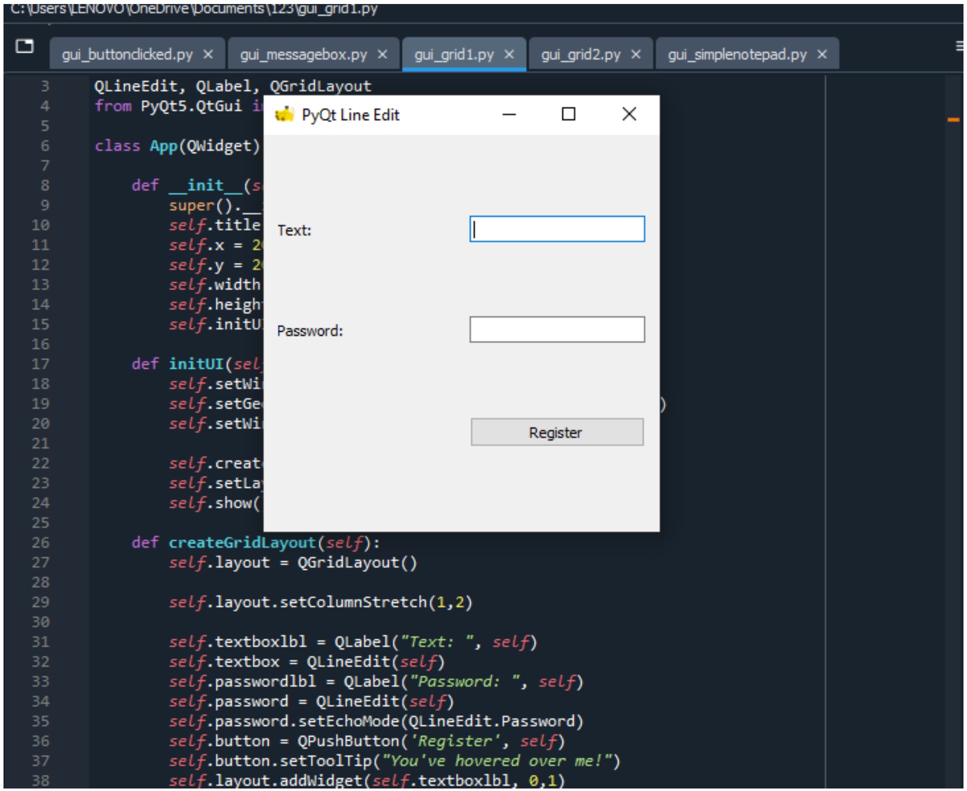
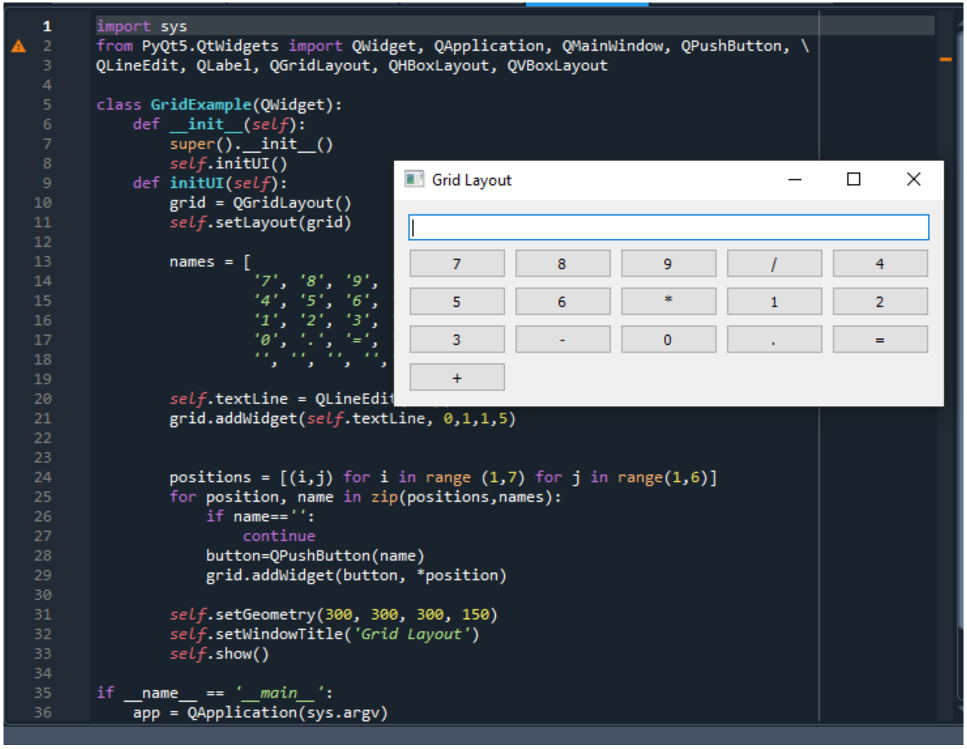
Hands-on Activity 6.3 GUI Design: Layout and Styling	
Buenafe, Dhafny Francisco,Lauper Xavier Efa, Christian	4/20/2022
BSCPE12S1	Engr. Roman Richard

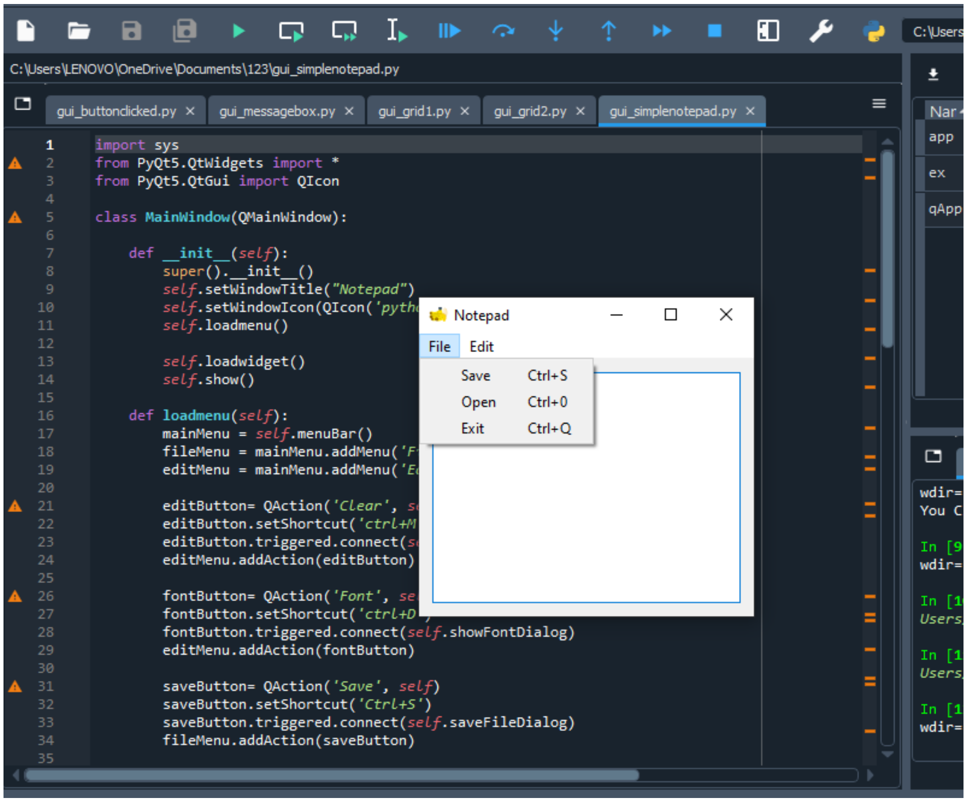
Procedures:



```
QLineEdit, QLabel, QGridLayout
from PyQt5.QtGui i

class App(QWidget)

    def __init__(s
        super().__
        self.title
        self.x = 2
        self.y = 2
        self.width
        self.heigh
        self.initU

    def initUI(sel
        self.setWi
        self.setGe                              )
        self.setWi

        self.creat
        self.setLa
        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)
```



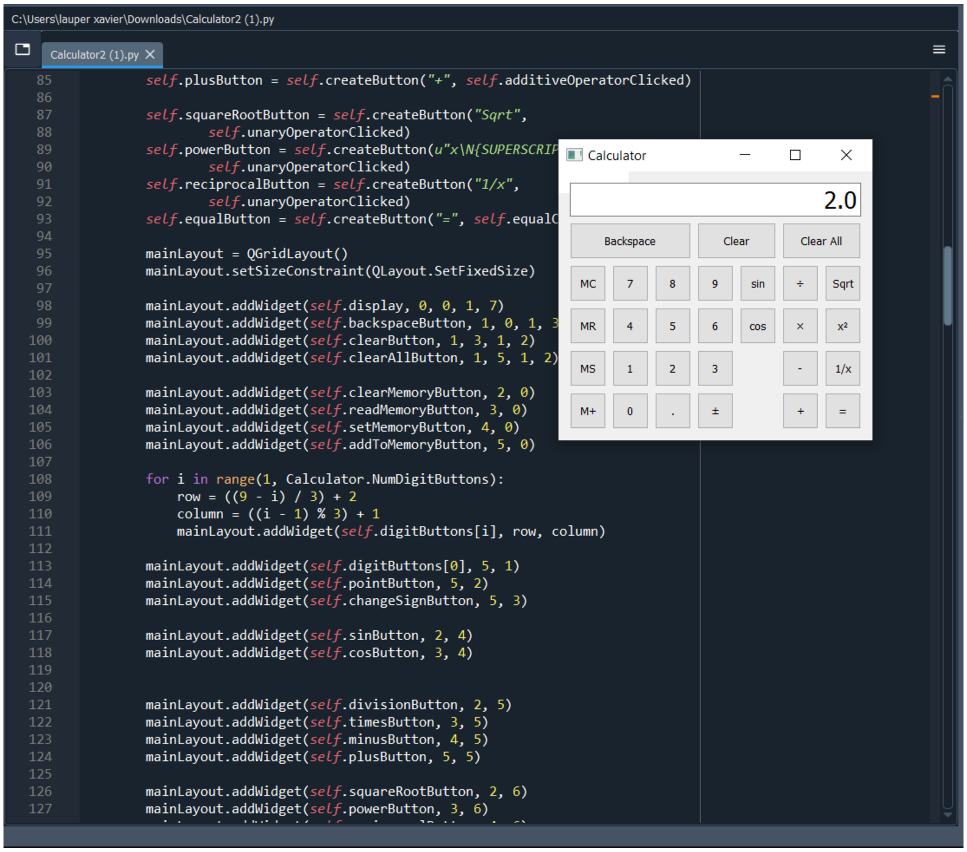
```
import sys
from PyQt5.QtWidgets import QWidget, QApplication, QMainWindow, QPushButton, \
QLineEdit, QLabel, QGridLayout, QHBoxLayout, QVBoxLayout

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 = QLineEdi
        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)
```



```
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('pyth
        self.loadmenu()

        self.loadwidget()
        self.show()

    def loadmenu(self):
        mainMenu = self.menuBar()
        fileMenu = mainMenu.addMenu('F
        editMenu = mainMenu.addMenu('E

        editButton= QAction('Clear', s
        editButton.setShortcut('ctrl+M
        editButton.triggered.connect(s
        editMenu.addAction(editButton)

        fontButton= QAction('Font', se
        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)
```

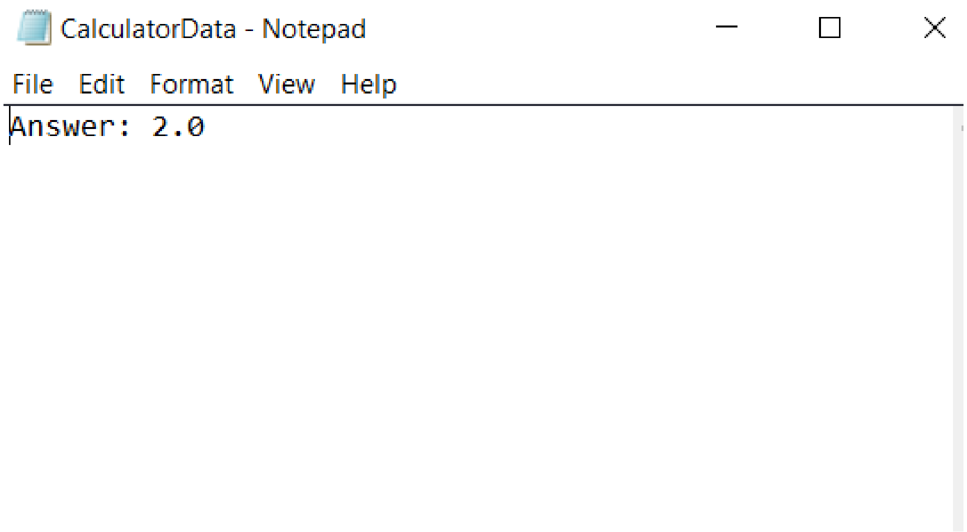
```
wdir=
You C

In [9
wdir=

In [1
Users

In [1
Users

In [1
wdir=
```

Supplementary Task:

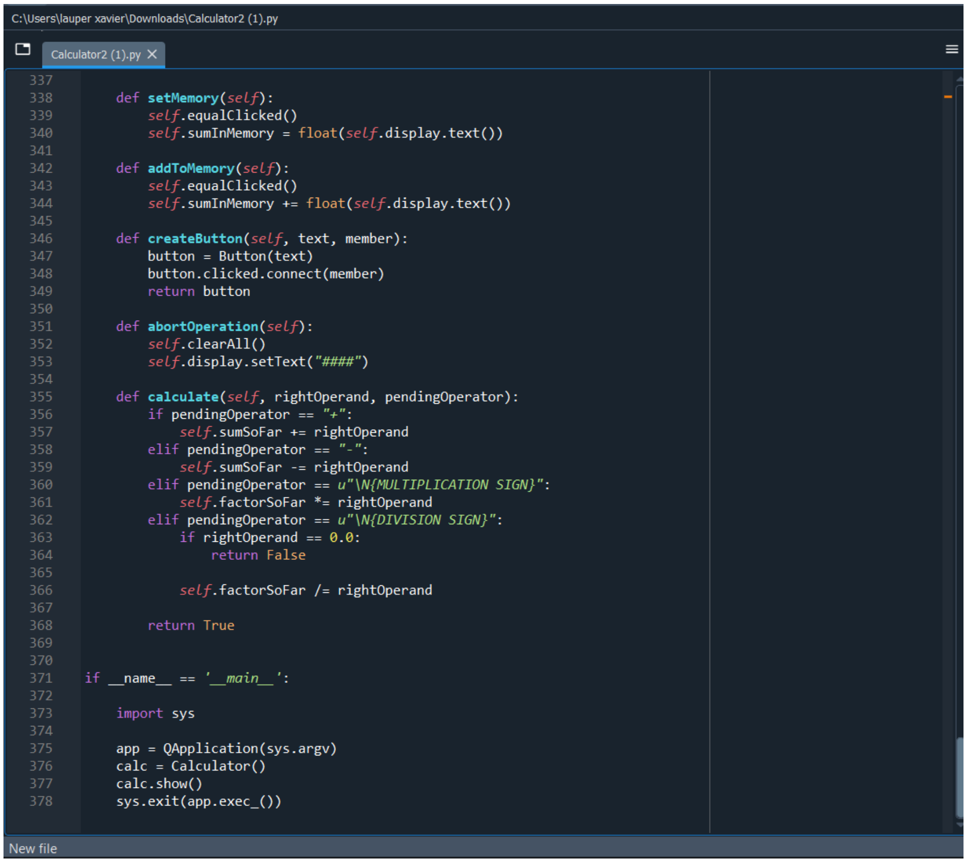


C:\Users\lauper xavier\Downloads\Calculator2 (1).py

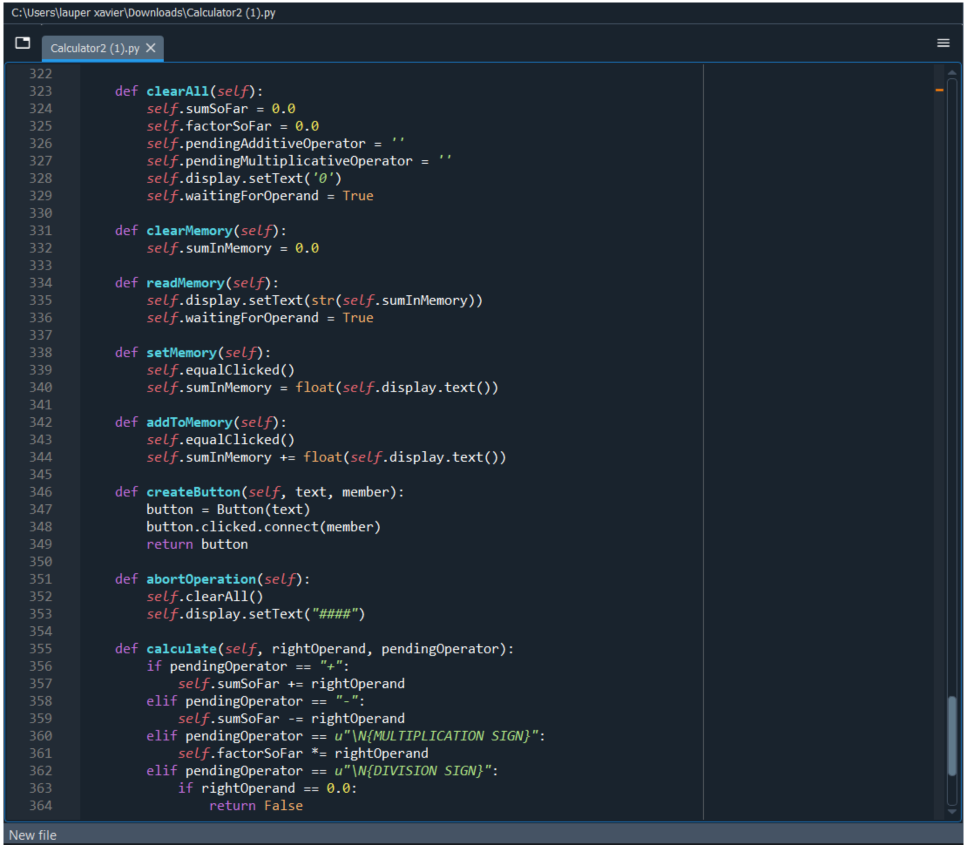
```
        self.plusButton = self.createButton("+", self.additiveOperatorClicked)

        self.squareRootButton = self.createButton("Sqrt",
                self.unaryOperatorClicked)
        self.powerButton = self.createButton(u"x\N{SUPERSCRIP
                self.unaryOperatorClicked)
        self.reciprocalButton = self.createButton("1/x",
                self.unaryOperatorClicked)
        self.equalButton = self.createButton("=", self.equalC

        mainLayout = QGridLayout()
        mainLayout.setSizeConstraint(QLayout.SetFixedSize)

        mainLayout.addWidget(self.display, 0, 0, 1, 7)
        mainLayout.addWidget(self.backspaceButton, 1, 0, 1, 3
        mainLayout.addWidget(self.clearButton, 1, 3, 1, 2)
        mainLayout.addWidget(self.clearAllButton, 1, 5, 1, 2)

        mainLayout.addWidget(self.clearMemoryButton, 2, 0)
        mainLayout.addWidget(self.readMemoryButton, 3, 0)
        mainLayout.addWidget(self.setMemoryButton, 4, 0)
        mainLayout.addWidget(self.addToMemoryButton, 5, 0)

        for i in range(1, Calculator.NumDigitButtons):
            row = ((9 - i) / 3) + 2
            column = ((i - 1) % 3) + 1
            mainLayout.addWidget(self.digitButtons[i], row, column)

        mainLayout.addWidget(self.digitButtons[0], 5, 1)
        mainLayout.addWidget(self.pointButton, 5, 2)
        mainLayout.addWidget(self.changeSignButton, 5, 3)

        mainLayout.addWidget(self.sinButton, 2, 4)
        mainLayout.addWidget(self.cosButton, 3, 4)


        mainLayout.addWidget(self.divisionButton, 2, 5)
        mainLayout.addWidget(self.timesButton, 3, 5)
        mainLayout.addWidget(self.minusButton, 4, 5)
        mainLayout.addWidget(self.plusButton, 5, 5)

        mainLayout.addWidget(self.squareRootButton, 2, 6)
        mainLayout.addWidget(self.powerButton, 3, 6)
```



```
Answer: 2.0
```

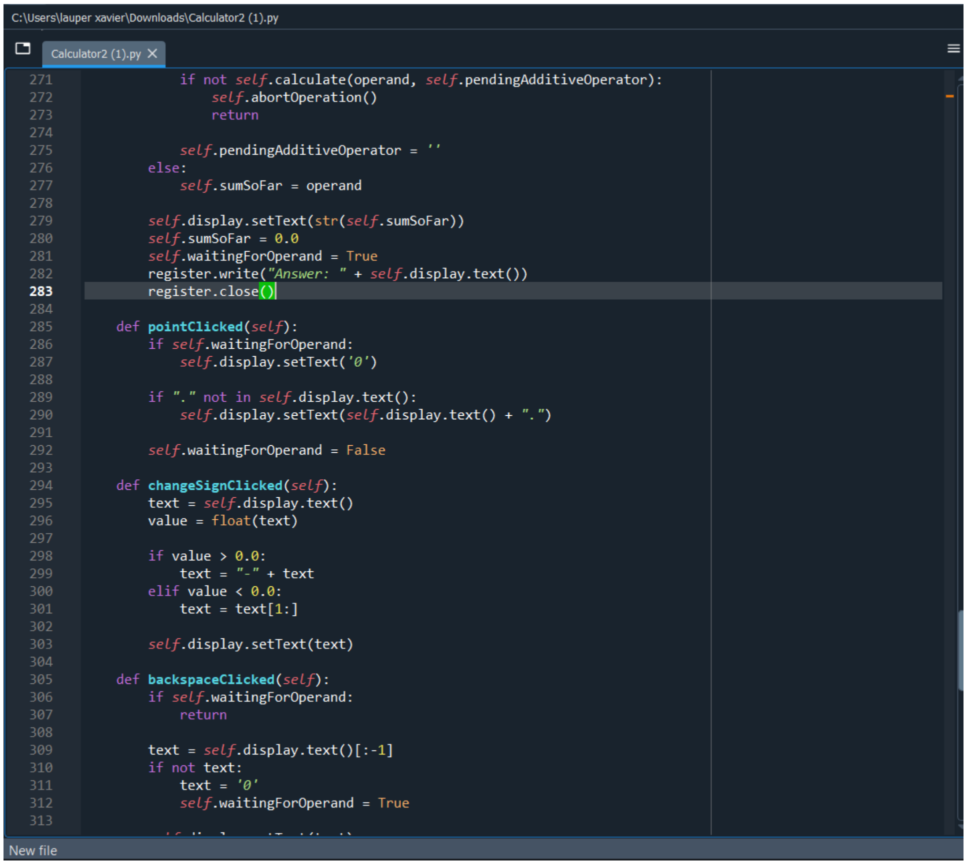


```python

    def setMemory(self):
        self.equalClicked()
        self.sumInMemory = float(self.display.text())

    def addToMemory(self):
        self.equalClicked()
        self.sumInMemory += float(self.display.text())

    def createButton(self, text, member):
        button = Button(text)
        button.clicked.connect(member)
        return button

    def abortOperation(self):
        self.clearAll()
        self.display.setText("####")

    def calculate(self, rightOperand, pendingOperator):
        if pendingOperator == "+":
            self.sumSoFar += rightOperand
        elif pendingOperator == "-":
            self.sumSoFar -= rightOperand
        elif pendingOperator == u"\N{MULTIPLICATION SIGN}":
            self.factorSoFar *= rightOperand
        elif pendingOperator == u"\N{DIVISION SIGN}":
            if rightOperand == 0.0:
                return False

            self.factorSoFar /= rightOperand

        return True


if __name__ == '__main__':

    import sys

    app = QApplication(sys.argv)
    calc = Calculator()
    calc.show()
    sys.exit(app.exec_())
```

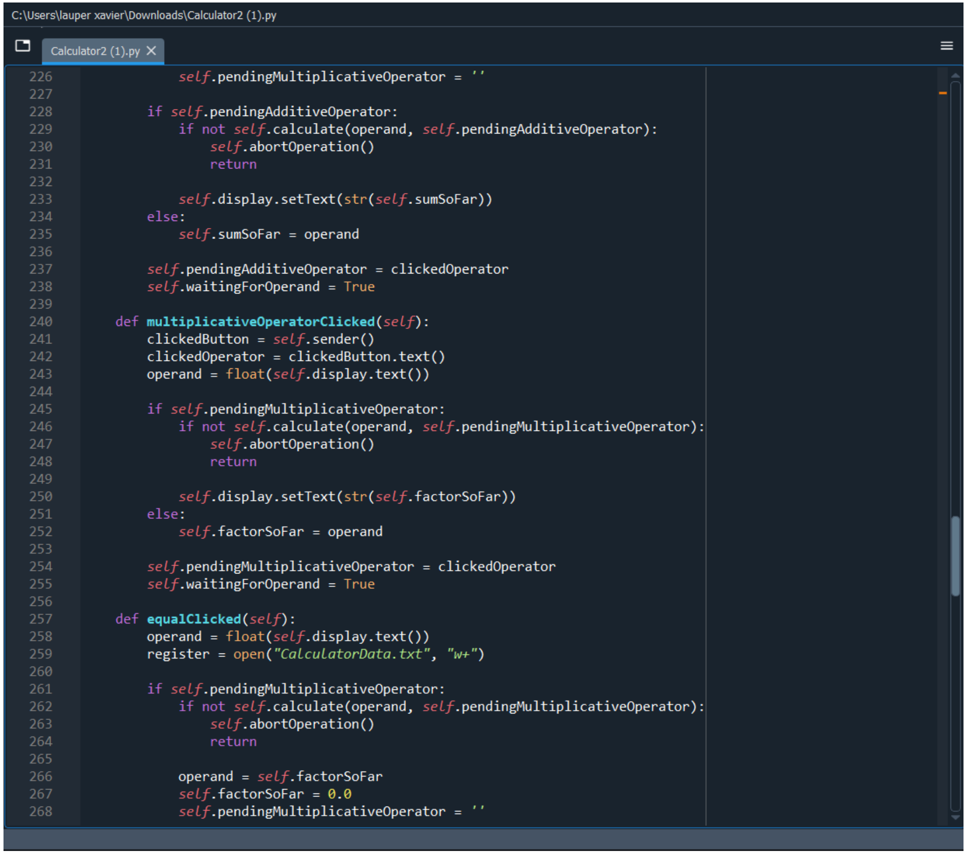


```python

    def clearAll(self):
        self.sumSoFar = 0.0
        self.factorSoFar = 0.0
        self.pendingAdditiveOperator = ''
        self.pendingMultiplicativeOperator = ''
        self.display.setText('0')
        self.waitingForOperand = True

    def clearMemory(self):
        self.sumInMemory = 0.0

    def readMemory(self):
        self.display.setText(str(self.sumInMemory))
        self.waitingForOperand = True

```

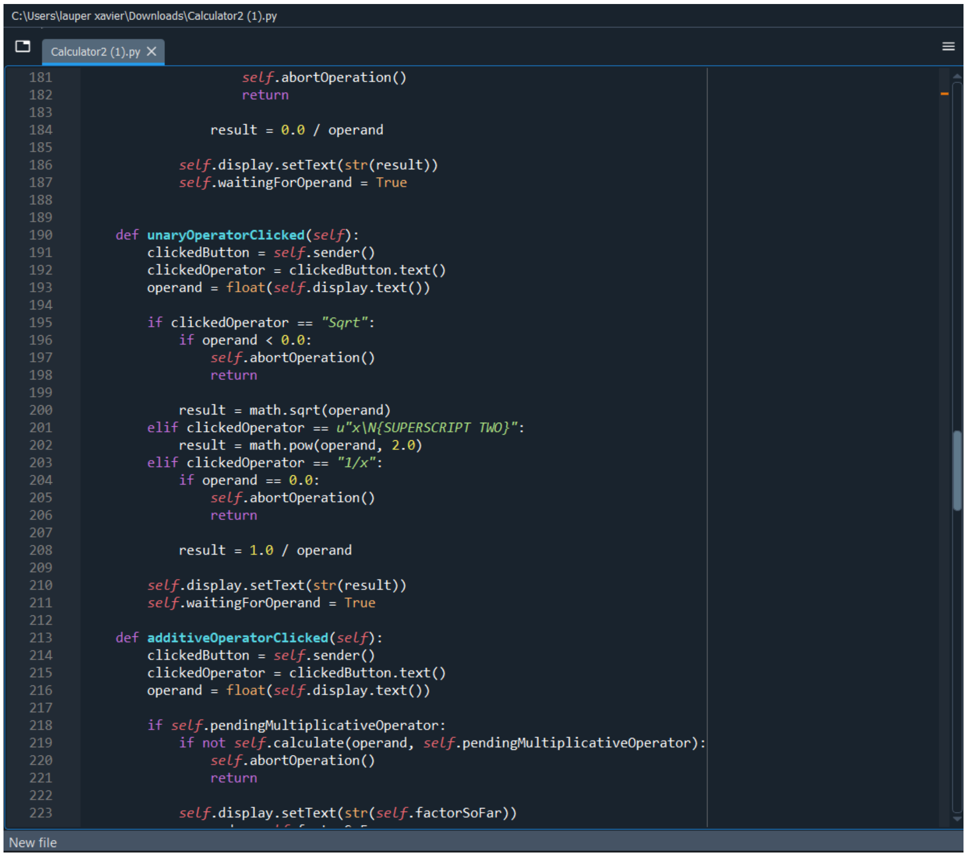
```python
    def setMemory(self):
        self.equalClicked()
        self.sumInMemory = float(self.display.text())

    def addToMemory(self):
        self.equalClicked()
        self.sumInMemory += float(self.display.text())

    def createButton(self, text, member):
        button = Button(text)
        button.clicked.connect(member)
        return button

    def abortOperation(self):
        self.clearAll()
        self.display.setText("####")

    def calculate(self, rightOperand, pendingOperator):
        if pendingOperator == "+":
            self.sumSoFar += rightOperand
        elif pendingOperator == "-":
            self.sumSoFar -= rightOperand
        elif pendingOperator == u"\N{MULTIPLICATION SIGN}":
            self.factorSoFar *= rightOperand
        elif pendingOperator == u"\N{DIVISION SIGN}":
            if rightOperand == 0.0:
                return False
```



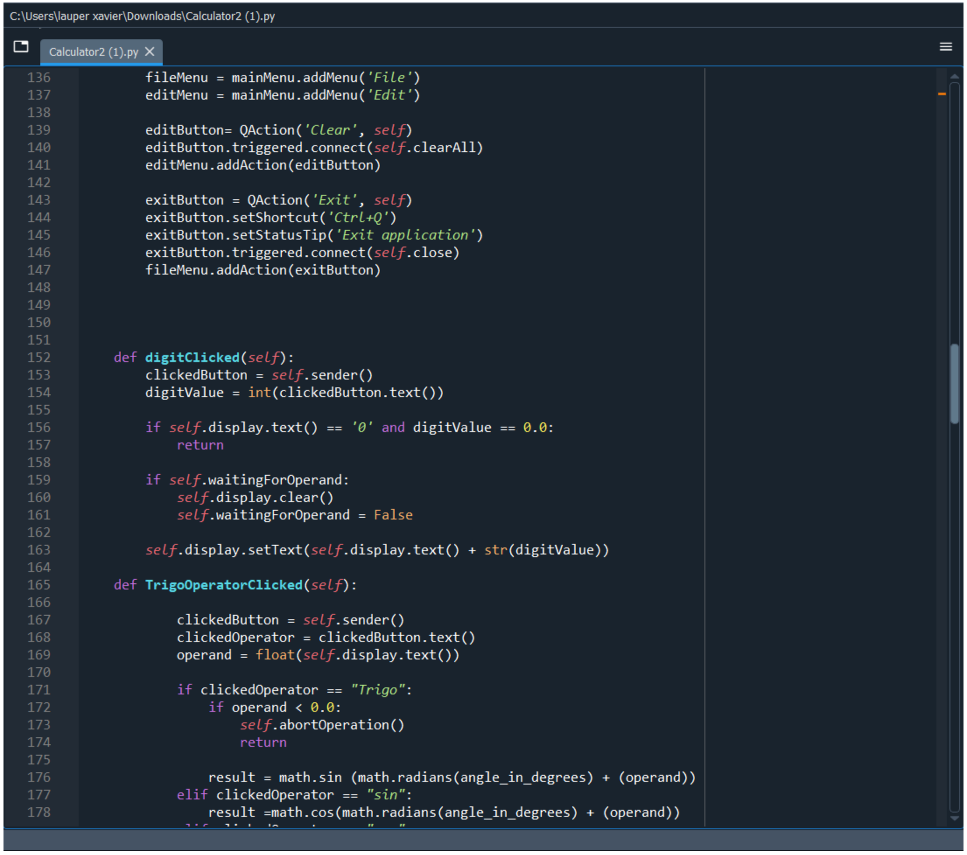
```
            if not self.calculate(operand, self.pendingAdditiveOperator):
                self.abortOperation()
                return

            self.pendingAdditiveOperator = ''
        else:
            self.sumSoFar = operand

        self.display.setText(str(self.sumSoFar))
        self.sumSoFar = 0.0
        self.waitingForOperand = True
        register.write("Answer: " + self.display.text())
        register.close()

    def pointClicked(self):
        if self.waitingForOperand:
            self.display.setText('0')

        if "." not in self.display.text():
            self.display.setText(self.display.text() + ".")

        self.waitingForOperand = False

    def changeSignClicked(self):
        text = self.display.text()
        value = float(text)

        if value > 0.0:
            text = "-" + text
        elif value < 0.0:
            text = text[1:]

        self.display.setText(text)

    def backspaceClicked(self):
        if self.waitingForOperand:
            return

        text = self.display.text()[:-1]
        if not text:
            text = '0'
            self.waitingForOperand = True

        self.display.setText(text)
```



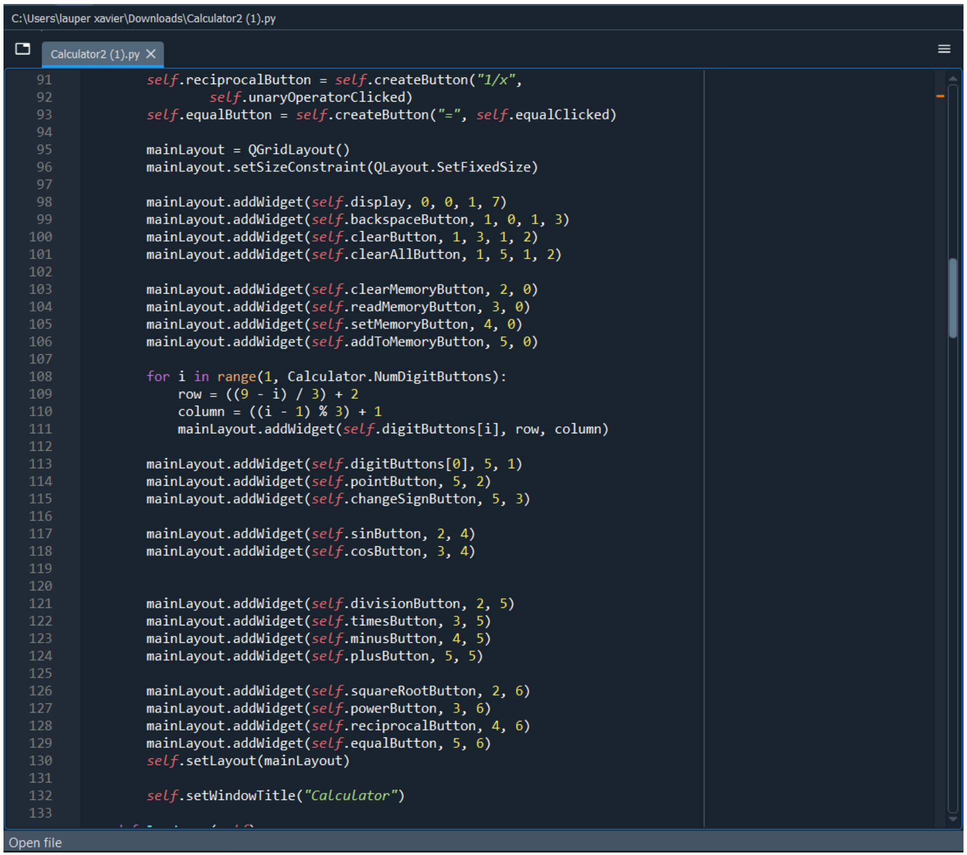
```
            self.pendingMultiplicativeOperator = ''

        if self.pendingAdditiveOperator:
            if not self.calculate(operand, self.pendingAdditiveOperator):
                self.abortOperation()
                return

            self.display.setText(str(self.sumSoFar))
        else:
            self.sumSoFar = operand

        self.pendingAdditiveOperator = clickedOperator
        self.waitingForOperand = True

    def multiplicativeOperatorClicked(self):
        clickedButton = self.sender()
        clickedOperator = clickedButton.text()
        operand = float(self.display.text())

        if self.pendingMultiplicativeOperator:
            if not self.calculate(operand, self.pendingMultiplicativeOperator):
                self.abortOperation()
                return

            self.display.setText(str(self.factorSoFar))
        else:
            self.factorSoFar = operand

        self.pendingMultiplicativeOperator = clickedOperator
        self.waitingForOperand = True

    def equalClicked(self):
        operand = float(self.display.text())
        register = open("CalculatorData.txt", "w+")

        if self.pendingMultiplicativeOperator:
            if not self.calculate(operand, self.pendingMultiplicativeOperator):
                self.abortOperation()
                return

            operand = self.factorSoFar
            self.factorSoFar = 0.0
            self.pendingMultiplicativeOperator = ''
```



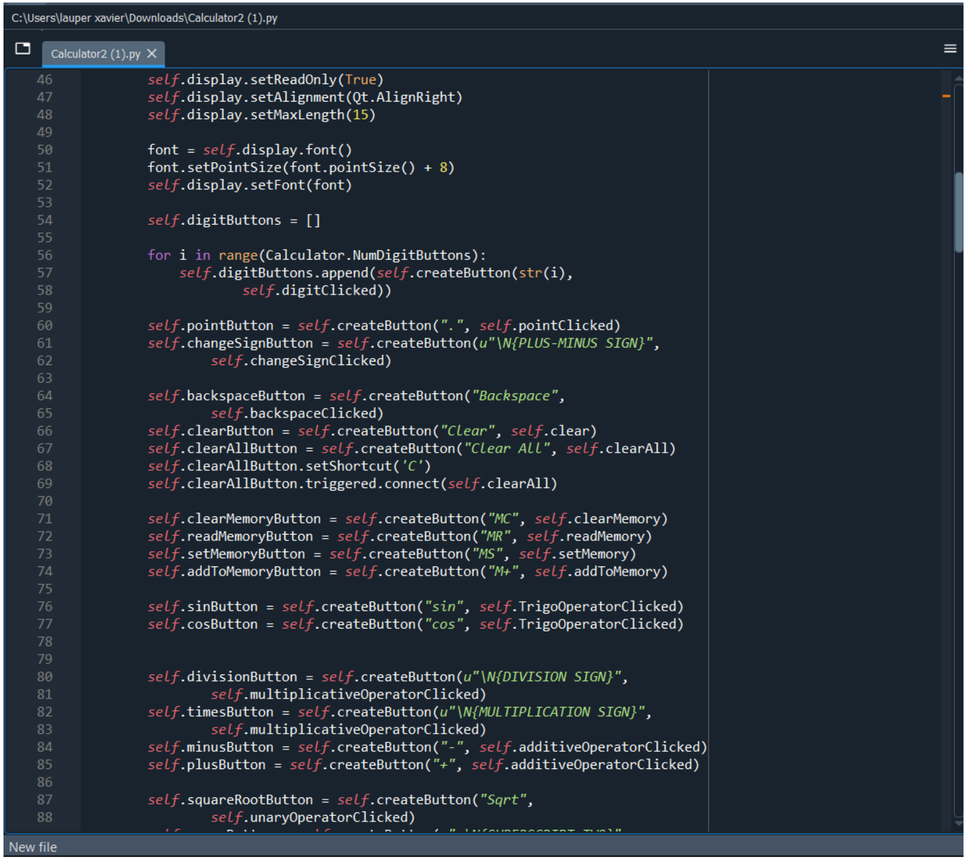
```
                    self.abortOperation()
                    return

                result = 0.0 / operand

            self.display.setText(str(result))
            self.waitingForOperand = True


    def unaryOperatorClicked(self):
        clickedButton = self.sender()
        clickedOperator = clickedButton.text()
        operand = float(self.display.text())

        if clickedOperator == "Sqrt":
            if operand < 0.0:
                self.abortOperation()
                return

            result = math.sqrt(operand)
        elif clickedOperator == u"x\N{SUPERSCRIPT TWO}":
            result = math.pow(operand, 2.0)
        elif clickedOperator == "1/x":
            if operand == 0.0:
                self.abortOperation()
                return

            result = 1.0 / operand

        self.display.setText(str(result))
        self.waitingForOperand = True

    def additiveOperatorClicked(self):
        clickedButton = self.sender()
        clickedOperator = clickedButton.text()
        operand = float(self.display.text())

        if self.pendingMultiplicativeOperator:
            if not self.calculate(operand, self.pendingMultiplicativeOperator):
                self.abortOperation()
                return

            self.display.setText(str(self.factorSoFar))
```



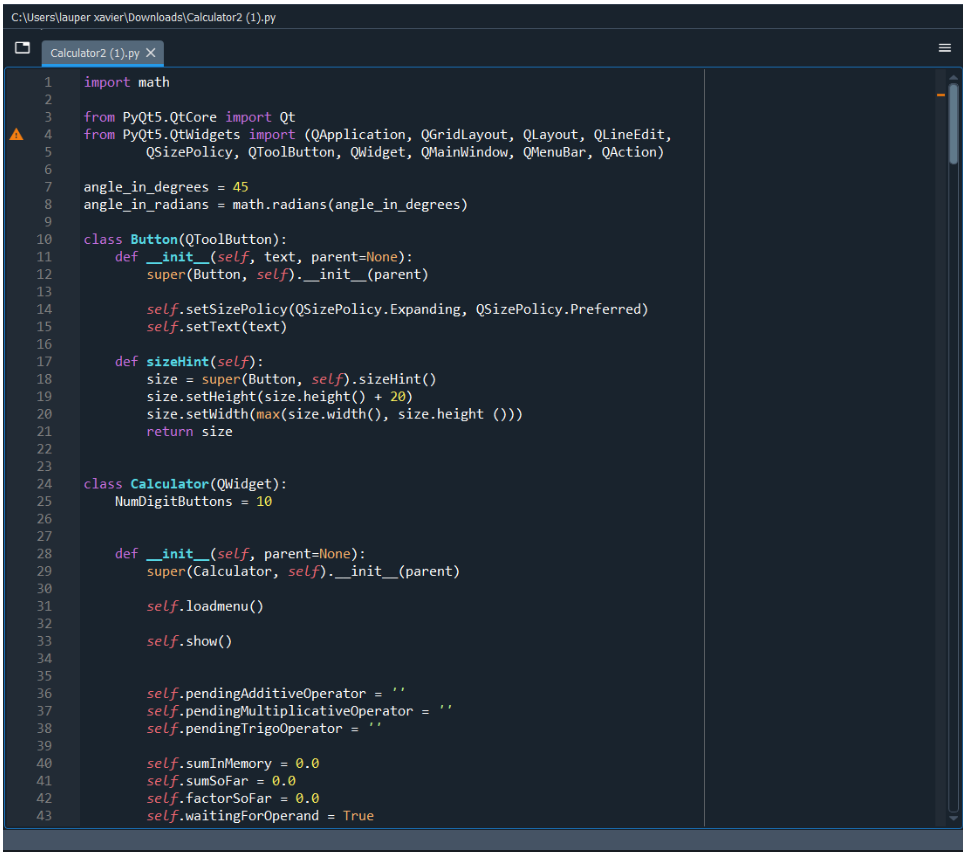
```
        fileMenu = mainMenu.addMenu('File')
        editMenu = mainMenu.addMenu('Edit')

        editButton= QAction('Clear', self)
        editButton.triggered.connect(self.clearAll)
        editMenu.addAction(editButton)

        exitButton = QAction('Exit', self)
        exitButton.setShortcut('Ctrl+Q')
        exitButton.setStatusTip('Exit application')
        exitButton.triggered.connect(self.close)
        fileMenu.addAction(exitButton)




    def digitClicked(self):
        clickedButton = self.sender()
        digitValue = int(clickedButton.text())

        if self.display.text() == '0' and digitValue == 0.0:
            return

        if self.waitingForOperand:
            self.display.clear()
            self.waitingForOperand = False

        self.display.setText(self.display.text() + str(digitValue))

    def TrigoOperatorClicked(self):

            clickedButton = self.sender()
            clickedOperator = clickedButton.text()
            operand = float(self.display.text())

            if clickedOperator == "Trigo":
                if operand < 0.0:
                    self.abortOperation()
                    return

                result = math.sin (math.radians(angle_in_degrees) + (operand))
            elif clickedOperator == "sin":
                result =math.cos(math.radians(angle_in_degrees) + (operand))
```



```
        self.reciprocalButton = self.createButton("1/x",
                self.unaryOperatorClicked)
        self.equalButton = self.createButton("=", self.equalClicked)

        mainLayout = QGridLayout()
        mainLayout.setSizeConstraint(QLayout.SetFixedSize)

        mainLayout.addWidget(self.display, 0, 0, 1, 7)
        mainLayout.addWidget(self.backspaceButton, 1, 0, 1, 3)
        mainLayout.addWidget(self.clearButton, 1, 3, 1, 2)
        mainLayout.addWidget(self.clearAllButton, 1, 5, 1, 2)

        mainLayout.addWidget(self.clearMemoryButton, 2, 0)
        mainLayout.addWidget(self.readMemoryButton, 3, 0)
        mainLayout.addWidget(self.setMemoryButton, 4, 0)
        mainLayout.addWidget(self.addToMemoryButton, 5, 0)

        for i in range(1, Calculator.NumDigitButtons):
            row = ((9 - i) / 3) + 2
            column = ((i - 1) % 3) + 1
            mainLayout.addWidget(self.digitButtons[i], row, column)

        mainLayout.addWidget(self.digitButtons[0], 5, 1)
        mainLayout.addWidget(self.pointButton, 5, 2)
        mainLayout.addWidget(self.changeSignButton, 5, 3)

        mainLayout.addWidget(self.sinButton, 2, 4)
        mainLayout.addWidget(self.cosButton, 3, 4)


        mainLayout.addWidget(self.divisionButton, 2, 5)
        mainLayout.addWidget(self.timesButton, 3, 5)
        mainLayout.addWidget(self.minusButton, 4, 5)
        mainLayout.addWidget(self.plusButton, 5, 5)

        mainLayout.addWidget(self.squareRootButton, 2, 6)
        mainLayout.addWidget(self.powerButton, 3, 6)
        mainLayout.addWidget(self.reciprocalButton, 4, 6)
        mainLayout.addWidget(self.equalButton, 5, 6)
        self.setLayout(mainLayout)

        self.setWindowTitle("Calculator")

```



```
        self.display.setReadOnly(True)
        self.display.setAlignment(Qt.AlignRight)
        self.display.setMaxLength(15)

        font = self.display.font()
        font.setPointSize(font.pointSize() + 8)
        self.display.setFont(font)

        self.digitButtons = []

        for i in range(Calculator.NumDigitButtons):
            self.digitButtons.append(self.createButton(str(i),
                    self.digitClicked))

        self.pointButton = self.createButton(".", self.pointClicked)
        self.changeSignButton = self.createButton(u"\N{PLUS-MINUS SIGN}",
                self.changeSignClicked)

        self.backspaceButton = self.createButton("Backspace",
                self.backspaceClicked)
        self.clearButton = self.createButton("Clear", self.clear)
        self.clearAllButton = self.createButton("Clear All", self.clearAll)
        self.clearAllButton.setShortcut('C')
        self.clearAllButton.triggered.connect(self.clearAll)

        self.clearMemoryButton = self.createButton("MC", self.clearMemory)
        self.readMemoryButton = self.createButton("MR", self.readMemory)
        self.setMemoryButton = self.createButton("MS", self.setMemory)
        self.addToMemoryButton = self.createButton("M+", self.addToMemory)

        self.sinButton = self.createButton("sin", self.TrigoOperatorClicked)
        self.cosButton = self.createButton("cos", self.TrigoOperatorClicked)


        self.divisionButton = self.createButton(u"\N{DIVISION SIGN}",
                self.multiplicativeOperatorClicked)
        self.timesButton = self.createButton(u"\N{MULTIPLICATION SIGN}",
                self.multiplicativeOperatorClicked)
        self.minusButton = self.createButton("-", self.additiveOperatorClicked)
        self.plusButton = self.createButton("+", self.additiveOperatorClicked)

        self.squareRootButton = self.createButton("Sqrt",
                self.unaryOperatorClicked)
```



```
import math

from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import (QApplication, QGridLayout, QLayout, QLineEdit,
        QSizePolicy, QToolButton, QWidget, QMainWindow, QMenuBar, QAction)

angle_in_degrees = 45
angle_in_radians = math.radians(angle_in_degrees)

class Button(QToolButton):
    def __init__(self, text, parent=None):
        super(Button, self).__init__(parent)

        self.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Preferred)
        self.setText(text)

    def sizeHint(self):
        size = super(Button, self).sizeHint()
        size.setHeight(size.height() + 20)
        size.setWidth(max(size.width(), size.height ()))
        return size


class Calculator(QWidget):
    NumDigitButtons = 10


    def __init__(self, parent=None):
        super(Calculator, self).__init__(parent)

        self.loadmenu()

        self.show()


        self.pendingAdditiveOperator = ''
        self.pendingMultiplicativeOperator = ''
        self.pendingTrigoOperator = ''

        self.sumInMemory = 0.0
        self.sumSoFar = 0.0
        self.factorSoFar = 0.0
        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.”