Deferred Request Callbacks

One of the design principles of Flask is that response objects are created and passed down a chain of potential callbacks that can modify them or replace them. When the request handling starts, there is no response object yet. It is created as necessary either by a view function or by some other component in the system.

What happens if you want to modify the response at a point where the response does not exist yet? A common example for that would be a **`before_request()`** callback that wants to set a cookie on the response object.

One way is to avoid the situation. Very often that is possible. For instance you can try to move that logic into a **`after_request()`** callback instead. However, sometimes moving code there makes it more more complicated or awkward to reason about.

As an alternative, you can use **`after_this_request()`** to register callbacks that will execute after only the current request. This way you can defer code execution from anywhere in the application, based on the current request.

At any time during a request, we can register a function to be called at the end of the request. For example you can remember the current language of the user in a cookie in a **`before_request()`** callback:

```
from flask import request, after_this_request

@app.before_request
def detect_user_language():
    language = request.cookies.get('user_lang')

    if language is None:
        language = guess_language_from_request()

        # when the response exists, set a cookie with the language
        @after_this_request
        def remember_language(response):
            response.set_cookie('user_lang', language)
            return response

    g.language = language
```