國立虎尾科技大學

計算機程式 ag2 期末報告

PyQt5 事件導向計算器

PyQt5Event-DrivenCalculatorProject

學生：

設計一甲 40623104 曾昱瑄

設計一甲 40623105 陳微云

設計一甲 40623113 吳承哲

設計一甲 40623114 吳信億

設計一甲 40623115 吳隆廷

指導教授：嚴家銘

摘要

機械設計一甲-計算機程式第二組的期末簡報，透過Github網站協同，分工合作完成計算機介面與程式。

* 各學員根據抽籤結果決定處理之按鈕
* 編譯各種按鍵之處理方式
* 期末計算機程式之心得感想

本簡報重點在於練習計算機的邏輯與PyQt5程式的運用以及分工合作之重要性。

ag8計算機程式分配：

40623105 加減運算

40623104 記憶按鍵與其他按鍵

40623113 上一步、清除與等號按鍵

40623114 小數點與變號按鍵

40623115 乘除運算

|  |  |
| --- | --- |
| 摘要 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | P.1 |
| 目錄 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | P.2 |
| 1. 數字按鍵處理 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | P.3 |
| 1-1 數字按鍵程式 |  |
| 1. 加減運算按鍵處理 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | P.4 |
| 2-1 加減運算程式 |  |
| 1. 乘除運算按鍵處理 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | P.5 |
| 3-1 乘除運算程式 |  |
| 1. 小數點與變號按鍵處理 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | P.6 |
| 4-1小數點按鍵程式 |  |
| 4-2變號按鍵程式 |  |
| 1. 上一步、清除與等號按鍵處理 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | P.7 |
| 5-1上一步按鍵程式 |  |
| 5-2清除按鍵程式 |  |
| 5-3全部清除按鍵程式 |  |
| 5-4等號按鍵程式 |  |
| 1. 記憶按鍵與其他按鍵處理. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | P.8 |
| 6-1 記憶體按鍵程式 |  |
| 6-2 其他按鍵程式 |  |
| 1. 期末計算機心得與結論. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | P.9 |

目錄

1. 數字按鍵處理By 全組學員

1-1數字按鍵程式

* 點按數字按鍵, 將會送出該按鍵的訊號
* 儲存按鍵發出的訊號後，判斷視窗上是否有計算中的數字
* 如果無計算中的數字且訊號數字為字串0 則不動作
* 如果無計算中的數字且訊號數字不為字串0則顯示並堆疊顯示所按下之數字

程式如下：

number = [self.one, self.two, self.three, self.four, self.five, self.six, self.seven, self.eight, self.nine, self.zero]

for i in number:

i.clicked.connect(self.digitClicked)

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))

貳、 加減運算按鍵處理By40623105

●使用者按下加或減運算子按鍵時, 程式設定以 additiveOperatorClicked() 處理

進入 additiveOperatorClicked() 後, 必須先查是否有尚未運算的乘或除運算子, 因為必須先乘除後才能加減

●先處理乘與除運算後, 再處理加或減運算後, 將 sumSoFar 顯示在 display 後, 必須重置 sumSoFar 為 0, 表示運算告一段落

self.plus.clicked.connect(self.additiveOperatorClicked)

self.minus.clicked.connect(self.additiveOperatorClicked)

'''加或減按下後進行的處理方法'''

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))

operand = self.factorSoFar

self.factorSoFar = 0.0

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.wait = True

def multiplicativeOperatorClicked(self):

# 若有等待加或減的運算子, 執行運算

if self.pendingAdditiveOperator:

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

self.abortOperation()

return

self.pendingAdditiveOperator = ''

else:

self.sumSoFar = operand

self.display.setText(str(self.temp + self.sumSoFar))

self.sumSoFar = 0.0

self.waitingForOperand = True

def calculate(self, rightOperand, pendingOperator):

if pendingOperator == "+":

self.sumSoFar += rightOperand

elif pendingOperator == "-":

self.sumSoFar -= rightOperand

elif pendingOperator == "\*":

self.factorSoFar \*= rightOperand

elif pendingOperator == "/":

if rightOperand == 0.0:

return False

self.factorSoFar /= rightOperand

return True

def clearAll(self):

參、 乘除運算按鍵處理By40623115

3-1乘除運算程式

* 按下乘或除運算子按鍵時, 程式設定以 multiplicativeOperatorClicked() 處理
* 進入 multiplicativeOperatorClicked() 後, 無需檢查是否有尚未運算的加或減運算子, 因為乘除運算有優先權
* 先處理乘與除運算後, 再處理加或減運算, 將 sumSoFar 顯示在 display 後, 必須重置 sumSoFar 為 0, 表示運算告一段落

程式如下：

for i in multiply\_divide:

clicked.connect(self.multiplicativeOperatorClicked)

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

肆、 小數點與變號按鍵處理By40623114

4-1小數點按鍵程式

* 使用者按下小數點按鍵後, 以 pointClicked() 方法處理, 直接在 display字串中加上 "." 字串數值變號按鍵處理。

程式 :   
 self.pointButton.clicked.connect(self.pointClicked)

 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

4-2變號按鍵程式

* 使用者按下變號按鍵後, 由 changeSignClicked() 處理, 若顯示幕上為正值, 則在 display 字串最前面, 疊上 "-" 字串。
* 假如顯示幕上為負值, 則設法移除 display 上字串最前方的 "-" 字元。

程式 :

self.changeSignButton.clicked.connect(self.changeSignClicked)

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)

伍、上一步、清除與等號按鍵處理By40623113

* AC或C可以清除掉所有的記憶，
* AC   
  All Clear 全部清除鍵   
  按下 AC 按鍵，可將整個運算式清除。   
  範例： 235+882-762 ，按下 AC 按鍵，則整個運算資料接清除   
  C   
  Clear 清除鍵   
  按下 C 按鍵，可清除運算式的最後數據   
  範例： 235+882-762 ，按下 C 按鍵，則清除762這組數據
* = 可以鍵算出最後結果

Equal 等於

按下等於按鍵，可使前面運算得出結果

範例: 235+882-762，按下=按鍵，則會計算出355這組數據結果

def clear(self):

'''清除鍵按下後的處理方法'''

#留著前面的數字

if self.waitingForOperand:

#下面不會執行

return

#清除

self.display.setText('0')

self.waitingForOperand = True

def clearAll(self):

'''全部清除鍵按下後的處理方法'''

#重設預設值

self.sumSoFar = 0.0

self.factorSoFar = 0.0

self.pendingAdditiveOperator = ''

self.pendingMultiplicativeOperator = ''

self.display.setText('0')

self.waitingForOperand = True

def equalClicked(self):

'''等號按下後的處理方法'''

operand = float(self.display.text())

'''

同乘除

'''

if self.pendingMultiplicativeOperator:

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

self.abortOperation()

return

operand = self.factorSoFar

self.factorSoFar = 0.0

self.pendingMultiplicativeOperator = ''

'''

同加減

'''

if self.pendingAdditiveOperator:

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

以上就是寫出這些功能的程式

陸、記憶體按鍵與其他按鍵處理By40623104

6-1 記憶體按鍵程式

功能

* MC : Memory Clean，也就是將目前記憶的數字歸零。
* MR : Memory Recall的意思，將當前計算出來的數字呈現出來。
* MS : 無視目前記憶多少數字，直接以當前數字取代記憶中的數字。
* M+ : 記憶當前的數字，「加入累加數字當中」。

按鍵處理

* clearMemory() 方法與 "MC" 按鍵對應, 清除記憶體中所存 sumInMemory 設為 0
* readMemory() 方法與 "MR" 按鍵對應, 功能為讀取記憶體中的數值, 因此將 sumInMemory 顯示在 display, 作為運算數
* setMemory() 方法則與 "MS" 按鍵對應, 功能為設定記憶體中的數值，因此取 display 中的數字, 存入 sumInMemory
* addToMemory() 方法與 "M+" 按鍵對應, 功能為加上記憶體中的數值, 因此將 sumInMemory 加上 display 中的數值
* 因為 setMemory() 與 addToMemory() 方法, 都需要取用 display 上的數值, 因此必須先呼叫 equalClicked(), 以更新 sumSoFar 與 display 上的數值

導入的模組:

# 清除記憶

        self.clearMemoryButton.clicked.connect(self.clearMemory)

        # 讀取記憶

        self.readMemoryButton.clicked.connect(self.readMemory)

        # 設定記憶

        self.setMemoryButton.clicked.connect(self.setMemory)

# 加入記憶體

        self.addToMemoryButton.clicked.connect(self.addToMemory)

程式:

def clearMemory(self):

        self.sumInMemory = 0.0

    def readMemory(self):

        self.display.setText(str(self.sumInMemory))

        self.waitingForOperand = True

    def setMemory(self):

        self.equalClicked()

        self.sumInMemory = float(self.display.text())

    def addToMemory(self):

        self.equalClicked()

        self.sumInMemory += float(self.display.text())

    def abortOperation(self):

        self.clearAll()

        self.display.setText("####")

6-2其他按鍵程式

功能:

* Sqrt : 開根號
* X^2 : 平方
* 1/x : 倒數

按鍵處理:

* Sqrt, x^2 與 1/x 等按鍵的處理方法為 unaryOperatorClicked(), 與數字按鍵的點按回應相同, 透過 sender().text() 取得按鍵上的 text 字串
* unaryOperatorClicked() 方法隨後根據 text 判定運算子後, 利用 display 上的運算數進行運算後, 再將結果顯示在 display 顯示幕
* 若進行運算 Sqrt 求數值的平方根時, 顯示幕中為負值, 或 1/x 運算時, x 為 0, 都視為無法處理的情況, 以呼叫 abortOperation() 處理
* abortOperation() 方法則重置所有起始變數, 並在 display 中顯示 "####"
* 直接運算子處理結束前, 運算結果會顯示在 display 中, 而且運算至此告一段落, 計算機狀態應該要回復到等待新運算數的階段, 因此 waitingForOperand 要重置為 True

導入的模組:

# 單一運算子

   unaryOperator=[self.squareRootButton,self.powerButton,  self.recip rocalButton ]

   for i in unaryOperator:

       i.clicked.connect(self.unaryOperatorClicked)

程式:

    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 == "X^2":

            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

柒、 期末簡報心得與結論

40623104：在製作計算機的過程中，常常會發生很多錯誤，好險組長和組員以及室友們總是會很善良的幫助我，所以我才可以順利的完成自己分配到的部分。在這次的協同報告中學習到很多，也很感謝大家的幫助。

40623105：我在上傳時，常常遇到很多的錯誤，還好組長總是很熱心地幫助我，讓我可以順利的提交。組員們也都分工合作，使我們可以完成計算機。雖然我自己製作的時候，還是沒有很流暢，但是至少還是完成了，謝謝一路上幫助我的同學跟組長。

40623113：當我第一次上課的時候我就隱約知道阿=…這堂課是場硬仗，每次上課都是帶著緊張得心情去的，深怕當天內容不會，不過好在老師都會錄製當天的影片，使我能夠回家好好複習， 同學們也會互相幫忙，有幾天還熬夜趕工，真是煎熬的一堂課，後來分組後有一個很好的組長才能度過難關，當然TA與組員也是不可忽視的一環，從頭到尾也學到了很多程式語言的東西，原來電腦的世界是這麼廣闊。

40623114：由計算機程式當程式語言的第一步，只能在學習中慢慢了解每一行程式的意義；獨立完成簡直是不可能事，還好有強大的隊長支持著我們兩步兩步地往前走，成功結束這門課程。

40623115：在做完這些計算機按鍵後，我發現程式語言的的世界原來如此的廣大，從Python語言到eric6，了解到網路協同的重要性。