tkinter

tkinter介紹

tkinter是tk interface的縮寫, 而tk是一套跨平台的輕量GUI框架

簡單來說, tkinter是python 對於tk的一個轉接器

tkinter雖然是一個較為輕量的GUI開發工具, 但千萬別瞧不起他, 雖然沒有 "所見即所得"(直接拉來拉去) 的輔助介面, 但框架的設計簡潔簡單, 透過簡單的程式碼就可以快速的刻出簡潔大方地介面, 特別適合拿來快速開發簡單的 "圖形化小工具"

使用方法

tkinter的使用方法非常簡單, 但請大家一定要記得使用的秘訣

> Step 0. 創造Frame

> Step 1. 創造元件(指定放置的視窗)

> Step 2. 元件.排版()

元件

這裡列出一些常用的元件列表, 有一些我沒這麼常用的我就沒列出來了!

小元件	說明
Button	按鈕
Checkbutton	核取按鈕
Entry	文字輸入欄
Label	文字標籤
Listbox	列表選單
Menu	選單列的下拉式選單
MenuButton	選單的選項
Message	類似 Label ，可多行
OptionMenu	下拉式的選項選單
Radiobutton	單選按鈕
Scale	拉桿
Scrollbar	捲軸
Spinbox	微調器
Text	文字方塊

我最常使用的兩種Frame, 一種是最基本的Frame, 另外一種是帶標籤的Frame

Frame系列	說明
Frame	視窗
LabelFrame	文字標籤視窗

Frame

在使用tk的時候, 你要想像你的視窗有一個一個長方形(Frame), 請一定先把Frame創造出來, 再把元件放在合適的Frame上, 這樣你的排版才會比較輕鬆

排版

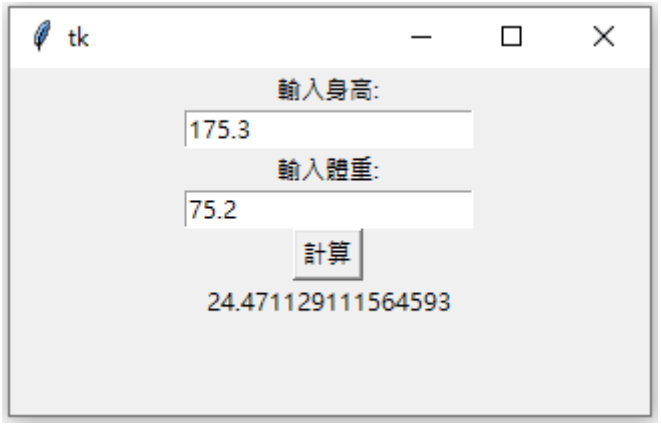
排版有三種方式, 一種是使用Grid(表格類型排版)和Pack(上->下/左->右)和Place(直接指定座標)

我會建議使用Pack排版較為好, 因為使用Grid或者Place的話經常會因為視窗變大變小而跑版

但是Pack要排的漂亮, 一定要配合Frame使用!

第一個 tkinter 程式

我們先來試一個比較簡單的 tkinter 程式, 這個程式會要求使用者輸入身高和體重, 並且幫他們計算BMI



In [ ]:

```python
import tkinter as tk

# Button需要連接一個命令, 來讓Button被點擊的時候有事做
def calculate():
    # 拿取兩個entry的值來做計算
    # 建議對e1, e2加上global修飾, 不然如果定義在上面會吃不到
    global e1, e2, result
    height, weight = e1.get(), e2.get()
    # 記得轉換成你要的型態
    height, weight = float(height), float(weight)
    bmi = weight / (height / 100) ** 2
    # 設定到結果裡面, 這裡我們需要動態設定text, 動態設定text的話使用[]做設定
    result["text"] = bmi

# 創造主視窗
window = tk.Tk()
# 可以透過 geometry 設置視窗大小
# 參數為 寬x高+右位移+下位移
window.geometry("500x500+300+300")

# 創造 Frame, 記得要指定老爸, 這時候老爸是主視窗(window)
f1 = tk.Frame(window)
# 立刻排版, pack預設是上到下
f1.pack()

# Label是單行標籤, 老爸是Frame喔, text是Label顯示的文字
l1 = tk.Label(f1, text="輸入身高:")
l1.pack()
# Entry是單行輸入
e1 = tk.Entry(f1)
e1.pack()

l2 = tk.Label(f1, text="輸入體重:")
l2.pack()

e2 = tk.Entry(f1)
e2.pack()

# Button是按鈕, command是連結的指令, 記得不要加(), 因為是點擊以後才執行
b1 = tk.Button(f1, text="計算", command=calculate)
b1.pack()

# 秀出result
result = tk.Label(f1, text="BMI結果")
result.pack()


# GUI程式事實上是一個永不結束的程式
# mainloop就是一個無窮迴圈, 直到你按下x才程式結束
window.mainloop()
```

物件導向

一般人的tkinter程式就長的上面這樣, 然後就會開始抱怨, 當程式出錯的時候根本找不到出錯的函式或者元件在哪裡? 又或者是想要修改一個元件的外觀, 卻根本找不到? 而且還有很多醜醜的global

為何會這樣呢? 主要原因是你沒有把該模組化的東西模組化. 白話來說, 就是該放在一起的東西就該放在一起, 也就是所謂的 "物件導向"

我們一起來把上面的程式改成物件導向吧!

In [ ]:

```
import tkinter as tk

# 從tk.Frame裡繼承出我們自己的Frame
class BmiFrame(tk.Frame):
    # 之前我們在init Frame的時候有指定老爸
    # 這裡當然也要讓老爸可以帶進來, 所以有parent參數
    def __init__(self, parent):
        # 使用繼承的原本init
        tk.Frame.__init__(self, parent)
        # 注意, 接下來的元件老爸是我們自己這個frame
        # 並且要把這些元件登記在我們的欄位底下
        self.l1 = tk.Label(self, text="輸入身高:")
        self.l1.pack()
        self.e1 = tk.Entry(self)
        self.e1.pack()
        self.l2 = tk.Label(self, text="輸入體重:")
        self.l2.pack()
        self.e2 = tk.Entry(self)
        self.e2.pack()
        # 這裡我也把calculate登記在我自己底下了
        self.b1 = tk.Button(self, text="計算", command=self.calculate)
        self.b1.pack()
        self.result = tk.Label(self, text="BMI結果")
        self.result.pack()

    # calculate當然也是我底下的一份子
    def calculate(self):
        # 你發現我們再也不用global了, 可以直接使用self.
        height, weight = self.e1.get(), self.e2.get()
        height, weight = float(height), float(weight)
        bmi = weight / (height / 100) ** 2
        self.result["text"] = bmi

window = tk.Tk()
window.geometry("500x500+300+300")

f1 = BmiFrame(window)
f1.pack()

window.mainloop()
```

美化一下

你會發現, 經過物件導向, 你把該收納在一起的東西收在一起了, 但你可能覺得介面還是有點醜, 沒關係, 我們來美化一下

In [ ]:

```
import tkinter as tk

# 我改使用LabelFrame, 並且設置text
class BmiFrame(tk.LabelFrame):
    def __init__(self, parent):
        tk.LabelFrame.__init__(self, parent, text="BMI")
        self.l1 = tk.Label(self, text="輸入身高:")
        self.l1.pack(expand=True, fill=tk.BOTH, padx=10, pady=10)
        self.e1 = tk.Entry(self)
        self.e1.pack(expand=True, fill=tk.BOTH, padx=10, pady=10)
        self.l2 = tk.Label(self, text="輸入體重:")
        self.l2.pack(expand=True, fill=tk.BOTH, padx=10, pady=10)
        self.e2 = tk.Entry(self)
        self.e2.pack(expand=True, fill=tk.BOTH, padx=10, pady=10)
        self.b1 = tk.Button(self, text="計算", command=self.calculate)
        self.b1.pack(expand=True, fill=tk.BOTH, padx=10, pady=10)
        self.result = tk.Label(self, text="BMI結果")
        self.result.pack(expand=True, fill=tk.BOTH, padx=10, pady=10)

    def calculate(self):
        height, weight = self.e1.get(), self.e2.get()
        height, weight = float(height), float(weight)
        bmi = weight / (height / 100) ** 2
        self.result["text"] = bmi


window = tk.Tk()
window.geometry("500x500+300+300")

# expand=True: 會隨著視窗變大變小調整位置
# fill=tk.BOTH: 視窗變大變小的時候元件也會隨著變大, 填滿視窗左右和上下
# padx, pady: 對外面做一個空間
f1 = BmiFrame(window)
f1.pack(expand=True, fill=tk.BOTH, padx=10, pady=10)

window.mainloop()
```

多執行緒

但我們這個程式有一個很大的缺陷, 如果這個calculate是一個執行很久的功能, 你會發現我們的GUI會卡住, 因為你的程式無法處理其他的GUI活動了, 他只能等這個很久的功能做完, 才能開始處理別的事, 所以在我們的GUI設計裡, 有一個鐵則中的鐵則

> 多執行緒: 一個程式可以同時雇用多個工人來處理不同的事, 我們就叫做多執行緒

> 主執行緒(主要工人)只處理GUI的活動(點擊,輸入...), 非GUI的活動就要起一個額外的執行緒來操作

In [ ]:

```python
import tkinter as tk
import threading
import time

class BmiFrame(tk.LabelFrame):
    def __init__(self, parent):
        tk.LabelFrame.__init__(self, parent, text="BMI")
        self.l1 = tk.Label(self, text="輸入身高:")
        self.l1.pack(expand=True, fill=tk.BOTH, padx=10, pady=10)
        self.e1 = tk.Entry(self)
        self.e1.pack(expand=True, fill=tk.BOTH, padx=10, pady=10)
        self.l2 = tk.Label(self, text="輸入體重:")
        self.l2.pack(expand=True, fill=tk.BOTH, padx=10, pady=10)
        self.e2 = tk.Entry(self)
        self.e2.pack(expand=True, fill=tk.BOTH, padx=10, pady=10)
        self.b1 = tk.Button(self, text="計算", command=self.calculate)
        self.b1.pack(expand=True, fill=tk.BOTH, padx=10, pady=10)
        self.result = tk.Label(self, text="BMI結果")
        self.result.pack(expand=True, fill=tk.BOTH, padx=10, pady=10)

    def calculate(self):
        def work():
            # 利用sleep來模擬一個長時間的功能
            time.sleep(5)
            height, weight = self.e1.get(), self.e2.get()
            height, weight = float(height), float(weight)
            bmi = weight / (height / 100) ** 2
            self.result["text"] = bmi
        # 還沒要執行, 別加(), 而且記得要使用start, 而非run
        threading.Thread(target=work).start()

window = tk.Tk()
window.geometry("500x500+300+300")

f1 = BmiFrame(window)
f1.pack(expand=True, fill=tk.BOTH, padx=10, pady=10)

window.mainloop()
```