

Building a Todo App with Flask in Python

Source: <https://stackabuse.com/building-a-todo-app-with-flask-in-python/>

To install CURL -> Open Powershell in administrator mode.
choco install curl -y

First thing to do is to create a virtual environment

```
py -m venv todo_flask
```

Once created, activate the virtual environment

```
source todo_flask/Scripts/activate
```

Once inside the virtual environment we will install flask,

```
pip install Flask
```

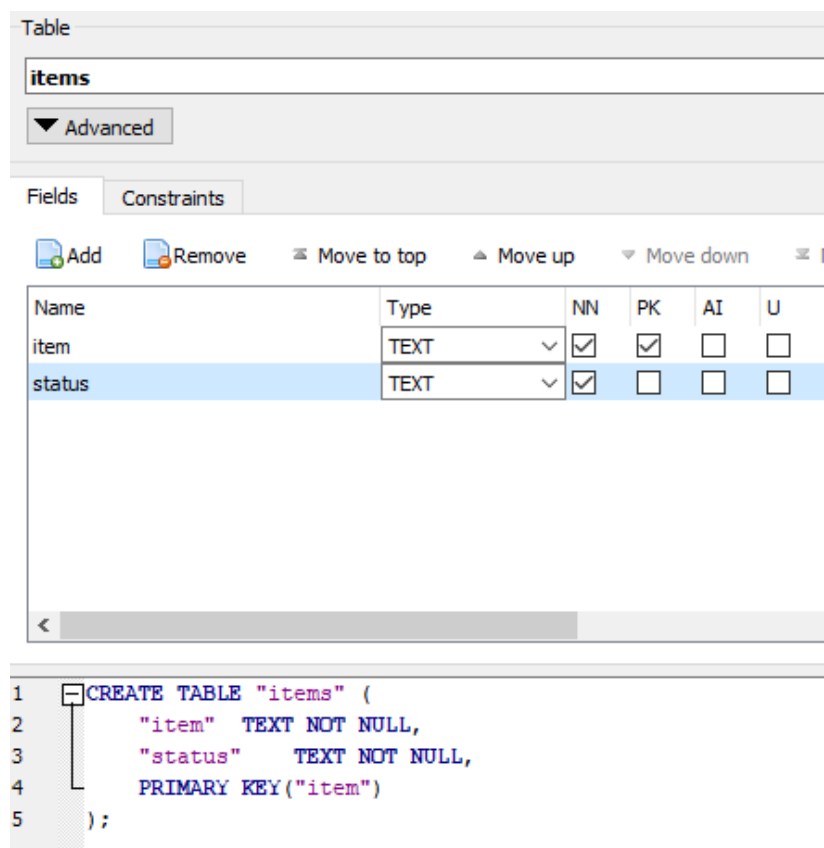
Change directory into todo_flask environment

Create an app.py file and helper.py file and attach commands to each.

Install a Database

Download "DB Browser for SQLite" -> [Here](#)

Open it and create a database table with the following



The screenshot shows the DB Browser for SQLite interface. At the top, the table name 'items' is entered. Below it, the 'Advanced' tab is selected. The 'Fields' tab is active, displaying a table with columns: Name, Type, NN, PK, AI, and U. The 'item' field is of type TEXT, not null, and is the primary key. The 'status' field is also of type TEXT, not null. Below the table, the SQL statement to create the table is shown:

```
1 CREATE TABLE "items" (  
2     "item" TEXT NOT NULL,  
3     "status" TEXT NOT NULL,  
4     PRIMARY KEY("item")  
5 );
```

Start the Flask application in the current terminal

FLASK_APP=app.py flask run

Open the link and check if it works.

Open another Bash and run the following commands

First command

```
curl -X POST http://127.0.0.1:5000/item/new -d '{"item": "Setting up Flask"}' -H 'Content-Type: application/json'
```

>Expected outcome: {"Setting up Flask": "Not Started"}

Second commands

```
curl -X POST http://127.0.0.1:5000/item/new -d '{"item": "Implement POST endpoint"}' -H 'Content-Type: application/json'
```

>Expected outcome: {"Implement POST endpoint": "Not Started"}

Third Command

```
curl -X GET http://127.0.0.1:5000/item/all
```

>Expected outcome: json {"count": 2, "items": [{"Setting up Flask", "Not Started"}, {"Implement POST endpoint", "Not Started"}]}

Fourth Command

```
curl -X GET http://127.0.0.1:5000/item/status?name=Setting+up+Flask
```

>Expected outcome: {"status": "Not Started"}

Fifth Command

```
curl -X PUT http://127.0.0.1:5000/item/update -d '{"item": "Setting up Flask", "status": "Completed"}' -H 'Content-Type: application/json'
```

>Expected outcome: {"Setting up Flask": "Completed"}

Sixth Command

```
curl -X DELETE http://127.0.0.1:5000/item/remove -d '{"item": "Setting up Flask"}' -H 'Content-Type: application/json'
```

>Expected outcome: {"item": "Temporary item to be deleted"}

Thoughts on this Assignment

For this application, I have made a documentation file that includes how to complete this assignment. While doing this assignment, I faced a lot of issues with the documentation. The person who made the article from Stack Abuse had so many issues with simple typos. He would leave out some routes from the CURL request which would make the app fail. I had to spend a good amount of time debugging the issues. I tried to make the database by just creating a random file but I had to create an actual database using DB Browser for SQLite. Curl and Windows are also a weird combination. I had many issues but figured it out. This program basically has different ways of adding, receiving, updating, and deleting items from a database. Basically it follows CRUD. Create, read, update, and delete.

Added feature

My Feature

There are not many features that I can think of adding so I thought since we have a delete one row, it would be great to have a delete all rows function. In the app.py I created a new function called delete_all_items() In this function delete_all_items() function is called from helper.py. The result is stored in res_data. The next line dumps all the items from the list on, I reused some code from the get_all_items function.

(Make sure you have data in the database before running this command)

Seventh Command

```
curl -X DELETE http://127.0.0.1:5000/item/removeall
```

Future

I plan to improve this application by adding new methods to sort the database whenever I get a chance to.