

# Python & GUI

# GUI

為甚麼要GUI, 什麼時候要GUI, 我要怎麼選擇?

PyQt 的 hello world

QT中UI常用的元件介紹

常用的元件的使用教學

PyQt thread

軟體部署 - 如何點兩下啟動程式

課程程式碼下載位置

[https://github.com/JuFengWu/py\\_qt](https://github.com/JuFengWu/py_qt)

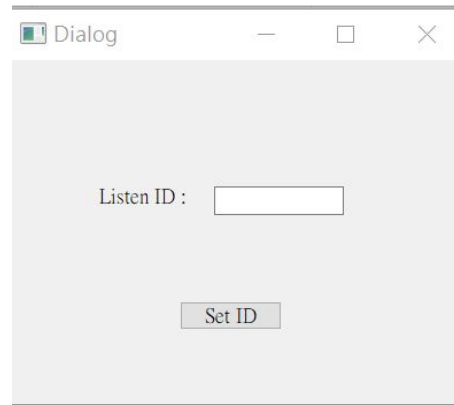
# 為甚麼要GUI

把程式拿給使用者

```
C:\opt>python windows_popup.py aaa  
aaa  
end main
```

使用者: 好醜, 我不會用

使用者會根據美醜來選擇程式  
公司老闆會叫你做GUI



使用者: 我會用

# 什麼時候要開發GUI

對一般使用者 -> 一定要開發GUI

對於會寫程式的人 -> 不一定要開發GUI

那我使用程式時  
要用GUI還是下CMD?

# 補充 - 一個新的開發軟體要學GUI還是CMD?

一些開發軟體ex Git 、 Docker同時有GUI和CMD要學哪一個？

	使用CMD	使用GUI
優點	彈性大	容易看狀態
缺點	要背指令	進版之之後、按鈕位置改變
建議	下指令改變狀態	看現在狀態、關係圖

# GUI的選擇

```
public class Frame extends JFrame{
    private final JLabel label;
    private final JButton button;
    private final JTextField textField;

    public Frame(){
        super("Hello Swing"); //建立標題名稱
        super.setLayout(null);

        label = new JLabel("Enter your name:");
        label.setBounds(140, 50, 100, 30); //設定x·y·寬·高
        add(label);

        button = new JButton("Click me!");
        button.setBounds(125, 200, 150, 25);
        add(button);

        textField = new JTextField(10); //設定10列
        textField.setBounds(140, 100, 120, 30);
        add(textField);
    }
}
```

JAVA Swing

```
class Application(ttk.Frame):
    def __init__(self, master):
        ttk.Frame.__init__(self, master)
        self.pack()

        self.button = ttk.Button(self)
        self.button["text"] = "Click Me!"
        self.button["command"] = self.popup_hello
        self.button.pack()

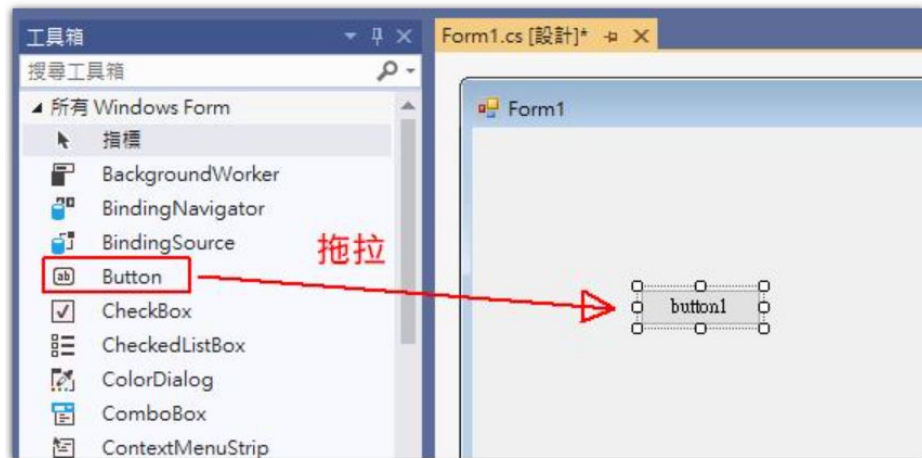
    def popup_hello(self):
        showinfo("Hello", "Hello Tk!")

root = tk.Tk()
app = Application(root)
root.mainloop()
```

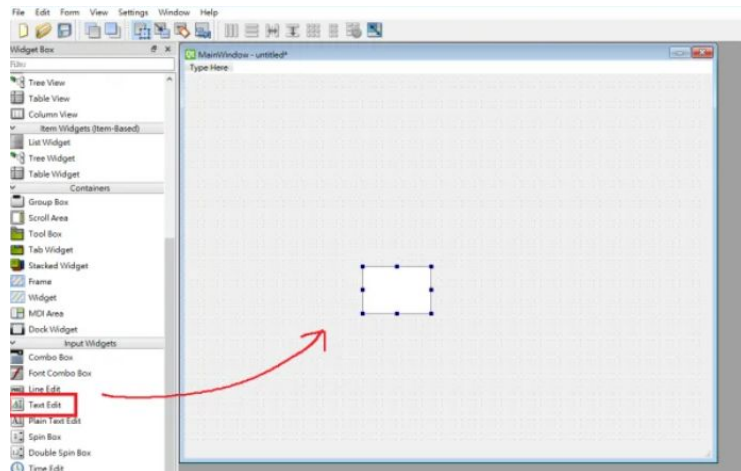
Python tkinter

GUI的設計沒有分開, 很難進行分工和複雜的設計, 不適合大專案

# GUI的選擇



C# winform or C# WPF



Qt

直接拖拉需要的元件  
可以將設計和程式分開

# GUI的選擇

	GUI設計元件分離	GUI設計元件沒有分離
例子	pyqt, pyslide, wxPython	pyqt, pyslide, Tkinter
優點	設計GUI元件容易	程式碼比較簡單
適合	大專案	小專案、實驗性質的程式
業界	常用	不常用

對岸的文章: 為什麼很多 python開發者寫GUI不用Tkinter, 而要選擇PyQt和wxPython或其他?

其他程式語言的選擇  
概念也是如此



# PyQt Hello world



寫了什麼先不管  
先跑成功再說

```
1  import sys
2  from PyQt5.QtWidgets import QApplication, QWidget, QLabel
3  from PyQt5.QtGui import QIcon
4  from PyQt5.QtCore import pyqtSlot
5
6  def window():
7      app = QApplication(sys.argv)
8      widget = QWidget()
9
10     textLabel = QLabel(widget)
11     textLabel.setText("Hello World!")
12     textLabel.move(110,85)
13
14     widget.setGeometry(50,50,320,200)
15     widget.setWindowTitle("PyQt5 Example")
16     widget.show()
17     sys.exit(app.exec_())
18
19 if __name__ == '__main__':
20     window()
```


[https://github.com/JuFengWu/py\\_qt/blob/master/hello\\_world\\_pyqt/hello\\_world\\_py\\_qt.py](https://github.com/JuFengWu/py_qt/blob/master/hello_world_pyqt/hello_world_py_qt.py)

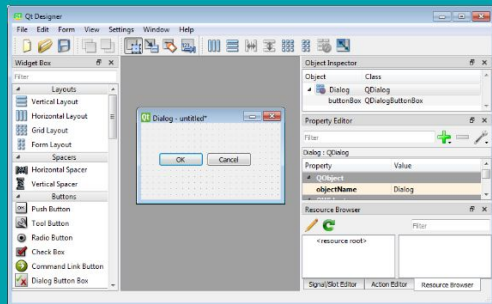
No module named 'PyQt5' -> pip install pyqt5

# 設計第一個UI - Qt designer

下載Qt Designer & 安裝吧

fman build system

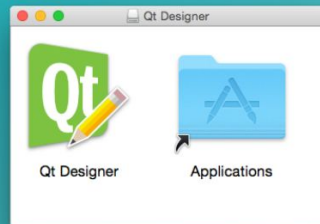
Features Documentation Book Pro 



## Qt Designer Download

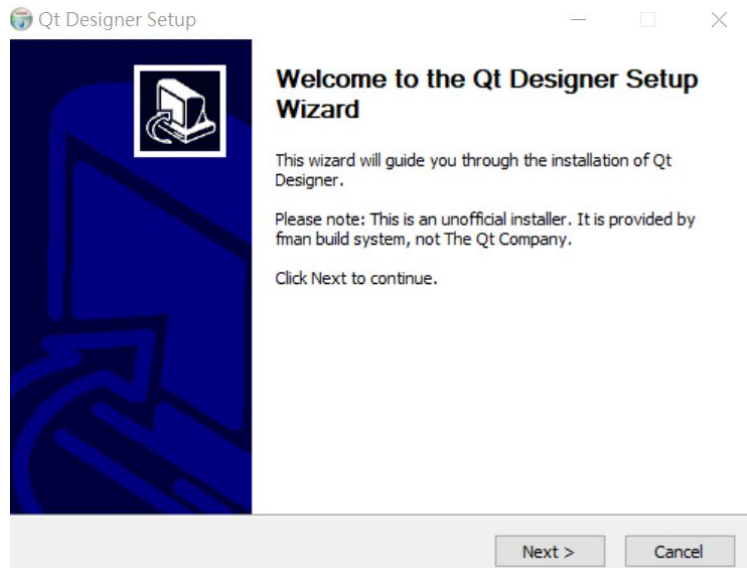
Install Qt Designer on Windows or  
Mac.

Tiny download: Only 40MB!



<https://build-system.fman.io/qt-designer-download>

# 安裝Qt Designer



# 選擇widget

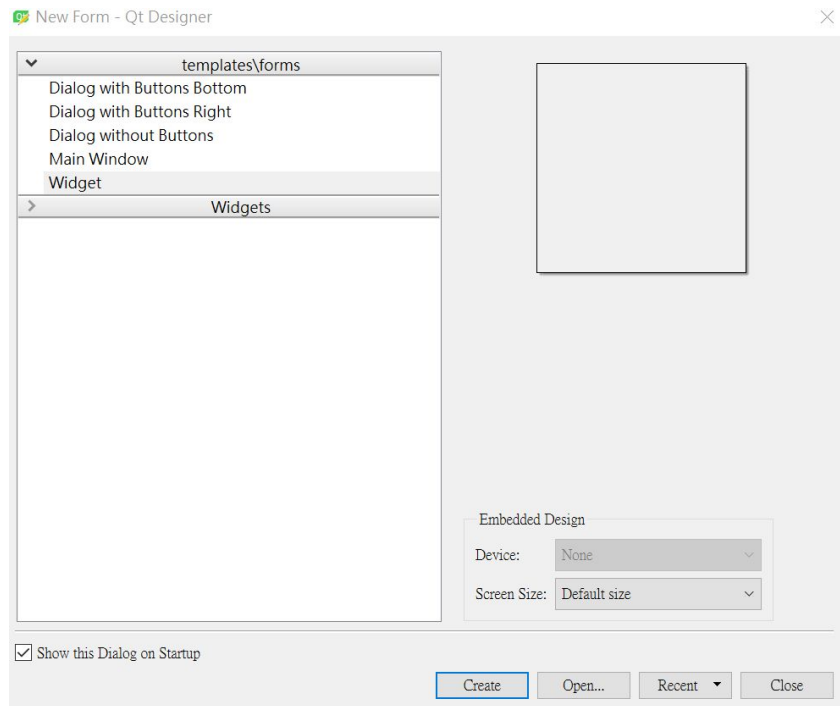
Widget : 所有的視窗

Dialog : 彈出error的那一個視窗

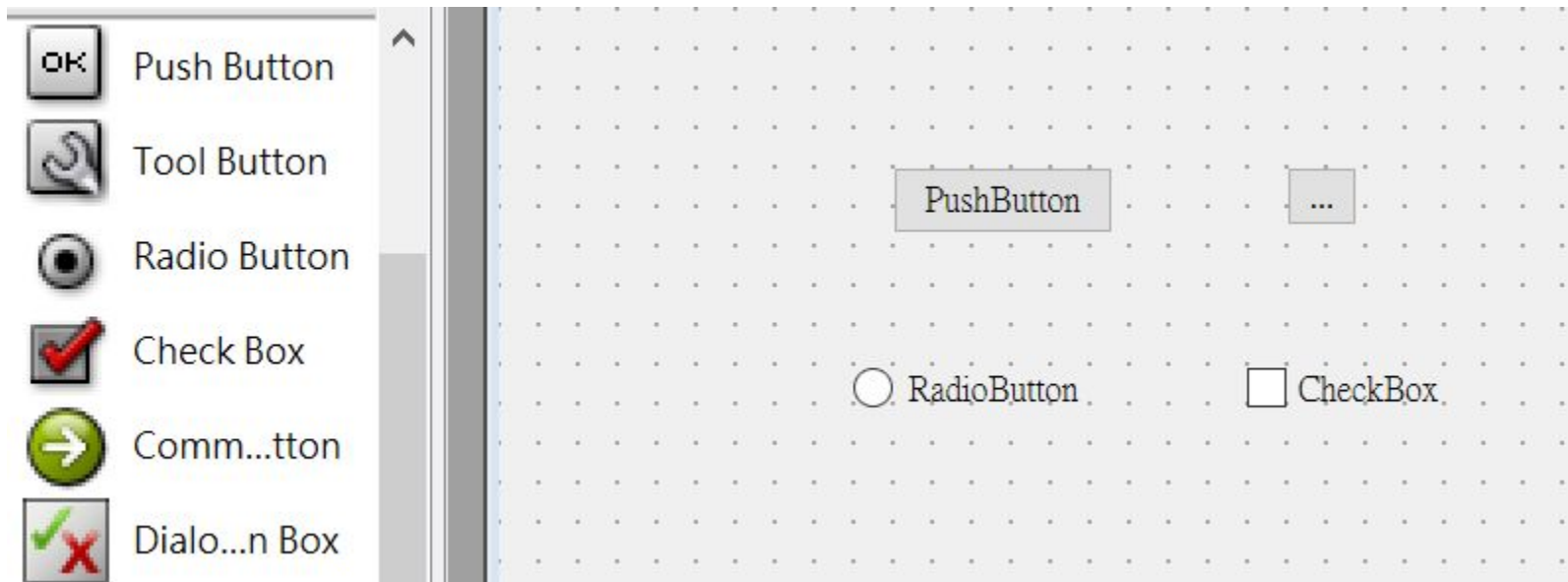
Main window : 主視窗

Main window可以包含很多Widget

我們選擇Dialog



## Qt designer 常用的元件 - 按鈕類



# Qt designer 常用的元件 - 其他類

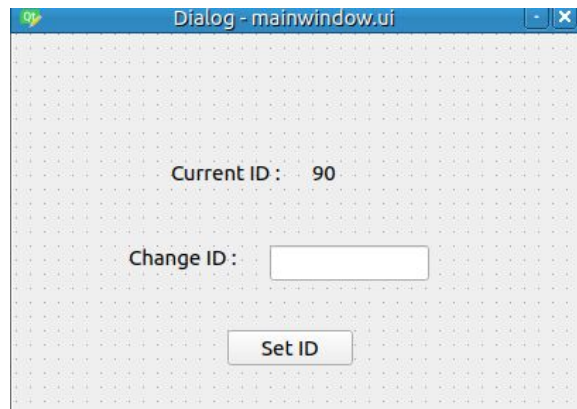
Line editor : 讓使用者輸入文字

TextLabel : 顯示說明



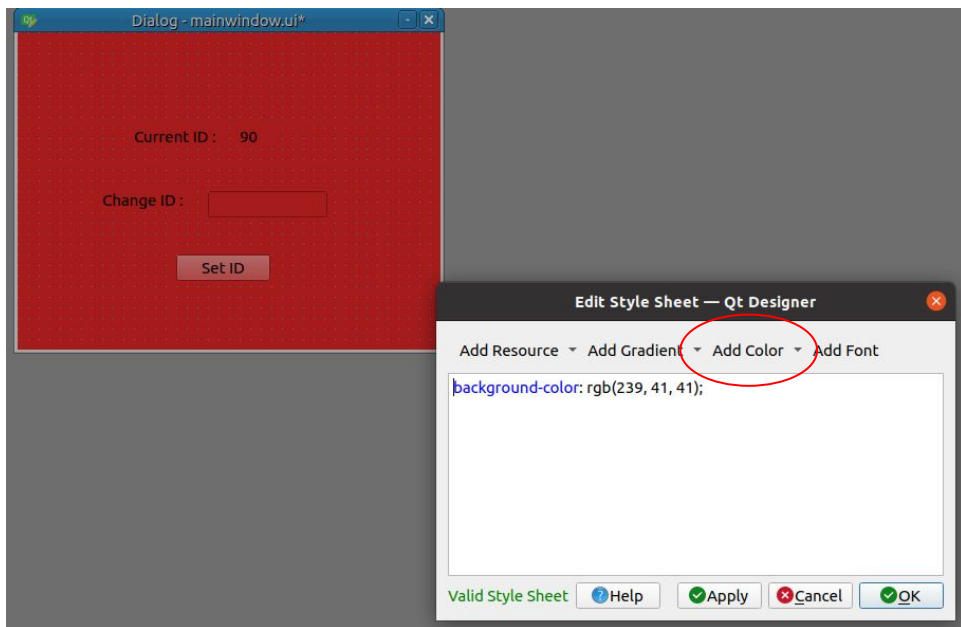
# Qt designer

開始來設計第一個人機界面吧



# 換一下顏色

使用change style sheet換顏色



也可以直接改色碼

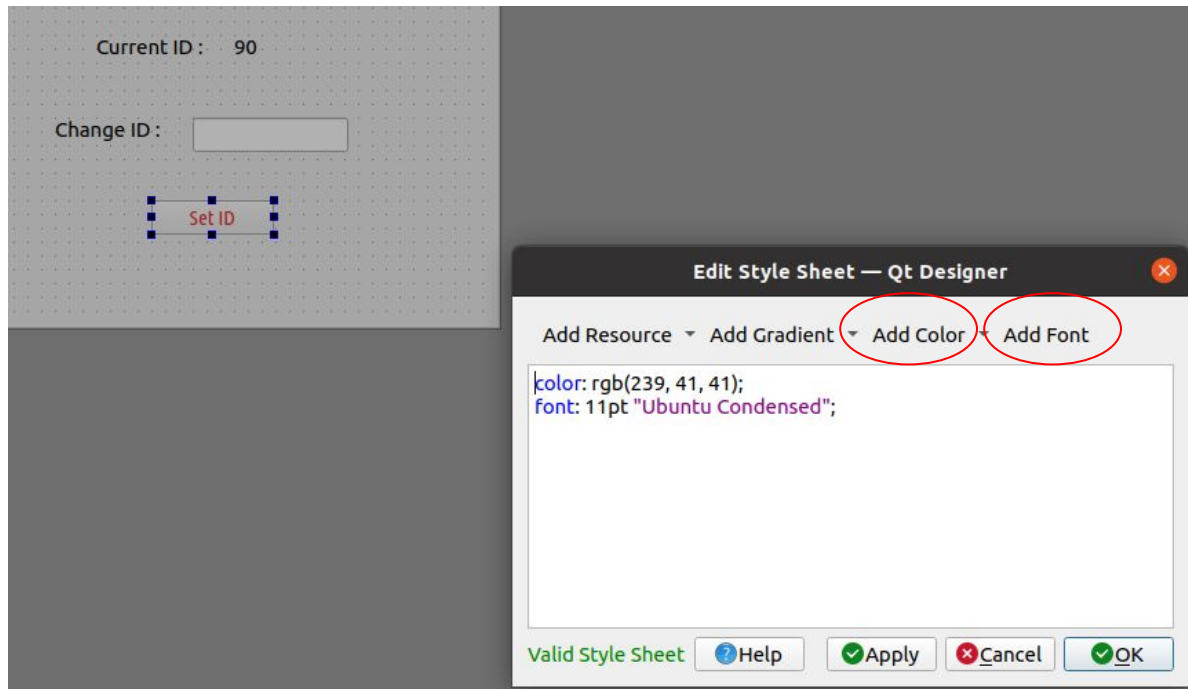


<b>styleSheet</b>	background-color: rgb(239, 41, 41);
▶ locale	English, United States
▶ windowFilePath	
▶ inputMethodHi...	ImhNone
▼ QDialog	

按鈕的顏色也是這樣做

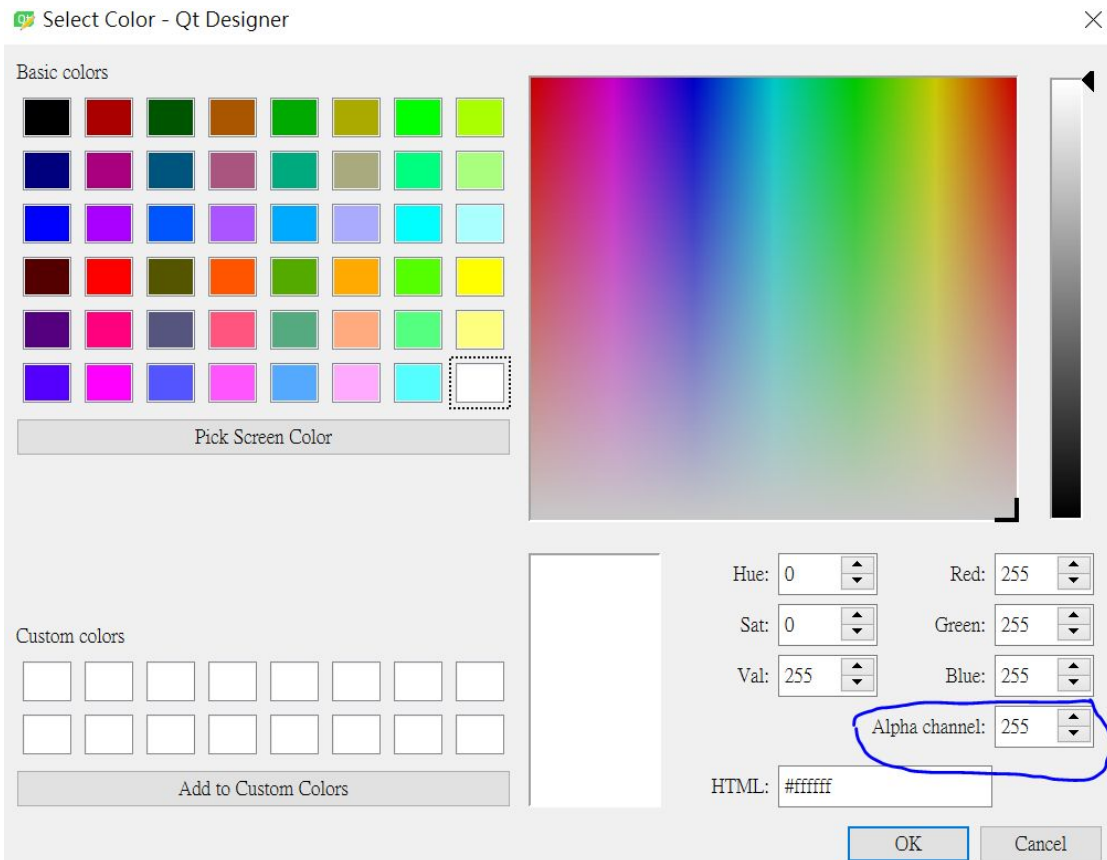


# 改一下字體和文字顏色



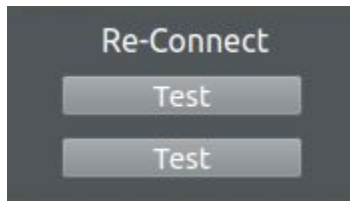
# 如果要透明的話

設定alpha為0



# 實做的時間到了

我們來嘗試畫出這一張圖吧



背景顏色 `rgb(85, 87, 83)`

按鈕背景顏色 `rgb(130, 135, 139)`

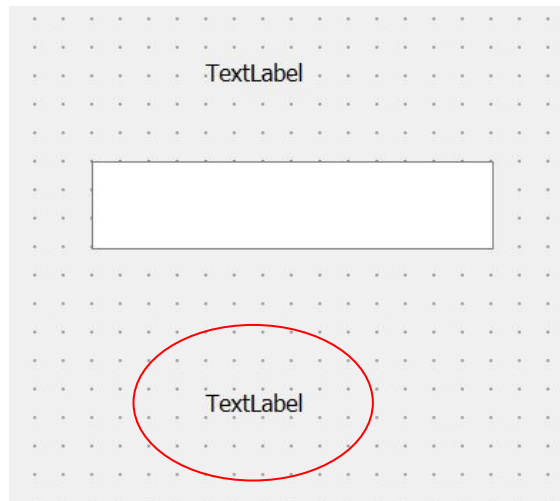
按鈕文字白色 `rgb(255, 255, 255)`

按鈕文字大小 11

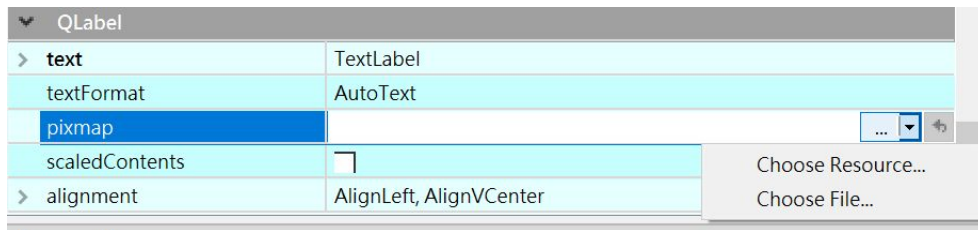
說明文字大小 12

# 如果想要放入圖片？

Step1 放入一個label



Step2 選擇pixmap-> choose file



Step3 設定大小

# 設計好界面，然後呢？

## UI檔案變成python的檔案

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
  <class>Dialog</class>
  <widget class="QDialog" name="Dialog">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>400</width>
        <height>300</height>
      </rect>
    </property>
    <property name="windowTitle">
      <string>Dialog</string>
    </property>
    <widget class="QPushButton" name="setIDButton">
      <property name="geometry">
        <rect>
          <x>150</x>
          <y>210</y>
          <width>89</width>
          <height>25</height>
        </rect>
      </property>
      <property name="text">
        <string>Set ID</string>
      </property>
    </widget>
    <widget class="QLabel" name="showIDLabel">
      <property name="geometry">
```

自動產生

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'mainwindow.ui'
#
# Created by: PyQt5 UI code generator 5.14.1
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(400, 300)
        self.setIDButton = QtWidgets.QPushButton(Dialog)
        self.setIDButton.setGeometry(QtCore.QRect(150, 210, 89, 25))
        self.setIDButton.setObjectName("setIDButton")
        self.showIDLabel = QtWidgets.QLabel(Dialog)
        self.showIDLabel.setGeometry(QtCore.QRect(110, 90, 81, 17))
        self.showIDLabel.setObjectName("showIDLabel")
        self.IDLabel = QtWidgets.QLabel(Dialog)
        self.IDLabel.setGeometry(QtCore.QRect(210, 90, 31, 17))
```

pyuic5 -x mainwindow.ui -o mainwindow.py

pyuic5: command not found 怎麼辦？

# 進入到範例程式碼

怎麼辦，好像很多都看不懂

```
def buttonClick():
    #time.sleep(5) #假設動作很久，睡5秒
    lineEditText = ui.lineEdit.text()
    if(not lineEditText.isnumeric()):
        message_box = QMessageBox()
        message_box.setWindowTitle("error")
        message_box.setInformativeText("please enter a interger number")
        message_box.exec_()
    else:
        ui.label.setText(lineEditText)

def buttonClick2():
    print("xxx")

app = QApplication(sys.argv)
widget = QWidget()
ui = Ui_Form()
ui.setupUi(widget)
ui.pushButton.clicked.connect(buttonClick)
ui.pushButton.clicked.connect(buttonClick2)
widget.show()
sys.exit(app.exec_())
```

沒關係，我們先從重點開始

simple\_test/pyqt\_button

在公司拿到新的程式，就是這樣的感覺

# 進入到範例程式碼

首先是按下按鈕

```
def buttonClick(self):  
    #time.sleep(5) #假設動作很久，睡5秒  
    IDlineEditText = ui.IDlineEdit.text()  
    if(not IDlineEditText.isnumeric()):  
        message_box = QMessageBox()  
        message_box.setWindowTitle ("error")  
        message_box.setInformativeText("please enter a interger number")  
        message_box.exec_()  
    else:  
        ui.IDlabel.setText(IDlineEditText)
```

按下按鈕的反應

```
app = QApplication(sys.argv)  
widget = QWidget()  
ui = Ui_Dialog()  
ui.setupUi(widget)  
setIDButton = ui.setIDButton  
setIDButton.clicked.connect(buttonClick)  
widget.show()  
sys.exit(app.exec_())
```

把按鈕和method連結

# 進入到範例程式碼(進階)

怎麼辦，好像很多都看不懂

```
from PyQt5 import QtWidgets
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
import sys
from mainwindow import Ui_Dialog

class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)
        self.ui_connection()

    def ui_connection(self):
        self.setIDButton = self.ui.setIDButton
        self.setIDButton.clicked.connect(self.buttonClick)

    def buttonClick(self):
        IDlineEditText = self.ui.IDlineEdit.text()
        if(not IDlineEditText.isnumeric()):
            message_box = QMessageBox()
            message_box.setWindowTitle("error")
            message_box.setInformativeText("please enter a interger number")
            message_box.exec_()
        else:
            self.ui.IDlabel.setText(IDlineEditText)
```

沒關係，我們先從重點開始

simple\_test/pyqt\_button

在公司拿到新的程式，就是這樣的感覺



# 範例程式碼(進階)

首先是按下按鈕

```
from PyQt5 import QtWidgets
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
import sys
from mainwindow import Ui_Dialog
```

```
class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)
        self.ui_connection()
```

```
def ui_connection(self):
    self.setIDButton = self.ui.setIDButton
    self.setIDButton.clicked.connect(self.buttonClick)
```

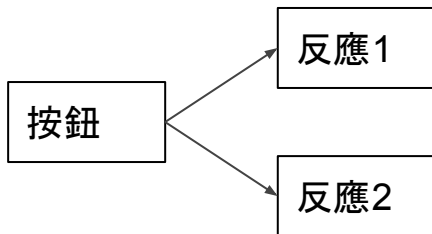
把按鈕和method連結

```
def buttonClick(self):
    IDlineEditText = self.ui.IDlineEdit.text()
    if(not IDlineEditText.isnumeric()):
        message_box = QMessageBox()
        message_box.setWindowTitle("error")
        message_box.setInformativeText("please enter a interger number")
        message_box.exec_()
    else:
        self.ui.IDlabel.setText(IDlineEditText)
```

按下按鈕的反應

# 一個按鈕多個反應

這是標準的observer pattern



```
def buttonClick(self):
    #time.sleep(5) #假設動作很久，睡5秒
    IDlineEditText = ui.IDlineEdit.text()
    if(not IDlineEditText.isnumeric()):
        message_box = QMessageBox()
        message_box.setWindowTitle ("error")
        message_box.setInformativeText("please enter a interger number")
        message_box.exec_()
    else:
        ui.IDlabel.setText(IDlineEditText)
def buttonClick2(self):
    print("xxx")

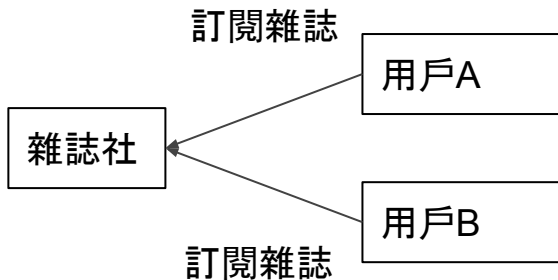
app = QApplication(sys.argv)
widget = QWidget()
ui = Ui_Dialog()
ui.setupUi(widget)
setIDButton = ui.setIDButton
setIDButton.clicked.connect(buttonClick)
setIDButton.clicked.connect(buttonClick2)
widget.show()
sys.exit(app.exec_())
```

# 程式架構 observer pattern概念

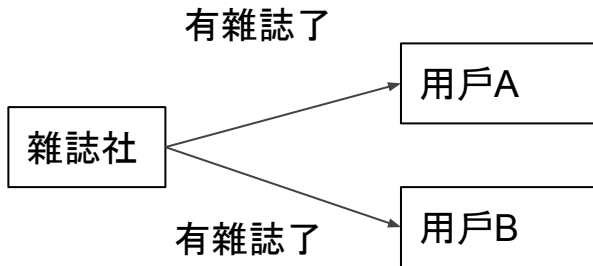
步驟1 A訂閱B

步驟2 當B做事的時候, 會通知A

步驟1



步驟2

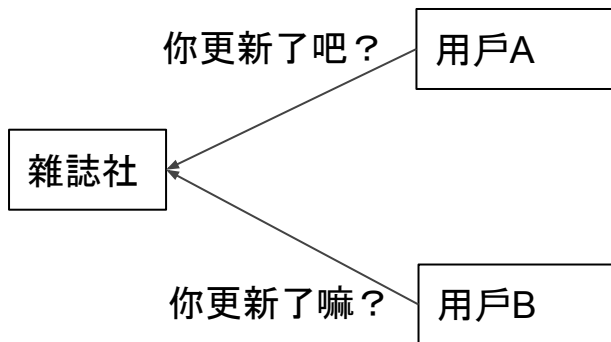


這樣有什麼好處？

就像訂閱雜誌

# 如果沒有Observer的概念

用Observer省資源, 以及加速

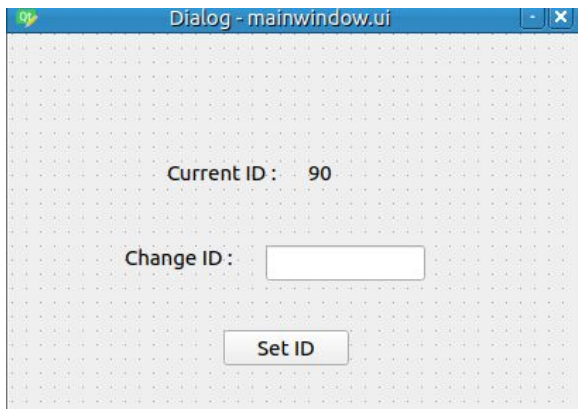


不用Observer的話

1. 要一直花資源去問
2. 雜誌社要花時間去處理詢問(DDOS)
3. 更新的時候, 沒辦法第一時間知道

# 按下的動作

1. 從lineEdit取資料
2. 判斷是不是數字
3. 不是數字，製作Message box
4. 數字的話，修改ID



```
def buttonClick(self):  
    #time.sleep(5) #假設動作很久，睡5秒  
    IDlineEditText = ui.IDlineEdit.text()  
    if(not IDlineEditText.isnumeric()):  
        message_box = QMessageBox()  
        message_box.setWindowTitle ("error")  
        message_box.setInformativeText("please enter a interger number")  
        message_box.exec_()  
    else:  
        ui.IDlabel.setText(IDlineEditText)
```

# LineEditor vs TextEditor

	Line Editor	Text Editor
行數	1行	多行
取得文字	text()	toPlainText()

```
def buttonClick():  
    ui.lineEdit.setText("hello\naaa")  
    print(ui.lineEdit.text())  
def buttonClick2():  
    ui.textEdit.setText("aaa\nbbb")  
    print(ui.textEdit.toPlainText())
```

hello aaa

aaa  
bbb

## 按下圖片的反應

在實務上，為了滿足設計師的需求，通常是設定按下圖片，而非button

---

```
mousePressEvent(QMouseEvent *event)
```

---

```
mouseReleaseEvent(QMouseEvent *event)
```

---

當滑鼠按下去or滑鼠放開時進行動作

# 按下圖片的反應

將mousePressEvent註冊buttonClick的動作

```
def buttonClick(event):  
    lineEditText = ui.lineEdit.text()  
    if(not lineEditText.isnumeric()):  
        message_box = QMessageBox()  
        message_box.setWindowTitle ("error")  
        message_box.setInformativeText("please enter a interger number")  
        message_box.exec_()  
    else:  
        ui.label.setText(lineEditText)  
        print(str(event.x()) + "," + str(event.y()) + "," + str(event.button()))  
  
app = QApplication(sys.argv)  
widget = QWidget()  
ui = Ui_Form()  
ui.setupUi(widget)  
  
ui.label_2.mousePressEvent = buttonClick
```



# 按下圖片的反應

因為只能左鍵按下，才能有反應

這部分可以藉由實驗得知其數值，或是上網查

Qt::LeftButton	0x00000001	The left button is pressed, or an event refers to the left button. (The left button may be the right button on left-handed mice.)
Qt::RightButton	0x00000002	The right button.
Qt::MidButton	0x00000004	The middle button.

[http://man.hubwiz.com/docset/Qt\\_5.docset/Contents/Resources/Documents/doc.qt.io/qt-5/qt.html#MouseButton-enum](http://man.hubwiz.com/docset/Qt_5.docset/Contents/Resources/Documents/doc.qt.io/qt-5/qt.html#MouseButton-enum)

小練習：設定按下左鍵才有反應

# 動手的時間到了

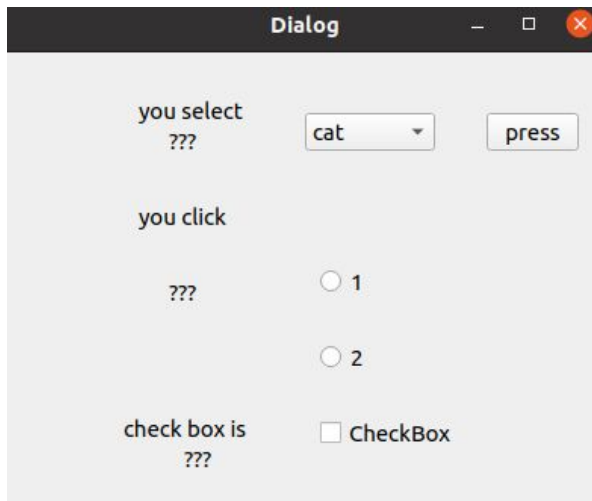
輸入兩個數字，按下+ or -  
顯示相加or相減的數字

Ans  
???

Value 1      Value 2

# Radio Button和Check button

1. radio 設定互斥
2. radio box和check box點下的時候要有反應



radio\_button/pyqt\_test.py

```
def click_checkbox():
    if(self.checkBox.isChecked()):
        self.ui.label_4.setText("check")
    else:
        self.ui.label_4.setText("not check")

def click_radio_button(self):
    if(self.buttonGroup.checkedId() == 1):
        self.ui.label_2.setText("click 1")
    elif(self.buttonGroup.checkedId() == 2):
        self.ui.label_2.setText("click 2")
    else:
        print("error occur")

def click_combo_button():
    ui.label_6.setText(ui.comboBox.currentText())

app = QApplication(sys.argv)
widget = QWidget()
ui = Ui_Dialog()
ui.setupUi(widget)

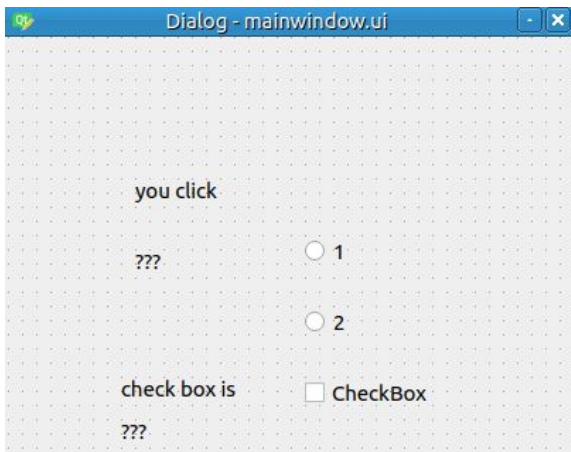
checkBox = ui.checkBox
checkBox.clicked.connect(click_checkbox)

buttonGroup = QButtonGroup(widget)
buttonGroup.addButton(ui.radioButton_1)
buttonGroup.addButton(ui.radioButton_2)
ui.radioButton.clicked.connect(click_radio_button)
ui.radioButton_2.clicked.connect(click_radio_button)

ui.comboBox.addItem('cat')
ui.comboBox.addItem('dog')
ui.comboBox.addItem('bird')
ui.pushButton.clicked.connect(click_combo_button)
widget.show()
sys.exit(app.exec_())
```

# Radio Button和Check button(進階)

1. radio 設定互斥
2. radio box和check box點下的時候要有反應



radio\_button/pyqt\_radio\_button.py

```
def ui_connection(self):
    self.checkBox = self.ui.checkBox
    self.checkBox.clicked.connect(self.click_checkbox)
    self.buttonGroup = QButtonGroup(self)
    self.buttonGroup.addButton(self.ui.radioButton, 1)
    self.buttonGroup.addButton(self.ui.radioButton_2, 2)
    self.ui.radioButton.clicked.connect(self.click_radio_button)
    self.ui.radioButton_2.clicked.connect(self.click_radio_button)

def click_checkbox(self):
    if(self.checkBox.isChecked()):
        self.ui.label_4.setText("check")
    else:
        self.ui.label_4.setText("not check")

def click_radio_button(self):
    if(self.buttonGroup.checkedId() == 1):
        self.ui.label_2.setText("click 1")
    elif(self.buttonGroup.checkedId() == 2):
        self.ui.label_2.setText("click 2")
    else:
        print("error occurt")
```

# 動手的时间到了

把+號和-號改成radio button

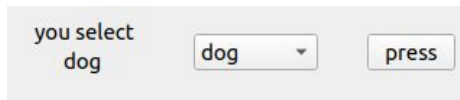
Ans  
???

Value 1   Value 2

☐ +   ☐ -

# 下拉選單 combobox

下拉選單也是很常用的一個工具



```
ui.comboBox.addItem(['cat', 'dog', 'bird'])  
ui.pushButton.clicked.connect(click_combo_button)
```

# 動手的時間到了

增加一個combo box  
讓別人選擇要value1-value2  
還是value2-value1

Ans  
???

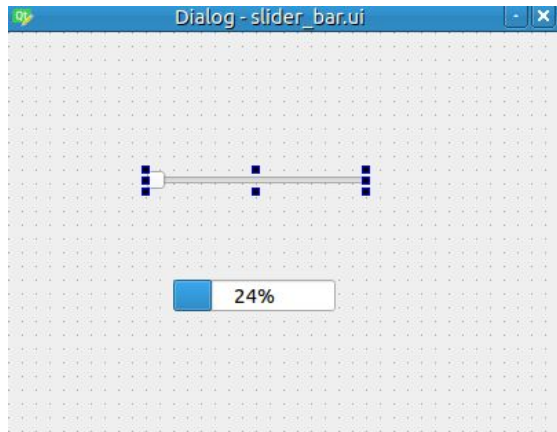
Value 1 Value 2

=

+ -

# pyqt進度條

## 拉拉桿改變進度條



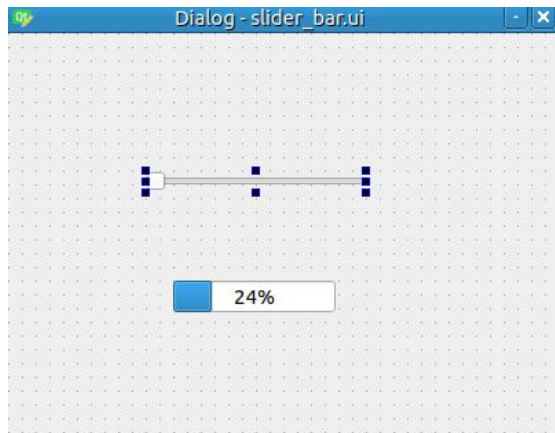
```
def value_change():
    ui.progressBar.setValue(ui.horizontalSlider.value())
def show_release():
    message_box = QMessageBox()
    message_box.setWindowTitle("final slidbar value")
    message_box.setInformativeText(str(ui.horizontalSlider.value()))
    message_box.exec_()

app = QApplication(sys.argv)
widget = QWidget()
ui = Ui_Dialog()
ui.setupUi(widget)
ui.progressBar.setMinimum(0)
ui.progressBar.setMaximum(100)
ui.progressBar.setValue(65)
ui.horizontalSlider.setMaximum(120)
ui.horizontalSlider.valueChanged.connect(value_change)
ui.horizontalSlider.sliderReleased.connect(show_release)
widget.show()
sys.exit(app.exec_())
```



# pyqt進度條(進階)

拉拉桿改變進度條



```
def ui_connection(self):
    self.ui.progressBar.setMinimum(0)
    self.ui.progressBar.setMaximum(100)
    self.ui.progressBar.setValue(65)
    self.ui.horizontalSlider.setMaximum(120)
    self.ui.horizontalSlider.valueChanged.connect(self.value_change)
    self.ui.horizontalSlider.sliderReleased.connect(self.show_release)

def value_change(self):
    self.ui.progressBar.setValue(self.ui.horizontalSlider.value())

def show_release(self):
    message_box = QMessageBox()
    message_box.setWindowTitle("final slidbar value")
    message_box.setInformativeText(str(self.ui.horizontalSlider.value()))
    message_box.exec_()
```

## 動手的时间到了

## 加一個拉罷表示百分比

## 加或是減的數字會乘上百分比

Ans

???

percentage      Value 1      Value 2

☐                  

☐ +      ☐ -

# 設定資料夾和文件

folder

get folder

file

get file

## 設定資料夾和文件

```
def buttonClick():  
    outputFolderName = QtWidgets.QFileDialog.getExistingDirectory()  
    print(outputFolderName)  
    ui.lineEdit.setText(outputFolderName)  
  
def buttonClick2():  
    #filename , _ = QtWidgets.QFileDialog.getOpenFileNames(filter='py (*.py)')  
    filename , _ = QtWidgets.QFileDialog.getOpenFileNames()  
    ui.lineEdit_2.setText(filename[0])
```

[https://github.com/JuFengWu/py\\_qt/blob/master/folder\\_and\\_file/file\\_folder.py](https://github.com/JuFengWu/py_qt/blob/master/folder_and_file/file_folder.py)

# 檔案讀取和分類

## 檔案讀取的方式

```
path = 'target.txt'
f = open(path, 'r')
for line in f.readlines():
    print(line)
    splitValue = line.split(",")
    for single in splitValue:
        print(single)
    print("end split")
f.close()
```

[https://github.com/JuFengWu/py\\_qt/blob/master/folder\\_and\\_file/read\\_file.py](https://github.com/JuFengWu/py_qt/blob/master/folder_and_file/read_file.py)

# 小試身手(1)

將資料夾所有的file顯示在combo box上

folder

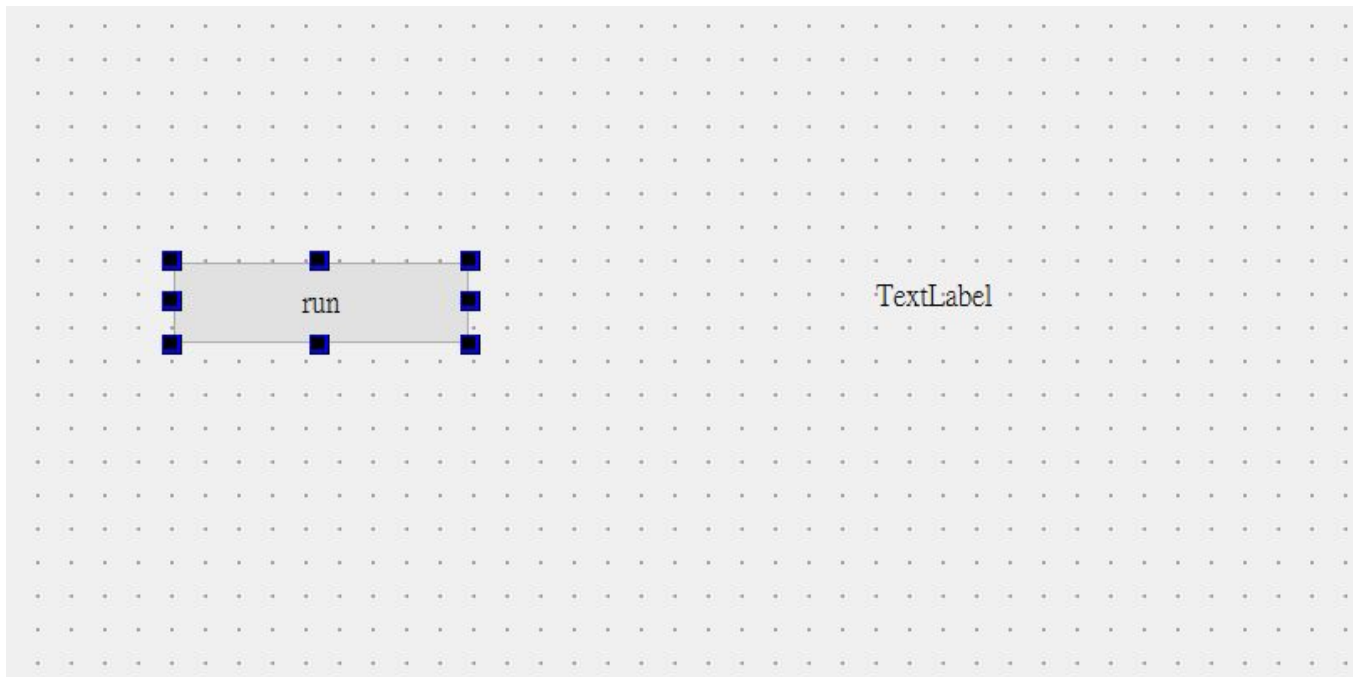
folder files

get folder

run

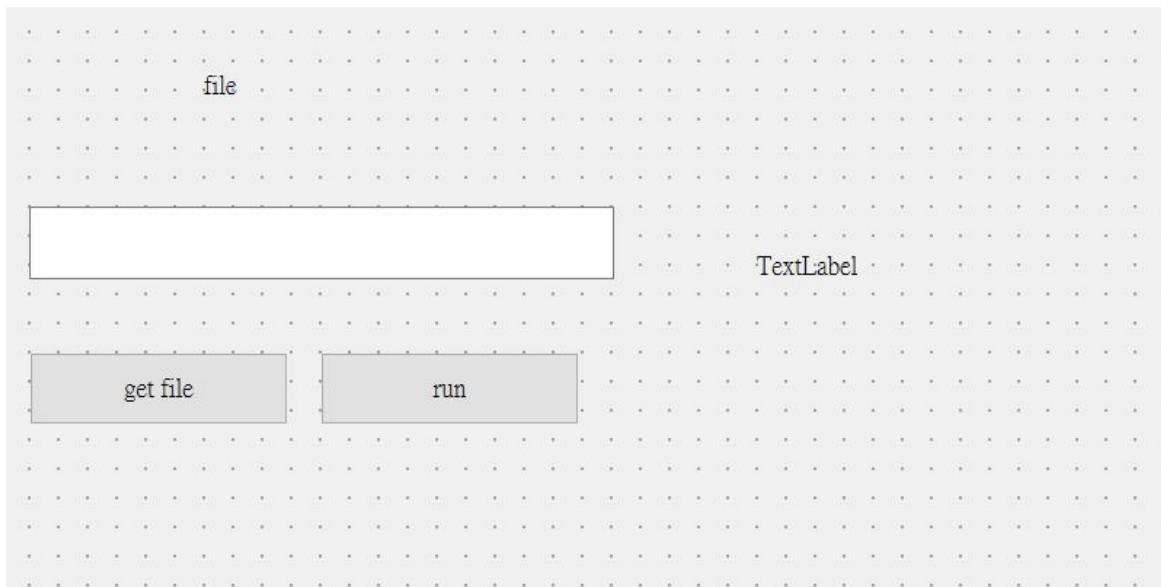
## 小試身手(2)-1

按下按鈕讀取檔案，將內容顯示在label上



## 小試身手(2)-2

將文字檔案讀取，顯示在text label上



你好  
這是pyqt



## 小試身手(2)-進階

利用split, 將所有數字依順序放入combobox內

file

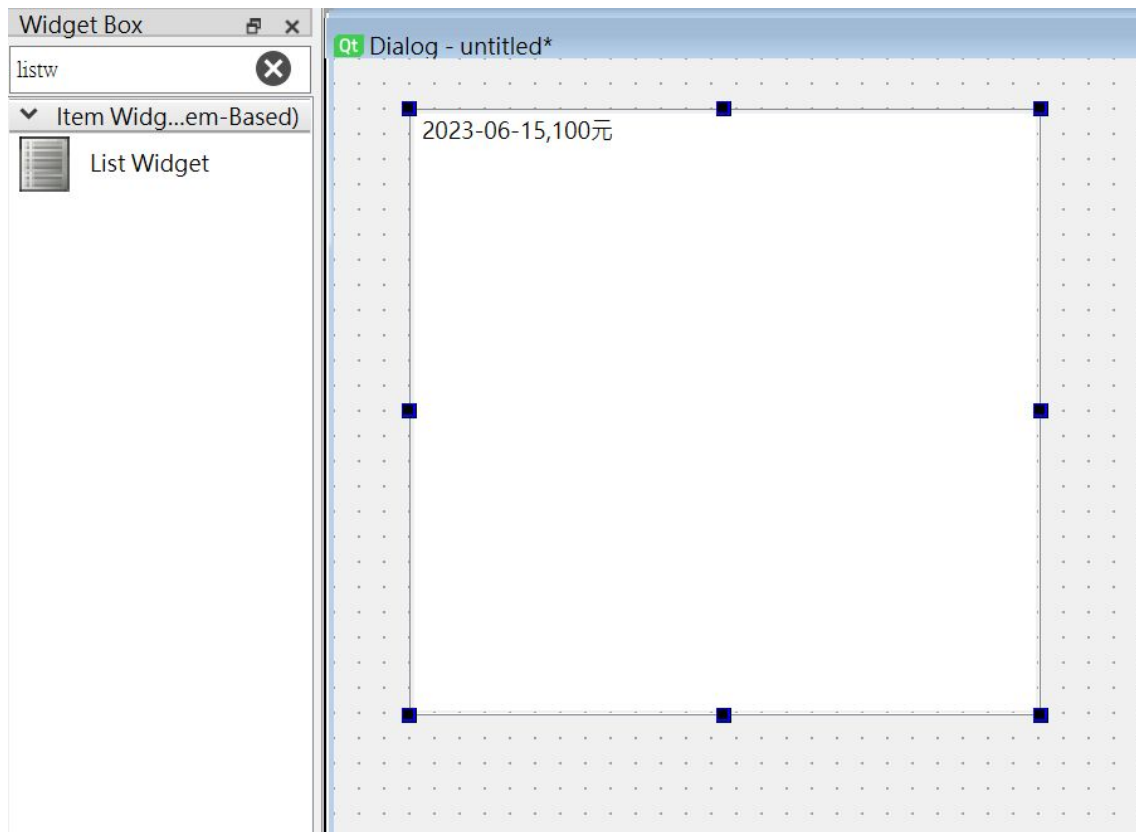
▼

get file

run

```
9,8,6,7,5  
2,4,3,8,10
```

# listWidget



# listWidget

```
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
import sys
from listwidget import Ui_Dialog
import time

def click_button():
    value = ui.listWidget.currentRow()

    print("Current Selected Row : " + str(value))

app = QApplication(sys.argv)
t=QTranslator()
t.load('tra_chinese.qm')
app.installTranslator(t)
widget = QWidget()
ui = Ui_Dialog()
ui.setupUi(widget)

ui.listWidget.addItem('A')
ui.listWidget.addItem('B')
ui.listWidget.addItem('C')
ui.listWidget.addItem('D')
ui.listWidget.addItem('X')
item = ui.listWidget.takeItem(1)    # 取得第二個項目，也就是 B
ui.listWidget.removeItemWidget(item) # 移除第二個項目

ui.pushButton.clicked.connect(click_button)

widget.show()
sys.exit(app.exec_())
```

[https://github.com/JuFengWu/py\\_qt/blob/master/listWidget/control.py](https://github.com/JuFengWu/py_qt/blob/master/listWidget/control.py)

# 小試身手-製作留言板

文字內容

輸入

移除

# 小試身手-製作留言板(進階)

文字內容

輸入

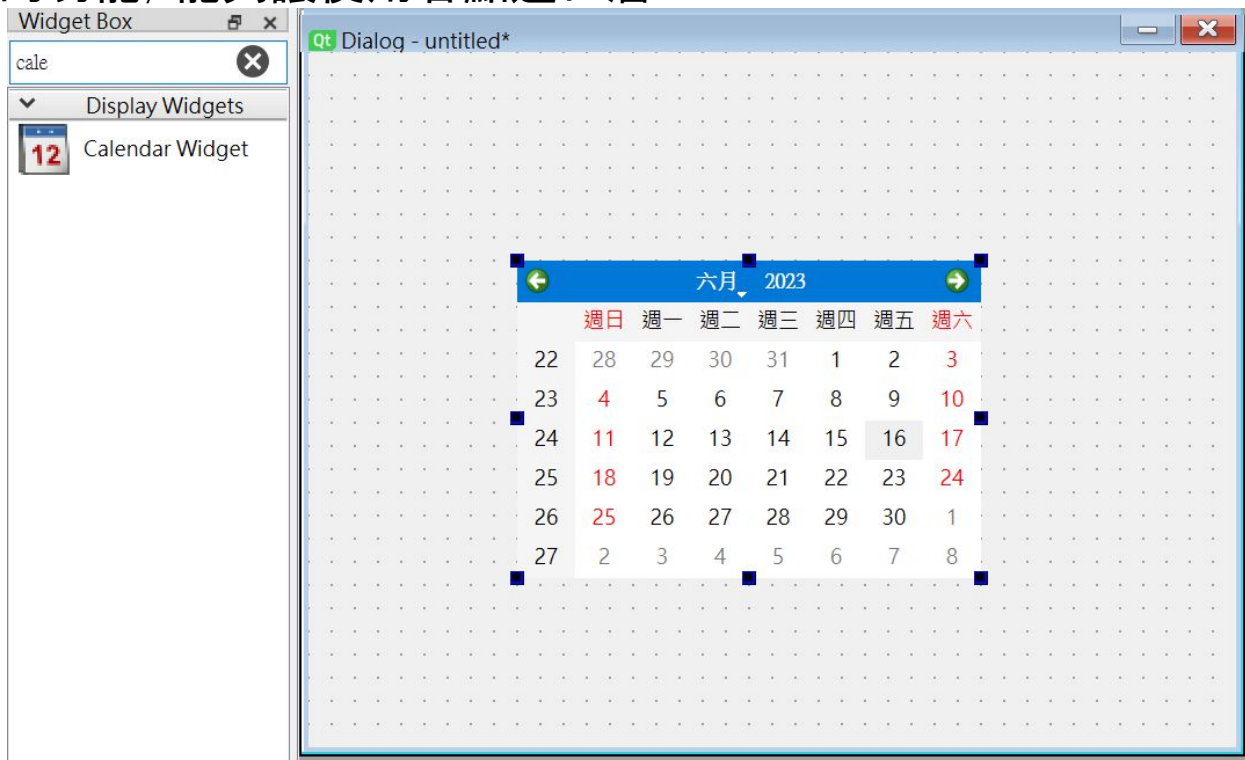
移除

匯入

存檔

# 日曆

pyqt有日曆的功能，能夠讓使用者點選日曆



# 日曆的反應

## 點選日曆的callback

```
selectDay = ui.calendarWidget.selectedDate()

print("pyQT day",selectDay)
print("normal day", selectDay.toString(Qt.ISODate))
cell_format = QTextCharFormat()
if backgroundColor:
    cell_format.setBackground(QColor("red"))
else:
    cell_format.setBackground(QColor("white"))
ui.calendarWidget.setDateTextFormat(selectDay, cell_format)

    ui.calendarWidget.setGridVisible(True)
    ui.calendarWidget.clicked.connect(showDate)
```

[https://github.com/JuFengWu/py\\_qt/blob/master/calendar/calendar.py](https://github.com/JuFengWu/py_qt/blob/master/calendar/calendar.py)

# 小作業

紀錄

2023-06-15,100元

← 六月 2023 →

	週日	週一	週二	週三	週四	週五	週六
22	28	29	30	31	1	2	3
23	4	5	6	7	8	9	10
24	11	12	13	14	15	16	17
25	18	19	20	21	22	23	24
26	25	26	27	28	29	30	1
27	2	3	4	5	6	7	8

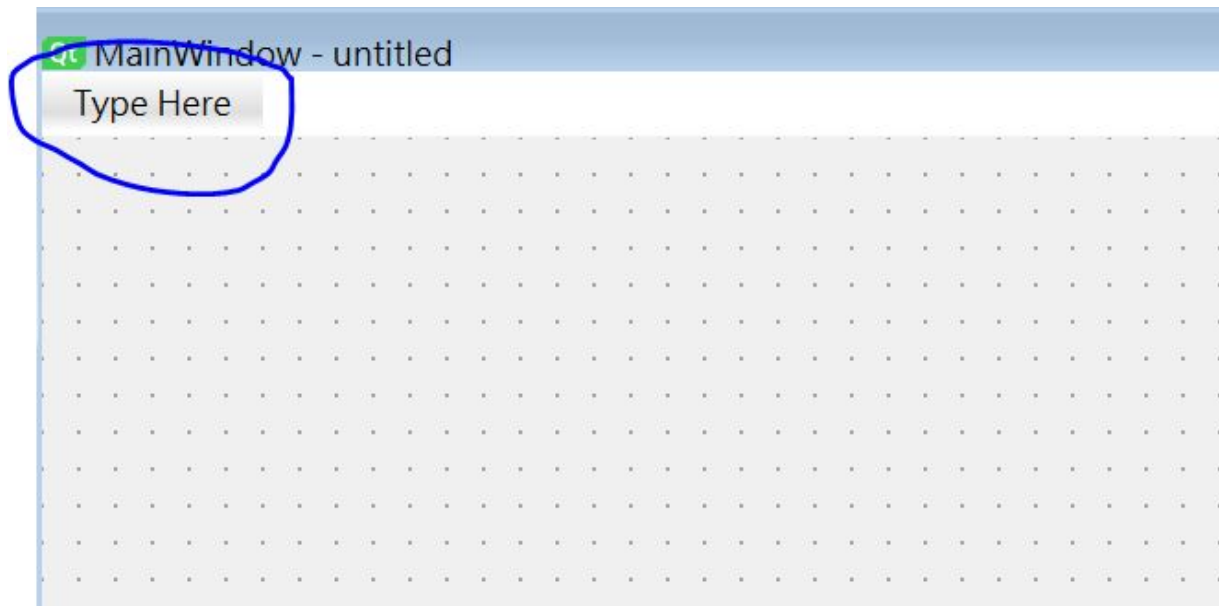
金額

確定紀錄

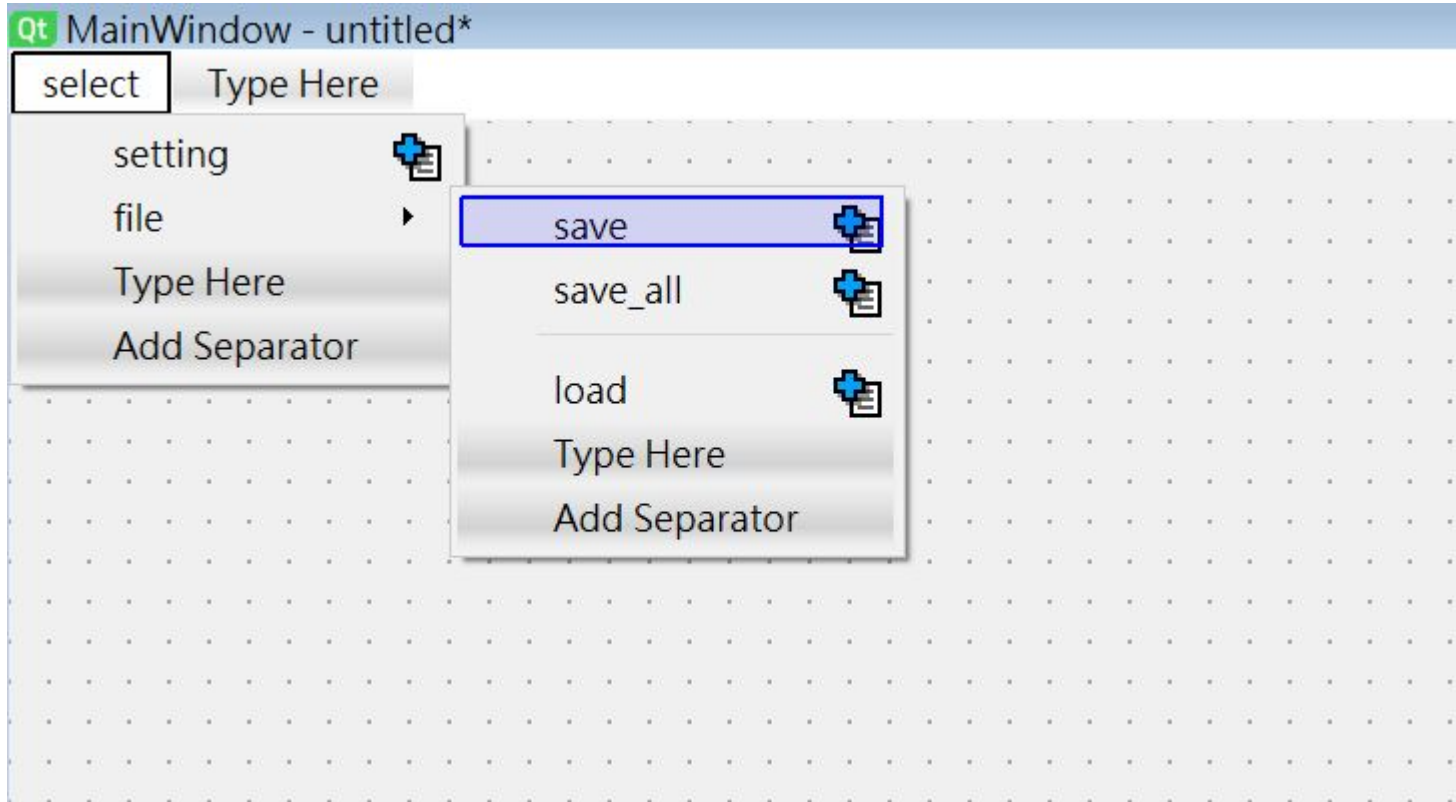


# mainwindow的使用 - menubar

mainwindow專用的功能



# 增加menubar功能



## 設定menubar的反應

```
def setting():  
    ui.label.setText("click setting")  
  
def save():  
    ui.label.setText("click save")  
  
def save_all():  
    ui.label.setText("click save_all")  
  
def load():  
    ui.label.setText("click load")  
  
app = QtWidgets.QApplication(sys.argv)  
MainWindow = QtWidgets.QMainWindow()  
ui = Ui_MainWindow()  
ui.setupUi(MainWindow)  
  
ui.actionsetting.triggered.connect(setting)  
ui.actionsave.triggered.connect(save)  
ui.actionsave_all.triggered.connect(save_all)  
ui.actionload.triggered.connect(load)
```

# 小試身手的時候到了

將記帳軟體+menu bar進行資料的存檔和開啟

紀錄

六月, 2023							
	週日	週一	週二	週三	週四	週五	週六
22	28	29	30	31	1	2	3
23	4	5	6	7	8	9	10
24	11	12	13	14	15	16	17
25	18	19	20	21	22	23	24
26	25	26	27	28	29	30	1
27	2	3	4	5	6	7	8

金額

2023-06-15,100元

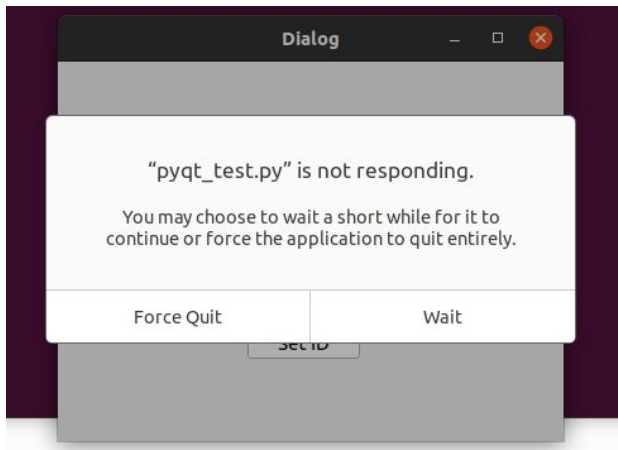
確定紀錄

# pyqt & thread

有一些動作按了按鈕之後，會花很多時間處理

ex爬蟲

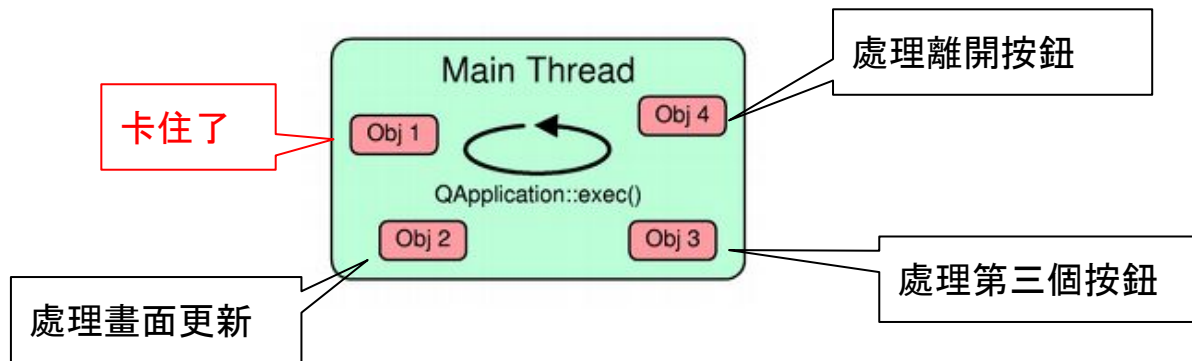
我們寫在button\_click中UI會卡住，按下x也不會動，還有可能會掛掉



```
def buttonClick(self):  
    time.sleep(5) #假設動作很久，睡5秒  
    IDlineEditText = self.ui.IDlineEdit.text()  
    if(not IDlineEditText.isnumeric()):  
        message_box = QMessageBox()  
        message_box.setWindowTitle("error")  
        message_box.setInformativeText("please enter a interger number")  
        message_box.exec_()  
    else:  
        self.ui.IDlabel.setText(IDlineEditText)
```

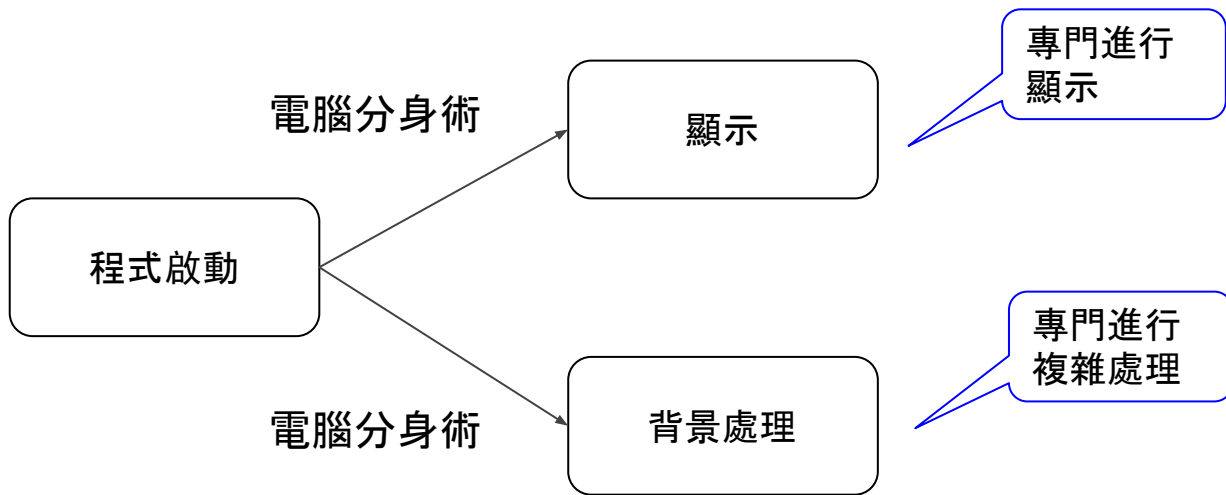
# 一個按鈕卡住會影響其他的按鈕?!

在QT的內部，程式的執行方式如下



# 這時候要如何處理？

把複雜的程式分開



這方法叫做Thread

電腦分身術  
怎麼達成？

# 電腦分身術達成方法

方法1：8核心的CPU

1號CPU做第一件事情，2號CPU做第二件事情

方法2：CPU做第一件事情，做到一半暫停做第二件事情

以1GHz的CPU，一秒可以做10億次運算

所以如果程式會執行很久要讓他休息去做別的事情

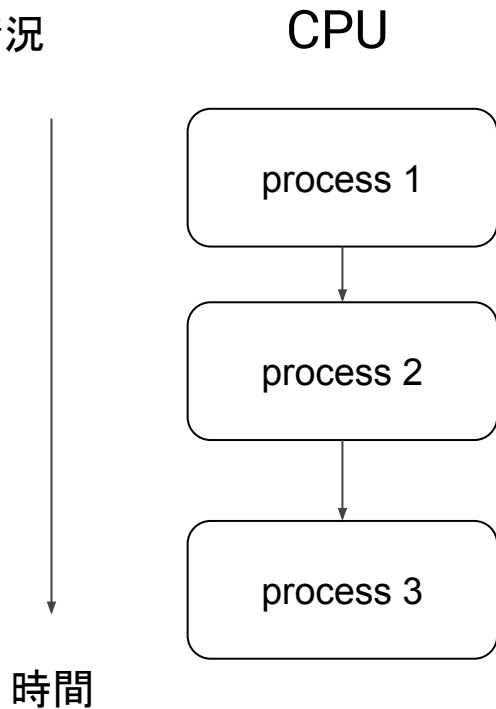
用方法1還是方法2

由作業系統決定

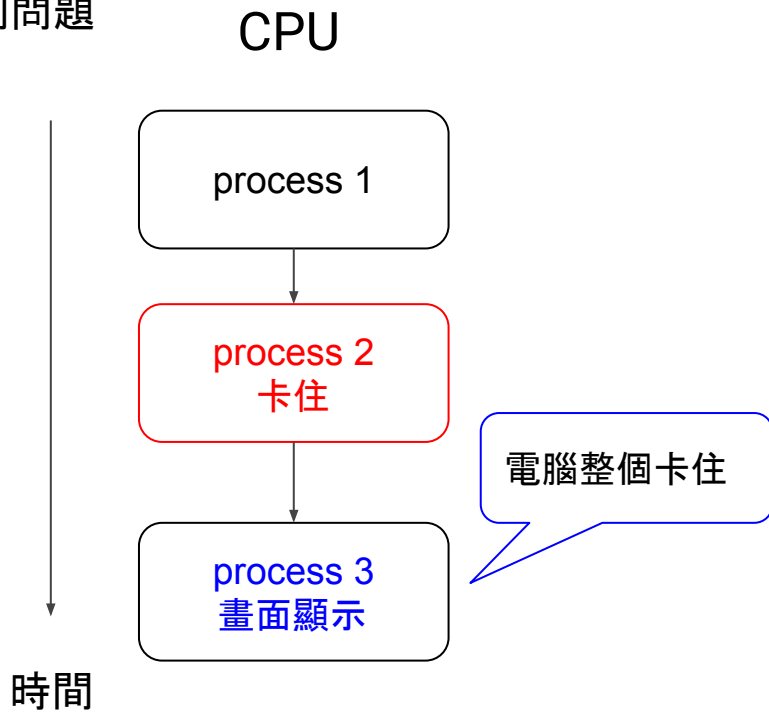


# 電腦分身術失敗

正常情況



會遇到問題



很多人不知道

# 電腦分身術失敗常見的程式寫法

1. for loop 太大
2. while loop 太大
3. 在等待其他東西 busy wait

```
a = 0
b = 0
while True:
    a = a+1
    if a==65535 :
        a = 0
        b = b+1
        if b == 65536 :
            break
```

做了65536x65535次

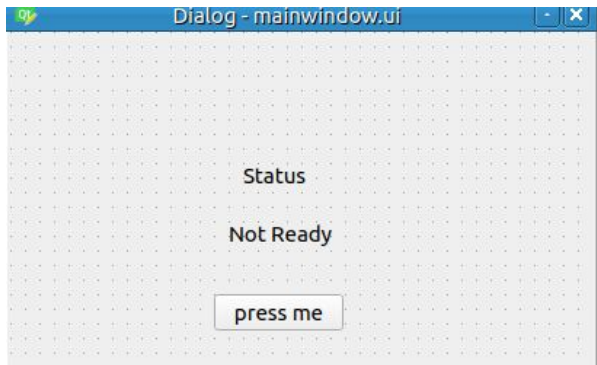
做很大的事情時  
迴圈內適度加上sleep

通常sleep(0.001)  
1ms

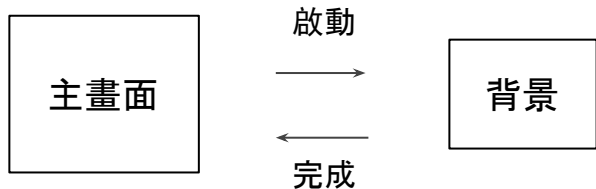
把CPU的使用權  
暫時給其他人

# pyqt thread

按下按鈕啟動thread  
thread完成後通知主畫面



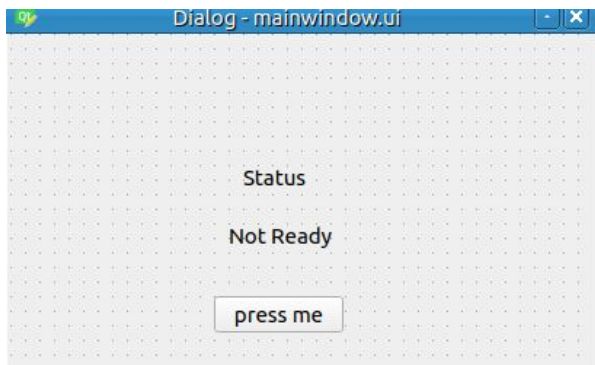
```
def run():  
    sleep(1)  
    ui.label_2.setText("end run")  
def click_button():  
    runningThread = threading.Thread(target = run)  
    runningThread.start()  
    ui.label_2.setText("start run")  
  
app = QApplication(sys.argv)  
widget = QWidget()  
ui = Ui_Dialog()  
ui.setupUi(widget)  
ui.pushButton.clicked.connect(click_button)  
  
widget.show()  
sys.exit(app.exec_())
```



qt\_thread/pyqt\_thread.py

# pyqt thread(進階)

按下按鈕啟動thread  
thread完成後通知主畫面



啟動



背景



完成

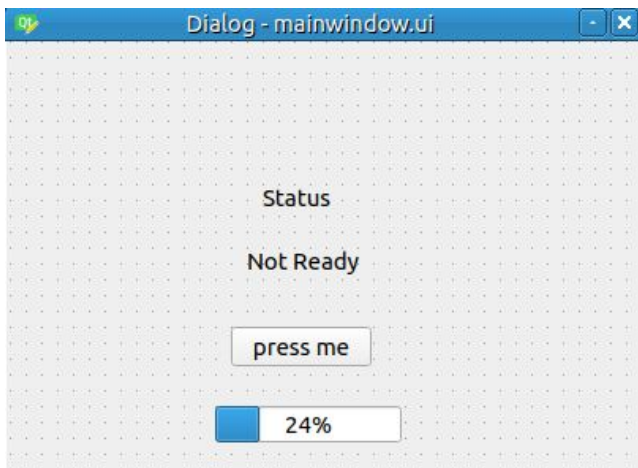
```
class RunningThread(QThread):
    def __init__(self, parent, signal):
        super(RunningThread, self).__init__(parent=parent)
        self.signal = signal
    def run(self):
        sleep(5)
        self.signal.emit("end running")

class MainWindow(QMainWindow):
    signal = pyqtSignal(str)
    def __init__(self):
        super(MainWindow, self).__init__()
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)
        self.ui_connection()

        self.signal.connect(self.finish_running)
        self.runningThread = RunningThread(self, self.signal)
    def ui_connection(self):
        self.ui.pushButton.clicked.connect(self.click_button)
    def click_button(self):
        self.runningThread.start()
        self.ui.label_2.setText("start running")
    @pyqtSlot(str)
    def finish_running(self, msg):
        self.ui.label_2.setText(msg)
```

# pyqt thread

利用進度條表示狀況



如果不使用進階的方式寫會導致crash

# 利用 pyqt timer取代thread

```
number = 0
global timer
def count_sheep():
    print("do timer")
    global number
    number += 1
    process_show(number)
    if(number==5):
        timer.stop()
        number = 0

def process_show(currentNumber):
    ui.progressBar.setValue(currentNumber*20)

def click_button():
    timer.start(500)
    ui.label_2.setText("start running")

app = QApplication(sys.argv)
widget = QWidget()
ui = Ui_Dialog()
ui.setupUi(widget)
ui.pushButton.clicked.connect(click_button)

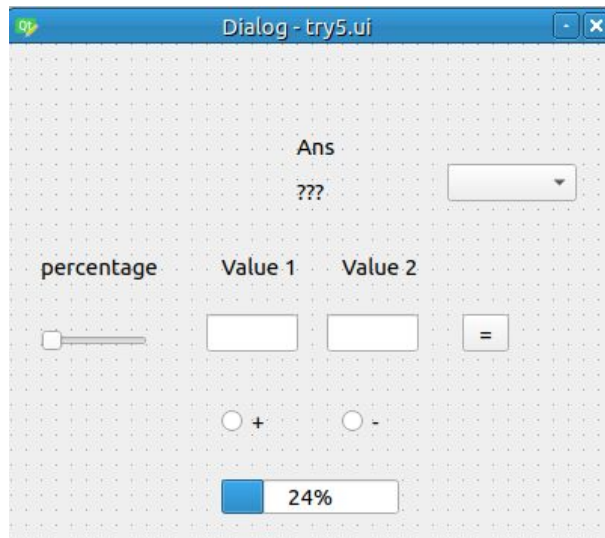
timer = QTimer()
timer.timeout.connect(count_sheep)

widget.show()
sys.exit(app.exec_())
```

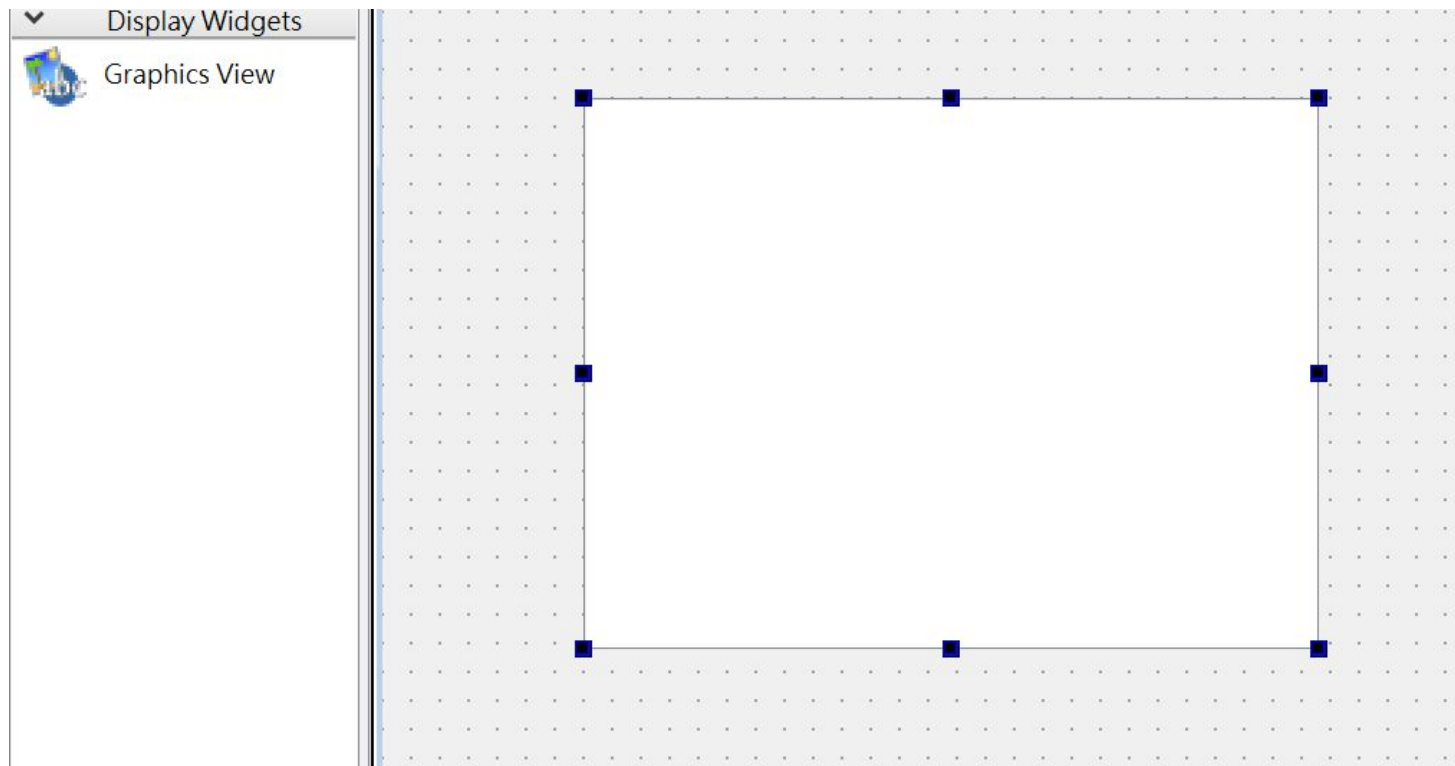
[https://github.com/JuFengWu/py\\_qt/blob/master/qt\\_thread\\_process/timer.py](https://github.com/JuFengWu/py_qt/blob/master/qt_thread_process/timer.py)

# 動手的時間到了

1. 把計算放入thread
2. 計算的時候故意要算5秒 (給客戶試用版, 解鎖算更快)
3. 每一秒process bar都要有顯示進度



# Qt和plot



基本的資料視覺化工具matplotlib會了嘛?



# 利用FigureCanvas進行pyqt和連結

```
class MyFigureCanvas(FigureCanvas):  
  
    def __init__(self, parent=None, width=11, height=5, dpi=100):  
        fig = Figure(figsize=(width, height), dpi=100)  
        FigureCanvas.__init__(self, fig)  
        self.setParent(parent)  
        self.axes = fig.add_subplot(111)  
  
    def test(self):  
        x = [1,2,3,4,5,6,7,8,9]  
        y = [23,21,32,13,3,13,13,3,1]  
        self.axes.plot(x, y)  
  
    def test2(self,x,y):  
        self.axes.plot(x, y)
```

## 畫出資料

```
def buttonClick2():  
    dr = MyFigureCanvas()  
    x = [1,2,3,4,5,6,7,8,9]  
    y = [12,4,17,19,45,21,11,2,3]  
    dr.test2(x,y)  
    graphicscene = QtWidgets.QGraphicsScene()  
    graphicscene.addWidget(dr)  
    ui.graphicsView.setScene(graphicscene)  
    ui.graphicsView.show()
```

[https://github.com/JuFengWu/py\\_qt/blob/master/qt\\_plot/control.py](https://github.com/JuFengWu/py_qt/blob/master/qt_plot/control.py)

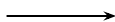
# Qt多國語言

老闆：我們要賣台灣、日本、德國，需要多國語言！

工程師A：那我們就四份程式，每一個不同語言！

資深工程師：直接使用QT的多語言切換套件

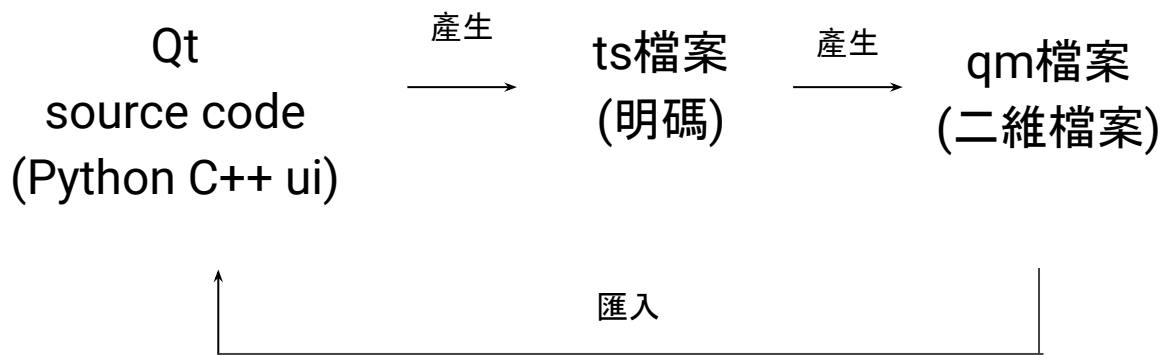
本章節主角  
QT Linguist



除了QT  
之前提到的C#也有支援多語言

# Qt多國語言

## QT多國語言的流程



在寫程式的時候就必須要標注  
哪一些文字需要轉換語言



# 下載Linguist

## 5.15.2

Qt Linguist (including lupdate/lupdate-pro, lrelease/lrelease-pro, and lconvert) 5.15.2 for Windows

Last 32-bit release

### ▼ Assets 3

 <a href="#">linguist_5.15.2.zip</a>	13.1 MB	05 Dec 2020
 <a href="#">Source code</a> (zip)		05 Dec 2020
 <a href="#">Source code</a> (tar.gz)		05 Dec 2020

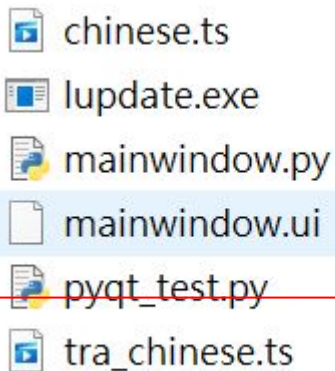


<https://github.com/thurask/Qt-Linguist/releases>

# 產生ts檔案

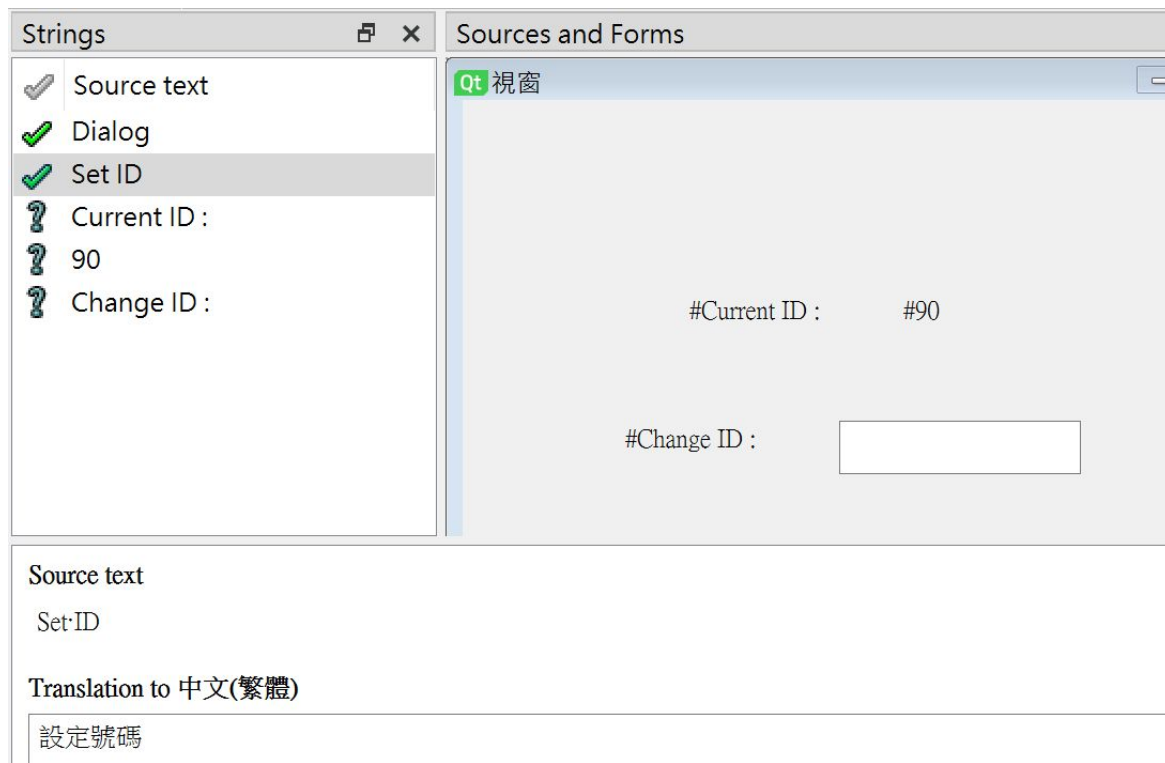
將lupdate放到ui檔案旁邊，開啟cmd下指令

```
D:\個人資料夾\P. 作品\python\liguest>lupdate.exe/mainwindow.ui -ts tra_chinese.ts  
Updating 'tra_chinese.ts'...  
Found 5 source text(s) (5 new and 0 already existing)
```








產生相對應的ts檔案

# 利用linguist撰寫翻譯



# 產生qm檔案

```
D:\個人資料夾\P. 作品\python\liguest>lrelease tra_chinese.ts tra_chinese.qm
Updating 'tra_chinese.qm'...
    Generated 4 translation(s) (4 finished and 0 unfinished)
    Ignored 1 untranslated source text(s)
Updating 'tra_chinese.qm'...
    Generated 4 translation(s) (4 finished and 0 unfinished)
```

-  mainwindow.py
-  mainwindow.ui
-  pyqt\_test.py
-  tra\_chinese.qm
-  tra\_chinese.ts

產生相對應的qm檔案



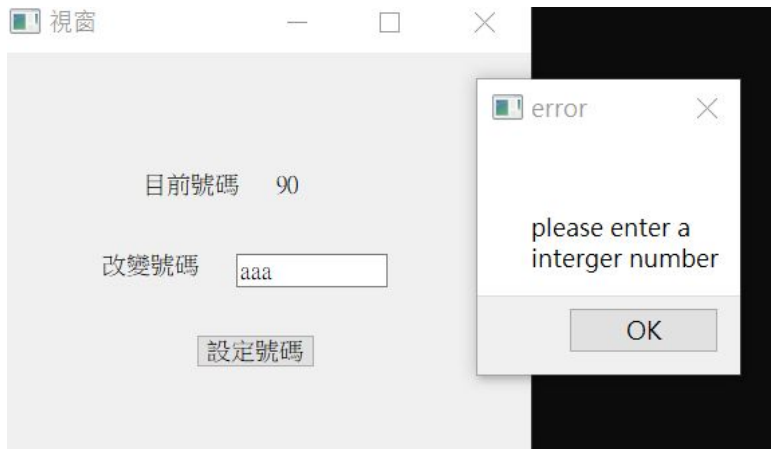
# 匯入譯本



```
if __name__ == "__main__":  
    app = QApplication(sys.argv)  
    t = QTranslator()  
    t.load('tra_chinese.qm')  
    app.installTranslator(t)  
    window = MainWindow()  
  
    window.show()  
  
    sys.exit(app.exec_())
```

## 但是有問題(只能使用進階)

我們python寫的文字還是英文  
這部分必須也要進行翻譯



# python的文字變成ts檔案(只能使用進階)

- 1.需要翻譯的地方加上self.tr()
- 2.使用pylupdate5的python指令產生ts檔案

```
def buttonClick(self):  
    #time.sleep(5) #假設動作很久，睡5秒  
    IDlineEditText = ui.IDlineEdit.text()  
    if(not IDlineEditText.isnumeric()):  
        message_box = QMessageBox()  
        message_box.setWindowTitle (widget.tr("error"))  
        message_box.setInformativeText(widget.tr("please enter a interger number"))  
        message_box.exec_()   
    else:  
        ui.IDlabel.setText(IDlineEditText)
```

因為是python的程式  
所以要用pylupdate5

# python的文字變成ts檔案(進階)

- 1.需要翻譯的地方加上self.tr()
- 2.使用pylupdate5的python指令產生ts檔案

```
if(not IDlineEditText.isnumeric()):  
    message_box = QMessageBox()  
    message_box.setWindowTitle (self.tr("error"))  
    message_box.setInformativeText(self.tr("please enter a interger number"))  
    message_box.exec_()
```

```
>pylupdate5 pyqt_test.py -ts tmp_chinese.ts
```

因為是python的程式  
所以要用pylupdate5

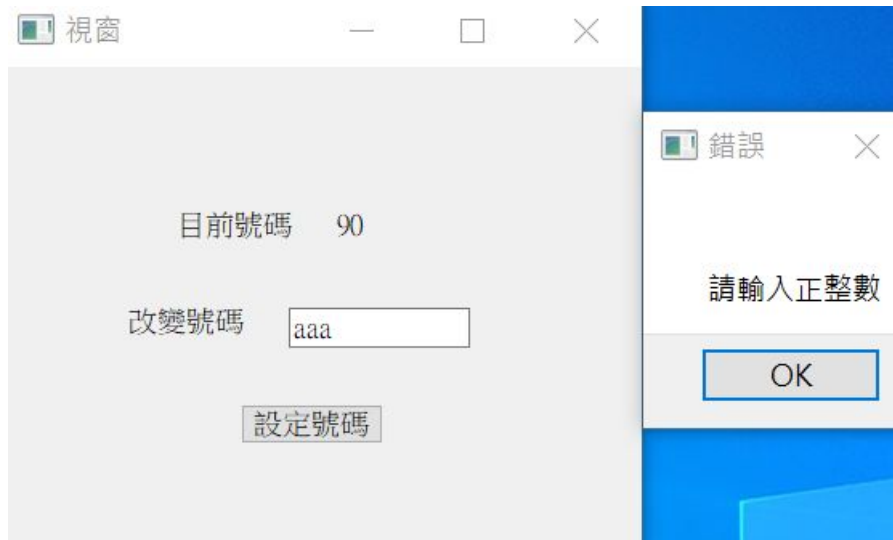
# 把新產生的ts檔案貼到原本的ts檔案

```
<!--
<name>MainWindow</name>
<message>
  <location filename="../../../../??H????/P. ??~/python/liguest/pyqt_test.py" line="24"/>
  <source>error</source>
  <translation type="unfinished"></translation>
</message>
<message>
  <location filename="../../../../??H????/P. ??~/python/liguest/pyqt_test.py" line="25"/>
  <source>please enter a interger number</source>
  <translation type="unfinished"></translation>
</message>
```

```
  <translation type="unfinished"></translation>
</message>
<message>
  <location filename="mainwindow.ui" line="75"/>
  <source>Change ID :</source>
  <translation>改變號碼</translation>
</message>
</context>
<context>
  <name>MainWindow</name>
  <message>
    <location filename="../../../../??H????/P. ??~/python/liguest/pyqt_test.py" line="24"/>
    <source>error</source>
    <translation>錯誤</translation>
  </message>
  <message>
    <location filename="../../../../??H????/P. ??~/python/liguest/pyqt_test.py" line="25"/>
    <source>please enter a interger number</source>
    <translation>請輸入正整數</translation>
```

# 最後結果

成功將python的文字也翻譯成中文

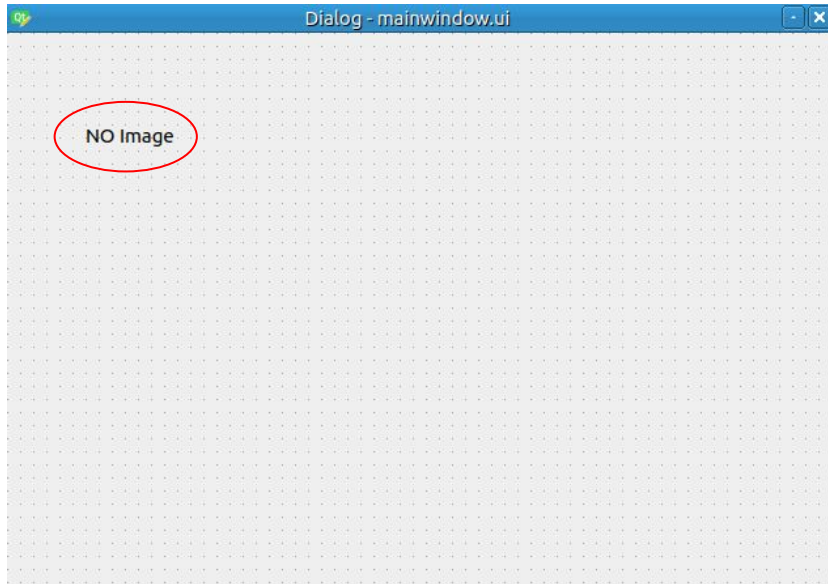


# 實做的時間到了

把上個程式加入中文版的功能

# 補充pyqt + opencv

把label轉成影像



```
def ui_connection(self):  
    image = cv2.imread(self.imageFile)  
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)  
    qImage = QImage(image, image.shape[1], image.shape[0], image.strides[0], QImage.Format_RGB888)  
    self.ui.imageLabel.setPixmap(QPixmap.fromImage(qImage))  
    self.ui.imageLabel.setFixedSize(image.shape[1], image.shape[0])
```



## 補充 thread 注意事項

1. thread 要確實關閉起來的機制

while(True) -> 不好

while(isRun) -> 有離開的機制

2. Thread 出 bug 會很棘手

Thread 出 bug 特徵是一下上一行掛掉, 一下下一行掛掉

3. 因為 thread 不好 debug, 建議一個程式不要開太多

慘痛經驗: 一個 bug 要花三個月去找

# 點兩下啟動程式

1. pyinstaller安裝
  - >常見的一個方法, 但是打包有時候會出問題, 要注意dependency
2. 寫一個shell檔案
  - >程式碼會給人家看到, 但是會了可以做很多系統的事情

# pyinstaller安裝

```
C:\opt>pip3 install pyinstaller
Collecting pyinstaller
  Downloading pyinstaller-5.1-py3-none-win_amd64.whl (1.2 MB)
    |████████████████████████████████████████| 1.2 MB 819 kB/s
Collecting pyinstaller-hooks-contrib>=2021.4
  Downloading pyinstaller_hooks_contrib-2022.7-py2.py3-none-any.whl (234 kB)
    |████████████████████████████████████████| 234 kB 6.4 MB/s
Collecting pefile>=2017.8.1; sys_platform == "win32"
  Downloading pefile-2022.5.30.tar.gz (72 kB)
    |████████████████████████████████████████| 72 kB ...
```

# pyinstaller使用

直接下cmd就可以了

```
C:\opt\radio_button>pyinstaller pyqt_test.py
62 INFO: PyInstaller: 5.1
62 INFO: Python: 3.8.3
62 INFO: Platform: Windows-10-10.0.19041-SP0
62 INFO: wrote C:\opt\radio_button\pyqt_test.spec
62 INFO: UPX is not available.
86 INFO: Extending PYTHONPATH with paths
['C:\\opt\\radio_button']
398 INFO: checking Analysis
398 INFO: Building Analysis because Analysis-00.toc is
```


















# pyinstaller使用

可以使用-F -i 去更換成喜歡的執行檔頭貼  
記得windows只吃ico檔

```
C:\opt\radio_button>pyinstaller -F -i C:\opt\radio_button\fb.ico pyqt_test.py
62 INFO: PyInstaller: 5.1
62 INFO: Python: 3.8.3
62 INFO: Platform: Windows-10-10.0.19041-SP0
78 INFO: wrote C:\opt\radio_button\pyqt_test.spec
78 INFO: UPX is not available.
78 INFO: Extending PYTHONPATH with paths
['C:\\opt\\radio_button']
```

# pyinstaller產生執行檔案

dist\檔案名稱資料夾內

 PyQt5	2022/6/27 下午 03:26	檔案資料夾	
 _bz2.pyd	2022/6/27 下午 03:23	Python Extension M...	85 KB
 _hashlib.pyd	2022/6/27 下午 03:23	Python Extension M...	46 KB
 _lzma.pyd	2022/6/27 下午 03:23	Python Extension M...	248 KB
 _socket.pyd	2022/6/27 下午 03:23	Python Extension M...	78 KB
 base_library.zip	2022/6/27 下午 03:26	壓縮的 (zipped) 資料...	776 KB
 d3dcompiler_47.dll	2022/6/27 下午 03:24	應用程式擴充	4,077 KB
 libcrypto-1_1.dll	2022/6/27 下午 03:23	應用程式擴充	3,338 KB
 libEGL.dll	2022/6/27 下午 03:24	應用程式擴充	25 KB
 libGLSv2.dll	2022/6/27 下午 03:24	應用程式擴充	3,306 KB
 MSVCP140.dll	2022/6/27 下午 03:24	應用程式擴充	559 KB
 MSVCP140_1.dll	2022/6/27 下午 03:24	應用程式擴充	31 KB
 opengl32sw.dll	2022/6/27 下午 03:24	應用程式擴充	20,433 KB
 pyqt_test.exe	2022/6/27 下午 03:26	應用程式	1,017 KB
 python3.dll	2022/6/27 下午 03:24	應用程式擴充	58 KB
 python38.dll	2022/6/27 下午 03:23	應用程式擴充	4,110 KB
 Ot5Core.dll	2022/6/27 下午 03:24	應用程式擴充	5.883 KB

# 產生執行檔要注意的問題

1. 檔案名稱資料夾內的所有檔案都要在exe旁邊
2. 給別人的話要注意license的問題
3. 別人可以從dll之類的去反推你的程式用了什麼dependency

## 寫bat(windows)/shell(linux)

- shell或是bat檔案 = 開一個cmd直接下命令
- 我們平常下什麼命令就寫什麼
- 如果要讓使用者的電腦安裝套件的話, 可以使用這樣的方式
- 比較沒有license的疑慮(因為是連結到相對的資料夾中)
- python的程式就是原始程式碼給別人看
- 換python檔案比較方便



# 開始使用bat檔案

1. 產生一個檔案, 副檔名改成bat檔案
2. 編輯bat檔案
3. 寫指令
4. 點兩下bat檔

```
cd C:\opt\radio_button  
python pyqt_test.py
```

1. 轉到相對應的資料夾
2. 執行py檔案

linux的shell使用方式也是如此

# 使用bat檔案進行安裝

注意以下事項

1. 幫人家安裝or更新套件的時候, 有可能會影響到其他的程式  
->建議使用anaconda or docker等將環境切乾淨
2. 有一些環境變數的設定要注意, 改了之後也有可能會影響到其他程式

# 動手的時間到了

1. 將上述的程式用pyinstaller或打包安裝
2. 寫一個shell進行執行
3. 將shell&程式上傳到github
4. 寫說明要如何使用這一個程式

組員互相交換github帳號  
下載對方的程式，看說明來執行