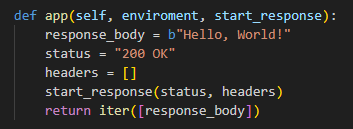
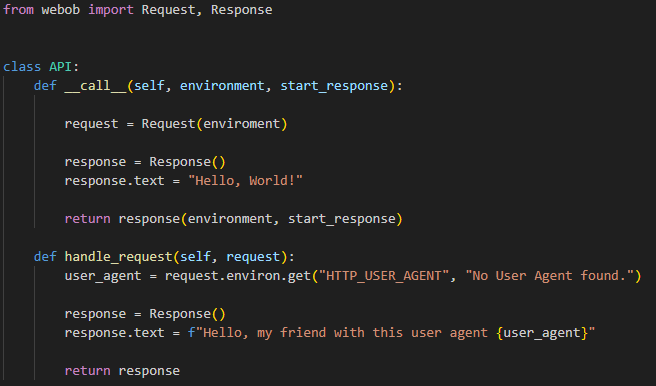
**ANOTAÇÕES PARA O PROJETO PRINCIPAL**



Essa é a forma mais básica de criar um código para um framework web que retornará um “*Hello, World!*” em uma página web. A ferramenta usada para criar o servidor WSGI foi a waitress. Segue abaixo outras anotações:

**environment:** contém várias informações importantes do cliente: url informada, navegador, parâmetros, IPs e outros itens relacionados

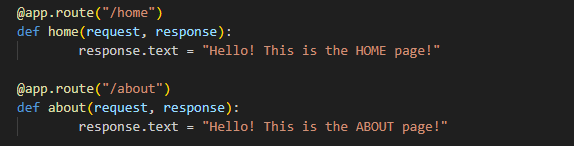
**start\_response:** é uma função callback que é necessária para iniciar uma resposta HTTP, seguido dela temos a resposta de conteúdo que é retornada pelo servidor.

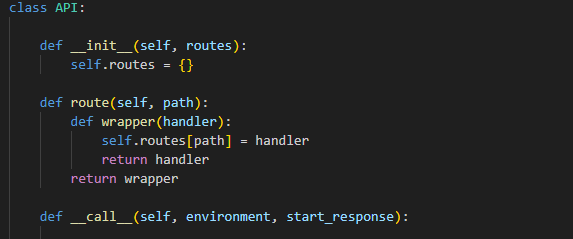


**WebOb:** Biblioteca que fornece objetos HTTP, que é o caso do código acima é Request e Response, para lidar com requisições e respostas do servidor.

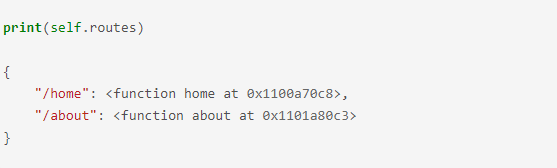
**\_\_call\_\_:** Uma função que faz o mesmo que a função app mostrada anteriormente, mas esta é feita utilizando os objetos Request e Response da biblioteca WebOb.

**handle\_request:** Função para lidar com requests. (Obs.: Na figura acima, o “response = Response(environment)” deveria ser “response = self.handle\_request(request)” para rodar direito e receber a função.

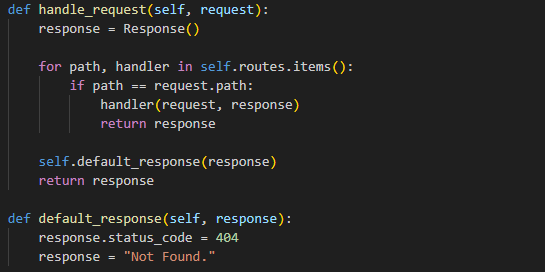




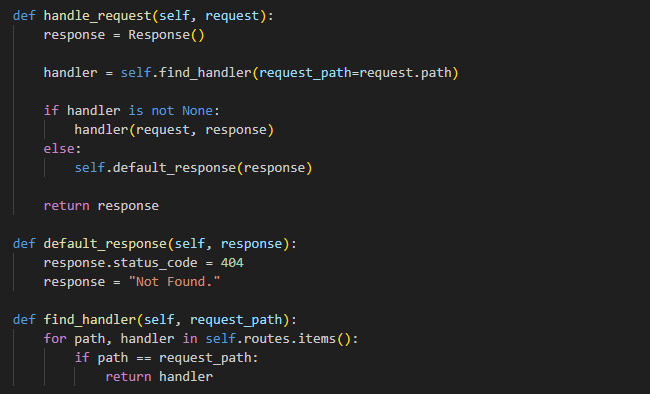
Na figura acima, estamos criando novas rotas que dão acesso a diferentes páginas da mesma aplicação. A função **route** tem o dever de armazenar paths com chaves e handlers como valores das chaves. Ficando mais ou menos assim:



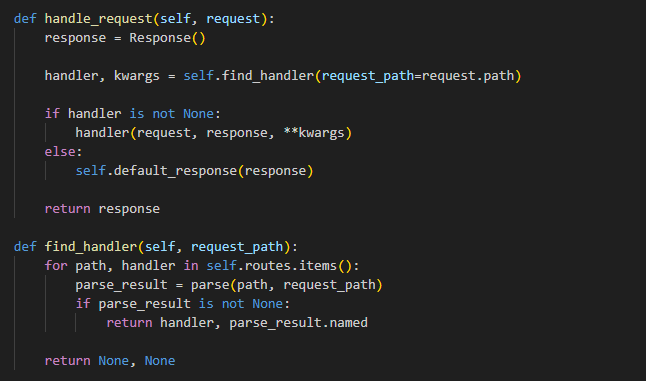
Agora, quando uma request chega, precisamos verificar seu path, encontrar um handler apropriado, chamar esse handler e retornar uma response apropriada. Como vocês podem verificar abaixo:



Como podem ver, a função recebe um request, verifica se existe um path que bate com esse request, então invoca um handler que dará uma response específica. Além disso, foi criada uma função de erro padrão, a **default\_response**, ela que informará que a página não foi encontrada.

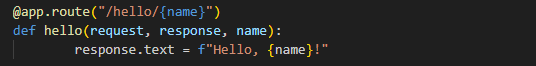


Essa função **find\_handler** é só uma forma de encontrar o handler caso não tenha nenhum. Além disso, ele diminui o trabalho da função **handle\_request**.

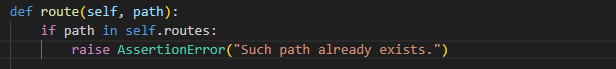


**\*\*kwargs:** Variável que permite a passagem de vários argumentos para alguma função.

**parse:** Veio da biblioteca de mesmo nome serve para relacionar informações textuais, no caso, para saber qual nome exibir na página hello/{nome}.



A nova função **find\_handler** retorna seu handler como anteriormente, mas também retorna um parse\_result (Esse que será usado para exibir o nome). Já na nova função **handle\_request** adicionamos o \*\*kwargs, que receberá as quantas variáveis textuais de nome o usuário quiser e que serão exibidas na página hello/{nome}.



Só uma condição para casa o criador do código criar mais de uma rota com o mesmo nome.