Exercises

1. Instructions

For these exercises, you should be checking your code by running the server and testing your routes.

**Exercise 1**

First thing's first, Let's add a simple route to make sure the server works.

Create a route /sanity in which accessing this route should return a simple string: "Server is up and running smoothly"

Now, let's allow the user to get some files from the server.

Open a dist directory in the root of your file. Add an images folder and an audio folder with some files in them and serve them in server.py.

To validate the code, try to access your files in the dist directory.

Try to access an image and an audio file.

**Note**: please check your routes are really working!

1. Exercise 2

Add the following data to a store.py file:

store = [

{ "name": "table", "inventory": 3, "price": 800 },

{ "name": "chair", "inventory": 16, "price": 120 },

{ "name": "couch", "inventory": 1, "price": 1200 },

{ "name": "picture frame", "inventory": 31, "price": 70 }

]

Now add a /priceCheck route in server.py which has one parameter: name

When this route is accessed, it should return the price of the item that was asked for.

So if someone goes to localhost:3000/priceCheck/couch, they should receive an **object** in response: {price: 1200}

If the item *doesn't* exist, the route should respond with {price: None}

1. Exercise 3

Add another python file: client.py that allows the user to input the name of a piece of furniture.

By now you should have the following files:

- server.py (the server)

- client.py (the client file that sends requests to the server)

- store.py (the data)

Now, in the client file make a GET request to /priceCheck with that input

After the request returns, print the price (using python print function).

**Note**: please check your routes are really working!

1. Exercise 4

Create another route in your server.py file called /buy which has one parameter: name

Accessing this route should reduce the inventory of that item by 1.

So if someone makes a request to localhost:3000/buy/chair, it should reduce the inventory of chair to 15. Make sure your route responds with the updated item.

1. Exercise 5

Let's add a client call, in the client.py file, that will use the /buy endpoint.

First ask the user what item does he want to buy (you can use the input method to get the user input).

Next use the requests library to make an HTTP request to buy that item.

After the request is complete, print the user a message on the page: *Congratulations, you've just bought {item} for {price}. There are {inventory} left now in the store.*

**Note**: please check your routes are really working!

1. Exercise 6

Create another route in your server.py called /sale

This route should accept an **optional parameter** called admin which takes a boolean

If the route is accessed with the admin param, it should reduce the price of only the items with an inventory greater than 10 by 50%

so that if we try to purchase an item on sale we will get the reduced price.

So once someone makes a request to localhost:3000/sale?admin=true, it should change the price of chair to 60, and of picture frame to 35.

If after that we make another buy request to: http://localhost:3000/buy/chair we should see as a result:

Congratulations, you've just bought a chair for 60...

**Hint**: The query parameter can not be a boolean but string only.

**Note:** You must end the **request** with sending the store back to the client

1. Extension 1

On the client side, define a variable called money.

When the user tries to buy something, **first** make a request to the /priceCheck route to see if the user has enough money.

If the user has enough money, complete the /buy request, and reduce the appropriate amount from money - otherwise, print a message telling the user they should get a job.

1. Extension 2

This one should be fun.

Use the [sched](https://docs.python.org/3/library/sched.html) package to make a request to the /priceCheck/chair route every 3 seconds.

Each time this request is made, you should check whether the price of a chair is lower than the last time you checked.

If it is lower, make a request to the /buy/chair route, and print "bought chair for less"

Otherwise, print "still waiting for a price drop..."

Let the interval run a few times and make sure it's not buying anything.

Then in your browser go to the /sale route - this should drop the price of chair - and within three seconds you should see "bought chair for less" in your console.

ראש הטופס