic buttons for a calculator but has no function but its not clean and it can be stretched wide.

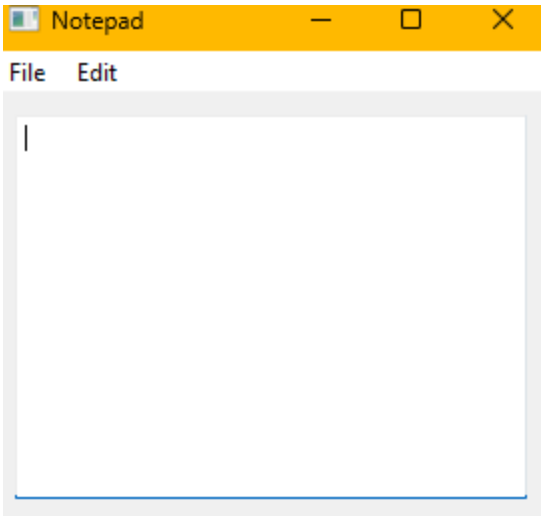
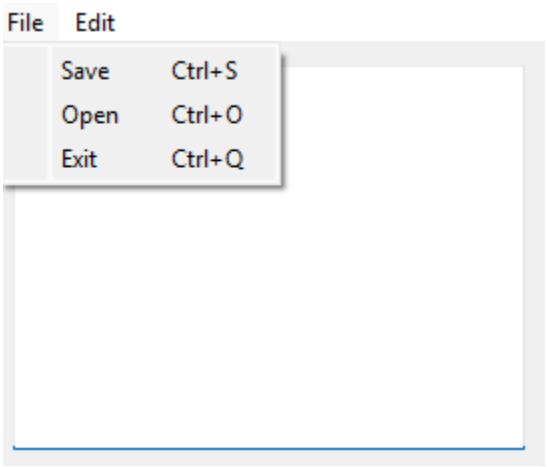
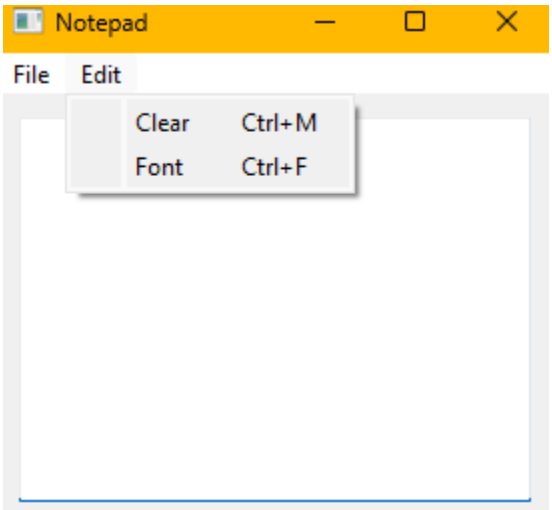
Code & Output	

```
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.i
```

```
co"))
        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.cle
artext)
        editMenu.addAction(editButton)

        fontButton = QAction('Font',
self)

fontButton.setShortcut("ctrl+F")

fontButton.triggered.connect(self.sho
wFontDialog)
        editMenu.addAction(fontButton)

        saveButton = QAction('Save',
self)

saveButton.setShortcut("ctrl+S")

saveButton.triggered.connect(self.sav
eFileDialog)
        fileMenu.addAction(saveButton)

        openButton = QAction('Open',
self)

openButton.setShortcut("Ctrl+O")

openButton.triggered.connect(self.ope
nFileNameDialog)
        fileMenu.addAction(openButton)

        exitButton = QAction('Exit',
self)

exitButton.setShortcut("Ctrl+Q")
        exitButton.setStatusTip('Exit
application')
```

```
exitButton.triggered.connect(self.clo
se)
        fileMenu.addAction(exitButton)

    def showFontDialog(self):
        font, ok =
QFontDialog.getFont()
        if ok:

self.notepad.text.setFont(font)

    def saveFileDialog(self):
        options =
QFileDialog.Options()
        # options |=
QFileDialog.DontUseNativeDialog
        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.toPlainT
ext())

    def openFileNameDialog(self):
        options =
QFileDialog.Options()
        # options |=
QFileDialog.DontUseNativeDialog
        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.cl
eartext)

        self.initUI()
        self.setLayout(self.layout)
        windowLayout = QVBoxLayout()

windowLayout.addWidget(self.horizonta
lGroupBox)
        self.show()

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

self.layout.addWidget(self.text)

#self.layout.addWidget(self.clearbtn)

self.horizontalGroupBox.setLayout(sel
f.layout)

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

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

	  
Observation	The Window has the same functions of the notepad that Operating System provided like font, clear, Save, Open, Exit .

Supplementary Activity:

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

class SciCal(QMainWindow):
   def __init__(self):
       super().__init__()
       self.loadmenu()
       self.initUI()

   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 saveFileDialog(self):
       options = QFileDialog.Options()
       fileName, _ = QFileDialog.getSaveFileName(self, "Save calculator
file", "",
                                                 "Text Files (*.txt);;All
files (*)",
                                                 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 calculator
```

```
file", "",
                                                        "Text Files (*.txt);;All
files (*)",
                                                        options=options)
        if fileName:
            with open(fileName, 'r') as file:
                data = file.read()
                self.textLine.setText(data)

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

    def initUI(self):
        centralWidget = QWidget(self)
        self.setCentralWidget(centralWidget)

        grid = QGridLayout()
        centralWidget.setLayout(grid)

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

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

        # Create buttons and add to layout
        positions = [(i, j) for i in range(1, 5) for j in range(5)]
        for position, name in zip(positions, names):
            button = QPushButton(name)
            grid.addWidget(button, *position)
            button.clicked.connect(lambda _, b=name: self.buttonClicked(b))

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

    def buttonClicked(self, text):
        if text == 'Clear':
            self.cleartext()
        elif text == '=':
            self.calculate_result()
        elif text == 'sin':
            self.calculate_trig(math.sin)
        elif text == 'cos':
            self.calculate_trig(math.cos)
        elif text == 'exp':
            self.calculate_exp()
        else:
            self.textLine.setText(self.textLine.text() + text)

    def calculate_result(self):
```

```
        try:
            result = str(eval(self.textLine.text()))
            self.textLine.setText(result)
        except Exception:
            self.textLine.setText("Error")

    def calculate_trig(self, func):
        try:
            value = float(self.textLine.text())
            result = str(func(math.radians(value)))  # Convert to radians
            self.textLine.setText(result)
        except Exception:
            self.textLine.setText("Error")

    def calculate_exp(self):
        try:
            value = float(self.textLine.text())
            result = str(math.exp(value))
            self.textLine.setText(result)
        except Exception:
            self.textLine.setText("Error")

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

Conclusion:

In the activity, I've learned a lot about GUI design using PyQt5, which makes creating interactive applications much more straightforward. For example, my scientific calculator not only performs basic calculations but also includes trigonometric functions like sine and cosine, showing how you can integrate more complex math into a user-friendly interface. The Notepad app I created demonstrates how to manage text input, allowing users to save and open files easily. I also used layouts, like grid and box layouts, which help organize widgets neatly on the screen. these projects have given me a solid look in designing functional and aesthetically pleasing applications and deepening my understanding of Python and PyQt5.