مشروع: Simple Flight Reservation Desktop App (بالعربي خطوة بخطوة) باستخدام Tkinter و SQLite

هنمشي خطوة خطوة من الصفر لحد ما نشغل برنامج حجز طيران بسيط — **مناسب للمبتدئين** بواجهة رسومية. في الآخر هتلاقي كود كامل تقدر تنسخه كملف واحد `flight_app.py` ويشتغل فورًا.

المتطلبات

- Python 3.9+
- لا تحتاج لتثبيت أي مكتبات إضافية (نستخدم المكتبات المدمجة: `tkinter` و `sqlite3`).

ثم لتشغيل التطبيق بعد الانتهاء، احفظ الكود في ملف `flight_app.py`:

```
python flight_app.py
```

أهداف التطبيق

- إنشاء قاعدة بيانات SQLite محليًا تحتوي على جدول للحجوزات.
- واجهة رسومية بـ Tkinter لعمل **CRUD**: إضافة، عرض، تحديث، حذف.
- عرض الحجوزات في جدول (Treeview) + تعبئة النموذج تلقائيًا عند اختيار صف.
- بحث سريع بالاسم أو رقم الرحلة.

تصميم الحقول المقترحة

هنخزن البيانات الأساسية لكل حجز: - `id` (رقم تسلسلي تلقائي) — المفتاح الأساسي - `passenger_name` — اسم الراكب - `passport_no` — رقم جواز السفر - `flight_no` — رقم الرحلة - `origin` — من مطار - `destination` — إلى مطار - `flight_date` — تاريخ الرحلة (YYYY-MM-DD) - `seat_class` — فئة المقعد (Economy/Business) - `price` — السعر (عدد عشري)

خطوة 1) الكود الكامل — ملف واحد جاهز للتشغيل `flight_app.py`

ثم شغله `flight_app.py` في ملف باسم **انسخي/انسخ كل الكود التالي كما هو**.

```
import sqlite3
import tkinter as tk
from tkinter import ttk, messagebox
from datetime import datetime
```

```python
DB_NAME = "flights.db"

# ============================
# 1) طبقة البيانات: SQLite
# ============================

def get_conn():
    return sqlite3.connect(DB_NAME)


def init_db():
    conn = get_conn()
    cur = conn.cursor()
    cur.execute(
        """
        CREATE TABLE IF NOT EXISTS reservations (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            passenger_name TEXT NOT NULL,
            passport_no TEXT NOT NULL,
            flight_no TEXT NOT NULL,
            origin TEXT NOT NULL,
            destination TEXT NOT NULL,
            flight_date TEXT NOT NULL,
            seat_class TEXT NOT NULL,
            price REAL NOT NULL
        )
        """
    )
    conn.commit()
    conn.close()


def create_reservation(data):
    conn = get_conn()
    cur = conn.cursor()
    cur.execute(
        """
        INSERT INTO reservations (
            passenger_name, passport_no, flight_no, origin, destination,
            flight_date, seat_class, price
        ) VALUES (?, ?, ?, ?, ?, ?, ?, ?)
        """,
        (
            data["passenger_name"],
            data["passport_no"],
            data["flight_no"],
            data["origin"],
            data["destination"],
            data["flight_date"],
            data["seat_class"],
            float(data["price"]),
```

```
        ),
    )
    conn.commit()
    conn.close()


def update_reservation(res_id, data):
    conn = get_conn()
    cur = conn.cursor()
    cur.execute(
        """
        UPDATE reservations
        SET passenger_name=?, passport_no=?, flight_no=?, origin=?,
destination=?,
            flight_date=?, seat_class=?, price=?
        WHERE id=?
        """,
        (
            data["passenger_name"],
            data["passport_no"],
            data["flight_no"],
            data["origin"],
            data["destination"],
            data["flight_date"],
            data["seat_class"],
            float(data["price"]),
            res_id,
        ),
    )
    conn.commit()
    conn.close()


def delete_reservation(res_id):
    conn = get_conn()
    cur = conn.cursor()
    cur.execute("DELETE FROM reservations WHERE id=?", (res_id,))
    conn.commit()
    conn.close()


def fetch_reservations(keyword=None):
    conn = get_conn()
    cur = conn.cursor()
    if keyword:
        kw = f"%{keyword.strip()}%"
        cur.execute(
            """
            SELECT * FROM reservations
            WHERE passenger_name LIKE ? OR flight_no LIKE ?
            ORDER BY id DESC
```

```
            """,
            (kw, kw),
        )
    else:
        cur.execute("SELECT * FROM reservations ORDER BY id DESC")
    rows = cur.fetchall()
    conn.close()
    return rows


# ============================
# 2) أدوات مساعدة للتحقق من المدخلات
# ============================

def validate_date(yyyy_mm_dd: str) -> bool:
    try:
        datetime.strptime(yyyy_mm_dd, "%Y-%m-%d")
        return True
    except ValueError:
        return False


def validate_required_fields(data: dict) -> tuple[bool, str]:
    # تأكد من عدم ترك حقول مهمة فارغة
    required = [
        "passenger_name",
        "passport_no",
        "flight_no",
        "origin",
        "destination",
        "flight_date",
        "seat_class",
        "price",
    ]
    for key in required:
        if not str(data.get(key, "")).strip():
            return False, f"الحقل '{key}' مطلوب"
    # تحقق من التاريخ
    if not validate_date(data["flight_date"]):
        return False, "صيغة التاريخ يجب أن تكون YYYY-MM-DD مثلاً 01-10-2025"
    # تحقق من السعر رقم
    try:
        float(data["price"])
    except ValueError:
        return False, "السعر يجب أن يكون رقمًا"
    return True, ""


# ============================
# 3) الواجهة الرسومية Tkinter
# ============================
```

```
class FlightApp(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Flight Reservation App")
        self.geometry("1000x600")
        self.minsize(900, 560)
        self.configure(padx=10, pady=10)

        # إطار علوي للنموذج، وأسفله الجدول
        self.form_frame = ttk.LabelFrame(self, text="بيانات الحجز")
        self.form_frame.pack(fill="x", padx=5, pady=5)

        self.table_frame = ttk.LabelFrame(self, text="الحجوزات")
        self.table_frame.pack(fill="both", expand=True, padx=5, pady=5)

        self._build_form()
        self._build_table()
        self._build_search_bar()

        self.selected_id = None
        self.refresh_table()

    def _build_search_bar(self):
        bar = ttk.Frame(self)
        bar.pack(fill="x", pady=(0, 5))
        ttk.Label(bar, text="بحث بالاسم/رقم الرحلة:").pack(side="left")
        self.search_var = tk.StringVar()
        ent = ttk.Entry(bar, textvariable=self.search_var, width=30)
        ent.pack(side="left", padx=5)
        ttk.Button(bar, text="بحث", command=self.on_search).pack(side="left")
        ttk.Button(bar, text="كل السجلات",
command=self.on_clear_search).pack(side="left", padx=(5,0))

    def _build_form(self):
        # متغيرات النموذج
        self.var_name = tk.StringVar()
        self.var_passport = tk.StringVar()
        self.var_flight = tk.StringVar()
        self.var_origin = tk.StringVar()
        self.var_dest = tk.StringVar()
        self.var_date = tk.StringVar()
        self.var_class = tk.StringVar(value="Economy")
        self.var_price = tk.StringVar()

        # شبكة الحقول (labels + entries)
        # صف 1
        ttk.Label(self.form_frame, text="اسم الراكب").grid(row=0, column=0,
sticky="w", padx=5, pady=5)
        ttk.Entry(self.form_frame, textvariable=self.var_name,
width=28).grid(row=0, column=1, padx=5, pady=5)
```

```
        ttk.Label(self.form_frame, text="رقم الجواز").grid(row=0, column=2,
sticky="w", padx=5, pady=5)
        ttk.Entry(self.form_frame, textvariable=self.var_passport,
width=20).grid(row=0, column=3, padx=5, pady=5)

        ttk.Label(self.form_frame, text="رقم الرحلة").grid(row=0, column=4,
sticky="w", padx=5, pady=5)
        ttk.Entry(self.form_frame, textvariable=self.var_flight,
width=12).grid(row=0, column=5, padx=5, pady=5)

        # صف 2
        ttk.Label(self.form_frame, text="من (المطار)").grid(row=1, column=0,
sticky="w", padx=5, pady=5)
        ttk.Entry(self.form_frame, textvariable=self.var_origin,
width=20).grid(row=1, column=1, padx=5, pady=5)

        ttk.Label(self.form_frame, text="إلى (المطار)").grid(row=1, column=2,
sticky="w", padx=5, pady=5)
        ttk.Entry(self.form_frame, textvariable=self.var_dest,
width=20).grid(row=1, column=3, padx=5, pady=5)

        ttk.Label(self.form_frame, text="تاريخ الرحلة (YYYY-MM-DD)").grid(row=1,
column=4, sticky="w", padx=5, pady=5)
        ttk.Entry(self.form_frame, textvariable=self.var_date,
width=12).grid(row=1, column=5, padx=5, pady=5)

        # صف 3
        ttk.Label(self.form_frame, text="الفئة").grid(row=2, column=0,
sticky="w", padx=5, pady=5)
        cmb = ttk.Combobox(self.form_frame, textvariable=self.var_class,
values=["Economy", "Business"], state="readonly", width=18)
        cmb.grid(row=2, column=1, padx=5, pady=5)

        ttk.Label(self.form_frame, text="السعر").grid(row=2, column=2,
sticky="w", padx=5, pady=5)
        ttk.Entry(self.form_frame, textvariable=self.var_price,
width=12).grid(row=2, column=3, padx=5, pady=5)

        # أزرار CRUD
        btns = ttk.Frame(self.form_frame)
        btns.grid(row=2, column=4, columnspan=2, sticky="e", padx=5, pady=5)
        ttk.Button(btns, text="إضافة", command=self.on_add).pack(side="left",
padx=2)
        ttk.Button(btns, text="تحديث",
command=self.on_update).pack(side="left", padx=2)
        ttk.Button(btns, text="حذف",
command=self.on_delete).pack(side="left", padx=2)
        ttk.Button(btns, text="تفريغ النموذج",
command=self.clear_form).pack(side="left", padx=2)
```

```
        # ضبط الأعمدة
        for col in range(6):
            self.form_frame.grid_columnconfigure(col, weight=1)

    def _build_table(self):
        columns = (
            "id",
            "passenger_name",
            "passport_no",
            "flight_no",
            "origin",
            "destination",
            "flight_date",
            "seat_class",
            "price",
        )
        self.tree = ttk.Treeview(self.table_frame, columns=columns,
show="headings", height=12)
        # رؤوس الأعمدة
        headers = [
            ("id", "ID"),
            ("passenger_name", "الاسم"),
            ("passport_no", "الجواز"),
            ("flight_no", "الرحلة"),
            ("origin", "من"),
            ("destination", "إلى"),
            ("flight_date", "التاريخ"),
            ("seat_class", "الفئة"),
            ("price", "السعر"),
        ]
        for key, title in headers:
            self.tree.heading(key, text=title)
        # عرض أعمدة مناسب
        self.tree.column("id", width=50, anchor="center")
        self.tree.column("passenger_name", width=160)
        self.tree.column("passport_no", width=100, anchor="center")
        self.tree.column("flight_no", width=80, anchor="center")
        self.tree.column("origin", width=90, anchor="center")
        self.tree.column("destination", width=90, anchor="center")
        self.tree.column("flight_date", width=100, anchor="center")
        self.tree.column("seat_class", width=90, anchor="center")
        self.tree.column("price", width=80, anchor="e")

        self.tree.pack(fill="both", expand=True, side="left")
        # شريط تمرير
        scrollbar = ttk.Scrollbar(self.table_frame, orient="vertical",
command=self.tree.yview)
        scrollbar.pack(side="right", fill="y")
        self.tree.configure(yscrollcommand=scrollbar.set)

        # حدث اختيار صف
```

```
        self.tree.bind("<<TreeviewSelect>>", self.on_row_select)

    # ============ عمليات الواجهة ============
    def on_search(self):
        kw = self.search_var.get().strip()
        self.refresh_table(kw if kw else None)

    def on_clear_search(self):
        self.search_var.set("")
        self.refresh_table()

    def on_row_select(self, event=None):
        sel = self.tree.selection()
        if not sel:
            return
        item = self.tree.item(sel[0])
        values = item["values"]
        self.selected_id = values[0]
        # عبّي النموذج
        self.var_name.set(values[1])
        self.var_passport.set(values[2])
        self.var_flight.set(values[3])
        self.var_origin.set(values[4])
        self.var_dest.set(values[5])
        self.var_date.set(values[6])
        self.var_class.set(values[7])
        self.var_price.set(values[8])

    def clear_form(self):
        self.selected_id = None
        self.var_name.set("")
        self.var_passport.set("")
        self.var_flight.set("")
        self.var_origin.set("")
        self.var_dest.set("")
        self.var_date.set("")
        self.var_class.set("Economy")
        self.var_price.set("")
        self.tree.selection_remove(self.tree.selection())

    def get_form_data(self) -> dict:
        return {
            "passenger_name": self.var_name.get().strip(),
            "passport_no": self.var_passport.get().strip(),
            "flight_no": self.var_flight.get().strip(),
            "origin": self.var_origin.get().strip(),
            "destination": self.var_dest.get().strip(),
            "flight_date": self.var_date.get().strip(),
            "seat_class": self.var_class.get().strip(),
            "price": self.var_price.get().strip(),
        }
```

```
    def on_add(self):
        data = self.get_form_data()
        ok, msg = validate_required_fields(data)
        if not ok:
            messagebox.showerror("خطأ", msg)
            return
        try:
            create_reservation(data)
            messagebox.showinfo("تم", "تمت إضافة الحجز بنجاح")
            self.clear_form()
            self.refresh_table()
        except Exception as e:
            messagebox.showerror("خطأ", f"حدث خطأ أثناء الإضافة: {e}")

    def on_update(self):
        if not self.selected_id:
            messagebox.showwarning("تنبيه", "من فضلك اختر سجلًا من الجدول أولًا")
            return
        data = self.get_form_data()
        ok, msg = validate_required_fields(data)
        if not ok:
            messagebox.showerror("خطأ", msg)
            return
        try:
            update_reservation(self.selected_id, data)
            messagebox.showinfo("تم", "تم تحديث الحجز")
            self.refresh_table()
        except Exception as e:
            messagebox.showerror("خطأ", f"فشل التحديث: {e}")

    def on_delete(self):
        if not self.selected_id:
            messagebox.showwarning("تنبيه", "اختر سجلًا للحذف")
            return
        if messagebox.askyesno("تأكيد", "هل أنت متأكد من الحذف؟"):
            try:
                delete_reservation(self.selected_id)
                messagebox.showinfo("تم", "تم حذف الحجز")
                self.clear_form()
                self.refresh_table()
            except Exception as e:
                messagebox.showerror("خطأ", f"فشل الحذف: {e}")

    def refresh_table(self, keyword=None):
        # مسح الجدول
        for row in self.tree.get_children():
            self.tree.delete(row)
        # تعبئة
        for row in fetch_reservations(keyword):
            self.tree.insert("", tk.END, values=row)
```

```
# ============================
# 4) نقطة تشغيل التطبيق
# ============================
if __name__ == "__main__":
    init_db()
    app = FlightApp()
    app.mainloop()
```

خطوة 2) شرح سريع لما يحدث

1. **قاعدة البيانات**: أول ما البرنامج يشتغل ينفّذ `init_db()` ويعمل ملف `flights.db` وجدول `reservations` لو مش موجودين.
2. **النموذج**: حقول لإدخال البيانات + أزرار (إضافة/تحديث/حذف/تفريغ).
3. **الجدول**: `Treeview` لعرض السجلات. لما تختار صف، الحقول تتملي تلقائيًا.
4. **البحث**: مربع بسيط للبحث بالاسم أو رقم الرحلة.

خطوة 3) تحسينات اختيارية (لاحقًا)

- **فصل الكود إلى ملفات**: مثل `db.py` للبيانات و `ui.py` للواجهة.
- **التحقق المتقدم**: التأكد من عدم تكرار نفس رقم الجواز لنفس التاريخ... إلخ.
- **التصدير**: إضافة زر لتصدير النتائج إلى CSV.
- **تسجيل الدخول وصلاحيات**: مستخدم عادي/مدير.

خطوة 4) أوامر Git/GitHub (مختصر)

1. إنشاء مستودع محلي:

```
git init
git add flight_app.py
git commit -m "Initial commit: simple flight reservation app"
```

2. على GitHub أنشئ Repository جديد (عام/خاص)، ثم اربطه:

```
git remote add origin https://github.com/<username>/<repo>.git
git branch -M main
git push -u origin main
```

لو حبيتي/حبيت أقسم المشروع لملفات أو أضيف مميزات (CSV export، داكن/رسائل خطأ أفضل، فاتح، صفحات متعددة)، قولي لي وهنعدل الكود الحالي خطوة بخطوة.