

Hands-on Activity 6.3 GUI Design: Layout and Styling	
Buenafe, Dhafny Francisco,Lauper Xavier Efa, Christian	4/20/2022
BSCPE12S1	Engr. Roman Richard

Procedures:

C:\Users\LENOVO\OneDrive\Documents\123\gui_grid1.py

gui_buttonclicked.py x gui_messagebox.py x gui_grid1.py x gui_grid2.py x gui_simplenotepad.py x

```
3 QLineEdit, QLabel, QGridLayout
4 from PyQt5.QtGui import QLineEdit
5
6 class App(QWidget):
7
8     def __init__(self):
9         super().__init__()
10        self.title = 'PyQt Line Edit'
11        self.x = 200
12        self.y = 200
13        self.width = 300
14        self.height = 200
15        self.initUI()
16
17    def initUI(self):
18        self.setWindowTitle(self.title)
19        self.setGeometry(self.x, self.y, self.width, self.height)
20        self.setWindowIcon(QIcon(''))
21
22        self.createGridLayout()
23        self.setLayout(self.grid)
24        self.show()
25
26    def createGridLayout(self):
27        self.layout = QGridLayout()
28
29        self.layout.setColumnStretch(1,2)
30
31        self.textboxlbl = QLabel("Text: ", self)
32        self.textbox = QLineEdit(self)
33        self.passwordlbl = QLabel("Password: ", self)
34        self.password = QLineEdit(self)
35        self.password.setEchoMode(QLineEdit.Password)
36        self.button = QPushButton('Register', self)
37        self.button.setToolTip("You've hovered over me!")
38        self.layout.addWidget(self.textboxlbl, 0,1)
```

PyQt Line Edit

Text:

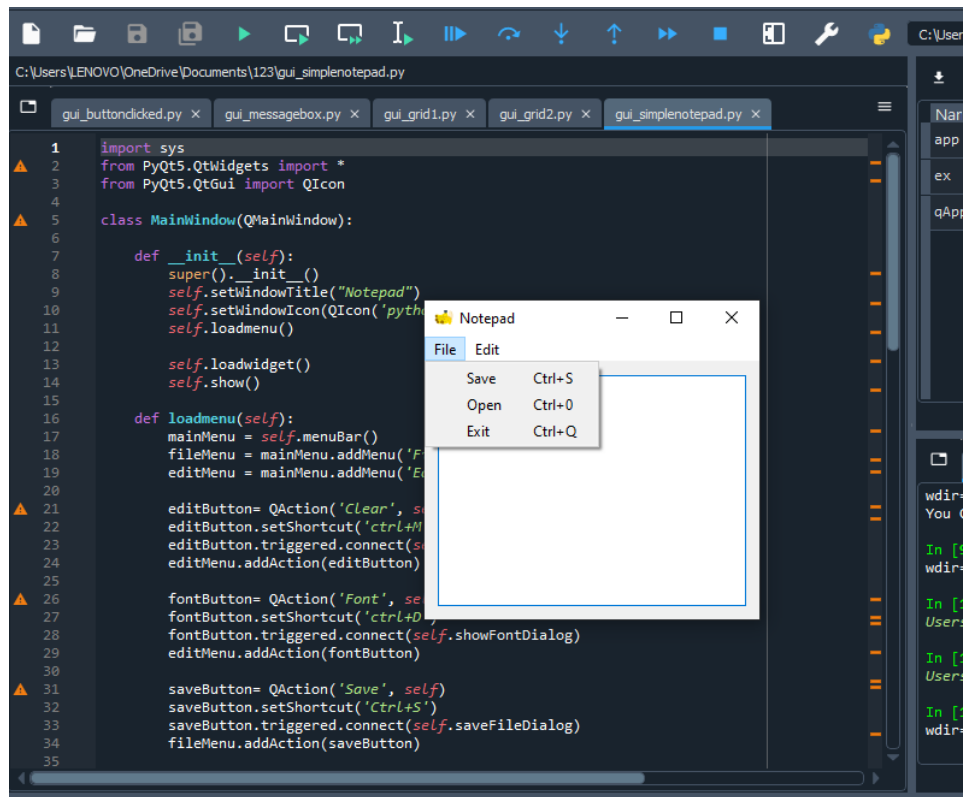
Password:

Register

```
1 import sys
2 from PyQt5.QtWidgets import QWidget, QApplication, QMainWindow, QPushButton, \
3 QLineEdit, QLabel, QGridLayout, QHBoxLayout, QVBoxLayout
4
5 class GridExample(QWidget):
6     def __init__(self):
7         super().__init__()
8         self.initUI()
9     def initUI(self):
10        grid = QGridLayout()
11        self.setLayout(grid)
12
13        names = [
14            '7', '8', '9',
15            '4', '5', '6',
16            '1', '2', '3',
17            '0', '.', '=',
18            ',', '-', '+',
19        ]
20
21        self.textline = QLineEdit()
22        grid.addWidget(self.textline, 0,1,1,5)
23
24        positions = [(i,j) for i in range(1,7) for j in range(1,6)]
25        for position, name in zip(positions, names):
26            if name == '':
27                continue
28            button = QPushButton(name)
29            grid.addWidget(button, *position)
30
31        self.setGeometry(300, 300, 300, 150)
32        self.setWindowTitle('Grid Layout')
33        self.show()
34
35 if __name__ == '__main__':
36     app = QApplication(sys.argv)
```

Grid Layout

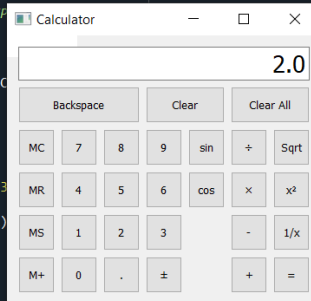
7	8	9	/	4
5	6	*	1	2
3	-	0	.	=
+				



Supplementary Task:

```
C:\Users\lauper xavier\Downloads\Calculator2 (1).py

Calculator2 (1).py X
85     self.plusButton = self.createButton("+", self.additiveOperatorClicked)
86
87     self.squareRootButton = self.createButton("Sqrt",
88         self.unaryOperatorClicked)
89     self.powerButton = self.createButton(u"x\N{SUPERSCRIP",
90         self.unaryOperatorClicked)
91     self.reciprocalButton = self.createButton("1/x",
92         self.unaryOperatorClicked)
93     self.equalButton = self.createButton("=", self.equalClicked)
94
95     mainLayout = QGridLayout()
96     mainLayout.setSizeConstraint(QLayout.SetFixedSize)
97
98     mainLayout.addWidget(self.display, 0, 0, 1, 7)
99     mainLayout.addWidget(self.backspaceButton, 1, 0, 1, 3)
100    mainLayout.addWidget(self.clearButton, 1, 3, 1, 2)
101    mainLayout.addWidget(self.clearAllButton, 1, 5, 1, 2)
102
103    mainLayout.addWidget(self.clearMemoryButton, 2, 0)
104    mainLayout.addWidget(self.readMemoryButton, 3, 0)
105    mainLayout.addWidget(self.setMemoryButton, 4, 0)
106    mainLayout.addWidget(self.addToMemoryButton, 5, 0)
107
108    for i in range(1, Calculator.NumDigitButtons):
109        row = ((i - 1) / 3) + 2
110        column = ((i - 1) % 3) + 1
111        mainLayout.addWidget(self.digitButtons[i], row, column)
112
113    mainLayout.addWidget(self.digitButtons[0], 5, 1)
114    mainLayout.addWidget(self.pointButton, 5, 2)
115    mainLayout.addWidget(self.changeSignButton, 5, 3)
116
117    mainLayout.addWidget(self.sinButton, 2, 4)
118    mainLayout.addWidget(self.cosButton, 3, 4)
119
120
121    mainLayout.addWidget(self.divisionButton, 2, 5)
122    mainLayout.addWidget(self.timesButton, 3, 5)
123    mainLayout.addWidget(self.minusButton, 4, 5)
124    mainLayout.addWidget(self.plusButton, 5, 5)
125
126    mainLayout.addWidget(self.squareRootButton, 2, 6)
127    mainLayout.addWidget(self.powerButton, 3, 6)
```



CalculatorData - Notepad

File Edit Format View Help

Answer: 2.0

```
C:\Users\lauper xavier\Downloads\Calculator2 (1).py

Calculator2 (1).py X

337
338     def setMemory(self):
339         self.equalClicked()
340         self.sumInMemory = float(self.display.text())
341
342     def addToMemory(self):
343         self.equalClicked()
344         self.sumInMemory += float(self.display.text())
345
346     def createButton(self, text, member):
347         button = Button(text)
348         button.clicked.connect(member)
349         return button
350
351     def abortOperation(self):
352         self.clearAll()
353         self.display.setText("####")
354
355     def calculate(self, rightOperand, pendingOperator):
356         if pendingOperator == "+":
357             self.sumSoFar += rightOperand
358         elif pendingOperator == "-":
359             self.sumSoFar -= rightOperand
360         elif pendingOperator == u"\N{MULTIPLICATION SIGN}":
361             self.factorSoFar *= rightOperand
362         elif pendingOperator == u"\N{DIVISION SIGN}":
363             if rightOperand == 0.0:
364                 return False
365             self.factorSoFar /= rightOperand
366
367         return True
368
369
370
371 if __name__ == '__main__':
372
373     import sys
374
375     app = QApplication(sys.argv)
376     calc = Calculator()
377     calc.show()
378     sys.exit(app.exec_())
```

New file

```
C:\Users\lauper xavier\Downloads\Calculator2 (1).py

Calculator2 (1).py X

322
323     def clearAll(self):
324         self.sumSoFar = 0.0
325         self.factorSoFar = 0.0
326         self.pendingAdditiveOperator = ''
327         self.pendingMultiplicativeOperator = ''
328         self.display.setText('0')
329         self.waitingForOperand = True
330
331     def clearMemory(self):
332         self.sumInMemory = 0.0
333
334     def readMemory(self):
335         self.display.setText(str(self.sumInMemory))
336         self.waitingForOperand = True
337
338     def setMemory(self):
339         self.equalClicked()
340         self.sumInMemory = float(self.display.text())
341
342     def addToMemory(self):
343         self.equalClicked()
344         self.sumInMemory += float(self.display.text())
345
346     def createButton(self, text, member):
347         button = Button(text)
348         button.clicked.connect(member)
349         return button
350
351     def abortOperation(self):
352         self.clearAll()
353         self.display.setText("####")
354
355     def calculate(self, rightOperand, pendingOperator):
356         if pendingOperator == "+":
357             self.sumSoFar += rightOperand
358         elif pendingOperator == "-":
359             self.sumSoFar -= rightOperand
360         elif pendingOperator == u"\N{MULTIPLICATION SIGN}":
361             self.factorSoFar *= rightOperand
362         elif pendingOperator == u"\N{DIVISION SIGN}":
363             if rightOperand == 0.0:
364                 return False
```

New file

```
C:\Users\lauper xavier\Downloads\Calculator2 (1).py

Calculator2 (1).py X
271         if not self.calculate(operand, self.pendingAdditiveOperator):
272             self.abortOperation()
273             return
274
275         self.pendingAdditiveOperator = ''
276     else:
277         self.sumSoFar = operand
278
279         self.display.setText(str(self.sumSoFar))
280         self.sumSoFar = 0.0
281         self.waitingForOperand = True
282         register.write("Answer: " + self.display.text())
283     register.close()
284
285     def pointClicked(self):
286         if self.waitingForOperand:
287             self.display.setText('0')
288
289         if "." not in self.display.text():
290             self.display.setText(self.display.text() + ".")
291
292         self.waitingForOperand = False
293
294     def changeSignClicked(self):
295         text = self.display.text()
296         value = float(text)
297
298         if value > 0.0:
299             text = "-" + text
300         elif value < 0.0:
301             text = text[1:]
302
303         self.display.setText(text)
304
305     def backspaceClicked(self):
306         if self.waitingForOperand:
307             return
308
309         text = self.display.text()[:-1]
310         if not text:
311             text = '0'
312         self.waitingForOperand = True
313
```

New file

```
C:\Users\lauper xavier\Downloads\Calculator2 (1).py

Calculator2 (1).py X
226         self.pendingMultiplicativeOperator = ''
227
228         if self.pendingAdditiveOperator:
229             if not self.calculate(operand, self.pendingAdditiveOperator):
230                 self.abortOperation()
231                 return
232
233             self.display.setText(str(self.sumSoFar))
234         else:
235             self.sumSoFar = operand
236
237         self.pendingAdditiveOperator = clickedOperator
238         self.waitingForOperand = True
239
240     def multiplicativeOperatorClicked(self):
241         clickedButton = self.sender()
242         clickedOperator = clickedButton.text()
243         operand = float(self.display.text())
244
245         if self.pendingMultiplicativeOperator:
246             if not self.calculate(operand, self.pendingMultiplicativeOperator):
247                 self.abortOperation()
248                 return
249
250             self.display.setText(str(self.factorSoFar))
251         else:
252             self.factorSoFar = operand
253
254         self.pendingMultiplicativeOperator = clickedOperator
255         self.waitingForOperand = True
256
257     def equalClicked(self):
258         operand = float(self.display.text())
259         register = open("CalculatorData.txt", "w+")
260
261         if self.pendingMultiplicativeOperator:
262             if not self.calculate(operand, self.pendingMultiplicativeOperator):
263                 self.abortOperation()
264                 return
265
266         operand = self.factorSoFar
267         self.factorSoFar = 0.0
268         self.pendingMultiplicativeOperator = ''
```

C:\Users\jauper xavier\Downloads\Calculator2 (1).py

```
Calculator2 (1).py X
181         self.abortOperation()
182         return
183
184         result = 0.0 / operand
185
186         self.display.setText(str(result))
187         self.waitingForOperand = True
188
189
190     def unaryOperatorClicked(self):
191         clickedButton = self.sender()
192         clickedOperator = clickedButton.text()
193         operand = float(self.display.text())
194
195         if clickedOperator == "Sqrt":
196             if operand < 0.0:
197                 self.abortOperation()
198                 return
199
200             result = math.sqrt(operand)
201         elif clickedOperator == u"x\N{SUPERSCRIPT TWO}":
202             result = math.pow(operand, 2.0)
203         elif clickedOperator == "1/x":
204             if operand == 0.0:
205                 self.abortOperation()
206                 return
207
208             result = 1.0 / operand
209
210         self.display.setText(str(result))
211         self.waitingForOperand = True
212
213     def additiveOperatorClicked(self):
214         clickedButton = self.sender()
215         clickedOperator = clickedButton.text()
216         operand = float(self.display.text())
217
218         if self.pendingMultiplicativeOperator:
219             if not self.calculate(operand, self.pendingMultiplicativeOperator):
220                 self.abortOperation()
221                 return
222
223         self.display.setText(str(self.factorSoFar))
```

New file

C:\Users\jauper xavier\Downloads\Calculator2 (1).py

```
Calculator2 (1).py X
136     fileMenu = mainMenu.addMenu('File')
137     editMenu = mainMenu.addMenu('Edit')
138
139     editButton= QAction('Clear', self)
140     editButton.triggered.connect(self.clearAll)
141     editMenu.addAction(editButton)
142
143     exitButton = QAction('Exit', self)
144     exitButton.setShortcut('Ctrl+Q')
145     exitButton.setStatusTip('Exit application')
146     exitButton.triggered.connect(self.close)
147     fileMenu.addAction(exitButton)
148
149
150
151
152     def digitClicked(self):
153         clickedButton = self.sender()
154         digitValue = int(clickedButton.text())
155
156         if self.display.text() == '0' and digitValue == 0.0:
157             return
158
159         if self.waitingForOperand:
160             self.display.clear()
161             self.waitingForOperand = False
162
163         self.display.setText(self.display.text() + str(digitValue))
164
165     def TrigoOperatorClicked(self):
166
167         clickedButton = self.sender()
168         clickedOperator = clickedButton.text()
169         operand = float(self.display.text())
170
171         if clickedOperator == "Trigo":
172             if operand < 0.0:
173                 self.abortOperation()
174                 return
175
176         result = math.sin (math.radians(angle_in_degrees) + (operand))
177         elif clickedOperator == "sin":
178             result =math.cos(math.radians(angle_in_degrees) + (operand))
```

C:\Users\lauper xavier\Downloads\Calculator2 (1).py

Calculator2 (1).py X

```
91     self.reciprocalButton = self.createButton("1/x",
92                                             self.unaryOperatorClicked)
93     self.equalButton = self.createButton("=", self.equalClicked)
94
95     mainLayout = QGridLayout()
96     mainLayout.setSizeConstraint(QLayout.SetFixedSize)
97
98     mainLayout.addWidget(self.display, 0, 0, 1, 7)
99     mainLayout.addWidget(self.backspaceButton, 1, 0, 1, 3)
100    mainLayout.addWidget(self.clearButton, 1, 3, 1, 2)
101    mainLayout.addWidget(self.clearAllButton, 1, 5, 1, 2)
102
103    mainLayout.addWidget(self.clearMemoryButton, 2, 0)
104    mainLayout.addWidget(self.readMemoryButton, 3, 0)
105    mainLayout.addWidget(self.setMemoryButton, 4, 0)
106    mainLayout.addWidget(self.addToMemoryButton, 5, 0)
107
108    for i in range(1, Calculator.NumDigitButtons):
109        row = ((9 - i) / 3) + 2
110        column = ((i - 1) % 3) + 1
111        mainLayout.addWidget(self.digitButtons[i], row, column)
112
113    mainLayout.addWidget(self.digitButtons[0], 5, 1)
114    mainLayout.addWidget(self.pointButton, 5, 2)
115    mainLayout.addWidget(self.changeSignButton, 5, 3)
116
117    mainLayout.addWidget(self.sinButton, 2, 4)
118    mainLayout.addWidget(self.cosButton, 3, 4)
119
120
121    mainLayout.addWidget(self.divisionButton, 2, 5)
122    mainLayout.addWidget(self.timesButton, 3, 5)
123    mainLayout.addWidget(self.minusButton, 4, 5)
124    mainLayout.addWidget(self.plusButton, 5, 5)
125
126    mainLayout.addWidget(self.squareRootButton, 2, 6)
127    mainLayout.addWidget(self.powerButton, 3, 6)
128    mainLayout.addWidget(self.reciprocalButton, 4, 6)
129    mainLayout.addWidget(self.equalButton, 5, 6)
130    self.setLayout(mainLayout)
131
132    self.setWindowTitle("Calculator")
133
```

Open file

C:\Users\lauper xavier\Downloads\Calculator2 (1).py

Calculator2 (1).py X

```
46     self.display.setReadOnly(True)
47     self.display.setAlignment(Qt.AlignRight)
48     self.display.setMaxLength(15)
49
50     font = self.display.font()
51     font.setPointSize(font.pointSize() + 8)
52     self.display.setFont(font)
53
54     self.digitButtons = []
55
56     for i in range(Calculator.NumDigitButtons):
57         self.digitButtons.append(self.createButton(str(i),
58                                                     self.digitClicked))
59
60     self.pointButton = self.createButton(".", self.pointClicked)
61     self.changeSignButton = self.createButton(u"\N{PLUS-MINUS SIGN}",
62                                             self.changeSignClicked)
63
64     self.backspaceButton = self.createButton("Backspace",
65                                             self.backspaceClicked)
66     self.clearButton = self.createButton("Clear", self.clear)
67     self.clearAllButton = self.createButton("Clear ALL", self.clearAll)
68     self.clearAllButton.setShortcut('C')
69     self.clearAllButton.triggered.connect(self.clearAll)
70
71     self.clearMemoryButton = self.createButton("MC", self.clearMemory)
72     self.readMemoryButton = self.createButton("MR", self.readMemory)
73     self.setMemoryButton = self.createButton("MS", self.setMemory)
74     self.addToMemoryButton = self.createButton("M+", self.addToMemory)
75
76     self.sinButton = self.createButton("sin", self.TrigoOperatorClicked)
77     self.cosButton = self.createButton("cos", self.TrigoOperatorClicked)
78
79
80     self.divisionButton = self.createButton(u"\N{DIVISION SIGN}",
81                                             self.multiplicativeOperatorClicked)
82     self.timesButton = self.createButton(u"\N{MULTIPLICATION SIGN}",
83                                         self.multiplicativeOperatorClicked)
84     self.minusButton = self.createButton("-", self.additiveOperatorClicked)
85     self.plusButton = self.createButton("+", self.additiveOperatorClicked)
86
87     self.squareRootButton = self.createButton("Sqrt",
88                                             self.unaryOperatorClicked)
```

New file


```
C:\Users\jauper xavier\Downloads\Calculator2 (1).py
Calculator2 (1).py X
1 import math
2
3 from PyQt5.QtCore import Qt
4 from PyQt5.QtWidgets import (QApplication, QGridLayout, QLayout, QLineEdit,
5     QSizePolicy, QToolButton, QWidget, QMainWindow, QMenuBar, QAction)
6
7 angle_in_degrees = 45
8 angle_in_radians = math.radians(angle_in_degrees)
9
10 class Button(QToolButton):
11     def __init__(self, text, parent=None):
12         super(Button, self).__init__(parent)
13
14         self.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Preferred)
15         self.setText(text)
16
17     def sizeHint(self):
18         size = super(Button, self).sizeHint()
19         size.setHeight(size.height() + 20)
20         size.setWidth(max(size.width(), size.height ()))
21         return size
22
23
24 class Calculator(QWidget):
25     NumDigitButtons = 10
26
27     def __init__(self, parent=None):
28         super(Calculator, self).__init__(parent)
29
30         self.loadmenu()
31
32         self.show()
33
34
35         self.pendingAdditiveOperator = ''
36         self.pendingMultiplicativeOperator = ''
37         self.pendingTrigoOperator = ''
38
39         self.sumInMemory = 0.0
40         self.sumSoFar = 0.0
41         self.factorSoFar = 0.0
42         self.waitingForOperand = True
```

Conclusion:

on the supplementary task which is required to have a Arithmetic operations as well as exponential operation, sin, and cosine we researched for the solution of sin and cosine and based on the desktop calculator to determine a proper arithmetic operations. We manage to put a file menu and an option to exit by having a base on the previous gui_simplenotepad. After we finished on the supplementary task we are able to Create a GUI program with layout and stylesheets.

"I accept responsibility for my role in ensuring the integrity of the work submitted by the group in which I participated."