HTTP 状态码解释

这段代码使用 Python 的 requests 库向天气服务网站 wttr.in 发送请求，并获取响应的状态码。让我详细解释一下：

python

import requests　*# 导入 requests 库，用于发送 HTTP 请求*

url = "https://wttr.in/Beijing?format=j1"　*# 定义请求的 URL，获取北京天气信息的 JSON 格式数据*

res = requests.get(url)　*# 发送 GET 请求并获取响应对象*

print(res.status_code)　*# 打印 HTTP 响应状态码*

状态码的含义

status_code 是 HTTP 响应中的一个标准部分，它表示服务器对请求的处理结果。状态码是一个三位数字，分为五个类别：

1xx - 信息性状态码

表示请求已被接收，需要继续处理

2xx - 成功状态码

表示请求已成功被服务器接收、理解并接受

- **200 OK** - 请求成功
- **201 Created** - 资源已被创建
- **202 Accepted** - 请求已接受但尚未处理完成

3xx - 重定向状态码

表示需要客户端采取进一步的操作才能完成请求

- **301 Moved Permanently** - 资源已永久移动到新位置
- **302 Found** - 资源临时移动到不同位置
- **304 Not Modified** - 资源未修改，可使用缓存版本

4xx - 客户端错误状态码

表示客户端可能出错，妨碍了服务器的处理

- **400 Bad Request** - 请求无效或无法被服务器理解
- **401 Unauthorized** - 需要身份验证
- **403 Forbidden** - 服务器理解请求但拒绝执行
- **404 Not Found** - 请求的资源不存在

5xx - 服务器错误状态码

表示服务器在处理请求的过程中有错误或异常状态

- **500 Internal Server Error** - 服务器内部错误
- **502 Bad Gateway** - 服务器作为网关或代理时收到无效响应
- **503 Service Unavailable** - 服务器暂时不可用

为什么 res.status_code 能获取状态码

1. 当你使用 **requests.get(url)** 发送请求时，requests 库会：
    - 创建一个 HTTP 请求
    - 通过网络发送到服务器
    - 接收服务器的响应
2. 服务器响应包含：
    - 状态行（包含状态码）
    - 响应头（包含元数据）
    - 响应体（包含实际内容）
3. requests 库将这些信息封装到一个 Response 对象中（这里是 **res** 变量）
4. **status_code** 是 Response 对象的一个属性，它提取并存储了服务器返回的状态码

Response 对象的完整属性详解

在 requests 库中，当你发送一个 HTTP 请求后，会返回一个 Response 对象，它包含了服务器返回的所有信息。以下是 Response 对象的主要属性：

核心属性

1. `status_code`

HTTP 状态码，表示请求的成功或失败状态（如 200 表示成功，404 表示未找到等）。

2. `headers`

包含服务器响应头的字典，提供了关于响应的元数据信息。

3. `text`

以字符串形式返回响应内容。requests 会自动根据响应头中的编码信息解码内容。

4. `content`

以字节形式返回响应内容。适用于非文本内容（如图片、文件等）。

5. `json()`

如果响应内容是 JSON 格式，此方法会将其解析为 Python 字典或列表。

6. `url`

最终请求的 URL，可能与原始请求 URL 不同（由于重定向）。

7. `history`

一个包含所有重定向响应对象的列表，按从最早到最近的顺序排列。

8. `reason`

与状态码对应的文本描述（如 "OK" 对应 200，"Not Found" 对应 404）。

9. `cookies`

服务器返回的 Cookies，是一个 RequestsCookieJar 对象。

10. `encoding`

响应内容的编码。可以修改此属性来改变文本的编码方式。

11. `elapsed`

从发送请求到收到响应所经过的时间。

12. `request`

返回与此响应相关联的 PreparedRequest 对象。

实际应用建议

在实际开发中，处理响应时应该：

1. 总是检查状态码或使用 **Response.raise_for_status()**
2. 根据内容类型选择适当的方式处理响应内容
3. 考虑使用上下文管理器确保资源正确释放
4. 处理可能的异常