**1. 代码整体框架**

**这个代码的目标是获取一个中国城市的天气数据。整体结构分为三个主要部分：**

**• 获取城市的地理坐标：通过 OpenWeatherMap 的地理编码 API 获取城市的经纬度。**

**• 获取天气数据：通过获取到的城市坐标（经纬度）查询天气数据。**

**• 程序入口：用户输入城市名，程序通过获取城市坐标和天气数据来输出结果。**

**2. 导入库**

**import requests**

**import os**

**#from dotenv import load\_dotenv**

**• requests: 这是一个 Python 库，用于简化 HTTP 请求的发送，尤其是 GET、POST 请求。它帮助我们方便地访问网络资源，例如 OpenWeatherMap API。**

**• os: 这是 Python 的标准库，用于与操作系统交互。我们用它来获取环境变量（比如 API 密钥）和设置环境变量。**

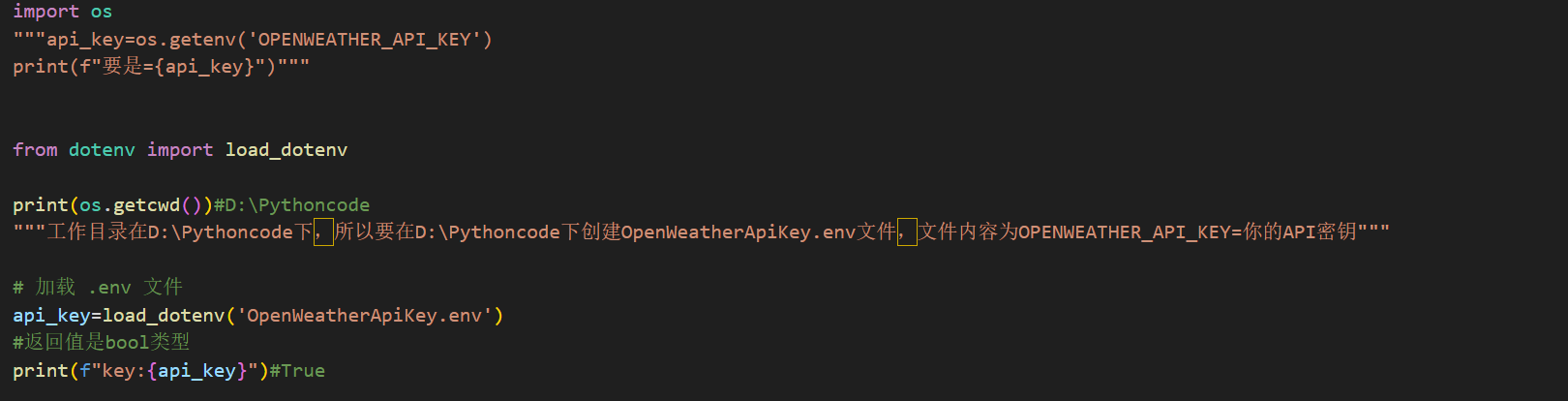
**• dotenv: 这是一个第三方库，帮助我们从 .env 文件中加载环境变量。这可以帮助我们把敏感数据（如 API 密钥）从代码中分离出去，避免硬编码。**

**3. 加载 .env 文件**

**#load\_dotenv()**

**#• load\_dotenv()：加载当前目录中的 .env 文件。这个文件通常包含环境变量，比如 API 密钥等敏感信息。在这个代码中，我们假设 .env 文件里有 OPENWEATHER\_API\_KEY 这一环境变量，用于存储 OpenWeatherMap API 的密钥。**

**使用load\_dotenv注意.env文件位置**



**4. 获取城市的地理坐标**

**def get\_coordinates(city, api\_key):**

**try:**

**url = f"https://api.openweathermap.org/geo/1.0/direct?q={city}&limit=1&appid={api\_key}"**

**print(f"正在请求城市坐标: {url}")**

**response = requests.get(url, timeout=10)**

**response.raise\_for\_status()**

**data = response.json()**

**print(f"获取到的城市坐标数据: {data}")**

**if data:**

**lat = data[0]["lat"]**

**lon = data[0]["lon"]**

**return lat, lon**

**else:**

**print("错误：未找到该城市，请检查名称是否正确。")**

**return None, None**

**except requests.exceptions.RequestException as e:**

**print(f"请求发生错误：{e}")**

**return None, None**

**功能：获取城市的经纬度**

**• 参数：**

**• city: 用户输入的城市名（例如“北京”）。**

**• api\_key: 从环境变量中获取的 OpenWeatherMap API 密钥。**

**• 过程：**

**1. 构造一个 URL，使用 OpenWeatherMap 的地理编码 API 来请求城市的坐标。URL 中包含城市名称 (q={city})、请求返回的最大数量 (limit=1) 和 appid={api\_key}（API 密钥）。**

**2. 使用 requests.get(url, timeout=10) 向 API 发起 GET 请求，timeout=10 表示请求最长等待 10 秒，超时则抛出异常。**

**3. 如果请求成功，response.raise\_for\_status() 会检查 HTTP 状态码是否为 200（即请求是否成功）。如果失败，会抛出异常。**

**4. response.json() 将响应内容解析为 JSON 格式的数据（整个JSON响应通常是对象（用`{}`包裹），包含多个字段，其中某些字段可能是数组(列表)（如`weather`）），从中提取出城市的纬度（lat）和经度（lon）。**

**5. 如果获取到的城市坐标数据为空，返回 None, None 并提示用户城市未找到。**

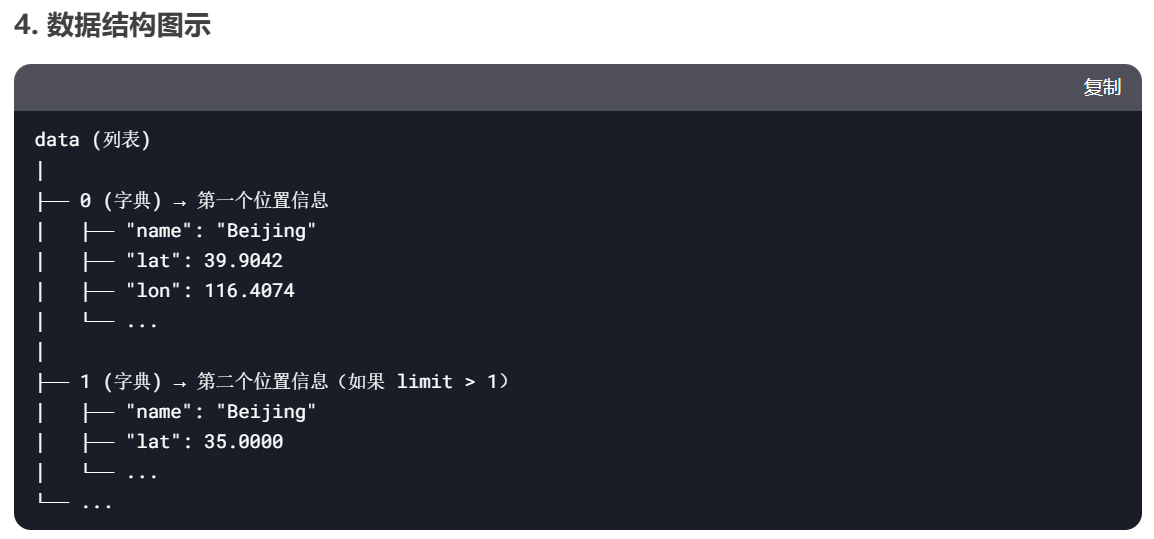
**关键单词：**

**• url: 指定请求的 API 地址。**

**• timeout: 请求的超时时间（单位为秒）。**

**• raise\_for\_status(): 检查请求的 HTTP 状态码是否为 200，若不是则抛出异常。**

**• response.json(): 解析响应内容为 JSON 格式。**



**5. 获取天气数据**



**def get\_weather(lat, lon, api\_key):**

**weather\_url = "https://api.openweathermap.org/data/2.5/weather"**

**params = {**

**"lat": lat,**

**"lon": lon,**

**"appid": api\_key,**

**"units": "metric", # 摄氏温度**

**"lang": "zh\_cn" # 中文描述**

**}**

**try:**

**print(f"正在请求天气数据: {weather\_url}, 参数: {params}")**

**response = requests.get(weather\_url, params=params, timeout=10) # 设置超时**

**response.raise\_for\_status()**

**return response.json()**

**except requests.exceptions.RequestException as e:**

**print(f"获取天气失败：{e}")**

**return None**

**功能：根据经纬度获取天气信息**

**• 参数：**

**• lat: 城市的纬度。**

**• lon: 城市的经度。**

**• api\_key: 从环境变量中获取的 OpenWeatherMap API 密钥。**

**• 过程：**

**1. 构造一个 URL 来请求 OpenWeatherMap 的天气数据 API。**

**2. params 是传递给 API 的参数，包含：**

**• "lat" 和 "lon"：城市的经纬度。**

**• "appid"：API 密钥。**

**• "units": "metric"：指定温度单位为摄氏度。**

**• "lang": "zh\_cn"：指定返回中文的天气描述。**

**3. 发送 GET 请求，使用 timeout=10 防止请求长时间阻塞。**

**4. 如果请求成功，通过 response.json() 解析返回的 JSON 数据。**

**5. 如果请求失败，捕获异常并打印错误信息。**

**关键单词：**

**• params: 传递给 API 的查询参数。**

**• response.json(): 将响应的 JSON 数据解析为 Python 字典。**

**• timeout: 防止请求长时间挂起。**





**6. 主程序入口**

**def main():**

**api\_key = os.getenv('OPENWEATHER\_API\_KEY')**

**if not api\_key:**

**print("错误：没有找到 API Key，请检查环境变量设置。")**

**return**

**city = input("请输入中国城市名（例如：北京）：").strip()**

**# 获取城市的坐标**

**lat, lon = get\_coordinates(city, api\_key)**

**if lat is not None and lon is not None:**

**# 获取天气数据**

**weather\_data = get\_weather(lat, lon, api\_key)**

**if weather\_data:**

**print(f"城市：{city}")**

**print(f"天气：{weather\_data['weather'][0]['description']}")**

**print(f"温度：{weather\_data['main']['temp']}°C")**

**else:**

**print("无法获取该城市的天气信息。")**

**else:**

**print("未能获取城市坐标，请检查网络连接或 API 设置。")**





**功能：程序的主控制逻辑**

**• 过程：**

**1. 使用 os.getenv('OPENWEATHER\_API\_KEY') 获取环境变量中的 API 密钥。如果没有获取到 API 密钥，则提示错误信息并结束程序。**

**2. 使用 input() 获取用户输入的城市名，并去除前后的空格。**

**3. 调用 get\_coordinates(city, api\_key) 获取城市的经纬度。**

**4. 如果成功获取到坐标（lat, lon 不为 None），调用 get\_weather(lat, lon, api\_key) 获取天气数据。**

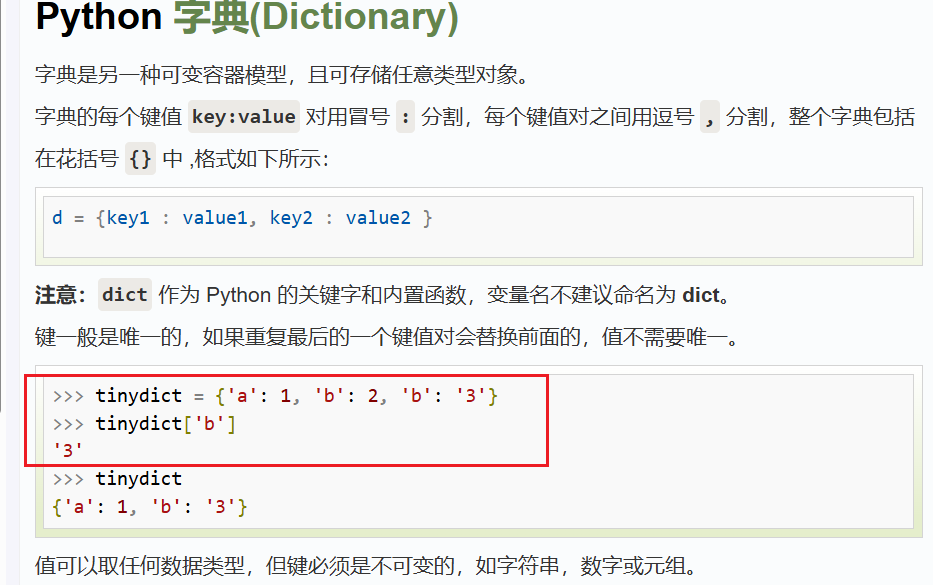
**5. 如果成功获取天气数据，打印城市名称、天气描述和当前温度。如果获取失败，则打印错误信息。**

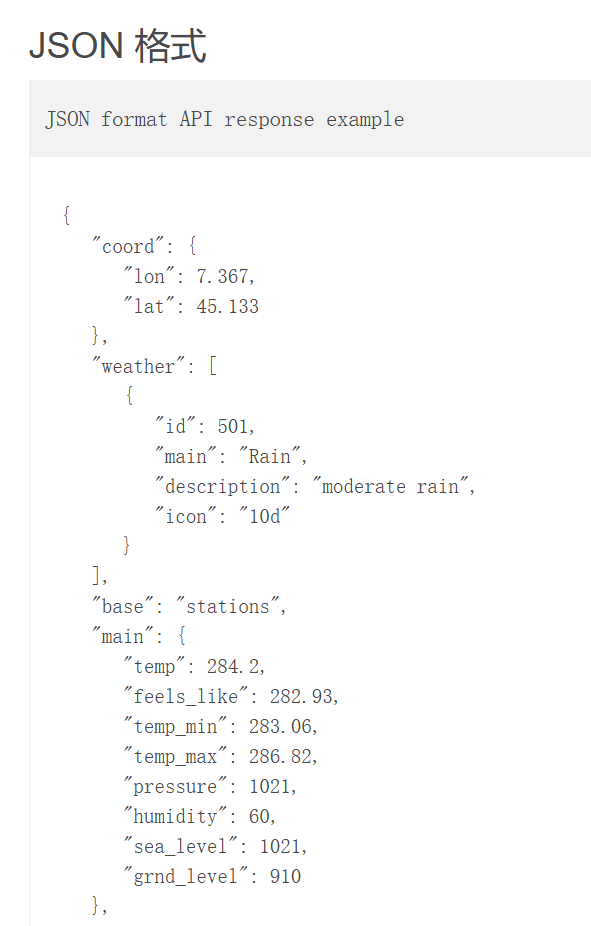
**关键单词：**

**• input(): 用于接收用户输入。**

**• os.getenv(): 获取环境变量中的值。**

**• print(): 打印信息到控制台。**







**7. 程序启动**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**• if \_\_name\_\_ == "\_\_main\_\_": 这行代码用于判断是否是直接运行该脚本。如果是，它会执行 main() 函数。**

**• main() 函数是程序的入口点，包含了整个程序的逻辑。**

**总结**

**整体框架结构：**

**1. API 请求：使用 requests 向 OpenWeatherMap 发送请求，获取城市的坐标和天气数据。**

**2. 错误处理：使用 try-except 捕获网络请求中的异常，防止程序崩溃。**

**3. 用户交互：通过 input() 获取用户输入的城市，并根据经纬度查询天气数据。**

**4. 环境变量：通过 .env 文件存储 API 密钥，避免硬编码密钥。**

**每一段代码的作用：**

**• get\_coordinates()：通过城市名称获取经纬度。**

**• get\_weather()：根据经纬度获取天气数据。**

**• main()：程序主逻辑，处理用户输入、调用相关函数并输出结果。**

**如果你有任何其他问题，欢迎继续提问！**

**1. 检查虚拟环境的配置**

**确保你在 VS Code 中已经选择了正确的 Python 解释器（即你虚拟环境中的 Python 解释器）。**

**步骤：**

**1. 打开 VS Code。**

**2. 按 Ctrl + Shift + P 打开命令面板。**

**3. 输入 Python: Select Interpreter 并选择你虚拟环境中的 Python 解释器，路径应类似于：**

**D:\pythoncode\Lib\.venv\Scripts\python.exe**

**确保你选择了正确的虚拟环境 Python 解释器**

**调试的配置文件如图program：${debug那个文件，写那个名字}**

