天气信息查询应用设计与实现

自12班　阿拉帕提·吐尔逊　2019010363

2023 年 12 月 14 日

1　总体要求

设计并实现一个天气信息查询应用，通过调用国内和国际天气API接口，提供用户友好的图形用户界面，支持实时天气查询、图表展示和用户收藏功能。

2　实现的功能

- 城市搜索:用户可以自行输入城市名称来搜索天气信息。
- 天气显示:

  (1)显示当日未来 24h (或者当日 24h)的天气状况，包括温度、湿度、风力风向等。

  (2) 展示所选城市的不同日期(包括今日和未来几日)对应的天气状况，包括温度、湿度、风力风向等。
- 数据更新:应用应能够定时从 API 获取最新数据，保持信息的实时性。
- 错误处理:当搜索不到相关城市或者API 服务不可用时，应用应能够给用户合适的反馈。
- 多地点保存:用户可以保存多个地点的天气信息，便于快速切换查看。
- 天气趋势图表:利用图表展示历史和预测的天气数据趋势，如温度曲线图等。
- 多语言支持：课根据用户偏好，切换应用中英文显示。

3　模块讲解

3.1　依赖库

- requests：用来获取API返回的JSON数据。
- tkinter：用来创建图形用户界面。
- matplotlib：用来绘制天气趋势图。
- json：用来解析一个JSON数据文件，该文件是OpenWeatherMap提供的地理信息。

- datetime：用来获取当前时间。
- os：用来获取当前路径。

3.2 API接口选择

在本实验中，选择的国内API为高德地图天气API，https://restapi.amap.com/v3/weather/weatherInfo?key=&city=&extensions=all，包括国内的3241个省市区的天气信息供选择，天气信息的类别分别有白天/黑夜气温、湿度、风力风向、白天/黑夜天气情况等信息。可以提供未来四日的天气预报。数据量简约，查询速度快，适合本实验的要求。

国际API为OpenWeatherMap API，http://api.openweathermap.org/data/2.5/forecast?lat=lat&lon=lon&appid=appid。包括国际上239个国家对应的城市和地区的天气信息，天气信息类别有气温、体感温度、风力风向、经纬度、压强、湿度等信息。可以提供未来六日，间隔3小时的精确天气预报。数据量较大，查询速度较慢，但是提供了更多的天气信息。

3.3 系统设计架构

设计一个基于tkinter的图形用户界面，通过requests库调用API接口获取天气数据。核心模块包括Weather(GlobalWeather)、WeatherGet(GlobalWeatherGet)、MyFavoriteCity以及WeatherGUI。

1. Weather类和GlobalWeather类分别表示国内和国际城市的天气信息的数据结构，包括城市、日期、温度等信息。由于两种天气信息各有异同，因此没有设计继承类，而分别设计了两个类别。
2. WeatherGet和GlobalWeatherGet类负责从API获取天气信息，并将其转化为Weather和GlobalWeather对象。
    - WeatherGet类和GlobalWeatherGet类都包含__init__()、update_weather()、add_weather()、get_weather()等公有函数，负责将API返回的JSON数据转化为Weather和GlobalWeather对象，用来存储和查询。
3. MyFavoriteCity类用于管理用户收藏的城市和相应的天气信息。
    - MyFavoriteCity类包含__init__()、add()、delete()、get_classofcity()、get_city_codes()、get_city_latlon()、get_favor_cityls()、__del__()等公有函数，负责管理用户收藏的城市和相应的天气信息。
    - 在__init__()中，在类初始化时，从文件中读取用户以前收藏的城市信息，存储在列表中。在__del__()中，在类销毁时，将用户收藏的城市信息写入文件。以便下次使用时读取。从而实现了数据的持久化。
4. WeatherGUI类是图形用户界面的核心，负责展示天气信息、图表绘制和用户交互。
    - WeatherGUI类包含__init__()、update_weather()、create_menu()、national()、international()、show_favorite()、chinese()、english()等主要的公有函数。
    - update_weather()函数负责定时从API获取天气信息，并更新界面上的天气信息。保证数据的实时性。

- create_menu()函数负责创建菜单栏，包括国内/国际天气查询、收藏列表、语言切换等功能。
- national()函数创建国内天气查询窗口，调用WeatherGet类获取天气信息，并在界面上展示。
- international()函数创建国际天气查询窗口，调用GlobalWeatherGet类获取天气信息，并在界面上展示。
- show_favorite()函数创建收藏列表窗口，调用MyFavoriteCity类获取用户收藏的城市信息和天气信息，并在界面上展示。
- chinese()和english()函数分别切换界面语言为中文和英文。

3.4 核心模块详解

1. **national()**

   该函数主要负责创建国内天气查询窗口的所有组建，并根据用户的交互，调用相关函数显示天气信息。

   城市和日期下拉框回调函数

   - city_combobox_selected()函数和date_combobox_selected()函数分别是界面中城市下拉框和日期下拉框的回调函数，前者将调用WeatherGet类获取天气信息，而后使用update_weather_info()函数更新界面天气信息，后者将直接调用update_weather_info()函数更新天气信息。

```python
# 城市下拉框选择事件
def city_combobox_selected(self, event):
    self.citysearch_entry.delete(0, tk.END) # 清空搜索框中的内容
    selected_date = self.date.get()  # 获取当前选择的日期
    city_name = self.city.get()  # 获取用户选择的新城市名
    # 使用天气 API 获取新的天气信息
    self.weather_get = WeatherGet(self.city_code[city_name]['adcode'])
    new_weather = self.weather_get.get_weather(selected_date)
    # 更新 GUI 显示的天气信息
    self.update_weather_info(new_weather)
# 日期下拉框选择事件
def date_combobox_selected(self, event):
    selected_date = self.date.get()  # 获取当前选择的日期
    new_weather = self.weather_get.get_weather(selected_date)
    # 更新 GUI 显示的天气信息
    self.update_weather_info(new_weather)
```

图 1: city_combobox_selected()函数和date_combobox_selected()函数

   - update_weather_info()函数负责更新界面上的天气信息，包括实时温度、湿度、风向等。

```python
# 更新 GUI 上的天气信息显示，根据传入的天气信息（例如温度、湿度等）
def update_weather_info(self, weather_info):
    self.temperature_entry.delete(0, tk.END)  # 清空温度 Entry 中的内容
    # ............
    self.week_entry.insert(0, weather_info.week)

    self.draw_tempreture(self.maincanvas, self.weather_get)
    # 更新上次更新的时间
    self.last_update_time = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    self.last_update_time_label.config(text = self.last_update_time)
```

图 2: update_weather_info()函数

搜索框回调函数

- search()函数将实时根据用户输入查找库中的可查询的城市信息，并把可能的城市信息在搜索显示框中显示。

```python
# 搜索框输入事件
def search(self,event):
    # 获取输入框的内容
    content = self.citysearch_entry.get()
    # 在城市代码中搜索可能的城市
    possible_cities = [city for city in self.city_code.keys() if content in city]
    # 更新搜索结果标签
    self.search_results_listbox.delete(0, tk.END)
    for city in possible_cities:
        self.search_results_listbox.insert(tk.END, city)
```

图 3: search()函数

- on_select()函数是搜索显示框的回调函数，将用户选择的城市信息填入搜索框中。

```python
# 搜索结果列表框点击事件
def on_select(self,event):
    selected_city = self.search_results_listbox.get(
        self.search_results_listbox.curselection())
    # 将选中的城市填充到输入框中
    self.citysearch_entry.delete(0, tk.END)
    self.citysearch_entry.insert(0, selected_city)
```

图 4: on_select()函数

- citysearch_button_clicked()函数是搜索按钮的回调函数，将根据用户输入的城市信息调用WeatherGet类获取天气信息。

```python
# 搜索按钮点击事件
def citysearch_button_clicked(self):
    self.search_results_listbox.delete(0, tk.END)
    city_name = self.citysearch_entry.get()
    if city_name in self.city_code.keys():
        self.city.set(city_name)
        self.city_combobox['values'] = list(self.city_code.keys())
        self.city_combobox.current(list(self.city_code.keys()).index(city_name))
        self.weather_get = WeatherGet(self.city_code[city_name]['adcode'])
        new_weather = self.weather_get.get_weather(self.date.get())
        self.update_weather_info(new_weather)
    else:
        messagebox.showinfo('错误', '无法获取天气信息')
```

图 5: citysearch_button_clicked()函数

天气趋势图

- draw_tempreture()函数负责根据已获取的天气信息绘制未来几天的天气趋势图，并把图片嵌入到窗口中。

```python
# 根据当前选择的城市天气信息，画出温度变化图
def draw_tempreture(self, tmp_canvas, weather_get):
    x_values = self.weather_get.datelist
    y1_values = []
    y2_values = []
    for x_value in x_values:
        y1_values.append(int(weather_get.get_weather(x_value).daytemperature))
        y2_values.append(int(weather_get.get_weather(x_value).nighttemperature))
    fig = Figure(figsize=(5,2), dpi=100)
    a = fig.add_subplot(111)
    a.plot(x_values, y1_values, label='day')
    a.plot(x_values, y2_values, label='night')
    # 在每个点上显示温度数值
    for i, txt in enumerate(y1_values):
        a.annotate(txt, (x_values[i], y1_values[i]))
    for i, txt in enumerate(y2_values):
        a.annotate(txt, (x_values[i], y2_values[i]))
    a.set_xlabel('date')
    a.set_ylabel('temperature/℃')
    a.set_axis_on()
    a.grid(True, axis='y')
    canvas = FigureCanvasTkAgg(fig, master=tmp_canvas)  # A tk.DrawingArea.
    canvas.draw()
    canvas.get_tk_widget().place(x=20, y=220, width=500, height=200)
```

图 6: draw_tempreture()函数

2. **international()**

该函数创建国际天气查询窗口的所有组建，并根据用户的交互，调用相关函数显示天气信息。

国家、城市、时间下拉框回调函数

- country_combobox_selected()函数是国家下拉框的回调函数，将更新城市下拉框城市列表，而后调用GlobalWeatherGet类获取天气信息。

```
# 国家下拉框选择事件
def country_combobox_selected(self, event):
    self.international_city_combobox['values'] = list( # 根据选定的城市列表更新城市下拉框
        national_citys(self.international_countrys, self.countryfullnames[self.country.get()]))
    self.country2city.set(list(national_citys( # 设置默认城市为城市列表中的第一个
        self.international_countrys, self.countryfullnames[self.country.get()]))[0])
    self.citylat, self.citylon = citys_lat_lon( # 获取城市的经纬度
        self.international_countrys, self.countryfullnames[self.country.get()], self.country2city.get())
    # 根据经纬度获取天气信息
    self.globalweather_get = GlobalWeatherGet(self.citylat, self.citylon)
    self.globalweather = self.globalweather_get.get_weather(self.international_time.get())
    # 更新天气信息
    self.international_weather_update()
    # 画出温度变化图
    self.draw_international_tempreture(self.secondcanvas, self.globalweather_get)
```

图 7: country_combobox_selected()函数

- city_combobox_selected()函数是城市下拉框的回调的函数，将根据已选的国家、城市调用GlobalWeatherGet类获取天气信息。

```
# 国际城市下拉框选择事件
def international_city_combobox_selected(self, event):
    self.citylat, self.citylon = citys_lat_lon( # 获取城市的经纬度
        self.international_countrys, self.countryfullnames[
            self.country.get()], self.country2city.get())
    # 根据经纬度获取天气信息
    self.globalweather_get = GlobalWeatherGet(self.citylat, self.citylon)
    self.globalweather = self.globalweather_get.get_weather(self.international_time.get())
    # 更新天气信息
    self.international_weather_update()
    # 画出温度变化图
    self.draw_international_tempreture(self.secondcanvas, self.globalweather_get)
```

图 8: city_combobox_selected()函数

- international_time_combobox_selected()是日期下拉框的回调函数，将根据已选的日期已获取的天气信息查找并更新天气信息。

```
# 国际日期下拉框选择事件
def international_time_combobox_selected(self, event):
    # 获取当前选择的日期
    selected_time = self.international_time.get()
    # 根据时间获取天气信息
    self.globalweather = self.globalweather_get.get_weather(selected_time)
    # 更新天气信息
    self.international_weather_update()
```

图 9: international_time_combobox_selected()函数

- international_weather_update()函数负责更新界面上的天气信息，包括温度、湿度、风向等。

```python
def international_weather_update(self):
    # 更新 GUI 显示的天气信息
    self.international_temperature_entry.delete(0, tk.END)  # 清空温度 Entry 中的内容
    # ..............
    self.international_wind_entry.insert(0, self.globalweather.wind)  # 显示新的风向信息

    # 更新上次更新的时间
    self.last_update_time = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    self.last_update_time_label.config(text = self.last_update_time)
```

图 10: international_weather_update()函数

天气趋势图

- draw_internationa_tempreture()函数负责根据已获取的天气信息绘制未来几天的天气趋势图，并把图片嵌入到窗口中。

```python
# 根据当前选择的城市天气信息，画出温度变化图
def draw_international_tempreture(self, tmp_canvas, globalweather_get):
    x_values = list(globalweather_get.timelist)
    y1_values = []
    for x_value in x_values:
        y1_values.append(globalweather_get.get_weather(x_value).temp_max)
    fig = Figure(figsize=(35,2), dpi=100)
    a = fig.add_subplot(111)
    # 只显示x_values的形式是2023-12-1 18:00:00，只留下12:1 18
    x_values = [x_value[8:10] + '/' + x_value[11:13]+'' for x_value in x_values]
    a.plot(x_values, y1_values, label='time')
    # 在每个点上显示温度数值
    for i, txt in enumerate(y1_values):
        a.annotate(txt, (x_values[i], y1_values[i]))
    a.set_xlabel('date')
    a.set_ylabel('temperature/℃')
    a.set_axis_on()
    a.grid(True, axis='y')
    # 创建一个可滚动的画布
    scroll_canvas = Canvas(tmp_canvas)
    scroll_canvas.place(x=50, y=220, width=500, height=200)
    # 创建一个滚动条
    scrollbar = Scrollbar(tmp_canvas, orient="horizontal", command=scroll_canvas.xview, width = 30)
    scrollbar.place(x=50, y=430, width=500, height=20)
    # 将滚动条连接到画布
    scroll_canvas.configure(xscrollcommand=scrollbar.set)
    # 创建一个Frame，将其添加到画布中
    frame = Frame(scroll_canvas)
    scroll_canvas.create_window((500,200), window=frame, anchor='center')
    # 将FigureCanvasTkAgg添加到Frame中
    canvas = FigureCanvasTkAgg(fig, master=frame)
    canvas.draw()
    canvas.get_tk_widget().pack(side="left", fill="both", expand=True) # 将画布填充到Frame中
    # 更新画布的滚动区域
    frame.update_idletasks()
    scroll_canvas.configure(scrollregion=scroll_canvas.bbox('all'))
```

图 11: draw_international_tempreture()函数

3. 收藏夹

- show_favorite()函数创建收藏列表窗口，调用MyFavoriteCity类获取用户收藏的城市信息和天气信息，并在界面上展示。

```python
# 查看收藏
def show_favorite(self):
    """
    show the window of the favorite weather information
    """
    if len(self.my_favorite_city.get_favor_cityls()) == 0:
        messagebox.showinfo('提示', '收藏夹为空')
    else:
        self.maincanvas.pack_forget()
        self.secondcanvas.pack_forget()
        self.myfavcanvas.pack()
        self.myfavcanvas.update() # 刷新窗口
        self.cityinlist = self.my_favorite_city.get_favor_cityls()
        self.my_fav_city = tk.StringVar()  # 用来存储用户选择的城市名
        self.my_fav_city.set(self.cityinlist[0])
        self.fav_city_label = tk.Label(self.myfavcanvas, text='城市：', bg='white')
        self.fav_city_label.place(x=20, y=20, width=50, height=20)
        self.fav_city_combobox = ttk.Combobox(self.myfavcanvas, textvariable= self.my_fav_city, state='re
        self.fav_city_combobox['values'] =  self.cityinlist
        self.fav_city_combobox.place(x=80, y=20, width=100, height=20)
        self.win1_delete_button = tk.Button(self.myfavcanvas, text='删除', command=self.delete_favorite)
        self.win1_delete_button.place(x=200, y=20, width=40, height=20)
        self.fav_city_combobox_selected(0)
        self.fav_city_combobox.bind('<<ComboboxSelected>>', self.fav_city_combobox_selected)
```

图 12: show_favorite()函数

- add_favorite()函数是添加收藏按钮的回调函数，在国内和国际天气信息显示界面时，将收藏的城市信息存储到列表中。

```python
# 添加到收藏
def add_favorite(self):
    #如果当前页面是self.maincanvas，则添加到收藏
    if self.maincanvas.winfo_ismapped():
        cityinfo = {}
        cityinfo['class'] = self.weather.__class__.__name__
        cityinfo['cityname'] = self.city.get()
        cityinfo['code'] = self.city_code[self.city.get()]['adcode']
        self.my_favorite_city.add(cityinfo)
        messagebox.showinfo('提示', '添加成功')
    # 如果当前页面是self.secondcanvas，将国际类的天气信息添加到收藏
    elif self.secondcanvas.winfo_ismapped():
        self.citylat, self.citylon = citys_lat_lon(
            self.international_countrys, self.countryfullnames[self.country.get()], s
        cityinfo = {}
        cityinfo['class'] = self.globalweather.__class__.__name__
        cityinfo['cityname'] = self.country2city.get()
        cityinfo['lat'] = self.citylat
        cityinfo['lon'] = self.citylon
        self.my_favorite_city.add(cityinfo)
        messagebox.showinfo('提示', '添加成功')
    else:
        messagebox.showinfo('提示', '当前页面天气页面，无法添加')
```

图 13: add_favorite()函数

- delete_favorite()函数是删除收藏按钮的回调函数，在收藏列表界面时，将收藏的城市信息从列表中删除。

```python
# 从收藏中删除
def delete_favorite(self):
    self.my_favorite_city.delete(self.fav_city_combobox.get())
    # 如果收藏夹空的，则关掉my_favorite_window窗口
    if len(self.my_favorite_city.get_favor_cityls()) == 0:
        self.myfavcanvas.pack_forget()
        self.maincanvas.pack()
        messagebox.showinfo('提示', '删除成功')
    else:
        # 删除后清空 self.win1_city_combobox
        self.my_fav_city.set(list(self.my_favorite_city.get_favor_cityls())[0])
        # 重新加载收藏夹中的城市
        self.fav_city_combobox['values'] = list(self.my_favorite_city.get_favor_cityls())
        self.fav_city_combobox_selected(0)
```

图 14: delete_favorite()函数

- fav_city_combobox_selected()函数是收藏列表界面城市下拉框的回调函数，将根据已选的城市信息从API获取天气信息并更新界面。

```python
# 收藏界面城市下拉框选择事件
def fav_city_combobox_selected(self,event):
    # 清空页面上的信息，y = 60及以上的信息保留
    for widget in self.myfavcanvas.winfo_children():
        if widget.winfo_y() > 50:
            widget.destroy()
    # 如果选择的城市是国内类的天气信息
    if self.my_favorite_city.get_classofcity(self.my_fav_city.get()) == 'Weather'
        citycode = self.my_favorite_city.get_city_codes(self.my_fav_city.get())
        weatherget = WeatherGet(citycode)
        # ------------------------------
        # 根据用户选择的城市显示各个天气信息(略)
        # ------------------------------
        # 画气温变化图
        self.draw_tempreture(self.myfavcanvas, weatherget)
    elif :
        # ....................................
    else:
        messagebox.showinfo('提示', '无法获取天气信息')
    # 选择的语言
    if self.language == 'chinese':
        self.chinese()
    else:
        self.english()
```

图 15: fav_city_combobox_selected()函数

3.5 交互流程

1. 菜单栏

- 收藏：包含“添加至收藏和查看收藏”。选择添加至收藏可将当前城市添加至收藏列表，选择查看收藏可查看收藏列表。
- 语言/language：选择中文或英文。选择中文将切换界面语言为中文，选择英文将切换界面语言为英文。

- 国内/国际：选择国内将切换到国内天气查询界面，选择国际将切换到国际天气查询界面。

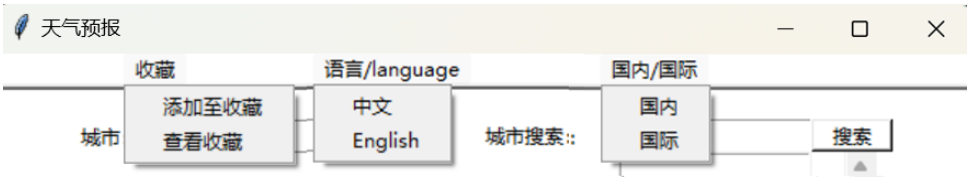

图 16: 菜单栏

2. 国内天气查询：

- 用户打开应用，默认显示北京市天气信息
- 在城市下拉框中选择城市或者在日期下拉框选择日期均能更新天气信息。
- 如果想搜索指定信息可在搜索框中输入城市信息，（搜索显示框将显示匹配的城市，点击可将其填入搜索框中），点击搜索按钮，即可更新天气信息。
- 显示查询结果，包括实时温度、湿度、风向等。
- 下方显示未来几天的天气趋势图和最近一次的更新时间。

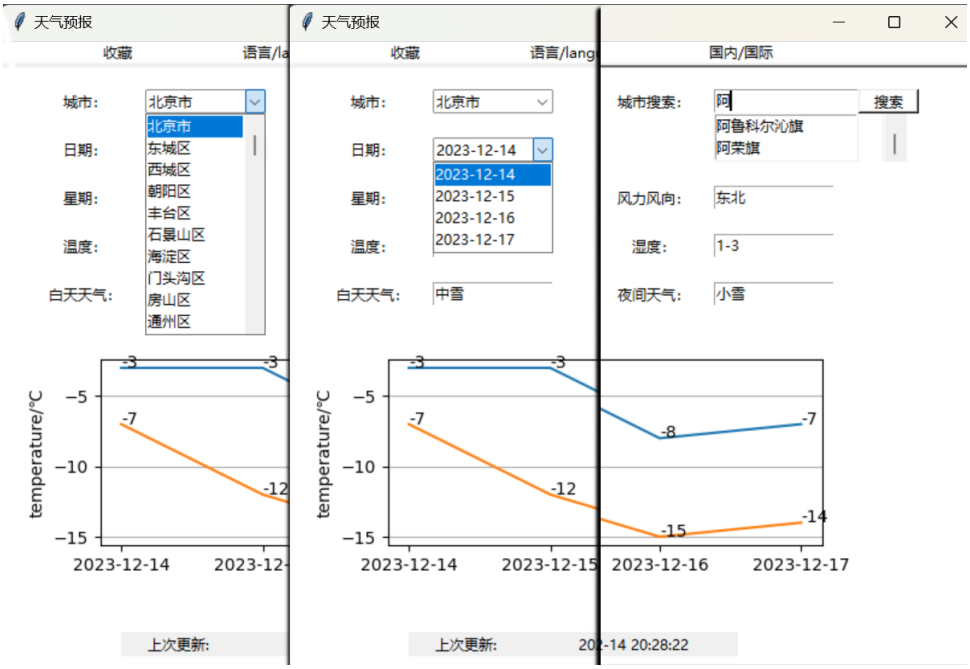

图 17: 国内天气查询

3. 国际天气查询：

- 点击菜单栏国际按钮，切换到国际天气查询界面。默认显示China，Beijing的天气信息。
- 在国家下拉框中选择国家，在城市下拉框选择城市，在时间下拉框选择时间均能更新界面天气信息。

- 界面显示查询结果，包括温度、湿度、风向等。
- 下方显示未来几天的天气趋势图（可通过滑条拖动查看）和最近一次的更新时间

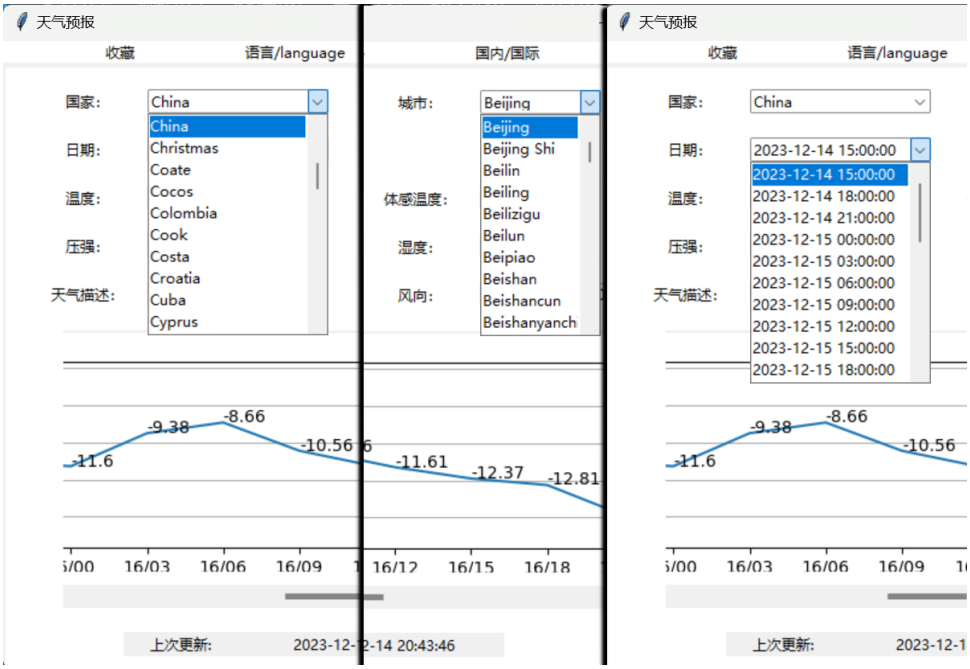

图 18: 国际天气查询

4. 查看收藏列表：

- 用户点击“添加收藏”按钮，将把当前城市添加至收藏列表。
- 用户点击“收藏列表”按钮，若已有收藏城市，将显示收藏列表界面。否则提示用户收藏列表为空。
- 收藏列表界面显示用户收藏的城市列表，在城市下拉框选择城市，可更新界面天气信息。
- 用户点击“删除”按钮将把当前城市从收藏列表中删除。
- 界面显示查询结果，包括温度、湿度、风向等。
- 下方显示未来几天的天气趋势图（国际城市的可通过滑条拖动查看）。

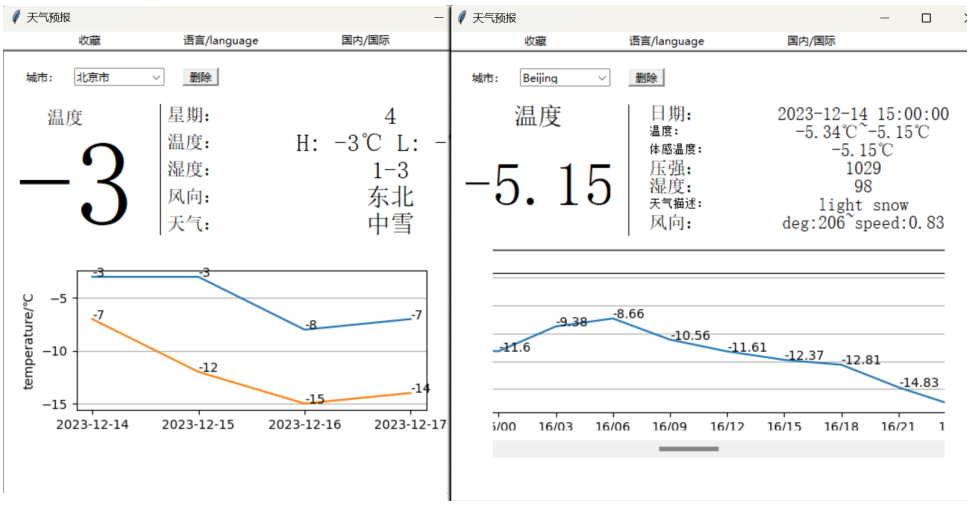

图 19: 收藏列表

5. 语言切换：

- 用户可以通过切换语言按钮选择中文或英文。
- 应用根据选择切换显示语言。

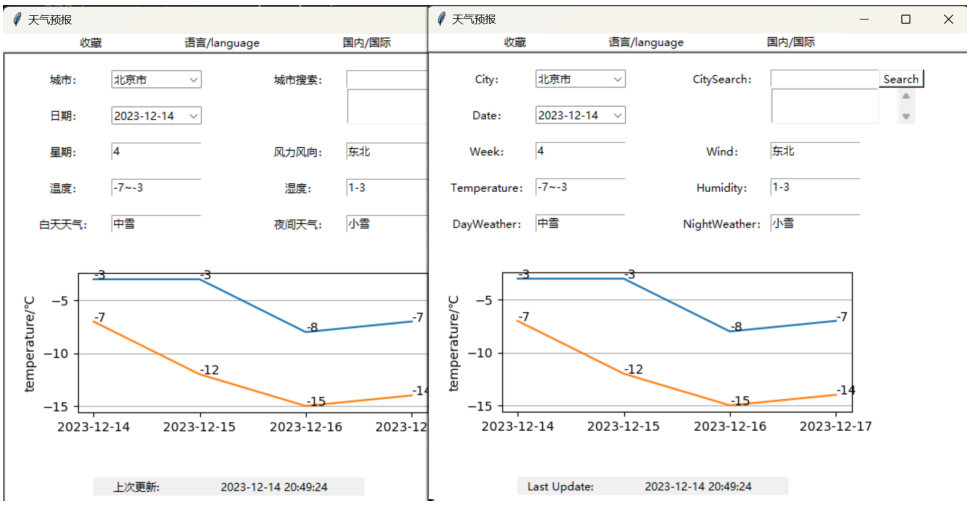

图 20: 语言切换

4 总结与体会

- 通过本次实验，我学会了如何使用requests库调用API接口获取天气信息，如何使用tkinter库创建图形用户界面，如何使用matplotlib库绘制天气趋势图。
- 我将课上学到的知识用到了实践上，对数据类型，类，图形界面等有了更加深刻的认识。
- 从写完第一版到最终版，我从一个程序员的角度考虑复用性、可读性、可维护性等问题，对代码进行了多次重构，使得代码更加简洁、高效。

- 在实验中，我遇到了很多问题，如如何将API返回的JSON数据转化为对象，如何将天气信息嵌入到图形界面中，如何实现数据的持久化等等。通过查阅资料，我一一解决了这些问题，对Python的使用更加熟练。
- 最后，感谢老师和助教平时的教导和解惑，让我对python的掌握更深了一些。