

程式設計 (二)

視窗應用程式開發

Ming-Hung Wang 王銘宏

tonymhwang@cs.ccu.edu.tw

Department of Computer Science and Information Engineering
National Chung Cheng University

Spring Semester, 2022

本章目錄

1. 介紹
2. Label 標籤元件
3. Button 標籤元件
4. messagebox 對話方塊元件
5. Entry 文字方塊元件
6. 事件處理函式

視窗應用程式

圖形使用者介面 (GUI, graphical user interface)

視窗 (window) 為最基本元件，所有使用者操作介面之元素都安置在此，
例如顯示文字訊息的標籤、執行特定動作的按鈕…等。

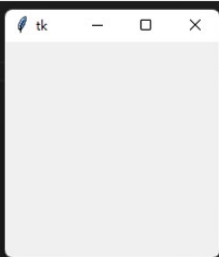
Tools Kit Interface 的簡寫
為 Python 標準 GUI 工具，安裝 Python 時會預設安裝
直接匯入即可使用 =>

```
import tkinter as tk
```

如何建立視窗

執行 tkinter 套件的 Tk() 建立視窗物件
執行 mainloop() 方法使視窗不斷運作，直到被關閉為止

```
1 import tkinter as tk
2
3 win = tk.Tk()
4 win.mainloop()
```



視窗常用屬性與方法

設定視窗屬性質，來改變視窗外觀及特性:

方法/屬性	說明
<code>title()</code>	設定標題文字，預設為「tk」
<code>mainloop()</code>	使視窗不斷運作，直到被關閉為止
<code>geometry()</code>	設定視窗的寬度、高度、X 座標、Y 座標 語法: 視窗名稱. <code>geometry('寬 x 高 +X+Y')</code> ，其中 x 為小寫 X 例如: <code>win.geometry('400x500+100+200')</code>
<code>maxsize()</code>	設定視窗可接受最大寬度及高度
<code>minsize()</code>	設定視窗可接受最小寬度及高度
<code>resizable()</code>	設定視窗是否可供調整寬度及高度

視窗常用屬性與方法

方法/屬性	說明
iconbitmap()	設定視窗圖示，可採用 ICO 圖檔格式
bg 或 background	設定視窗背景色屬性，可以使用 configure() 方法 例如:win.configure(bg='blue')
state()	可以設置視窗狀態，參數 'zoomed' 為最大化; 'icon' 為最小化; 'normal' 則為正常顯示 例如:win.state('zoomed')
iconify()	設定視窗最小化，等同 state('icon')
winfo_screenwidth()	程式執行時取得螢幕的寬度，單位為像素
winfo_screenheight()	程式執行時取得螢幕的高度，單位為像素

tkinter 套件提供多種元件 (widget)，每個元件皆有特定之功能以完成指定工作。

例如 Label 標籤元件可以顯示文字訊息；

Button 元件可以供使用者點按來執行特定功能；

Entry 文字方塊可接受使用者輸入的文數字資料。

其他元件如: messagebox、Scale、Spinbox、Scrollbar...

tkinter 套件常用元件-Label 標籤元件

使用 Label 標籤元件可以提供文字訊息，但無法接受使用者輸入。

語法:

```
物件變數 = tkinter.Label(容器物件, 參數 1= 值 1, 參數 2= 值 2,...)  
物件變數.pack()
```

pack() 為元件版面配置的方法之一，可將元件安置到視窗中。

tkinter 套件常用元件-Label 標籤元件

Label 元件常用屬性 (參數):

方法/屬性	說明
text	設定元件內的文字
width	設定元件寬度，單位為字元
height	設定元件高度，單位為字元
bg 或 background	設定元件背景顏色
fg 或 foreground	設定元件前景顏色，在 Label 即是指文字顏色
padx	設定元件內文字或內容與邊緣的水平間距，單位為像素
pady	設定元件內文字或內容與邊緣的垂直間距，單位為像素
font	設定元件字形、樣式、大小

tkinter 套件常用元件-Label 標籤元件

方法/屬性	說明
bitmap	設定元件顯示內建圖示，參數值如 'error'、'gray75'、'info'...
compound	設定元件圖示相對於文字的位置，參數值為 'left'、'right'、'top'、'bottom'、'center'
relief	設定元件邊框樣式，參數值如 'flat'(預設)、'groove'、'raised'...
anchor	設定元件的文字或內容在元件的位置，參數值為 'e'、'w'、'n'、's'、'ne'、'se'、'nw'、'sw'、'center'(預設)
cursor	設定元件的滑鼠指標樣式，參數值如 'arrow'(預設)、'circle'、'clock'、'cross'...
wraplength	設定標籤文字自動換行的寬度
justify	設定標籤文字最後一行的對齊位置，參數值為 'left'、'right'、'center'

以上介紹之 Label 元件屬性，除了 wraplength 及 justify 為專屬 Label 元件外，其餘屬性在各元件大都通用

tkinter 套件常用元件-Label 標籤元件

範例程式與結果:

```
1 import tkinter as tk
2
3 win = tk.Tk()
4 win.geometry('300x200')
5 win.title('Label')
6 win.configure(bg = 'blue')
7 tk.Label(win, text='Oh my God', fg='red',bg = 'yellow',bitmap='warning',\
8         compound='right',font=('Times',24,'bold')).pack()
9 msg = ('我的天我的天我的天我的天我的天我的天我的天我的天我的天我的天')
10 tk.Label(win,text = msg, width = 28, wraplength= 240, justify= 'left',\
11         pady =10, font = ('標楷體',14)).pack()
12 win.mainloop()
```



tkinter 套件常用元件-Button 按鈕元件

Button 按鈕元件是視窗應用程式常見的元件，介面中常常提供多個按鈕，讓使用者點選來執行特定的功能。

語法:

物件變數 = tkinter.Button(容器物件, 參數...)

簡例:

btnOK = tk.Button(win, text = '確定', command = fnOK)

tkinter 套件常用元件-Button 按鈕元件

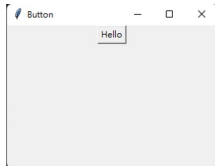
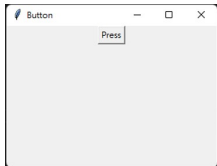
Button 元件許多參數跟 Label 相同，下表列出 Button 元件常用參數：

方法/屬性	說明
command	當按下按鈕時，會執行 command 所指定的函式
activebackground	設定按下按鈕時按鈕的背景顏色
activeforeground	設定按下按鈕時按鈕的文字前景顏色
state	設定按鈕元件是否能使用，參數值為 'normal'(可以使用，預設)、'disabled'(不能使用，按鈕文字以灰色顯示)
underline	指定在按鈕上的文字加底線，參數值為-1(不加，預設)、0(在第一個字加)、1(在第二個字加)...

tkinter 套件常用元件-Button 按鈕元件

簡例:

```
1  import tkinter as tk
2
3  def fnHi():
4      btnHi['text'] = 'Hello'
5  win = tk.Tk()
6  win.geometry('300x200')
7  win.title('Button')
8  btnHi = tk.Button(win, text = 'Press', command=fnHi)
9  btnHi.pack()
10 win.mainloop()
```



tkinter 套件常用元件-Button 按鈕元件

要在程式執行階段改變元件屬性值時，該元件配置時必須獨力一行敘述，不可在建立元件同時配置：

```
1 import tkinter as tk
2
3 def fnHi():
4     btnHi['text'] = 'Hello'
5 win = tk.Tk()
6 win.geometry('300x200')
7 win.title('Button')
8 btnHi = tk.Button(win, text = 'Press', command=fnHi).pack()
9 win.mainloop()
```

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\donal\anaconda3\lib\tkinter\__init__.py", line 1892, in __call__
    return self.func(*args)
  File "c:/Users/donal/Desktop/NFT/tesy.py", line 4, in fnHi
    btnHi['text'] = 'Hello'
TypeError: 'NoneType' object does not support item assignment
```


tkinter 套件常用元件-Button 按鈕元件

範例程式-簡易計數器:

```
1 import tkinter as tk
2 num = 0
3 def fnAdd():
4     global num        #宣告為全域變數
5     num += 1          #num加1
6     lblNum['text']=str(num) #重設標籤文字
7     if (num>0):        #若num大於0，就設歸零鈕可以使用
8         btnClear['state']='normal'
9
10 def fnClear():
11     global num
12     num = 0
13     lblNum['text']=str(num)
14     btnClear['state']='disabled' #設歸零鈕不能使用
15
16 win=tk.Tk()
17 win.geometry('300x200')
18 win.title('計數器')
19 win.configure(bg='white')
20 lblTitle=tk.Label(win, text = '計數器',font=('標楷體', 16),fg='white',bg='blue')
21 lblNum=tk.Label(win, text = '0',font=('微軟正黑體', 36))
22 btnAdd=tk.Button(win, text = '加 1',pady=5,padx=10,command=fnAdd)
23 btnClear=tk.Button(win, text = '歸零',pady=5,padx=10,command=fnClear,state='disabled')
24 lblTitle.pack(pady=10,fill='x')
25 lblNum.pack(pady=20,fill='x')
26 btnAdd.pack(pady=5, side='left',fill='x', expand=True)
27 btnClear.pack(pady=5, side='left',fill='x', expand=True)
28 win.mainloop()
```



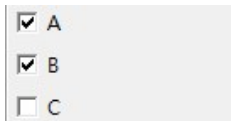
tkinter 套件常用元件-Button 按鈕元件

其他按鈕元件如 Radiobutton 選項按鈕元件:



A screenshot of a Tkinter Radiobutton widget. It contains two radio buttons. The first radio button is selected (filled with a black dot) and is followed by the text "參加". The second radio button is not selected (empty circle) and is followed by the text "不參加".

Checkbox 核取按鈕元件:



A screenshot of a Tkinter Checkbox widget. It contains three checkboxes arranged vertically. The first checkbox is checked (has a black checkmark) and is followed by the letter "A". The second checkbox is checked (has a black checkmark) and is followed by the letter "B". The third checkbox is not checked (empty box) and is followed by the letter "C".

上述兩種主要的差別在於 Radiobutton 元件只能從中擇一選取，而 Checkbox 元件則是每一個項目都能獨立勾選。

tkinter 套件常用元件-messagebox 對話方塊元件

messagebox 對話方塊元件可以提供顯示訊息的對話方塊，讓使用者了解程式的提示訊息，也可從中選取按鈕做回應。

語法:

變數 = tkinter.messagebox. 方法名稱 (title, message, 參數...)

建立 messagebox 對話方塊元件，不需用 pack() 等配置方法。
使用者在對話方塊中所選取之按鈕，會回傳至變數中。

tkinter 套件常用元件-messagebox 對話方塊元件

messagebox 元件顯示對話方塊可用的方法:

方法/屬性	說明
askokcancel	對話方塊提供「確定」及「取消」鈕，回傳值分別為 True 及 False
askquestion	對話方塊提供「是」及「否」鈕，回傳值分別為 'yes' 及 'no'
askretrycancel	對話方塊提供「重試」及「取消」鈕，回傳值分別為 True 及 False
askyesno	對話方塊提供「是」及「否」鈕，回傳值分別為 True 及 False
askyesnocancel	對話方塊提供「是」、「否」及「取消」鈕，回傳值分別為 True、False 及 None
showinfo	對話方塊提供「確定」鈕，回傳值為 'ok'
showerror	對話方塊提供「確定」鈕，回傳值為 'ok'
showwarning	對話方塊提供「確定」鈕，回傳值為 'ok'

tkinter 套件常用元件-messagebox 對話方塊元件

messagebox 元件常用的參數:

方法/屬性	說明
title	設定對話方塊的標題文字
message	設定對話方塊的訊息文字
icon	設定對話方塊的圖示，參數值為 'error'、'info'、'question'、'warning'
parent	設定對話方塊關閉後，顯示的作用視窗

tkinter 套件常用元件-messagebox 對話方塊元件

範例程式與結果:

```
1 import tkinter as tk
2 from tkinter import messagebox
3
4 def fnEnd():
5     res = tk.messagebox.askokcancel('注意','確定要結束程式嗎?')
6     if(res == True):      #若傳回值為 True
7         win.destroy()    #結束程式執行
8
9 win=tk.Tk()
10 win.geometry('300x150')
11 win.title('對話方塊')
12 lblTitle=tk.Label(win, text = '按鈕結束程式',font=('標楷體', 16))
13 btnEnd=tk.Button(win, text = '結束',pady=5,padx=10,command=fnEnd)
14 lblTitle.pack(pady=20)
15 btnEnd.pack(pady=5)
16 win.mainloop()
17
18
```



tkinter 套件常用元件-Entry 文字方塊元件

Entry 文字方塊元件可以接受使用者輸入的文字資料。

Entry 元件可以輸入、編輯和顯示文字資料，即具有讀寫功能，
不過只能接受單行文字。

語法:

物件變數 = tkinter.Entry(容器物件, 參數...)

簡例: entPw = tk.Entry(win, show = ‘#’)

tkinter 套件常用元件-Entry 文字方塊元件

Entry 文字方塊元件常用屬性:

參數	說明
state	設定是否能編輯元件的文字，參數值為 'normal'(可以輸入預設)、'disabled'(只能顯示文字)
textvariable	設定元件內文字的變數名稱
show	設定替代字元，常用於密碼輸入，例如參數值設為 '*', 使用者輸入文字時會顯示為「*」

tkinter 套件常用元件-Entry 文字方塊元件

在程式執行階段若要存取 Entry 文字內容，需要透過 textvariable 屬性，而首先要宣告一文字方塊變數元件。

簡例 1:

```
score = tk.IntVar()  
entScore = tk.Entry(win, textvariable = score)
```

簡例 2: name = tk.StringVar()

```
entName = tk.Entry(win, textvariable = name)
```

指定 textvariable 屬性後，可以使用 set() 方法來指定變數值，並且會顯示在 Entry 元件中。

語法:

變數物件.set(變數值)

簡例: `city = tk.StringVar()`
`entCity = tk.Entry(win, textvariable = city)`
`city.set('台北市')`

tkinter 套件常用元件-Entry 文字方塊元件

要取得使用者在 Entry 元件中輸入的資料，可以使用 `get()` 方法

語法:

變數物件.get(變數值)

簡例:

```
rate = tk.DoubleVar()
```

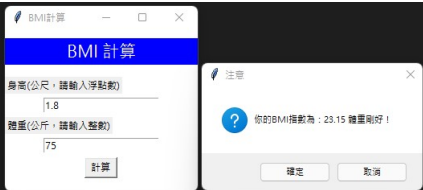
```
entRate = tk.Entry(win, textvariable = rate)
```

```
irate = rate.get()
```

tkinter 套件常用元件-Entry 文字方塊元件

範例程式與結果:

```
1 import tkinter.messagebox
2 import tkinter as tk
3
4 def fnBmi():
5     height = userH.get()    #用get方法取得身高
6     weight = userW.get()
7     bmi = round(weight / pow(height, 2), 2)
8     msg = ''
9     if bmi < 18.5:
10         msg = '體重過輕!'
11     elif bmi >= 24:
12         msg = '體重過重!'
13     else:
14         msg = '體重剛好!'
15     tk.messagebox.askokcancel('注意', '你的BMI指數為{:} {} {}'.format(bmi, msg))
16
17 win = tk.Tk()
18 win.title('BMI計算')
19 win.geometry('240x200')
20 win.configure(bg='white')
21 lblTitle = tk.Label(win, text='BMI 計算', font=('微軟正黑體', 16), fg='white', bg='blue')
22 lblTitle.pack(pady=10, fill='x')
23 tk.Label(win, text='身高(公尺, 請輸入浮點數)').pack(pady=5, anchor='w')
24 userH = tk.DoubleVar()    #宣告userH為浮點數物件
25 entH = tk.Entry(win, textvariable=userH).pack()    #textvariable參數值為userH
26 tk.Label(win, text='體重(公斤, 請輸入整數)').pack(pady=5, anchor='w')
27 userW = tk.IntVar()    #宣告userW為整數物件
28 entW = tk.Entry(win, textvariable=userW).pack()    #textvariable參數值為userW
29 btnCal = tk.Button(win, text='計算', command=fnBmi).pack(pady=5)
30 win.mainloop()
```



事件處理函式

事件 (event) 是物件受到外來因素影響，而發生的某種動作。
除了與用元件的 `command` 參數來指定事件處理函式外，
也可以使用 `bind()` 方法來綁定事件處理函式。

語法:

元件物件.`bind`(事件, 事件處理函式)

簡例:

`btnOK.bind('<Button-1>', fnClick)`

綁定事件後，若不再觸發事件時，可以用 `unbind()` 取消綁定。

語法:

元件物件.`unbind`(事件)

常用滑鼠事件:

事件	說明
<Button>	使用者按一下任意滑鼠鍵時所觸發的事件
<Button-1>	使用者按一下滑鼠左鍵時會觸發 <Button-1> 事件
<Button-2>	使用者按一下滑鼠中鍵時會觸發 <Button-2> 事件
<Button-3>	使用者按一下滑鼠右鍵時會觸發 <Button-3> 事件
<Double-Button-1>	使用者快按兩下滑鼠左鍵時會觸發 <Double-Button-1> 事件
<Double-Button-2>	使用者快按兩下滑鼠中鍵時會觸發 <Double-Button-2> 事件
<Double-Button-3>	使用者快按兩下滑鼠右鍵時會觸發 <Double-Button-3> 事件

事件處理函式

事件	說明
<ButtonRelease-1>	使用者放開滑鼠左鍵時會觸發 <ButtonRelease-1> 事件
<ButtonRelease-2>	使用者放開滑鼠中鍵時會觸發 <ButtonRelease-2> 事件
<ButtonRelease-3>	使用者放開滑鼠右鍵時會觸發 <ButtonRelease-3> 事件
<Enter>	當滑鼠指標進入元件時所觸發的事件
<Leave>	當滑鼠指標離開元件時所觸發的事件
<Motion>	當滑鼠指標在元件中移動時所觸發的事件
<B1-Motion>	當按住滑鼠左鍵拖曳滑鼠指標時會觸發 <B1-Motion> 事件
<B2-Motion>	當按住滑鼠中鍵拖曳滑鼠指標時會觸發 <B2-Motion> 事件
<B3-Motion>	當按住滑鼠右鍵拖曳滑鼠指標時會觸發 <B3-Motion> 事件

事件處理函式

建立滑鼠事件處理函式時，事件處理函式語法如下：

語法：

```
def 事件處理函式名稱 (事件參數)
```

語法中事件參數是必要參數，因為事件觸發時會將相關訊息，透過該參數傳遞給事件處理函式。

事件處理函式

範例程式與結果:

```
1 import tkinter as tk
2
3 def fnEnter(event):
4     lblTest['bg']='lightblue'
5
6 def fnLeave(event):
7     lblTest.config(text='試試看',bg='gray')
8
9 def fnMotion(event):
10    lblTest['text']='游標移動'
11
12 def fnClick(event):
13    global mx,my    #宣告mx,my為全域變數
14    mx=event.x      #紀錄按下時滑鼠游標的x坐標
15    my=event.y      #紀錄按下時滑鼠游標的y坐標
16
17 def fnB1Motion(event):
18    global mx,my    #宣告mx,my為全域變數
19    lblX=lblTest.winfo_rootx()-win.winfo_rootx()    #計算lblTest在視窗的x坐標
20    lblY=lblTest.winfo_rooty()-win.winfo_rooty()    #計算lblTest在視窗的y坐標
21    lblTest['text']='拖曳中...'
22    lblTest.place(x=lblX+(event.x-mx),y=lblY+(event.y-my))    #重設lblTest位置
23
24 win = tk.Tk()
25 win.title('滑鼠事件測試')
26 win.geometry('240x240')
27 mx=0
28 my=0
29 lblTest=tk.Label(win,text='試試看',width=8,height=2,relief='groove',bg='gray')
30 lblTest.place(x=80,y=100)
31 lblTest.bind('<Enter>',fnEnter)    #<Enter>事件綁定fnEnter事件處理函式
32 lblTest.bind('<Leave>',fnLeave)    #<Leave>事件綁定fnLeave事件處理函式
33 lblTest.bind('<Motion>',fnMotion)    #<Motion>事件綁定fnMotion事件處理函式
34 lblTest.bind('<Button-1>',fnClick)    #<Button-1>事件綁定fnClick事件處理函式
35 lblTest.bind('<B1-Motion>',fnB1Motion)    #<B1-Motion>事件綁定fnB1Motion事件處理函式
36 win.mainloop()
```

範例程式與結果：



事件處理函式

鍵盤事件：當使用者按鍵盤的任一個鍵時，會觸發作用元件的鍵盤相關事件，在事件處理函式中可做適當處理。

當某個元件取得 focus 時，就成為使用中的元件，簡稱「作用元件」。
要使元件成為作用元件，程式中使用 `focus_set()` 方法。

語法：

元件名稱.`focus_set()`

常用鍵盤事件:

事件	說明
<Key>	當使用者按下鍵盤鍵時所觸發的事件
< 按鍵 >	當使用者按下指定按鍵時所觸發的事件
<Alt-按鍵名稱 >	使用者按下指定的 'Alt' 組合按鍵時所觸發的事件
<Control-按鍵名稱 >	使用者按下指定的 'Ctrl' 組合按鍵時所觸發的事件
<Shift-按鍵名稱 >	使用者按下指定的 'Shift' 組合按鍵時所觸發的事件
<KeyPress>	當使用者按下鍵盤鍵時所觸發的事件
<KeyRelease>	當使用者放開鍵盤鍵時所觸發的事件
<FocusIn>	當元件取得焦點時所觸發的事件
<FocusOut>	當元件失去焦點時所觸發的事件

事件處理函式

簡例 1:

```
win.bind('<Key>', fnKeys)
```

簡例 2:

```
btnOK.focus_set()  
btnOK.bind('<a>', fnAClick)
```

簡例 3:

```
win.bind('<Alt-Up>', fnKeys)
```

簡例 4:

```
btnOK.bind('<Return>', fnEnter)
```

事件處理函式

鍵盤事件處理函式中的事件參數，所傳遞的資料主要為 widget、char、keycode、keysym、type。

下表為常用按鍵的 keycode 及 keysym：

鍵盤上的按鍵	keycode(鍵盤碼)	keysym(按鍵名稱)
0~9	48~57	0~9
A~Z	65~90	a~z、A~Z
F1~F12	112~123	F1~F12
←、→	37、39	Left、Right
↑、↓	38、40	Up、Down
Backspace、Del	8、46	BackSpace、Delete
Enter↵、空白鍵	13、32	Return、space
⇧Shift	16	Shift_L、Shift_R
Esc	27	Escape

事件處理函式

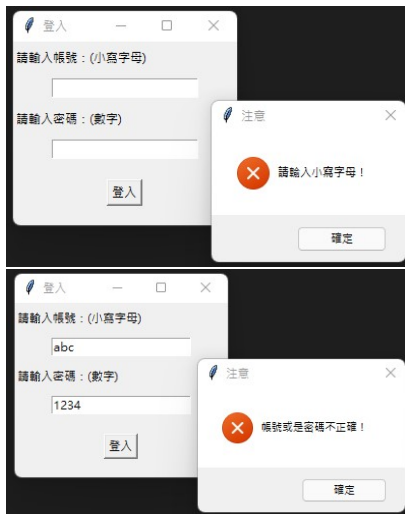
範例程式與結果:

```
1 import tkinter as tk
2 import tkinter.messagebox
3 def fnID(e):
4     code=e.keycode      #取得字元的鍵盤碼
5     if(code==8 or code==46):  #若是退位或刪除鍵就離開函式
6         return
7     if(e.keysym=='Return'):  #若是 Enter 鍵
8         entPW.focus_set()    #設entPW成為作用元件
9         return
10    id=userID.get()         #取得帳號字串
11    if(e.char.islower()==False):  #若輸入字元不是小寫字元
12        userID.set(id.replace(e.char,'')) #重設帳號字串將輸入字元以空字串取代
13        tk.messagebox.showerror('注意','請輸入小寫字母!')
14
15 def fnPW(e):
16     sym=e.keysym         #取得字元的按鍵名稱
17     if(sym=='BackSpace' or sym=='Delete'):  #若是退位或刪除鍵就離開函式
18         return
19     pw=userPW.get()
20     if(e.char.isdigit()==False):  #若輸入字元不是數字
21        userPW.set(pw.replace(e.char,'')) #重設密碼字串將輸入字元以空字串取代
22        tk.messagebox.showerror('注意','請輸入數字!')
```

範例程式與結果:

```
24 def fnCheck():
25     id=userID.get()
26     pw=userPW.get()
27     if (id == 'love' and pw == '1314'):#若帳號和密碼字符串都正確
28         tk.messagebox.showinfo('歡迎','帳號和密碼正確!')
29         win.destroy()
30     else:
31         tk.messagebox.showerror('注意','帳號或是密碼不正確!')
32         userID.set('') #清空帳號字符串
33         userPW.set('') #清空密碼字符串
34         entID.focus_set() #設entID成為作用元件
35
36 win = tk.Tk()
37 win.title('登入')
38 win.geometry('220x180')
39 tk.Label(win, text = '請輸入帳號:(小寫字母)').pack(anchor='w',pady=5)
40 userID=tk.StringVar()
41 entID= tk.Entry(win,textvariable=userID)
42 entID.pack(pady=5)
43 entID.bind('<KeyRelease>',fnID) # KeyRelease事件綁定fnID事件處理函式
44 entID.focus_set()
45 tk.Label(win, text = '請輸入密碼:(數字)').pack(anchor='w',pady=5)
46 userPW=tk.StringVar()
47 entPW= tk.Entry(win,textvariable=userPW)
48 entPW.pack(pady=5)
49 entPW.bind('<KeyRelease>',fnPW) # KeyRelease事件綁定fnPW事件處理函式
50 btnLogin = tk.Button(win, text='登入', command=fnCheck).pack(pady=15)
51 win.mainloop()
```


範例程式與結果：



事件處理函式

lambda 運算式: lambda 運算式是一種匿名的函式，可以依不同的引數值，來傳回運算結果，使運算結果可以重複運用。

有多個元件的事件處理函式程式碼類似，只有變數值不同時就可使用。

下面是 Button 元件的 command 參數使用語法：

語法：

```
def 函式名稱 (參數串列):  
    函式敘述區段
```

```
Button(容器, command = lambda: 函數名稱 (引數...)...)
```

事件處理函式

範例程式與結果:

```
1 import tkinter as tk
2 def fnKey(str):
3     global exp
4     exp+=str
5     lblExp.config(text=exp)
6
7 def fnCls():
8     global exp
9     exp=""
10    lblExp.config(text=exp)
11
12 def fnCal():
13     global exp
14     exp=str(eval(exp))
15    lblExp.config(text=exp)
16
17 win = tk.Tk()
18 win.title('簡易計算機')
19 win.geometry('180x140')
20 lblExp=tk.Label(win,text='',width=18,relief='raised',bg='yellow')
21 lblExp.grid(row=0,column=0,columnspan=4)
22 tk.Button(win,text='7',width=3,command=lambda:fnKey('7')).grid(row=1,column=0)
23 tk.Button(win,text='8',width=3,command=lambda:fnKey('8')).grid(row=1,column=1)
24 tk.Button(win,text='9',width=3,command=lambda:fnKey('9')).grid(row=1,column=2)
25 tk.Button(win,text='/',width=3,command=lambda:fnKey('/')).grid(row=1,column=3)
26 tk.Button(win,text='4',width=3,command=lambda:fnKey('4')).grid(row=2,column=0)
27 tk.Button(win,text='5',width=3,command=lambda:fnKey('5')).grid(row=2,column=1)
28 tk.Button(win,text='6',width=3,command=lambda:fnKey('6')).grid(row=2,column=2)
29 tk.Button(win,text='*',width=3,command=lambda:fnKey('*')).grid(row=2,column=3)
30 tk.Button(win,text='1',width=3,command=lambda:fnKey('1')).grid(row=3,column=0)
31 tk.Button(win,text='2',width=3,command=lambda:fnKey('2')).grid(row=3,column=1)
32 tk.Button(win,text='3',width=3,command=lambda:fnKey('3')).grid(row=3,column=2)
33 tk.Button(win,text='-',width=3,command=lambda:fnKey('-')).grid(row=3,column=3)
34 tk.Button(win,text='0',width=3,command=lambda:fnKey('0')).grid(row=4,column=0)
35 tk.Button(win,text='C',width=3,command=fnCls).grid(row=4,column=1)
36 tk.Button(win,text='=',width=3,command=fnCal).grid(row=4,column=2)
37 tk.Button(win,text='+',width=3,command=lambda:fnKey('+')).grid(row=4,column=3)
38 exp=""
39 win.mainloop()
```



參考資料：

資訊種子研究室. Python 全面攻略：從程式新人到開發設計的快速學習