```
import tkinter as tk

from tkinter import ttk, simpledialog, messagebox, Toplevel, Scrollbar

from datetime import datetime, timedelta

import calendar

from tkinter import Tk



class AdvancedCalendarApp:

    def __init__(self, root):

        self.root = root

        self.root.title("تقويم متقدم")

        self.root.geometry("800x700")

        self.root.configure(bg='white')



        # تاريخ اليوم

        self.current_date = datetime.now()

        self.selected_date = self.current_date.strftime('%Y-%m-%d')
        # أحداث

        self.events = {}

        self.new_event_dates = set()



        # إضافة حدث عيد الأضحى

        eid_al_adha_date = "2025-06-12"

        self.events[eid_al_adha_date] = ["عيد الأضحى"]

        self.new_event_dates.add(eid_al_adha_date)



        # إطار رئيسي

        main_frame = ttk.Frame(root, padding="10")

        main_frame.pack(fill=tk.BOTH, expand=True)



        # إطار حول التقويم

        calendar_frame = ttk.Frame(main_frame, padding="20", relief="sunken", 
borderwidth=2)
```

```
        calendar_frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)



        # أزرار للتنقل بين الأشهر والسنوات

        nav_frame = ttk.Frame(calendar_frame)

        nav_frame.pack(pady=10)



        self.prev_year_button = ttk.Button(nav_frame, text="<< السنة", command=self.prev_year)

        self.prev_year_button.pack(side=tk.LEFT)



        self.prev_month_button = ttk.Button(nav_frame, text="<< الشهر",
command=self.prev_month)

        self.prev_month_button.pack(side=tk.LEFT)



        self.month_year_label = ttk.Label(nav_frame, text="", font=("Arial", 16, 'bold'))

        self.month_year_label.pack(side=tk.LEFT, padx=10)
        self.next_month_button = ttk.Button(nav_frame, text="الشهر >>",
command=self.next_month)

        self.next_month_button.pack(side=tk.LEFT)



        self.next_year_button = ttk.Button(nav_frame, text="السنة >>", command=self.next_year)

        self.next_year_button.pack(side=tk.LEFT)



        # جدول التقويم

        self.calendar_table = ttk.Frame(calendar_frame)

        self.calendar_table.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)



        # منطقة لعرض الأحداث

        self.event_label = ttk.Label(main_frame, text="", font=("Arial", 12), anchor='w')

        self.event_label.pack(pady=10, fill=tk.BOTH, expand=True)



        # زر لإضافة حدث
        self.add_event_button = ttk.Button(main_frame, text="إضافة حدث",
command=self.add_event)

        self.add_event_button.pack(pady=10)



        # زر للبحث عن الأحداث

        self.search_event_button = ttk.Button(main_frame, text="بحث عن حدث",
command=self.search_event)

        self.search_event_button.pack(pady=10)



        # تخصيص الأنماط

        style = ttk.Style()

        style.configure("TButton", padding=6, relief="flat", background="#f0f0f0")

        style.configure("Event.TButton", padding=6, relief="flat", background="#ffdddd")

        style.configure("Today.TButton", padding=6, relief="flat", background="#a0a0a0")  #
اللون لليوم الحالي

        style.configure("EventDay.TButton", padding=6, relief="flat", foreground="blue")  # اللون
لأيام الأحداث

        style.configure("NewEventDay.TButton", padding=6, relief="flat", foreground="red")  #
اللون لأيام الحدث الجديد
```

```
        # تحديث واجهة المستخدم
        self.update_calendar()

    def update_calendar(self):
        # مسح الجدول الحالي
        for widget in self.calendar_table.winfo_children():
            widget.destroy()

        # اسم الشهر والسنة
        self.month_year_label.config(text=self.current_date.strftime('%B %Y'))

        # إنشاء رأس الجدول
        days = ["الأحد", "الإثنين", "الثلاثاء", "الأربعاء", "الخميس", "الجمعة", "السبت"]
        for day in days:
            ttk.Label(self.calendar_table, text=day, font=("Arial", 10, 'bold')).grid(row=0,
column=days.index(day))
```

```
        # الحصول على الأيام في الشهر الحالي

        first_day_of_month = self.current_date.replace(day=1)

        last_day_of_month = (first_day_of_month.replace(month=self.current_date.month + 1, day=1) - timedelta(days=1)) if self.current_date.month < 12 else self.current_date.replace(year=self.current_date.year + 1, month=1, day=1) - timedelta(days=1)



        start_day = first_day_of_month.weekday()  # اليوم الأول في الأسبوع (0=الإثنين، 1=الثلاثاء، ... 6=الأحد)

        days_in_month = (last_day_of_month - first_day_of_month).days + 1



        for i in range(start_day):

            ttk.Label(self.calendar_table, text="", width=4).grid(row=1, column=i)



        for day in range(1, days_in_month + 1):

            date_str = self.current_date.replace(day=day).strftime('%Y-%m-%d')

            button = ttk.Button(self.calendar_table, text=str(day), command=lambda d=day: self.select_date(d))
            button.grid(row=(day + start_day - 1) // 7 + 1, column=(day + start_day - 1) % 7)


            # تخصيص اللون لخلفية الأزرار

            if date_str in self.new_event_dates:
                button.configure(style="NewEventDay.TButton")
            elif date_str in self.events:
                button.configure(style="EventDay.TButton")
            else:
                button.configure(style="TButton")


            button.bind("<Enter>", self.on_hover)
            button.bind("<Leave>", self.on_leave)


        def select_date(self, day):
        self.selected_date = self.current_date.replace(day=day).strftime('%Y-%m-%d')
        self.show_events_for_date(self.selected_date)
```

```
def show_events_for_date(self, date):

    events = self.events.get(date, [])

    events_text = "\n".join(events) if events else "لا توجد أحداث لهذا التاريخ"

    self.event_label.config(text=f"التاريخ المحدد: {date}\n{events_text}")



    self.show_event_details(date)



def show_event_details(self, date):

    def delete_event(ev):

        if messagebox.askyesno("حذف الحدث", "هل أنت متأكد من حذف هذا الحدث؟"):

            self.events[date].remove(ev)

            if not self.events[date]:

                del self.events[date]

            self.show_events_for_date(date)

            self.update_calendar()
        event_window = Toplevel(self.root)

        event_window.title(f"أحداث لـ {date}")

        event_window.geometry("300x400")



        scrollbar = Scrollbar(event_window)

        scrollbar.pack(side=tk.RIGHT, fill=tk.Y)



        listbox = tk.Listbox(event_window, selectmode=tk.SINGLE,
yscrollcommand=scrollbar.set)

        listbox.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

        scrollbar.config(command=listbox.yview)



        for event in self.events.get(date, []):

            listbox.insert(tk.END, event)



        def on_select(event):
            selected_event = listbox.get(listbox.curselection())

            if messagebox.askyesno("تعديل الحدث", "هل تريد تعديل هذا الحدث؟"):

                new_event = simpledialog.askstring("تعديل الحدث", "أدخل تفاصيل الحدث:", 
initialvalue=selected_event)

                if new_event:

                    index = listbox.curselection()[0]

                    listbox.delete(index)

                    listbox.insert(index, new_event)

                    self.events[date][index] = new_event



        listbox.bind("<Double-1>", on_select)



        add_button = ttk.Button(event_window, text="إضافة حدث", command=lambda: 
self.add_event(date))

        add_button.pack(pady=5)



        delete_button = ttk.Button(event_window, text="حذف الحدث", command=lambda: 
delete_event(listbox.get(listbox.curselection())))
```

```
        delete_button.pack(pady=5)



    def search_event(self):

        search_term = simpledialog.askstring("بحث عن حدث", "أدخل نص البحث:")

        if search_term:

            results = []

            for date, events in self.events.items():

                results.extend([f"{date}: {event}" for event in events if search_term.lower() in
event.lower()])



            if results:

                search_result_window = Toplevel(self.root)

                search_result_window.title("نتائج البحث")

                search_result_window.geometry("300x400")



                scrollbar = Scrollbar(search_result_window)

                scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
                listbox = tk.Listbox(search_result_window, yscrollcommand=scrollbar.set)

                listbox.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

                scrollbar.config(command=listbox.yview)



                for result in results:

                    listbox.insert(tk.END, result)



                def on_select(event):

                    selected_result = listbox.get(listbox.curselection())

                    date_str = selected_result.split(":")[0]

                    self.select_date(datetime.strptime(date_str, '%Y-%m-%d').day)



                listbox.bind("<Double-1>", on_select)

            else:

                messagebox.showinfo("نتائج البحث", "لم يتم العثور على أي نتائج.")
    def on_hover(self, event):

        event.widget.configure(style="Hover.TButton")



    def on_leave(self, event):

        date_str = self.current_date.replace(day=int(event.widget.cget("text"))).strftime('%Y-
%m-%d')

        if date_str in self.new_event_dates:

            event.widget.configure(style="NewEventDay.TButton")

        elif date_str in self.events:

            event.widget.configure(style="EventDay.TButton")

        else:

            event.widget.configure(style="TButton")



    def prev_month(self):

        self.current_date = self.current_date.replace(day=1) - timedelta(days=1)

        self.update_calendar()
    def next_month(self):

        if self.current_date.month == 12:

            self.current_date = self.current_date.replace(year=self.current_date.year + 1, month=1, day=1)

        else:

            self.current_date = self.current_date.replace(month=self.current_date.month + 1, day=1)

        self.update_calendar()



    def prev_year(self):

        self.current_date = self.current_date.replace(year=self.current_date.year - 1, month=1, day=1)

        self.update_calendar()



    def next_year(self):

        self.current_date = self.current_date.replace(year=self.current_date.year + 1, month=1, day=1)

        self.update_calendar()
    def add_event(self, date=None):

        if date is None:

            date = self.selected_date

        event_text = simpledialog.askstring("إضافة حدث", "أدخل تفاصيل الحدث:")

        if event_text:

            if date in self.events:

                self.events[date].append(event_text)

            else:

                self.events[date] = [event_text]

            self.new_event_dates.add(date)

            self.show_events_for_date(date)

            self.update_calendar()



if __name__ == "__main__":

    root = tk.Tk()
```

```
app = AdvancedCalendarApp(root)

root.mainloop()
```