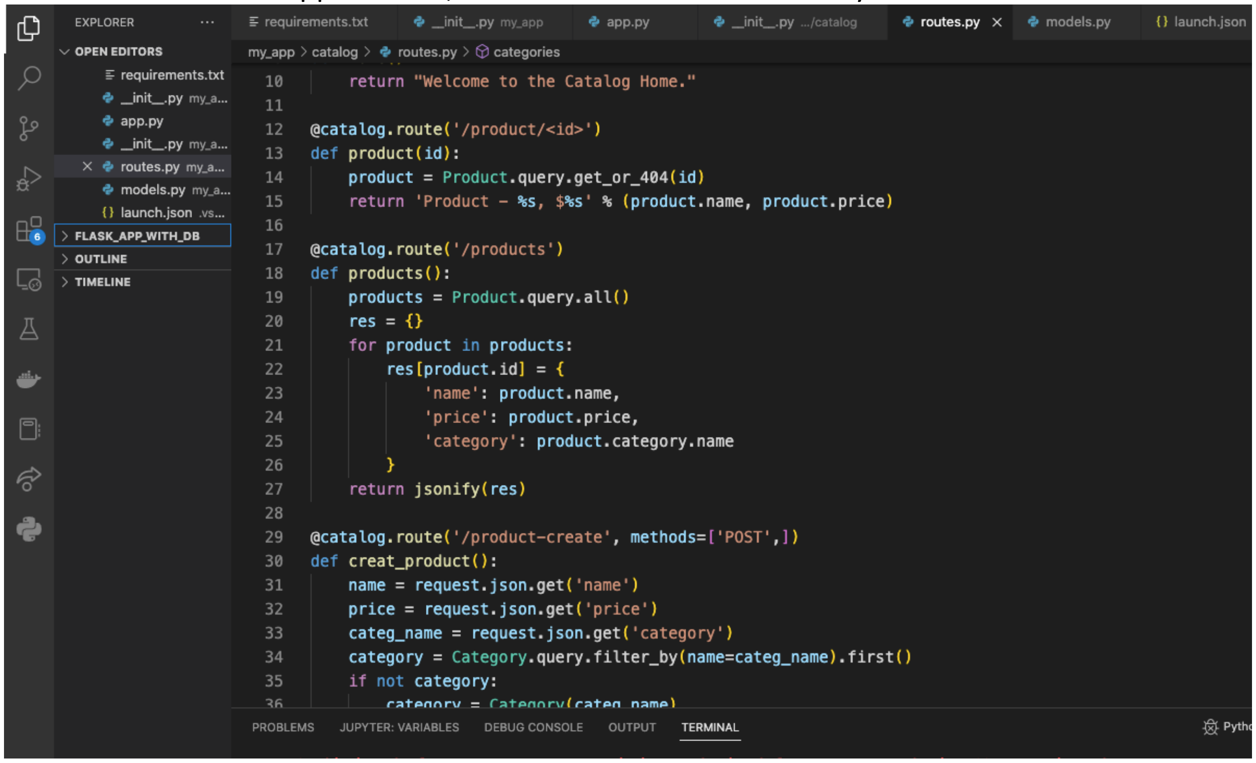
Note: I always make sure the top directory is the ‘default working directory’. By that, I mean, if a folder named ‘flask app with db’, then the vscode work directory tree should look like this:



```python
    return "Welcome to the Catalog Home."

@catalog.route('/product/<id>')
def product(id):
    product = Product.query.get_or_404(id)
    return 'Product - %s, $%s' % (product.name, product.price)

@catalog.route('/products')
def products():
    products = Product.query.all()
    res = {}
    for product in products:
        res[product.id] = {
            'name': product.name,
            'price': product.price,
            'category': product.category.name
        }
    return jsonify(res)

@catalog.route('/product-create', methods=['POST',])
def creat_product():
    name = request.json.get('name')
    price = request.json.get('price')
    categ_name = request.json.get('category')
    category = Category.query.filter_by(name=categ_name).first()
    if not category:
        category = Category(categ_name)
```

See the little blue box. This way it is easier for you to set up working directory.

First of all, you are going to install all the packages INSIDE a virtual environment which is covered the instruction of the class 3.
Once you have created a virtual environment and activated it, type this:
pip3 install -r requirements.txt

Next, you are going to try to run the flask application which will automate the configuration of database and generation of tables (or models/classes). So you want to first delete the database file.

Now, if you type ‘flask run’ in the terminal, you may get an error that says ‘flask application is not found’ or something like that. Again, in the instruction of the class 3, we learned that you need to TELL flask where the app is defined and/or installed. In this case, it is in the ‘main’ file (if you are looking at the first application named ‘all in one’ or in the ‘app’ file (if you are looking at the second application named ‘flask app with db’.
So the way for you to do this is that you enter this in the terminal:
export FLASK_APP= main (or app)

Then you are going to run the flask application by simply entering this in the terminal:

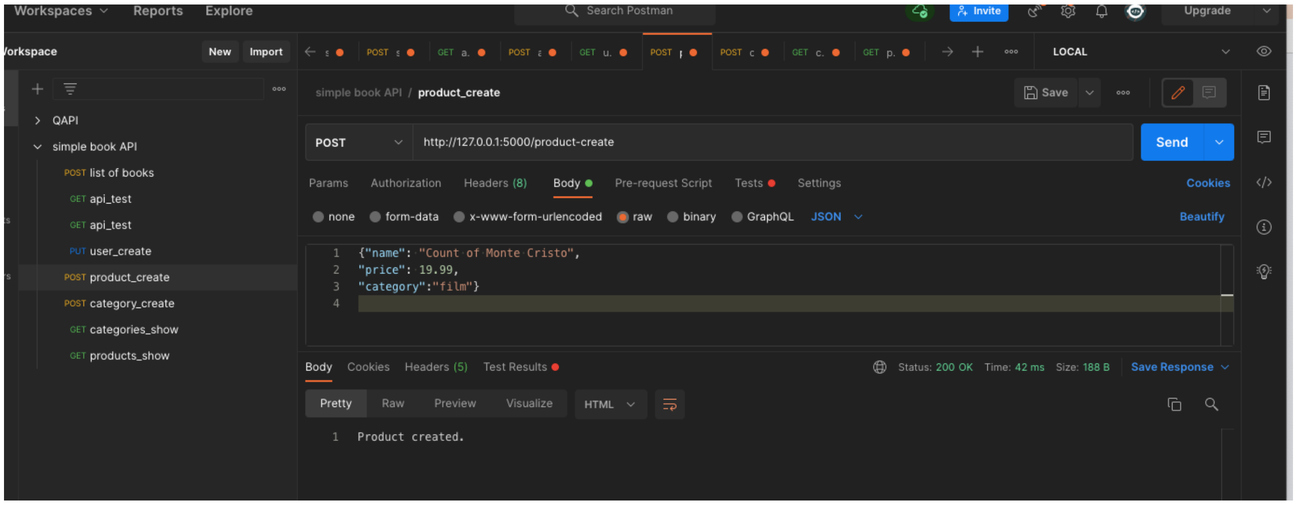
```
flask run
```

You should see a flask application up-running while a database file is being generated. If you take a look at inside that database file using some viewer gadget, you should see tables are all well defined but they are empty.

So you want to add a few objects into those tables. Instead of using SQLAlchemy directly as you have in the past two weeks, you now are going to hit endpoints where all these tasked are handled by those routes. If you take a look those routes functions inside the ‘routes’ file, you will see it better.

Let’s say you are looking at the application named ‘flask app with db’. You want to create a product or products, and you want to create a category or categories. Notice that category and product have ‘one to many’ relationship. So ideally, if you create a product, a category that is associated with it will be created accordingly. Thankfully, SQLAlchemy handles for you. How that exactly happens---you are going to have to look at where it happens. It is in the definitions of those models/classes/tables in the ‘models’ file.

How to create a product?

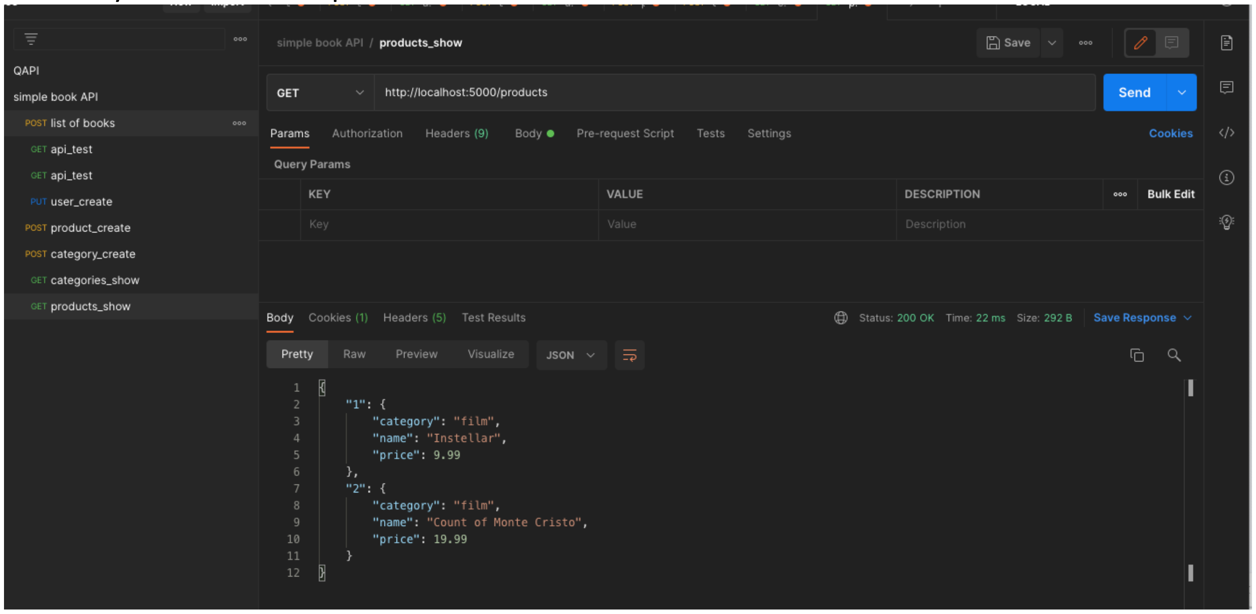


simple book API / product_create
POST
http://127.0.0.1:5000/product-create
Send
Body
raw
JSON
Status: 200 OK Time: 42 ms Size: 188 B

```
{"name": "Count of Monte Cristo",
"price": 19.99,
"category":"film"}
```

```
Product created.
```

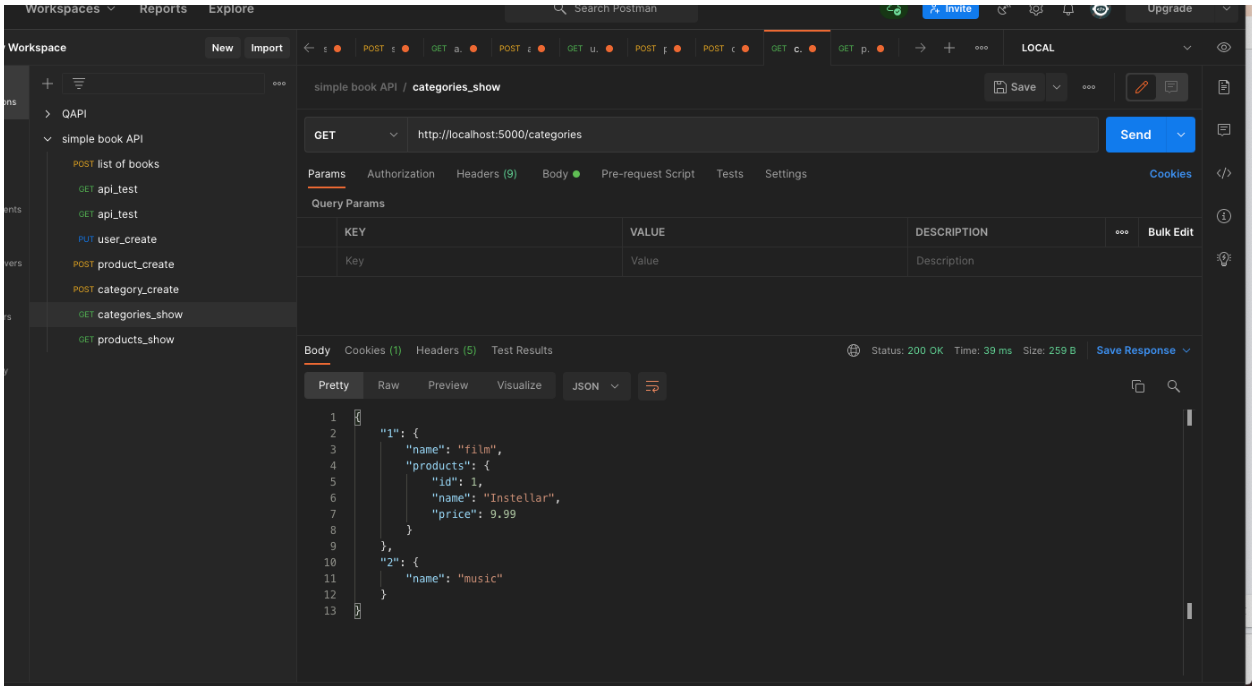
How do you show all the products?



simple book API / products_show
GET
http://localhost:5000/products
Send
Status: 200 OK Time: 22 ms Size: 292 B
JSON

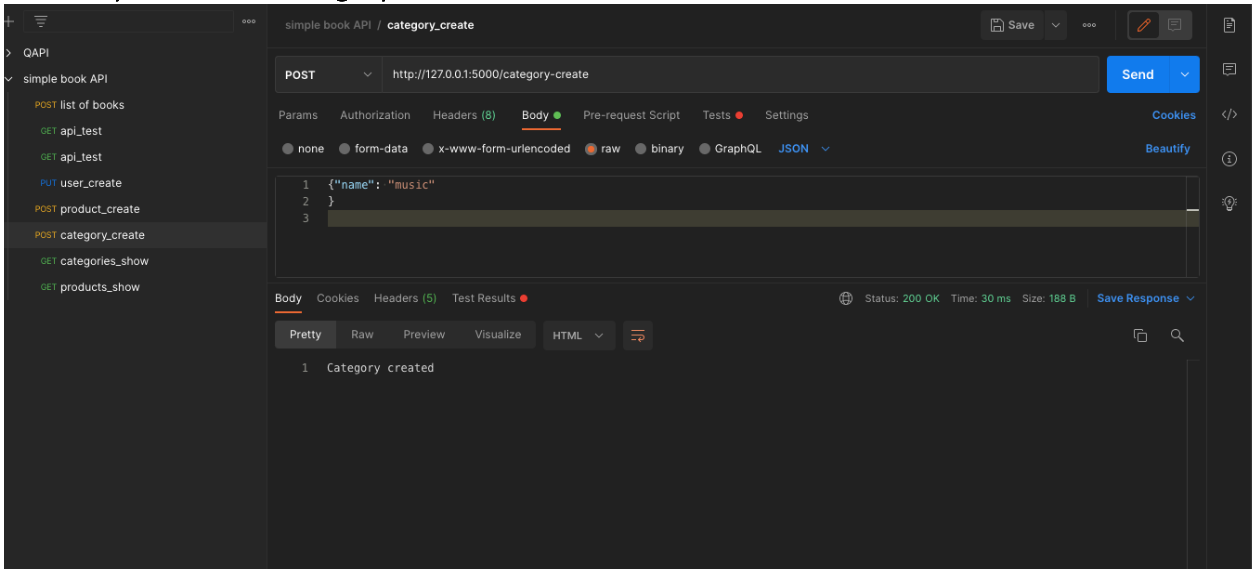
```
{
    "1": {
        "category": "film",
        "name": "Instellar",
        "price": 9.99
    },
    "2": {
        "category": "film",
        "name": "Count of Monte Cristo",
        "price": 19.99
    }
}
```

Now that you have created a product or a few, then by design, there should be some ‘category’ that have been created automatically.
How do you show all the categories?


simple book API / categories_show
GET
http://localhost:5000/categories
Send
Status: 200 OK Time: 39 ms Size: 259 B

```
{
    "1": {
        "name": "film",
        "products": {
            "id": 1,
            "name": "Instellar",
            "price": 9.99
        }
    },
    "2": {
        "name": "music"
    }
}
```

How do you create a category?


simple book API / category_create
POST
http://127.0.0.1:5000/category-create
Send
raw
JSON
Status: 200 OK Time: 30 ms Size: 188 B

```
{"name": "music"
}
```

```
Category created
```

This is how you create a flask application that talks to database on behalf of you so you don't have to (most of the time, aren't allowed to) interact with database directly. This is the whole point of API, application programming interface. A software or an application is created to act as an intermediary between software, servers, and/or database.

Why does all these ‘request’ as shown in the postman snapshot work the way they do?

Look at where all the ‘business logic’ is defined. It is defined in those functions in the ‘routes’ file.