

Python GUI

講師：隋建德

大綱

- GUI 簡介
- Tkinter
- Tkinter 元件
- thread觀念
- Rabboni GUI



GUI?

- 命令列介面
 - 早期電腦
 - 使用純文字來編輯文檔等等
 - Dos系統
- 圖形使用者介面(**Graphical User Interface**)
 - Windows、Mac...
 - 較易學習、理解
 - UI是一門學問
 - UI工程師

```

root@kali:~# cd /usr/portage/app-shells/bash
root@kali:~# cd /usr/portage/app-shells/bash && ls -la
total 130
drwxr-xr-x 33 portage portage 1824 Jul 25 18:06
drwxr-xr-x 1 root root 35808 Jul 25 18:06 chanelog
drwxr-xr-x 1 root root 27888 Jul 25 18:06 manifest
-rw-r--r-- 1 portage portage 4645 Apr 23 21:37 bash-3.1p17.ebuild
-rw-r--r-- 1 portage portage 5977 Mar 23 21:37 bash-3.2_p39.ebuild
-rw-r--r-- 1 portage portage 6151 Mar 23 21:37 bash-3.2_p48-r1.ebuild
-rw-r--r-- 1 portage portage 5988 Mar 23 21:37 bash-3.2_p48.ebuild
-rw-r--r-- 1 portage portage 5642 Apr 5 14:37 bash-4.0_p18-r1.ebuild
-rw-r--r-- 1 portage portage 6238 Apr 5 14:37 bash-4.0_p18.ebuild
-rw-r--r-- 1 portage portage 5648 Apr 14 05:52 bash-4.0_p17-r1.ebuild
-rw-r--r-- 1 portage portage 5532 Apr 8 18:21 bash-4.0_p17.ebuild
-rw-r--r-- 1 portage portage 5568 Jul 30 03:35 bash-4.0_p24.ebuild
-rw-r--r-- 1 root root 5568 Jul 25 09:43 bash-4.0_p28.ebuild
drwxr-xr-x 2 portage portage 2648 Jul 30 03:35 files
-rw-r--r-- 1 portage portage 1468 Jul 25 09:43 bash-4.0_p24.ebuild
root@kali:~# cd /usr/portage/app-shells/bash && cat metadata.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE metadata SYSTEM "http://www.gentoo.org/dtd/metadata.dtd">
<pkgmetadata>
  herdbase-system</herdbase>
  <use>
    (flag names="bashlogger" log ALL commands typed into bash; should ONLY be
    used in restricted environments such as honeypots/flag)
    (flag names="enable_dev/tcp/http/port redirection/flag)
    (flag names="plugins" add support for loading builtins at runtime via
    enable </flag>
  </use>
  <pkgmetadata>
    root@kali:~# cd /usr/portage/app-shells/bash && sudo /etc/init.d/bluetooth status
Bluetooth
Password:
< status: started
root@kali:~# cd /usr/portage/app-shells/bash && ping -q -c 1 en.wikipedia.org
PING rr.essas.wikiimedia.org (91.198.174.2) 56 bytes of data.
0 packets transmitted, 0 received, 80% packet loss, time 2ms
rtt min/avg/max/mdev = 49.820/49.820/49.820/0 ms
root@kali:~# cd /usr/portage/app-shells/bash && grep -i /dev/sda /etc/fstab | cut --fields=3
/dev/sda1
/dev/sda2 none
/dev/sda3 /
root@kali:~# cd /usr/portage/app-shells/bash && date
Sat Aug 8 02:42:24 MSD 2009
root@kali:~# cd /usr/portage/app-shells/bash && lsmod
Module Size Used by
rmdisk_wlan 23424 0
rmdisk_host 8596 1 rmdisk_wlan
cdc_ether 5672 1 rmdisk_host
usbnet 18688 3 rmdisk_wlan,rmdisk_host,cdc_ether
usbnet_lpc 38024 0
fglrx 2388128 20
parport 39648 1 parport_pc
ltdo_vet 12272 0
ltdo_1081 9380 0
root@kali:~# cd /usr/portage/app-shells/bash &&

```

GUI



- 基本要素
 - 視窗、圖示、按鈕
- 優點
 - 直觀易學習
 - 容易使用
- 影響
 - 使電腦大眾化
- 濫觴
 - GUI的習慣導致沒有GUI不行





GUI 開發語言

- 許多程式語言都有支援GUI開發
 - Java, C, Python.....
 - Android Studio(Java / Kotlin), XCODE(Swift)
- GUI運作效果
 - GUI 會牽扯到硬體等
 - 不對的語言在不對的硬體上實行 速度過慢
 - C on windows > Python on windows

Tkinter



GUI in Python

- 百家爭鳴
 - Tkinter, pyqt, kivy, PyGUI, Flexx....
- 各有個優點
 - Flexx 可跨瀏覽器
 - Kivy 內置許多介面控制項
 - PyQt使用者眾多
 - 商用板須授權付費



Tkinter

- 簡單易學
 - 比起其他LIB 較容易上手
- 程式碼精簡
 - 能以短短數行達到強大功能
- 跨平台
 - 可以在 windows/linux/mac 上執行
- 版本
 - Python2 Tkinter
 - Python3 tkinter

範例



BMI 計算器

BMI 計算器

身高 (m)

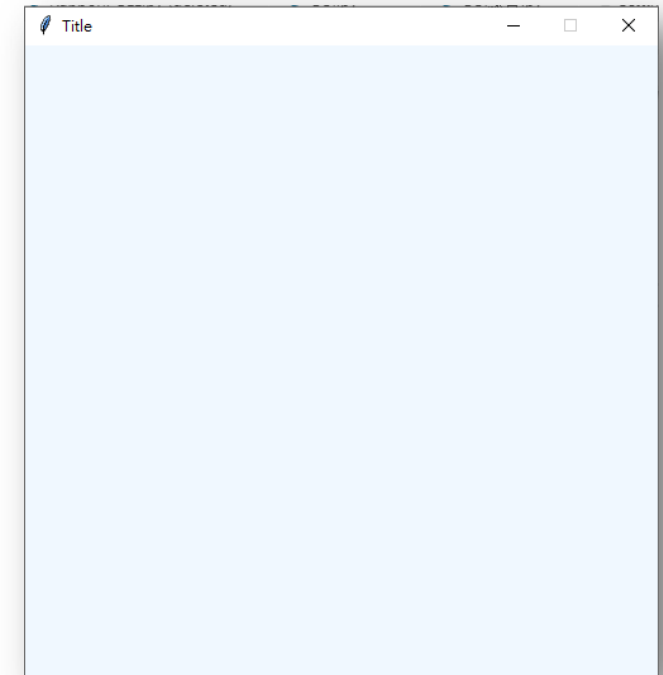
體重 (kg)

計算

建立視窗

- `root = tk.TK()`
 - 宣告一個視窗容器 `root`
- `root.title('Title')`
 - 給視窗名字
- `root.geometry('500x500')`
 - 視窗長寬
- `root.resizable(width=False, height=False)`
 - 限制視窗不可拉大拉小
- `root.configure(background='Aliceblue')`
 - 設定視窗的背景
 - `white, black, ...`
- `root.mainloop()`
 - 開啟視窗

```
1 import tkinter as tk
2 root = tk.Tk()
3 root.title('Title')
4 root.geometry('500x500')
5 root.resizable(width=False, height=False)
6 root.configure(background='Aliceblue')
7 root.mainloop()
8
```



Tkinter元件



Label 標籤

- 標籤

- 位於GUI上的字
- 常用來說明等等

- 9行

- 宣告一個lable 在root視窗上
- 給予文字內容
- 指定背景顏色
- 指定字體大小

- 10行

- place(x = 位置,y= 位置, anchor = 基準點) ## 精確的座標 nw->左上角3
- 可用pack(side = 'top')... 替代，但較不能精確指定位置

```
9 header_label = tk.Label(root, text='BMI 計算器',bg='Aliceblue', font=('Arial', 16))
10 header_label.place(x=200, y=50, anchor='nw')
11
12 height_label = tk.Label(root, text='身高 (m) ',bg='Aliceblue', font=('Arial', 16))
13 height_label.place(x=100, y=100, anchor='nw')
14
15 weight_label = tk.Label(root, text='體重 (kg) ',bg='Aliceblue', font=('Arial', 16))
16 weight_label.place(x=100, y=200, anchor='nw')
```

Label 標籤





Entry文字輸入格

- Entry 文字輸入格
 - 可在其中輸入文字
 - 常用在登入帳戶資料等等
- 18行
 - 宣告一個Entry物件在root視窗上
 - show : 顯示出的樣貌, None:不做任何處理
 - 指定字體
 - 指定輸入框寬度
- 20行
 - show = '*', 輸入文字變為*
 - 常用來當密碼輸入使用

```
18 height_entry = tk.Entry(root, show = None, font=('Arial', 16),width=10)
19 height_entry.place(x=250, y=100, anchor='nw')
20 weight_entry = tk.Entry(root, show = '*', font=('Arial', 16),width=10)
21 weight_entry.place(x=250, y=200, anchor='nw')
```

Entry文字輸入格

A screenshot of a Python Tkinter window titled "Title". The window has a light blue background and contains a BMI calculator interface. At the top center, the text "BMI 計算器" is displayed. Below it, there are two rows of input fields. The first row is labeled "身高 (m)" (Height in meters) and contains a text entry field with the value "180". The second row is labeled "體重 (kg)" (Weight in kilograms) and contains a text entry field with the value "**". The window has standard macOS-style window controls (red, yellow, and green buttons) in the top-left corner.



Text 視窗

- Text 視窗
 - 可以多行顯示
 - 類似Entry

```
26 t = tk.Text(root, font=('Arial', 16),width=10,height = 2)
27 t.place(x=250, y=400, anchor='nw')
28
29 t.insert('insert', "AioT")
```

- 26行
 - 宣告一個Text物件在root視窗上
 - 指定字體
 - 指定寬度高度
- 29行
 - 使用插入的方式
 - 插入 AioT字

Text 視窗

A screenshot of a Python Tkinter window titled "Title". The window has a light blue background and standard window controls (minimize, maximize, close) in the top right corner. The text "BMI 計算器" is centered at the top. Below it, there are two labels: "身高 (m)" and "體重 (kg)", each followed by a white text input field. At the bottom of the window, there is a white text area containing the text "AioT".

Title

BMI 計算器

身高 (m)

體重 (kg)

AioT

Button 按鈕

- Button 按鈕

- 人機互動的重要元件
- 可自訂觸發後事件
- 需要有command否則沒效
- 使用def func去定義動作

```
31 def end666():  
32     t.insert('end', "666")  
33  
34 calculate_btn = tk.Button(root, text='將666放在最後', command=end666, font=('Arial', 16))  
35 calculate_btn.place(x=230, y=300, anchor='nw')
```

- 31-32行

- 定義一個func決定button觸發後的行為

- 34行

- 宣告button物件在root視窗上
- 指定按鈕上文字
- command 指定觸發後執行的func

Button 按鈕



Title

BMI 計算器

身高 (m)

體重 (kg)

將666放在最後

AioT



Title

BMI 計算器

身高 (m)

體重 (kg)

將666放在最後

AioT666



BMI運算-自行完成

- 運算方式
 - `import math` ##將數學lib導入
 - `math(x,2)` ## 對x進行平方
 - `round(x,2)` ##只取到小數點以下第二位
- 將運算寫成func並用button觸發
 - 並使用`t.insert()`寫入textbox

$$\text{BMI} = \frac{\text{體重 (公斤)}}{\text{身高 (公尺)} \times \text{身高 (公尺)}}$$

BMI運算



Title

BMI 計算器

身高 (cm)

體重 (kg)



messagebox 訊息視窗

- messagebox 訊息視窗
 - 跳出視窗提醒使用者
 - 常用來跳出警告訊息
- 3行
 - 導入
- 14行
 - 秀出訊息視窗
 - 指定視窗title
 - 指定訊息
- 替換textbox顯示

```
3  import tkinter.messagebox
14  tkinter.messagebox.showinfo(title='BMI', message='BMI : '+str(bmi_value))
```

messagebox 訊息視窗

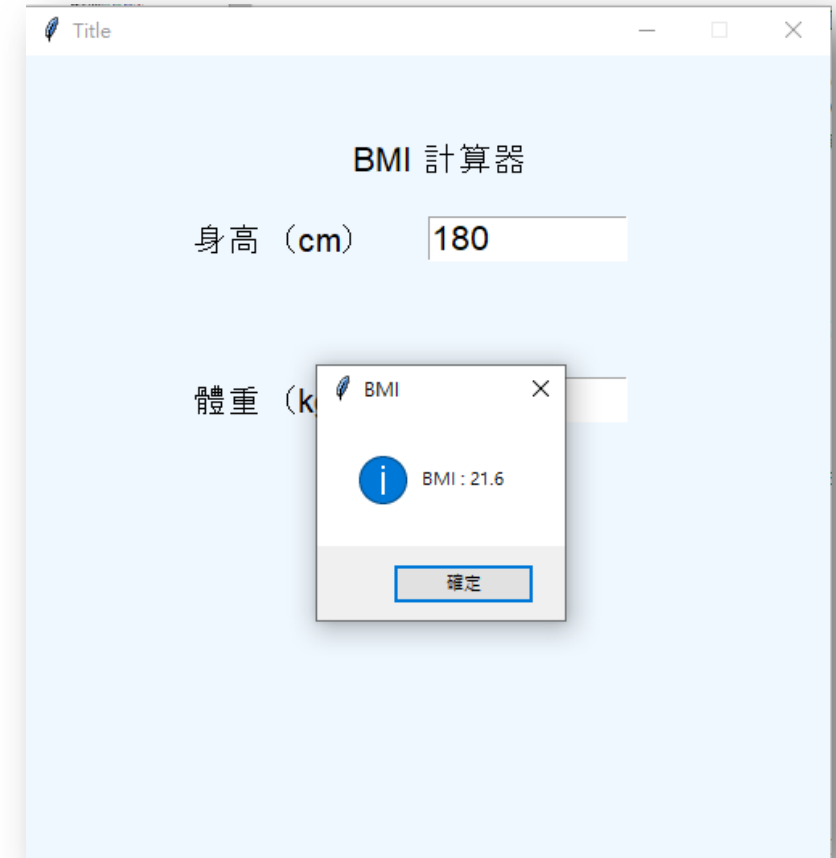
A screenshot of a Python GUI window titled "Title". It contains a label "BMI 計算器". Below it are two input fields: "身高 (cm)" with the value "180" and "體重 (kg)" with the value "70". At the bottom is a button labeled "計算".

BMI 計算器

身高 (cm) 180

體重 (kg) 70

計算

A screenshot of the same BMI Calculator GUI as on the left, but with a small "BMI" messagebox window overlaid. The messagebox has a blue information icon, the text "BMI : 21.6", and a "確定" (OK) button.

BMI 計算器

身高 (cm) 180

體重 (kg) 70

計算

BMI

BMI : 21.6

確定



其他

- Checkbutton
 - 勾選按鈕，只有兩種值
- Canvas
 - 圖形文件可以用來會置圖表和圖形
- Listbox
 - 一個可以選擇的列表
- Scrollbar
 - 配合canvas,listbox等等的滾動條



GUI on rabboni用途

- 方便debug
 - 運行演算法過程 結果顯示
- 方便呈現結果
 - 將結果以GUI方式顯示
 - 替換單調的文字顯示
- 方便執行
 - 將設定與輸入包在GUI內
 - 將GUI打包成EXE執行檔
 - 方便使用者使用

練習



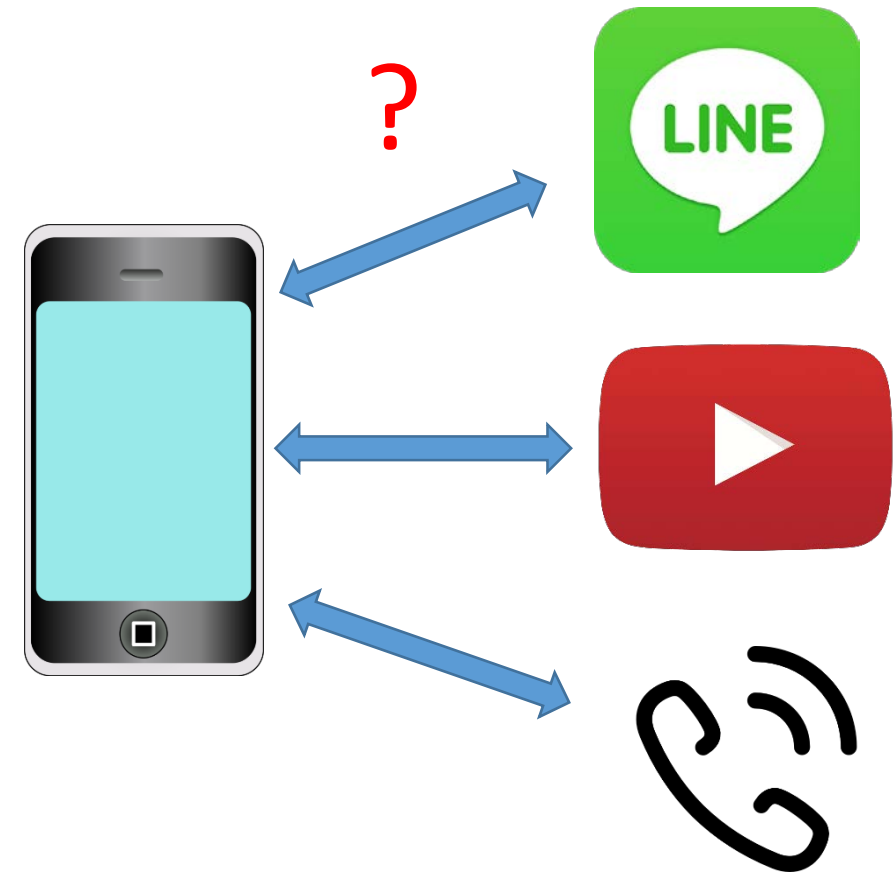
- 完成BMI App

Thread



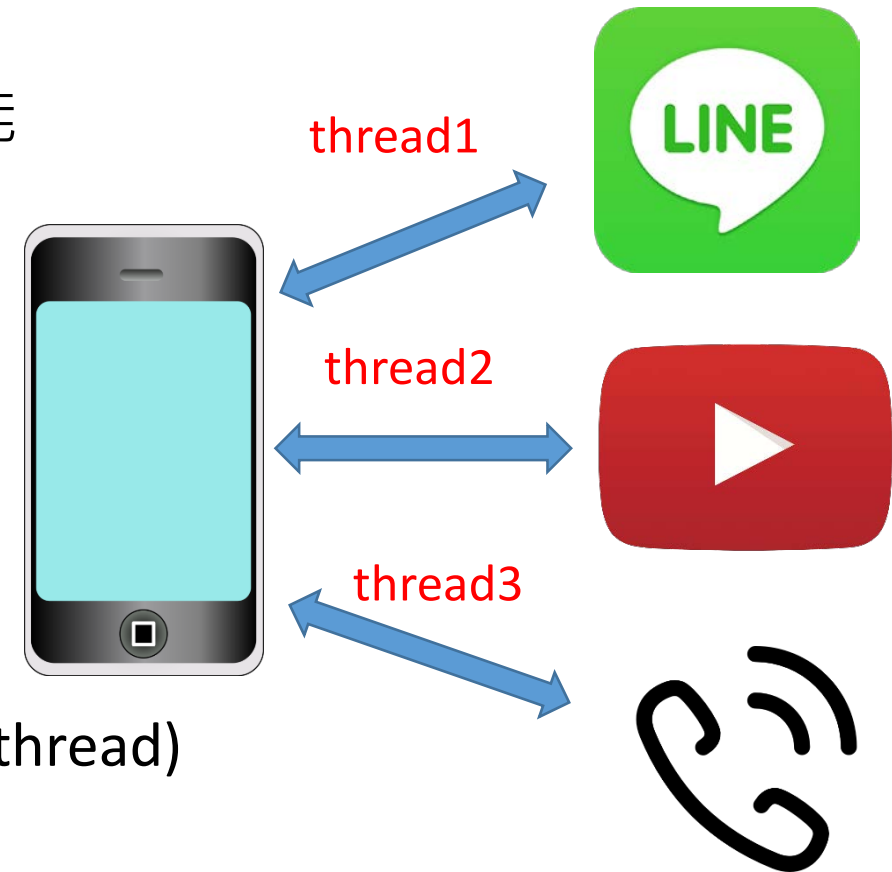
Thread?

- Thread 執行緒、線程
 - 多工
- 觀念
 - 手機擁有許多功能
 - 想同時時間執行
 - 打電話
 - LINE接收訊息
 - 背景播音樂
 - 如何執行?
 - 不同功能、APP、function各自開thread



Thread

- 用處
 - 可以在執行一個功能同時執行其他功能
 - GUI 介面本身占一個 thread
- GUI的問題
 - 按下button後，執行計算function
 - Function太複雜
 - **GUI介面卡住!!**
- 解決方法
 - 將function計算丟到“背景”去算(開新的thread)
 - 不佔到GUI介面的thread



將函數中改成複雜的函數

- 需要長時間計算的函數
 - 使用暫停時間模擬
- 第9行
 - 導入時間函式庫
- 第10~12行
 - 將原本的BMI公式改成模擬複雜的計算
 - `time.sleep(5)`
 - 執行到此行暫停五秒鐘
 - `print("Done")`
 - 停止五秒鐘後顯示
- 試試看會有什麼結果?
 - 按下按鈕後GUI介面卡住

```
9 | import time
10 | def calculate_bmi_number():
11 |     time.sleep(5)
12 |     print ("Done")
13 |
```



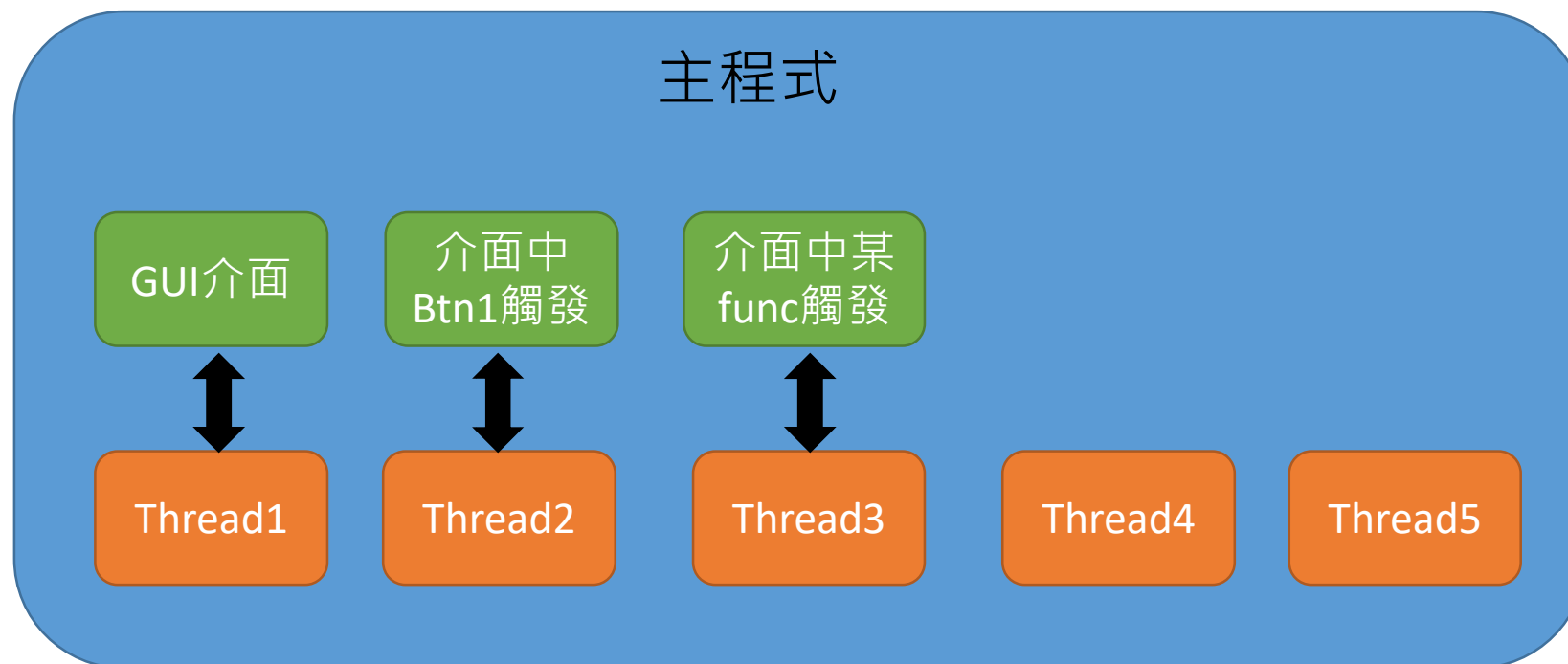
Thread實現

- 第7行
 - 先import 相關函式庫
- 14~19行
 - 建立一個thread function
 - 16:將丟進來的function、變數放入thread
 - 17:setDaemon(True)
 - 當主程序結束時是否將這個thread 一起結束?
 - 基本上都設為True，False可能會有不預期的結果(死循環)
 - 18:開始執行

```
7 | import threading

14 | def thread_it(func, *args):
15 |     # 創立
16 |     t = threading.Thread(target=func, args=args)
17 |     t.setDaemon(True)
18 |     t.start()
19 |     # t.join()
```

Thread 實現



Thread 實現

```
45 | calculate_btn = tk.Button(root, text='計算', command=lambda :thread_it(calculate_bmi_number), font=('Arial'
```

- 當Button觸發
 - 不希望事件用跟介面同一個thread執行，太複雜會讓介面卡住
 - 將計算的公式丟入thread執行
- 45行
 - Command = lambda : thread_it(...)
 - Lambda: 讓後方的function支援傳入參數功能
 - 因為我們需要將calculate function傳入 thread_it function
- 改為此函數後
 - GUI介面不卡住
 - 等待五秒後子線程完成

Rabboni GUI

Rabboni

MAC : **connect** Status : 0

Raw_hex:

AccX:

AccY:

AccZ:

GyrX:

GyrY:

GyrZ:





Rabboni

1 2

MAC : connect Status : 0

Raw_hex:

AccX:

AccY:

AccZ:

GyrX:

GyrY:

GyrZ:

流程圖



GUI thread

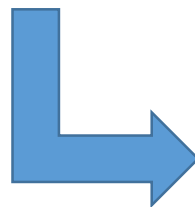
建立root視窗



按下button

連接、更新畫面thread

丟入
thread



依照MAC連接
讀取數值



將數值更新在介面上



如何更新畫面上數值？

- 使用StringVar()
- 第84行
 - 將Accx_var的數值設一個型態 tk.StringVar()
- 第85行
 - 將Accx這個label其中呈現的字串、數值設定為Accx_var
 - 之後只要改Accx_var 就會同步更改畫面的呈現

```
84 Accx_var = tk.StringVar()  
85 Accx_lab = tk.Label(root, textvariable=Accx_var, font=('Arial', 16), width=50, height=2)
```

Rabboni

MAC : **connect** Status : 0

Raw_hex:

AccX: **Accx_var**

AccY:

AccZ:

GyrX:

GyrY:

GyrZ:



將連接丟入thread



```
69 connect_button = tk.Button(root, text = "connect", command=lambda :thread_it(rabboni_creat), bg = "#6495ED", font = bold_16)
```

- 按下connect_button
 - 觸發藍芽的連接
 - 將連接的動作與讀數值丟給thread去執行

rabboni_creat()



- 第一周的內容
 - 將連接讀取數值放入此function
- 第22行
 - 將填入的mac碼讀出
- 第34~41行
 - 將讀取出的數值更新到相對應的變數

```
21 ∨ def rabboni_creat():
22     mac = mac_add_entry.get
23     print (mac)
24     rabo.connect(mac)#依照MAC連接
25     print (rabo.Status)
26     rabo.discover_characteristics()#掃描所有服務 可略過
27     rabo.print_char()#列出所有服務 可略過
28     rabo.read_data()#
29     status.config(text="Status : "+str(rabo.Status))
30 ∨ if rabo.Status == 1:
31 ∨     while True:#一直打印資料 直到結束程式
32         # rabo.print_data()#print資料
33
34         raw_var.set(rabo.Hex_data)
35         Accx_var.set(rabo.Accx)
36         Accy_var.set(rabo.Accy)
37         Accz_var.set(rabo.Accz)
38         Gyrx_var.set(rabo.Gyrx)
39         Gyry_var.set(rabo.Gyry)
40         Gyrz_var.set(rabo.Gyrz)
41         Cnt_var.set(rabo.Cur_Cnt)
42 ∨         if rabo.Status == 0:
43             break
44 ∨     else :
45         print ("未連接")
```

練習

- 嘗試看看修改程式碼
 - 顏色字體
 - 增加功能

