程式設計期末

▼ 使用套件

```
import tkinter as tk
from tkinter import ttk
import mysql.connector as myconn
import socket
import threading
import hashlib
from PIL import ImageTk, Image
import PySimpleGUI as sg
from tkinter import messagebox
from tkinter import messagebox as msgbox
import tkinter.font as tkFont
from tkinter import filedialog
from datetime import datetime
import tkinter as tk
from tkinter import ttk
from PIL import Image, ImageTk
import mysql.connector as myconn
import os
import mysql.connector as myconn
from tkinter import messagebox
import datetime
import re
from tkinter import *
```

▼ 帳號登入

- 介面


```
window = tk.Tk()
window.title("使用者登入")
window.geometry("300x200")
window.rowconfigure(0,weight=1)
window.rowconfigure(1,weight=1)
window.rowconfigure(2,weight=1)
window.rowconfigure(3,weight=1)
window.rowconfigure(4,weight=1)
window.columnconfigure(0,weight=1)
window.configure(background='#69829b')
fontStyle = tkFont.Font( family="Lucida Grande",size=15)

label_username = tk.Label(window, text="使用者名稱:",foreground='white',background='#69829b',font=fontStyle)
label_username.grid(row=0,column=0,pady=1,padx=2)
entry_username = tk.Entry(window)
entry_username.grid(row=1,column=0,pady=1,padx=2)


label_password = tk.Label(window, text="密碼:",foreground='white',background='#69829b',font=fontStyle)
label_password.grid(row=2,column=0,pady=1,padx=2)
entry_password = tk.Entry(window, show="*")
```

```
entry_password.grid(row=3,column=0,pady=1,padx=2)

button_login = tk.Button(window, text="登入",foreground='white',background='#1e4f80',width=10, command=login)
button_login.grid(row=4,column=0,pady=1,padx=2)

window.mainloop()
```

- 登入函式(連結資料庫)

```
dict1={}
def login():
    username = entry_username.get()
    password = entry_password.get()

    # 連接到SQL資料庫
    dbConn=myconn.connect(
        host="localhost",
        user="Kuo",
        password="D1146229",
        database="Kuo"
    )
    try:
            my_cursor=dbConn.cursor()
            sql = "SELECT * FROM `users2`  "
            my_cursor.execute(sql)
            result=my_cursor.fetchall()
            global dict1
            n=[]
            for row in result:
                for i in row:
                    n.append(i)

            for i in range(len(n)):
                if i%3==1:
                    dict1[n[i]]=n[i+1]
            m = hashlib.sha256()
            m.update(password.encode('utf-8'))
            pw_h = m.hexdigest()
            if username in dict1.keys():
                if dict1[username]==pw_h:
                    msgbox.showinfo("done",'登入成功')
                    open()

                    # 開始GUI事件迴圈
                    window.mainloop()
                else:
                    messagebox.showerror('Error!','登入失敗')
            else:
                messagebox.showerror('Error!','登入失敗')
    finally:
        # 關閉資料庫連接
        my_cursor.close()
```

▼ 主選單

- 介面

- 介面啟動函式

```
def open():
    window.destroy()
    root = tk.Tk()
    root.title("功能選單")
    root.geometry("300x200")
    root.configure(background='#69829b')
    root.rowconfigure(0,weight=1)
    root.rowconfigure(1,weight=1)
    root.rowconfigure(2,weight=1)
    root.columnconfigure(0,weight=1)
    root.columnconfigure(1,weight=1)
    fontStyle = tkFont.Font( family="Lucida Grande",size=15)
    button1 = tk.Button(root, text="帳號管理",foreground='white',background='#1e4f80',width=10, command=account)
    button1.grid(row=0,column=0,pady=1,padx=2)
    button2 = tk.Button(root, text="影像預覽",foreground='white',background='#1e4f80',width=10, command=image_browser)
    button2.grid(row=0,column=1,pady=1,padx=2)
    button3 = tk.Button(root, text="影像管理",foreground='white',background='#1e4f80',width=10, command=photo)
    button3.grid(row=1,column=0,pady=1,padx=2)
    button4 = tk.Button(root, text="電話篩選",foreground='white',background='#1e4f80',width=10, command=Phonenumber)
    button4.grid(row=1,column=1,pady=1,padx=2)
    button5 = tk.Button(root, text="聊天即時通",foreground='white',background='#1e4f80',width=10, command=lambda:ChatApp())
    button5.grid(row=2,column=0,pady=1,padx=2)
```

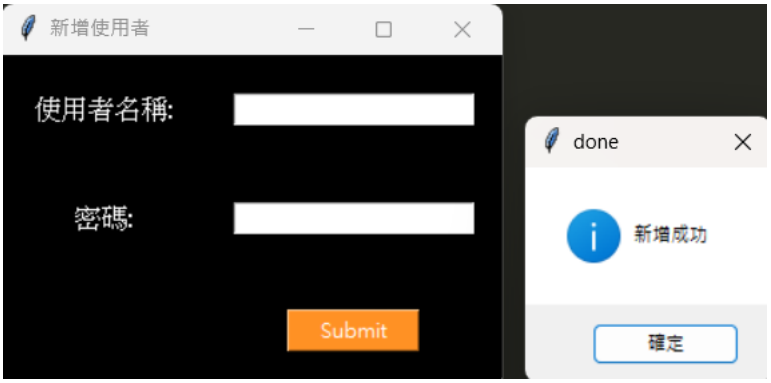
▼ 功能表

▼ 帳號管理

- 功能選擇介面


```
def account():
    root2 = tk.Tk()
    root2.title("帳號管理")
    root2.geometry("300x200")
    root2.configure(background='#69829b')
    root2.rowconfigure(0,weight=1)
    root2.rowconfigure(1,weight=1)
    root2.rowconfigure(2,weight=1)
    root2.rowconfigure(3,weight=1)
    root2.columnconfigure(0,weight=1)
    btn1 = tk.Button(root2, text="新增使用者",foreground='white',background='#1e4f80',width=10, command=create)
    btn1.grid(row=0,column=0,pady=1,padx=2)
    btn2 = tk.Button(root2, text="查詢使用者",foreground='white',background='#1e4f80',width=10, command=search)
    btn2.grid(row=1,column=0,pady=1,padx=2)
    btn3 = tk.Button(root2, text="修改使用者",foreground='white',background='#1e4f80',width=10, command=update)
    btn3.grid(row=2,column=0,pady=1,padx=2)
    btn4 = tk.Button(root2, text="刪除使用者",foreground='white',background='#1e4f80',width=10, command=dell)
    btn4.grid(row=3,column=0,pady=1,padx=2)
```

- 新增使用者

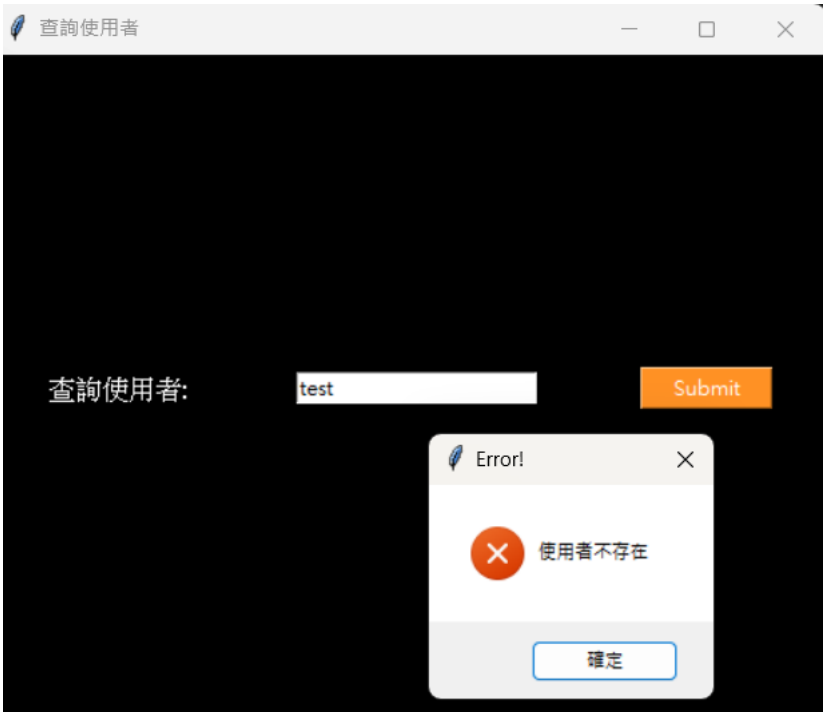



```
def create():
    root3 = tk.Tk()
    root3.title("新增使用者")
    root3.geometry("300x200")
    root3.configure(background='#000000')
    root3.rowconfigure(0,weight=1)
    root3.rowconfigure(1,weight=1)
    root3.rowconfigure(2,weight=1)
    root3.columnconfigure(0,weight=1)
    root3.columnconfigure(1,weight=1)
    name = tk.Label(root3, text="使用者名稱:",foreground='white',background='#000000',font=fontStyle)
    name.grid(row=0,column=0,pady=1,padx=2)
    nameentry = tk.Entry(root3)
    nameentry.grid(row=0,column=1,pady=1,padx=2)
    pw = tk.Label(root3, text="密碼:",foreground='white',background='#000000',font=fontStyle)
    pw.grid(row=1,column=0,pady=1,padx=2)
    pwentry = tk.Entry(root3)
    pwentry.grid(row=1,column=1,pady=1,padx=2)
    Sbm = tk.Button(root3, text="Submit",foreground='white',background='#ff9124',width=10, command=lambda:create_submit(nameentr
y,pwentry))
    Sbm.grid(row=2,column=1,pady=1,padx=2)
```

- 新增函式

```
def create_submit(nameentry,pwentry):
    bConn=myconn.connect(
    host="localhost",
    user="Kuo",
    password="D1146229",
    database="Kuo"
    )
    try:
        username=nameentry.get()
        password=pwentry.get()
        cursor = bConn.cursor()
        query = "INSERT INTO `users2` (username, password) VALUES (%s, %s)"
        m = hashlib.sha256()
        m.update(password.encode('utf-8'))
        pw_h = m.hexdigest()
        values = (username, pw_h)
        cursor.execute(query, values)
        bConn.commit()
        cursor.close()
        nameentry.delete(0, tk.END)
        pwentry.delete(0, tk.END)
        msgbox.showinfo("done",'新增成功')
    except:
        messagebox.showerror('Error!','新增失敗')
```

- 查詢使用者

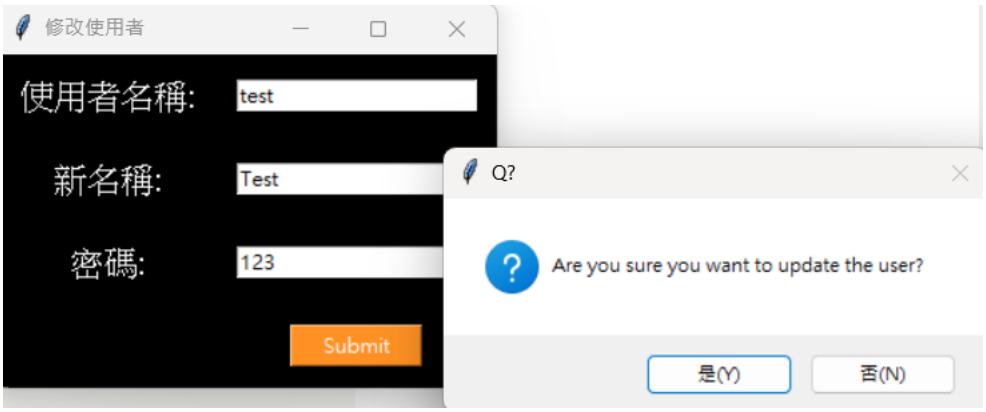



```
def search():
    root4 = tk.Tk()
    root4.title("查詢使用者")
    root4.geometry("500x400")
    root4.configure(background='#000000')
    root4.rowconfigure(0,weight=1)

    root4.columnconfigure(0,weight=1)
    root4.columnconfigure(1,weight=1)
    root4.columnconfigure(2,weight=1)

    userlab= tk.Label(root4, text="查詢使用者:",foreground='white',background='#000000',font=fontStyle)
    userlab.grid(row=0,column=0,pady=1,padx=2)
    nameentry = tk.Entry(root4)
    nameentry.grid(row=0,column=1,pady=1,padx=2)
    SBM = tk.Button(root4, text="Submit",foreground='white',background='#ff9124',width=10, command=lambda:search_(nameentry.get
()))
    SBM.grid(row=0,column=2,pady=1,padx=2)
```

- 查詢函式

```
def search_(username):

    if username in dict1.keys():
        msgbox.showinfo("done",'使用者存在')
    else:
        messagebox.showerror('Error!','使用者不存在')
```

- 修改使用者



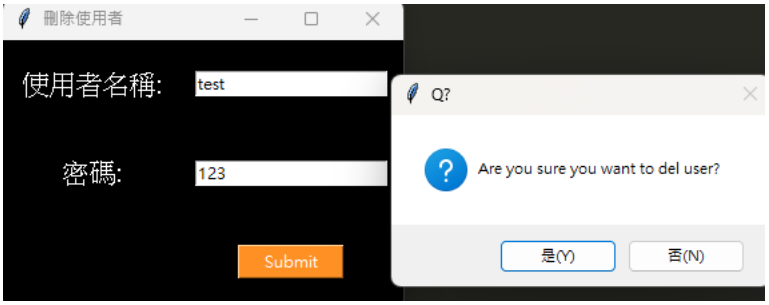

```
def update():
    root6 = tk.Tk()
    root6.title("修改使用者")
    root6.geometry("300x200")
    root6.configure(background='#000000')
    root6.rowconfigure(0,weight=1)
    root6.rowconfigure(1,weight=1)
    root6.rowconfigure(2,weight=1)
    root6.rowconfigure(3,weight=1)
    root6.columnconfigure(0,weight=1)
    root6.columnconfigure(1,weight=1)
    name = tk.Label(root6, text="使用者名稱:",foreground='white',background='#000000',font=fontStyle)
    name.grid(row=0,column=0,pady=1,padx=2)
    nameentry = tk.Entry(root6)
    nameentry.grid(row=0,column=1,pady=1,padx=2)
    pw = tk.Label(root6, text="新名稱:",foreground='white',background='#000000',font=fontStyle)
    pw.grid(row=1,column=0,pady=1,padx=2)
    pwentry = tk.Entry(root6)
    pwentry.grid(row=1,column=1,pady=1,padx=2)
    n_pw = tk.Label(root6, text="密碼:",foreground='white',background='#000000',font=fontStyle)
    n_pw.grid(row=2,column=0,pady=1,padx=2)
    n_pwentry = tk.Entry(root6)
    n_pwentry.grid(row=2,column=1,pady=1,padx=2)
    Sbm = tk.Button(root6, text="Submit",foreground='white',background='#ff9124',width=10, command=lambda:update_user(nameentry.g
et(),pwentry.get(),n_pwentry.get()))
    Sbm.grid(row=3,column=1,pady=1,padx=2)
```

- 修改函式

```
def update_user(username, new_username, pw):
    dbConn = myconn.connect(
        host="localhost",
        user="Kuo",
        password="D1146229",
        database="Kuo"
    )

    cursor = dbConn.cursor()


    query = "SELECT * FROM `users2` WHERE username = %s"
    cursor.execute(query, (username,))
    result = cursor.fetchone()

    if result:

        Msgbox = messagebox.askquestion('Q?', 'Are you sure you want to update the user?')
        if Msgbox == "yes":

            m = hashlib.sha256()
            m.update(pw.encode('utf-8'))
            h_pw = m.hexdigest()
            if dict1[username]==h_pw:

                query = "UPDATE `users2` SET username = %s WHERE username = %s"
                values = (new_username,  username)
```

```
                cursor.execute(query, values)
                dbConn.commit()

                messagebox.showinfo("Done", "修改使用者")
            else:
                messagebox.showwarning("Warning", "使用者密碼錯誤")
    else:
        messagebox.showwarning("Warning", "找不到使用者")

    cursor.close()
    dbConn.close()
```

- 刪除使用者




```
def dell():
    root5 = tk.Tk()
    root5.title("刪除使用者")
    root5.geometry("300x200")
    root5.configure(background='#000000')
    root5.rowconfigure(0,weight=1)
    root5.rowconfigure(1,weight=1)
    root5.rowconfigure(2,weight=1)
    root5.columnconfigure(0,weight=1)
    root5.columnconfigure(1,weight=1)
    name = tk.Label(root5, text="使用者名稱:",foreground='white',background='#000000',font=fontStyle)
    name.grid(row=0,column=0,pady=1,padx=2)
    nameentry = tk.Entry(root5)
    nameentry.grid(row=0,column=1,pady=1,padx=2)
    pw = tk.Label(root5, text="密碼:",foreground='white',background='#000000',font=fontStyle)
    pw.grid(row=1,column=0,pady=1,padx=2)
    pwentry = tk.Entry(root5)
    pwentry.grid(row=1,column=1,pady=1,padx=2)
    Sbm = tk.Button(root5, text="Submit",foreground='white',background='#ff9124',width=10, command=lambda:dell_submit(nameentry,p
wentry))
    Sbm.grid(row=2,column=1,pady=1,padx=2)
```

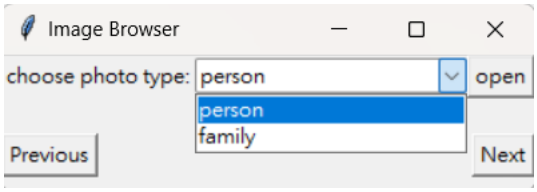
- 刪除函式

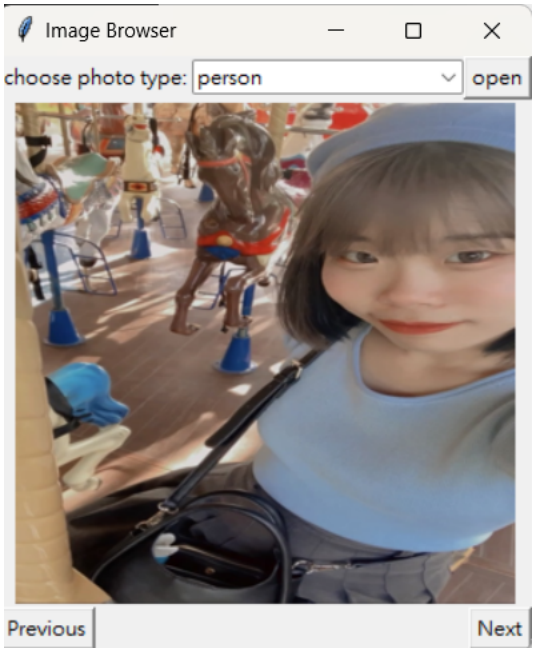
```
def dell_submit(nameentry,pwentry):
    bConn=myconn.connect(
    host="localhost",
    user="Kuo",
    password="D1146229",
    database="Kuo"
    )

    pw=pwentry.get()
    user=nameentry.get()
    cursor = bConn.cursor()
    m = hashlib.sha256()
    m.update(pw.encode('utf-8'))
    pw_h = m.hexdigest()

    if user in dict1.keys():
                if dict1[user]==pw_h:
                    Msgbox=messagebox.askquestion('Q?','Are you sure you want to del user?')
                    if Msgbox=="yes":
                        query = "DELETE FROM `users2` WHERE username = %s AND password = %s"
                        m = hashlib.sha256()
```

```
                    m.update(pw.encode('utf-8'))
                    pw_h = m.hexdigest()
                    values = (user, pw_h)
                    cursor.execute(query, values)
                    bConn.commit()
                    cursor.close()

                    msgbox.showinfo("done",'刪除成功')

else:
            messagebox.showerror('Error!','刪除失敗')
```

▼ 影像預覽

- 選單






```
#介面
def image_browser():
    root8 = tk.Toplevel()
    root8.title("Image Browser")

    _label = tk.Label(root8, text='choose photo type:')
    _label.grid(row=0, column=0, sticky="w")

    combol = ttk.Combobox(root8, values=['person', 'family'])
    combol.current(0)
    combol.grid(row=0, column=1, sticky="w")

    image_data = []
    current_index = 0

    open_button = tk.Button(root8, text="open", command=open_images)
    open_button.grid(row=0, column=2, sticky="e")

    image_label = tk.Label(root8)
    image_label.grid(row=1, column=0, columnspan=3)

    next_button = tk.Button(root8, text="Next", command=next_image)
    next_button.grid(row=2, column=2, sticky="e")

    previous_button = tk.Button(root8, text="Previous", command=previous_image)
    previous_button.grid(row=2, column=0, sticky="w")

    formation_label = tk.Label(root8, text="")
    formation_label.grid(row=2, column=1)

    root8.mainloop()
```

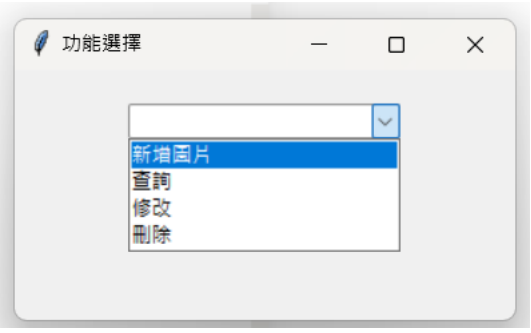
- 功能函式

```
def fetch_image_data():
    selected_type = combol.get()

    dbConn = myconn.connect(
        host="localhost",
        user="Kuo",
        password="D1146229",
        database="Kuo"
    )
    cursor = dbConn.cursor()
```

```
    query = "SELECT file_name, name, time FROM `photodt` WHERE type=%s"
    cursor.execute(query, (selected_type,))
    image_data = cursor.fetchall()

    # 關閉資料庫連接
    cursor.close()
    dbConn.close()

    return image_data

def show_image():
    nonlocal current_index
    if 0 <= current_index < len(image_data):
        image_path, image_name, image_time = image_data[current_index]
        image = Image.open(image_path)
        image = image.resize((300, 300))
        photo = ImageTk.PhotoImage(image)
        image_label.configure(image=photo)
        image_label.image = photo  # 保持對圖像物件的引用

        formation_label.config(text=f"Name: {image_name}\nType: {combol.get()}\nTime: {image_time}")

def next_image():
    nonlocal current_index
    if current_index < len(image_data) - 1:
        current_index += 1
        show_image()

def previous_image():
    nonlocal current_index
    if current_index > 0:
        current_index -= 1
        show_image()
def open_images():
    nonlocal image_data, current_index
    image_data = fetch_image_data()
    current_index = 0
    show_image()
```

▼ 影像管理

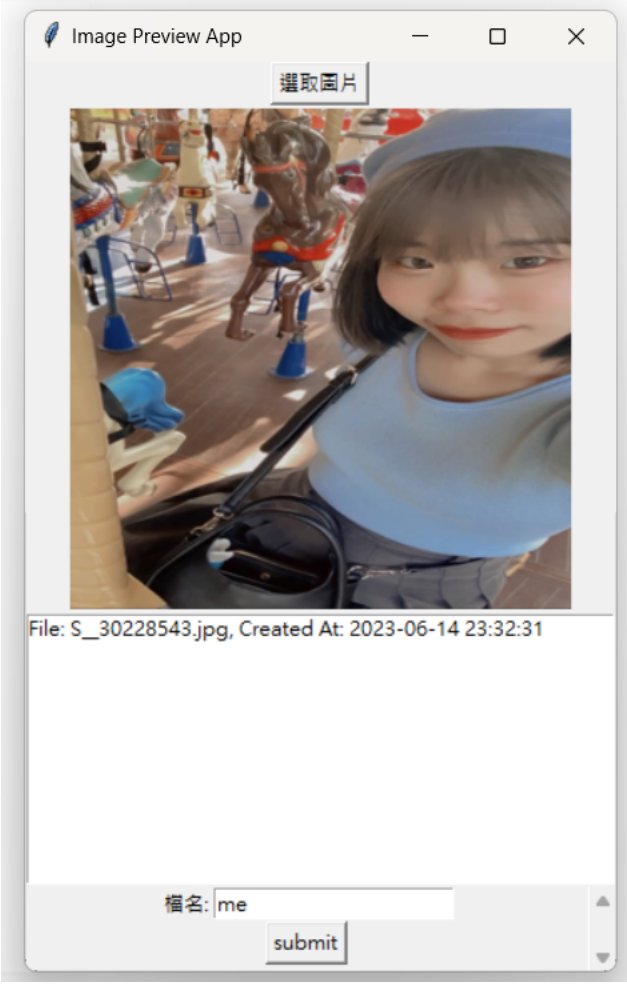
- 介面類別




```
class photo:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title("功能選擇")
        self.root.geometry("300x150")

        self.combobox = ttk.Combobox(self.root, values=["新增圖片", "查詢", "修改", "刪除"])
        self.combobox.pack(pady=20)

        self.button = tk.Button(self.root, text="執行", command=self.execute_function)
        self.button.pack()
        self.root.mainloop()
```

```
    def execute_function(self):
        selected_function = self.combobox.get()

        if selected_function == "新增圖片":
            open_image_preview()
        elif selected_function == "查詢":
            search_root()
        elif selected_function == "修改":
            update_image_root()
        elif selected_function == "刪除":
            delete_image_root()
```

- 新增圖片

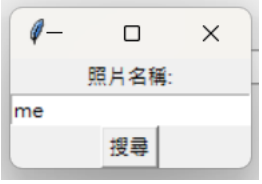


- 新增且瀏覽函式

```
def open_image_preview():
    root9 = tk.Toplevel()
    root9.title("Image Preview App")

    selected_image = None
    image = None  # 圖像物件

    def select_image():
        nonlocal selected_image
        filepath = filedialog.askopenfilename(initialdir="/", title="選取圖片",
                                              filetypes=(("JPEG files", "*.jpg;*.jpeg"), ("PNG files", "*.png"), ("All files", "*.*"
        if filepath:
            selected_image = filepath
```

```
            convert_to_png()
            show_image_preview()
            save_image_info(filepath)

    def convert_to_png():
        nonlocal selected_image
        filename, _ = os.path.splitext(selected_image)
        output_filepath = f"{filename}.png"
        img = Image.open(selected_image)
        img = img.resize((300, 300))
        img.save(output_filepath, "PNG")
        selected_image = output_filepath

    def show_image_preview():
        nonlocal image
        image = ImageTk.PhotoImage(file=selected_image)
        image_label.configure(image=image)

    def save_image_info(filepath):
        filename = os.path.basename(filepath)
        created_at = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        image_listbox.insert("end", f"File: {filename}, Created At: {created_at}")

    def update_database():
        nonlocal selected_image
        dbConn = myconn.connect(
            host="localhost",
            user="Kuo",
            password="D1146229",
            database="Kuo"
        )
        try:
            filename, _ = os.path.splitext(selected_image)
            created_at = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            output_filepath = f"{filename}.png"
            name = name_entry.get()
            cursor = dbConn.cursor()
            query = "INSERT INTO `photo` (file_name, name, time) VALUES (%s, %s, %s)"
            values = (output_filepath, name, created_at)
            cursor.execute(query, values)
            dbConn.commit()
            cursor.close()
            messagebox.showinfo("done", "新增成功")
        except:
            messagebox.showerror('Error!', '新增失敗')

    select_button = tk.Button(root9, text="選取圖片", command=select_image)
    select_button.pack()

    image_label = tk.Label(root9)
    image_label.pack()

    image_listbox = tk.Listbox(root9, width=50)
    image_listbox.pack()

    scrollbar = tk.Scrollbar(root9)
    scrollbar.pack(side="right", fill="y")

    image_listbox.config(yscrollcommand=scrollbar.set)
    scrollbar.config(command=image_listbox.yview)

    update_frame = tk.Frame(root9)
    update_frame.pack()
    label_name = tk.Label(update_frame, text="檔名:")
    label_name.pack(side="left")
    name_entry = tk.Entry(update_frame)
    name_entry.pack(side="right")
    submit_button = tk.Button(root9, text="submit", command=update_database)
    submit_button.pack()

    root9.mainloop()
```

- 搜尋圖片

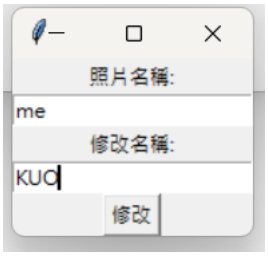



```
def search_images(name):
    root = tk.Toplevel()
    root.title("Photo")

    image_label = tk.Label(root)
    image_label.pack()

    dbConn = myconn.connect(
        host="localhost",
        user="Kuo",
        password="D1146229",
        database="Kuo"
    )
    cursor = dbConn.cursor()

    try:
        query = "SELECT file_name FROM `photo` WHERE name=%s"
        cursor.execute(query, (name,))
        image_path = cursor.fetchone()

        if image_path is not None:
            image = Image.open(image_path[0])
            image = image.resize((300, 300))
            photo = ImageTk.PhotoImage(image)
            image_label.configure(image=photo)
            image_label.image = photo
        else:
            messagebox.showinfo("Not Found", f"No image found for name '{name}'")
            root.destroy()
    except myconn.Error as error:
        messagebox.showerror("Database Error", f"Error retrieving image from database: {str(error)}")
        root.destroy()
    finally:
        cursor.close()
        dbConn.close()

    root.mainloop()
```

- 搜尋函式

```
def search_root():
    root = tk.Toplevel()
    root.title("search_images")
    name_label = tk.Label(root,text="照片名稱:")
    name_label.pack()
    entry_name = tk.Entry(root)
    entry_name.pack()
    submit_button = tk.Button(root, text="搜尋", command=lambda:search_images(entry_name.get()))
    submit_button.pack()

    root.mainloop()
```

- 修改圖片

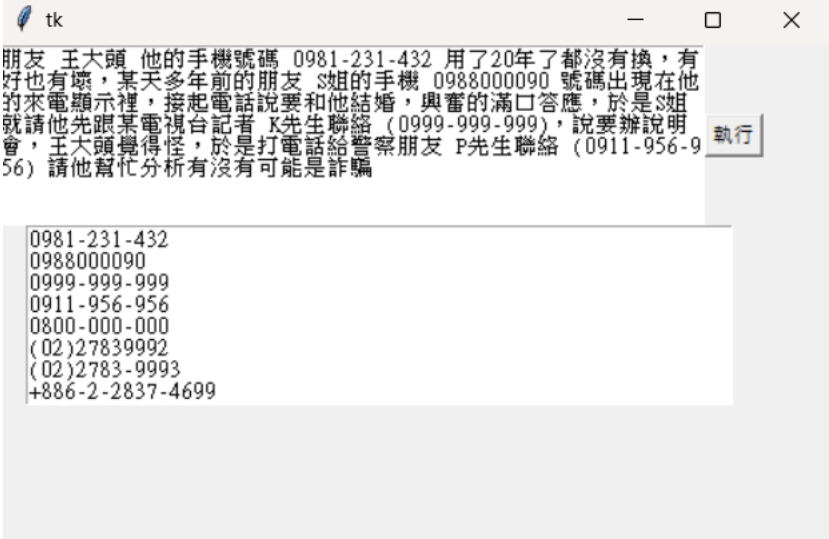



```
def update_image_root():
    root = tk.Toplevel()
    root.title("update_images")
    name_label = tk.Label(root,text="照片名稱:")
    name_label.pack()
    entry_name = tk.Entry(root)
    entry_name.pack()
    name2_label = tk.Label(root,text="修改名稱:")
    name2_label.pack()
    entry_name2 = tk.Entry(root)
    entry_name2.pack()
    submit_button = tk.Button(root, text="修改", command=lambda:update_image_name(entry_name.get(),entry_name2.get()))
    submit_button.pack()
```

- 修改函式

```
def update_image_name(old_name, new_name):
    dbConn = myconn.connect(
        host="localhost",
        user="Kuo",
        password="D1146229",
        database="Kuo"
    )
    cursor = dbConn.cursor()

    try:
        query = "SELECT file_name FROM `photo` WHERE name=%s"
        cursor.execute(query, (old_name,))
        image_path = cursor.fetchone()

        if image_path is not None:
            update_query = "UPDATE `photo` SET name=%s WHERE name=%s"
            cursor.execute(update_query, (new_name, old_name))
            dbConn.commit()
            messagebox.showinfo("Update Successful", f"Image name updated from '{old_name}' to '{new_name}'.")
        else:
            messagebox.showinfo("Not Found", f"No image found for name '{old_name}'")
    except myconn.Error as error:
        messagebox.showerror("Database Error", f"Error updating image name in database: {str(error)}")
    finally:
        cursor.close()
        dbConn.close()
```

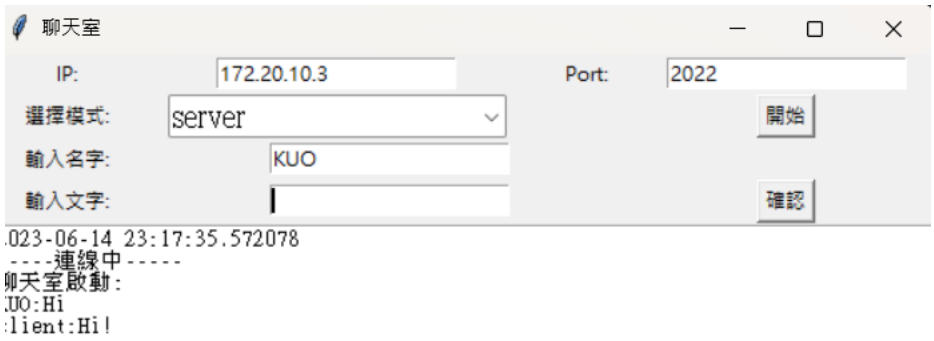
▼ 電話篩選

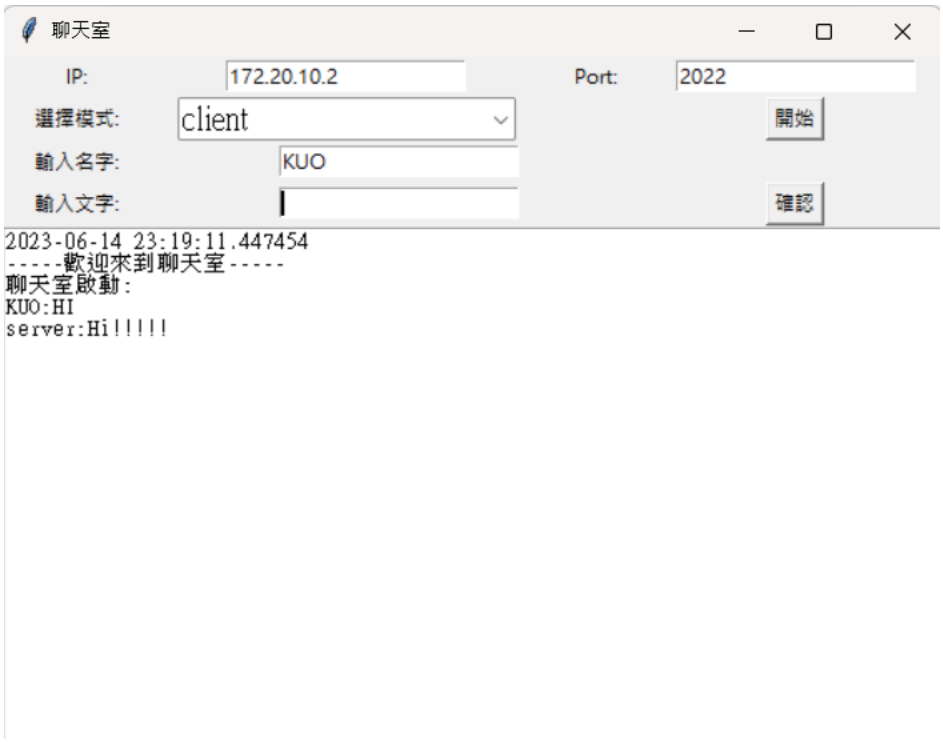


- 類別

```
class Phonenumber:
    def __init__(self):
        self.dbConn=myconn.connect(
        host="localhost",
        user="Kuo",
        password="D1146229",
        database="Kuo"
    )
        self.root = Tk()
        self.root.geometry('500x300')
        self.number_text = Text(self.root, width=60, height=8)
        self.number_text.grid(row=0, column=0)
        self.number_button = Button(self.root, text="執行", command=self.number_find)
        self.number_button.grid(row=0, column=1)
        self.result_text = Text(self.root, width=60, height=8)
        self.result_text.grid(row=1, column=0, columnspan=2)
        self.root.mainloop()

    def number_find(self):
        dbConn=myconn.connect(
        host="localhost",
        user="Kuo",
        password="D1146229",
        database="kuo"
    )
        cursor = dbConn.cursor()
        txt = self.number_text.get("1.0", "end-1c")
        phoneNum = re.findall(r'\d{4}-\d{3}-\d{3}|\(\d{2}\)\d{8}|\+\d{3}-\d{1}-\d{4}-\d{4}|\d{10}|\(\d{2}\)\d{4}-\d{4}', txt)
        if phoneNum:
            self.result_text.delete("1.0", "end")
            for phone in phoneNum:
                self.result_text.insert("end", phone + "\n")
        else:
            self.result_text.delete("1.0", "end")
            self.result_text.insert("end", "No data")
        article = txt
        number = "\n".join(phoneNum) if phoneNum else "No data"
        cursor = dbConn.cursor()
        query = "INSERT INTO `selectphone` (article, number) VALUES (%s, %s)"

        values = (article, number)
        cursor.execute(query, values)
        dbConn.commit()
        cursor.close()
```

▼ 聊天即時通





- 類別設定

```
class ChatApp:
        def __init__(self):
            self.name = ''
            self.user = ''
            self.s = None
```

```
            self.server = None
            self.clientAddr = None

            self.root7 = tk.Tk()
            self.root7.rowconfigure(0, weight=1)
            self.root7.rowconfigure(1, weight=1)
            self.root7.rowconfigure(2, weight=1)
            self.root7.rowconfigure(3, weight=1)
            self.root7.rowconfigure(4, weight=1)
            self.root7.columnconfigure(0, weight=1)
            self.root7.columnconfigure(1, weight=1)
            self.root7.columnconfigure(2, weight=1)
            self.root7.columnconfigure(3, weight=1)
            self.root7.title("聊天室")

            self.label_ip = tk.Label(self.root7, text="IP:")
            self.label_ip.grid(row=0, column=0, pady=1, padx=2)
            self.entry_ip = tk.Entry(self.root7)
            self.entry_ip.grid(row=0, column=1, pady=1, padx=2)

            self.label_port = tk.Label(self.root7, text="Port:")
            self.label_port.grid(row=0, column=2, pady=1, padx=2)

            self.entry_port = tk.Entry(self.root7)
            self.entry_port.grid(row=0, column=3, pady=1, padx=2)

            self.label_mode = tk.Label(self.root7, text="選擇模式:")
            self.label_mode.grid(row=1, column=0, pady=1, padx=2)

            self.combol2=ttk.Combobox(self.root7,values=['server','client'] ,font=fontStyle)
            self.combol2.current(0)
            self.combol2.grid(row=1,column=1,pady=1,padx=2)
            self.start_button = tk.Button(self.root7, text="開始", command=self.start)
            self.start_button.grid(row=1, column=3, pady=1, padx=2)

            self.label_name = tk.Label(self.root7, text="輸入名字:")
            self.label_name.grid(row=2, column=0, pady=1, padx=2)
            self.entry_name = tk.Entry(self.root7)
            self.entry_name.grid(row=2, column=1, pady=1, padx=2, columnspan=2)

            self.label_message = tk.Label(self.root7, text="輸入文字:")
            self.label_message.grid(row=3, column=0, pady=1, padx=2)
            self.entry_message = tk.Entry(self.root7)
            self.entry_message.grid(row=3, column=1, pady=1, padx=2, columnspan=2)

            self.button_send = tk.Button(self.root7, text="確認", command=self.send_message)
            self.button_send.grid(row=3, column=3, pady=1, padx=2)

            self.textbox = tk.Text(self.root7)
            self.textbox.grid(row=4, column=0, columnspan=4)
            self.user=''
            self.root7.mainloop()
```

- 啟動函式

```
def start(self):
            if self.combol2.current()==0:
                tr = threading.Thread(target=self.start_server)
                tr.start()

            elif self.combol2.current()==1:
                self.start_client()
```

- 啟動server

```
def start_server(self):
            time_now=datetime.datetime.now()
            self.name = ''
            self.s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
            self.textbox.insert(tk.INSERT, time_now)
            self.textbox.insert(tk.INSERT, "\n")
            self.textbox.insert(tk.INSERT, "-----連線中-----")
            self.textbox.insert(tk.INSERT, "\n")
            host = self.entry_ip.get()
            port = self.entry_port.get()
```

```
            self.server = (host, int(port))
            self.s.bind(self.server)
            msg, self.clientAddr = self.s.recvfrom(1024)
            self.name = msg.decode('utf-8')
            tr = threading.Thread(target=self.sever_recv, args=(self.s, self.server), daemon=True)

            try:
                tr.start()
            except ConnectionResetError:
                print('Error: someone left unexpect.')
```

- 啟動client

```
def start_client(self):
            time_now=datetime.datetime.now()
            self.name = ''
            self.textbox.insert(tk.INSERT, time_now)
            self.textbox.insert(tk.INSERT, "\n")
            self.textbox.insert(tk.INSERT, "-----歡迎來到聊天室-----")
            self.textbox.insert(tk.INSERT, "\n")
            self.s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
            host = self.entry_ip.get()
            port = int(self.entry_port.get())
            self.server = (host, port)
            tr = threading.Thread(target=self.client_recv, args=(self.s, self.server), daemon=True)
            tr.start()
```

- 傳送訊息

```
def send_message(self):
            if self.combol2.current()==0:
                ts = threading.Thread(target=self.send, args=(self.s, self.clientAddr))
                ts.start()
            elif self.combol2.current()==1:
                ts = threading.Thread(target=self.send, args=(self.s, self.server))
                ts.start()
```

```
def send(self, sock, addr):
            self.user=self.entry_name.get()
            string = self.entry_message.get()
            message = self.user + ':' + string
            data = message.encode("utf-8")
            sock.sendto(data, addr)
            self.textbox.insert(tk.END, message)
            self.textbox.insert(tk.END, "\n")
            self.entry_message.delete(0, tk.END)
```

- server收訊

```
def sever_recv(self, sock, addr):
            self.textbox.insert(tk.END, "聊天室啟動:")

            sock.sendto(self.name.encode('utf-8'), addr)
            while True:
                data, addr = sock.recvfrom(1024)
                recvMsg = data.decode('utf-8')
                self.textbox.insert(tk.END, recvMsg)
                self.textbox.insert(tk.END, "\n")
                if recvMsg.lower() == 'EXIT'.lower():
                    break
```

- client收訊

```
def client_recv(self, sock, addr):
            #self.user=self.entry_name.get()
            self.textbox.insert(tk.END, "聊天室啟動:")
            self.textbox.insert(tk.END, "\n")
```

```
            sock.sendto(self.name.encode('utf-8'), addr)
            while True:
                data, addr = sock.recvfrom(1024)
                recvMsg = data.decode('utf-8')
                self.textbox.insert(tk.END, recvMsg)
                self.textbox.insert(tk.END, "\n")
                if recvMsg.lower() == 'EXIT'.lower():
                    break
```