Các khái niệm cơ bản :

Front-end : là phần được tải từ bên trong trình duyệt web trên máy tính => là phần mà người dùng thấy và tương tác

Back-end : chạy trên server và chịu trách nhiệm xử lí dữ liệu

URL là một cách để định vị tài nguyên trên Internet ( tài nguyên như là pdf, web page, image,…)

Khi nhập URL và nhấn enter thì thời điểm này trình duyệt gửi yêu cầu đến máy chủ web lưu trang web và máy chủ sẽ xử lí yêu cầu sau đó gửi lại phản hồi cho máy khách hàng. Việc trao đổi dữ liệu này được xác định bởi giao thức http => xác định cách máy chủ và client có thể giao tiếp với nhau

Cách để máy chủ phản hồi lại client :

1. Tạo yêu cầu trên máy chủ và trả lại cho khách => dùng HTML (là ngôn ngữ đơn giản để biểu diễn các trang web và nội dung của nó

Như vậy là server sẽ trả lại tài liệu html cho client

1. Cách 2 là chỉ chi trả data cần thiết và để client tự tạo html => nó sẽ giải phóng server và giúp server hoạt động vs nhiều client hơn

Nếu mà chúng ta đưa việc tạo trang web cho máy khách thì máy chủ sẽ trở thành 1 cổng data. Trên server ta sẽ cung cấp các điểm cuối mà client có thể nói chuyện để lấy hoặc lưu các phần dữ liệu khác nhau



Các điểm cuối này đại diện cho giao diện mà người dùng giao tiếp với máy chủ

Máy chủ cung cấp API ( tưởng tượng giống mấy cái nút trên remote tivi, là để tương tác với tivi )

Như vậy ta dùng Django để xây dụng 1 API cho cửa hàng trực tuyến của chúng tôi, client có thể sử dụng API để lấy hoặc lưu data

**File \_\_init\_\_.py**



Xác định thư mục này là một gói

**File settings.py**



Nơi xác định cài đặt ứng dụng của mình

**File urls.py**



Nơi xác định các url của ứng dụng

**File manage.py**



Là 1 trình bao bọc xung quanh quản trị viên django => thay vì quản trị django thì ta sẽ sử dụng file này

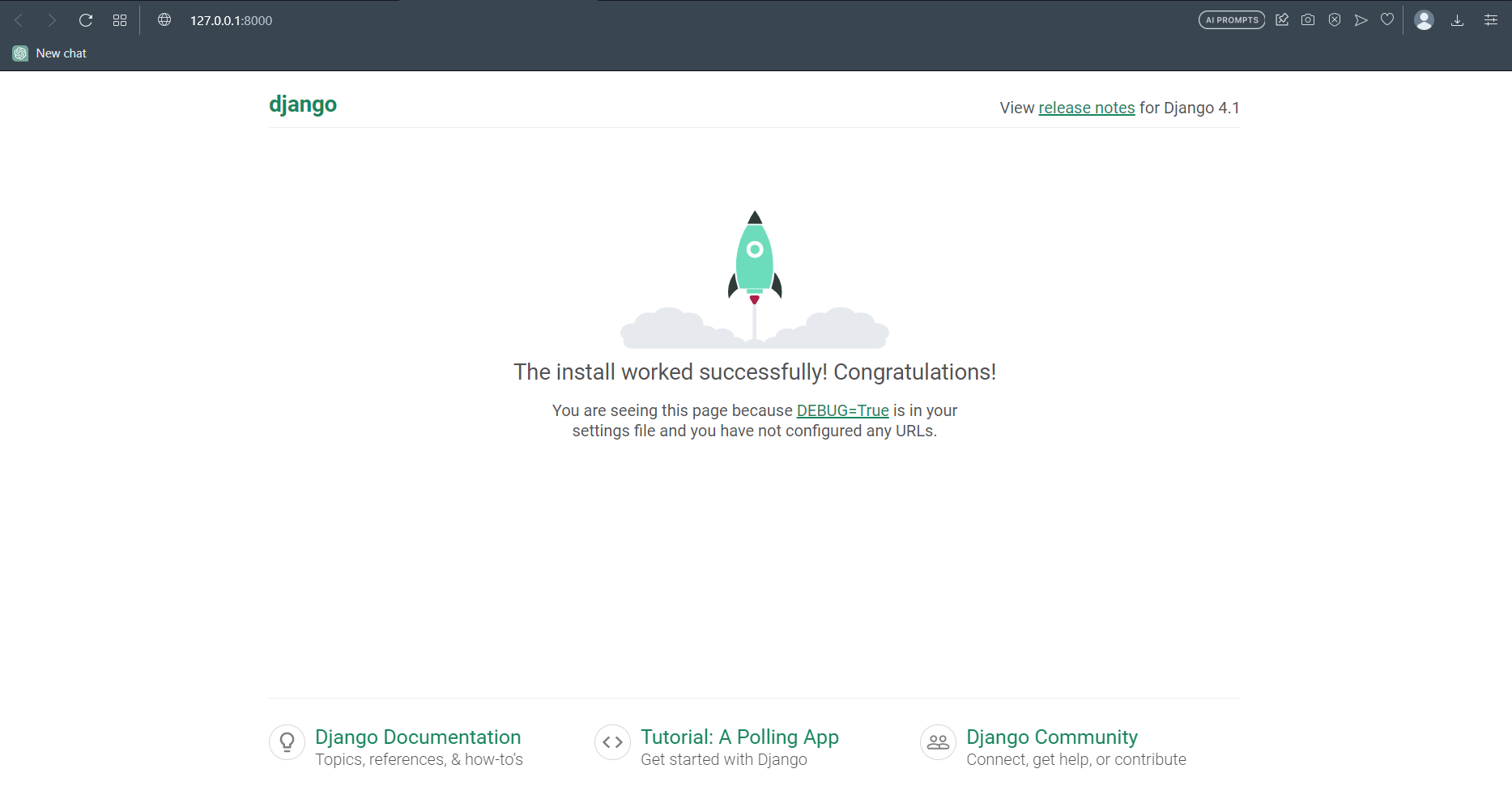
 chạy câu lệnh này trong cmd thì sẽ mặc định là dùng port 8000, tuy nhiên có thể điều chỉnh port này bằng cách thêm số ở sau câu lệnh trên

Vd : 

Khi chạy câu lệnh trên xong để mở một server, ta sẽ có đường dẫn như sau :



Chạy đường dẫn kia trên trình duyệt

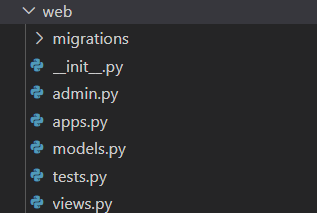


**Cách để chạy 1 app :**

**python manage.py startapp** <tên app>

Ví dụ : 

Sẽ xuất hiện :



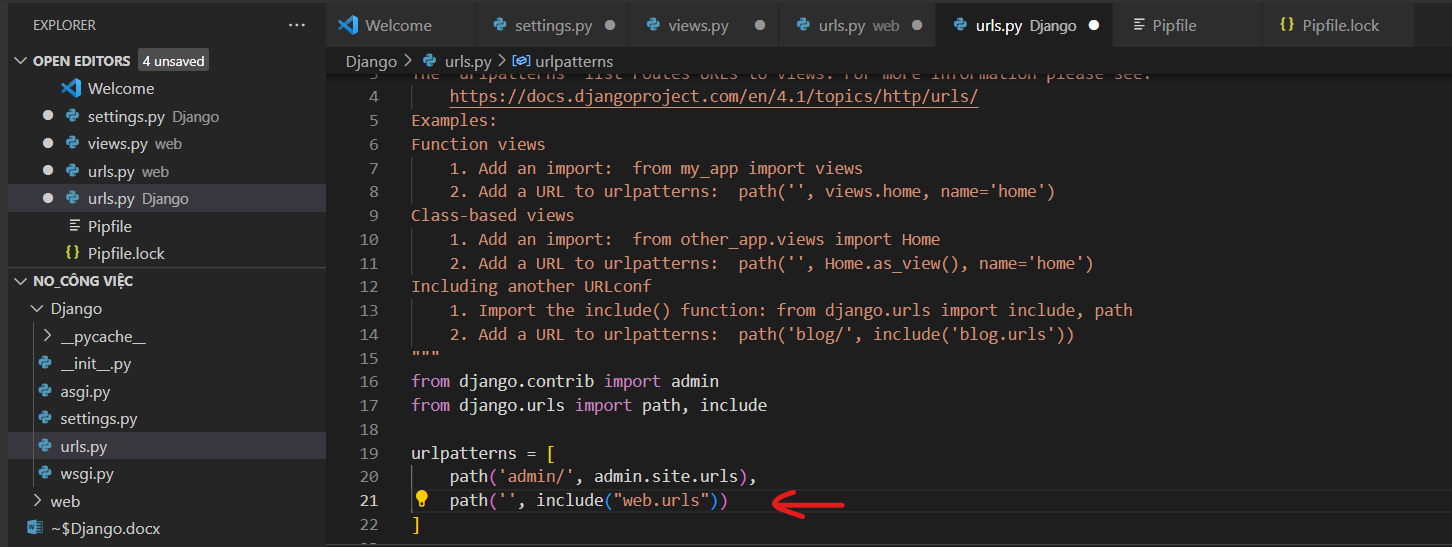
**Admin.py** là để định nghĩa giao diện của app

**app.py** là nơi để cấu hình app

**model.py** là nơi để các lớp model của app này, ta sẽ sử dụng các lớp model này để lấy dữ liệu từ database và gửi cho người dùng

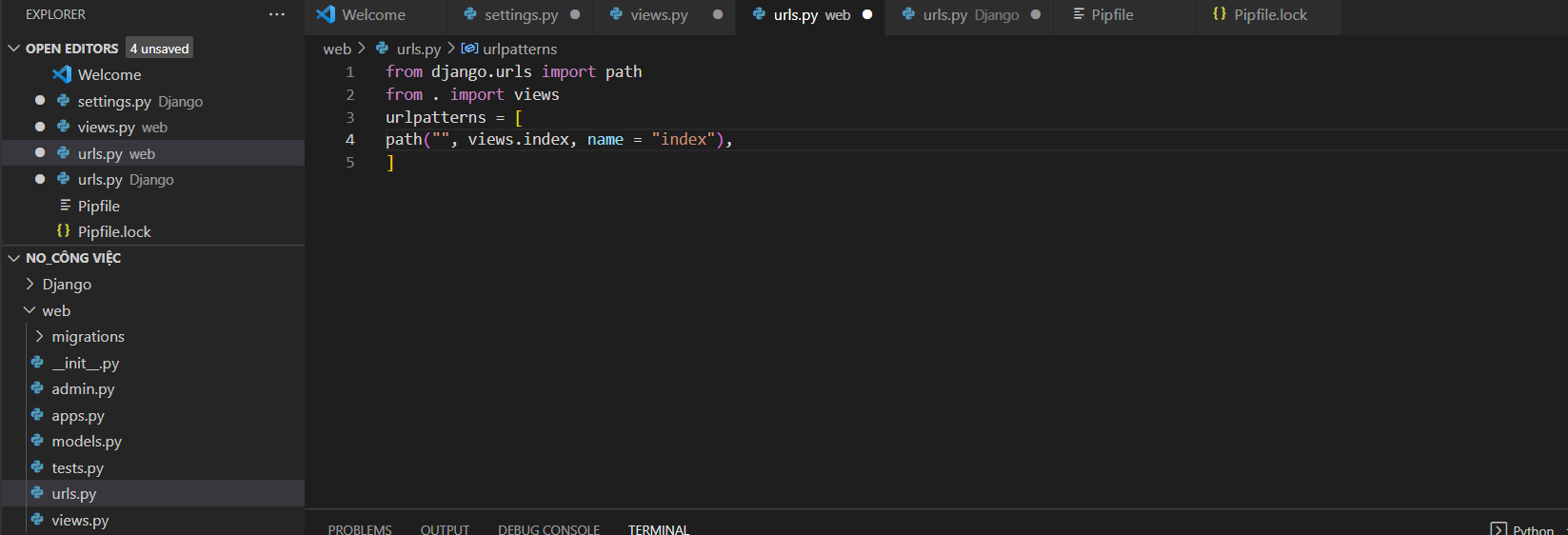
**tests.py** là nơi để viết các bài kiểm tra

**Views.py** là



Nếu để trống trong cái phần path thì mặc định sẽ được vào trang “web.urls”

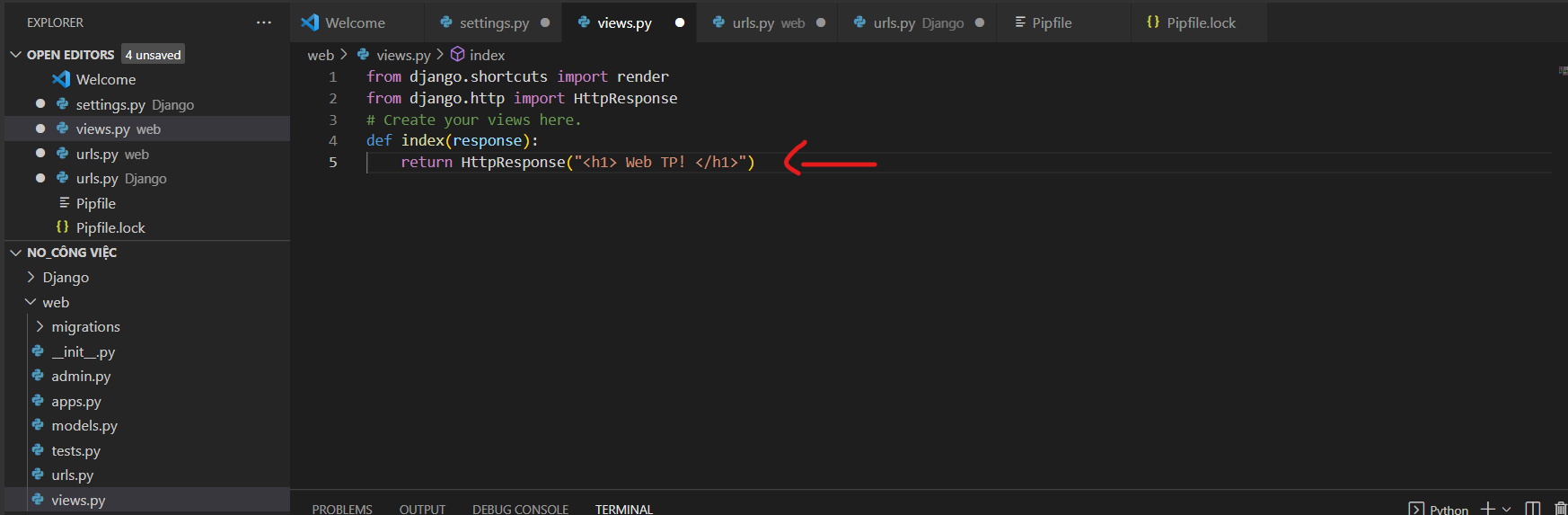
trong trang web.urls sẽ là :



Ta thấy cấu trúc   
**path(“”,view.index, name = “index”)**

Nghĩa là nó sẽ vào phần view, hàm con index để chạy

Dấu “” ở đầu sẽ là đường dẫn nào ta sẽ được nhận



Sẽ chạy cái hàm con index

Ví dụ ta muốn tạo một đường dẫn … /home/start là vào cái hàm con index đó

Bước 1, ở chỗ include ta thêm vào “home/”



Nghĩa là tất cả đường dẫn mà có dạng …./home/ thì nó sẽ vào file **urls** nằm trong folder **web** để kiếm

Thì sau đó vào file urls kiếm và tìm thấy 1 đường dẫn là start/ (kết hợp lại ta có …/home/start/)

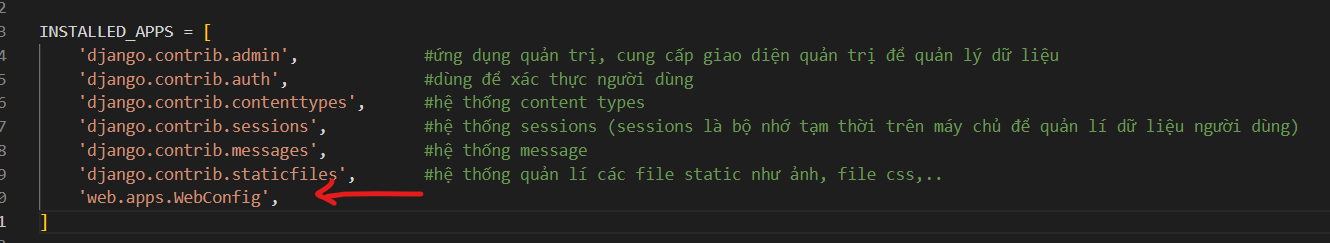


Sau khi đã tìm thấy chính xác đường dẫn thì nó sẽ bắt đầu vào file **view**, tìm hàm con **index** và thực hiện response

**Database và set up database truy xuất các mục tiêu**



Câu lệnh này được dùng để thay đổi cơ sở dữ liệu



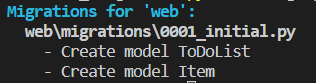
Khi khai báo thêm app, thì cần vào file **apps.py** để xem xem đang khai báo trong biến nào (vd ở đây là WebConfig) để điền vào **INSTALLED\_APP** cho đúng



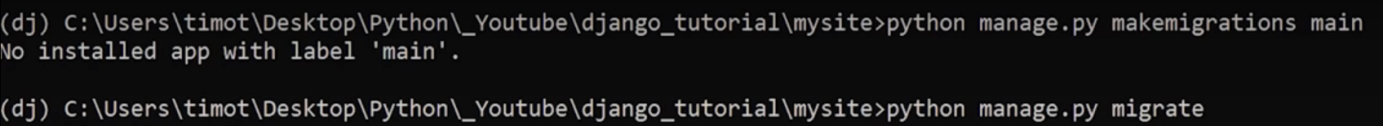
Sau đó vào file **models.py** để set up các class và đặc tính mình cần, ấn run file python và sau đó vào terminal nhập câu lệnh



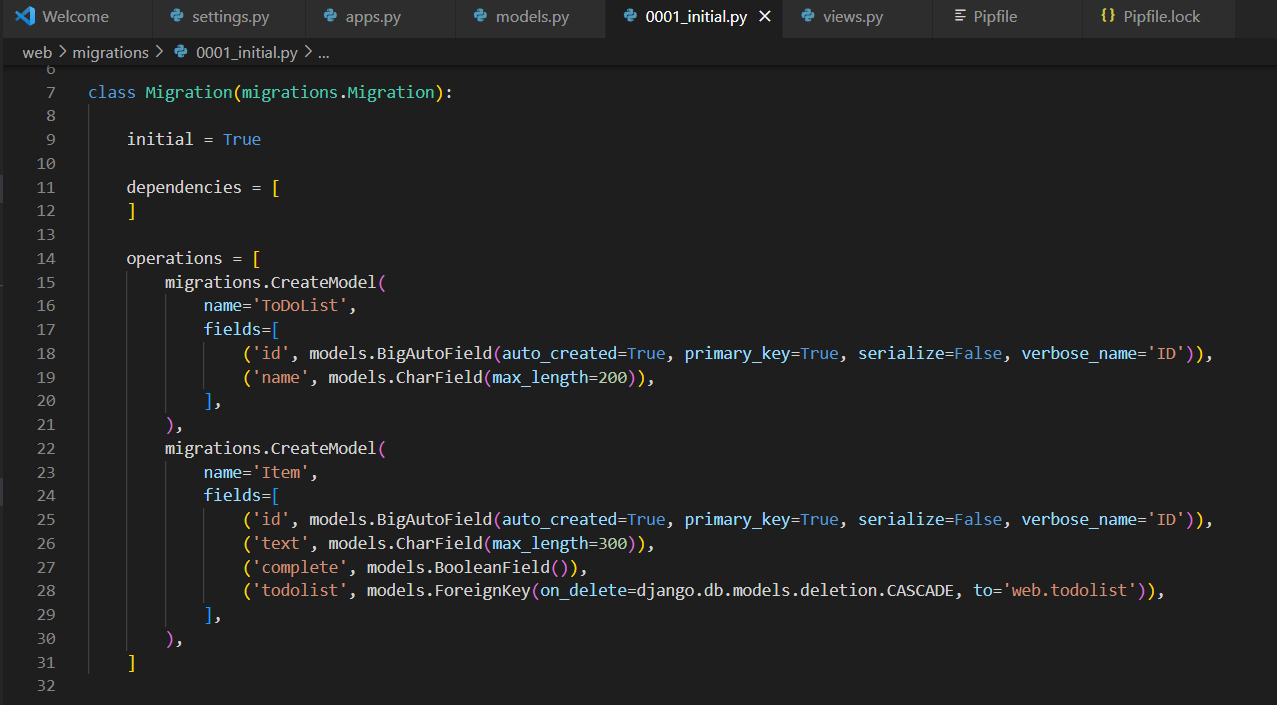
**Kết quả :**



Chạy cả 2 câu mới được



File **0001\_initial.py**





Câu lệnh này cho phép add thêm một số thứ vào database

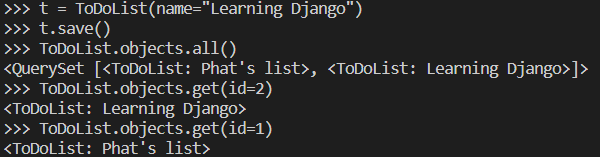
Kết quả :



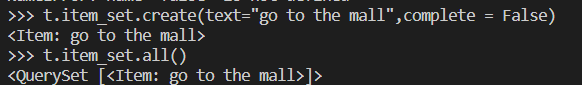
Sau đó ta tiến hành **from web.models import Item, ToDoList** để có thể tiến hành lưu các database

Ta muốn thêm dữ liệu vào database :





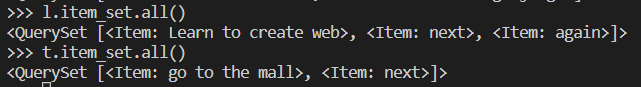
Tiếp đó tới class Item, do ta đã có liên kết từ class Item tới class ToDoList , cho nên cách tạo data sẽ khác đi một chút, sẽ thêm …\_set



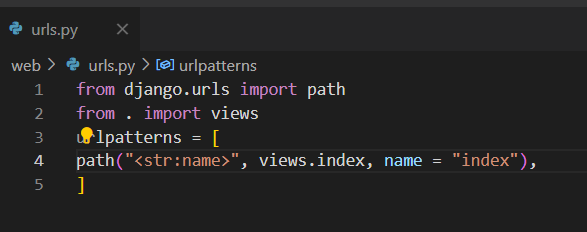
**Giả sử ToDoList có 2 object là “Phat’s list” và “Learning Django”**

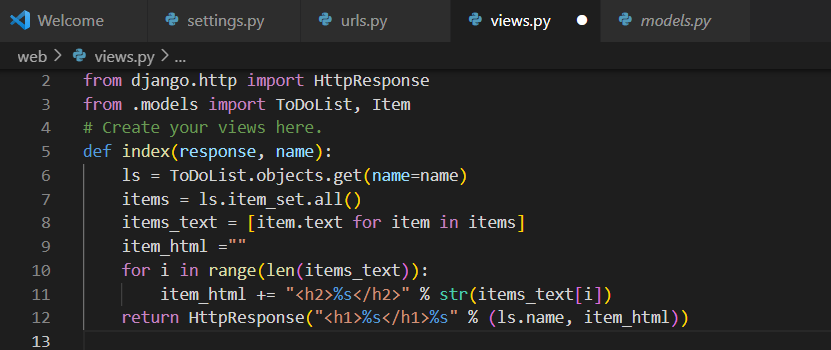


Mỗi object của ToDoList lại có các cái Item riêng



Cách để code cho viewer thấy các item của từng object



Path(”<str:name>”,…) là sẽ tạo ra đường dẫn có dạng …/<tên đường dẫn mình nhập>  


ls = ToDoList.objects.get(name=**name**) #cái biến **name** ở đây đã được lưu từ cái <str:name> ở hàm urls.py rồi

Và sẽ lấy cái tên đó để truy cập tới cái object ToDoList có tên tương ứng

Items = ls.item\_set\_all() là sẽ lấy toàn bộ các item trong cái object trên

Items\_text = … là lưu toàn bộ các item ở trên vào biến items\_text

Gọi biến item\_html để lưu các item theo từng dòng bằng hàm for

Return … xuất cái ls.name là tiêu đề <h1> và item\_html là các dòng nhỏ hơn <h2>

**Admin Dashboard : bảng quản trị**

Liên quan tới database và có thể xem tất cả các bảng khác nhau và có được list các danh mục => được gọi là truy vấn

Ta có lệnh <tên của danh sách>.**objects.filter(…)**

Thì filter có chức năng là tìm tất cả các object phù hợp với điều kiện nằm trong filter(..)

Vd :

Ta có 1 ToDoList với 2 object



Ta sử dụng câu lệnh

**ToDoList.objects.filter(name\_\_startswith=”Phat”)**



Kết quả chỉ trả về 1 đối tượng phù hợp với điều kiện đó

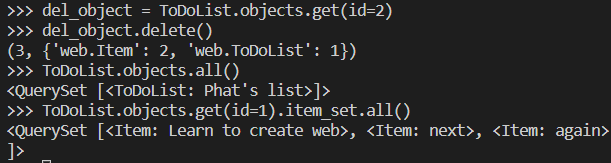
Còn nếu không có objects nào thỏa điều kiện trong filter, kết quả sẽ trả về rỗng



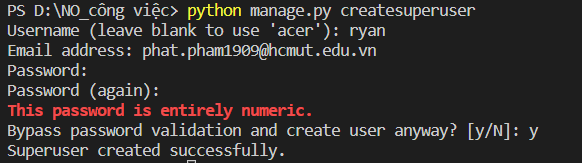
**Hàm để xóa**

**Cấu trúc :** <tên của danh sách>.objects.get(id=…).delete()

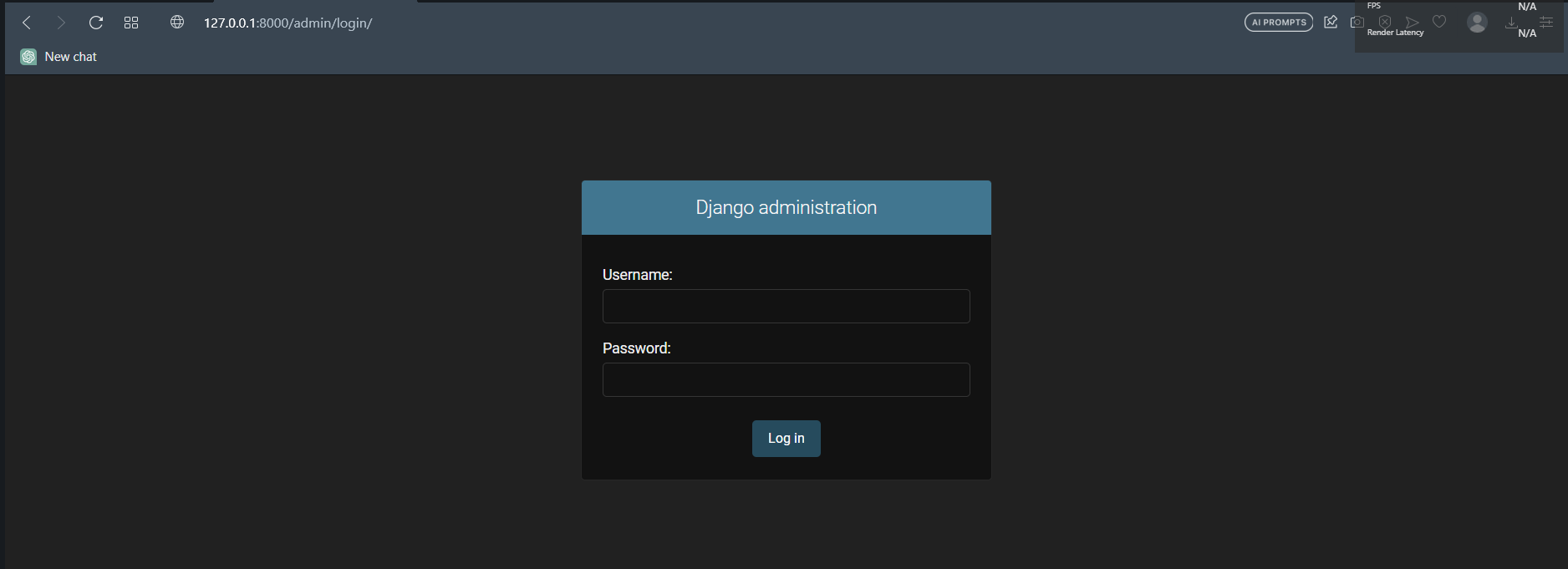
Vd: xóa đi objects có id = 2, chỉ còn lại id = 1 là Phat’s list và các item của n



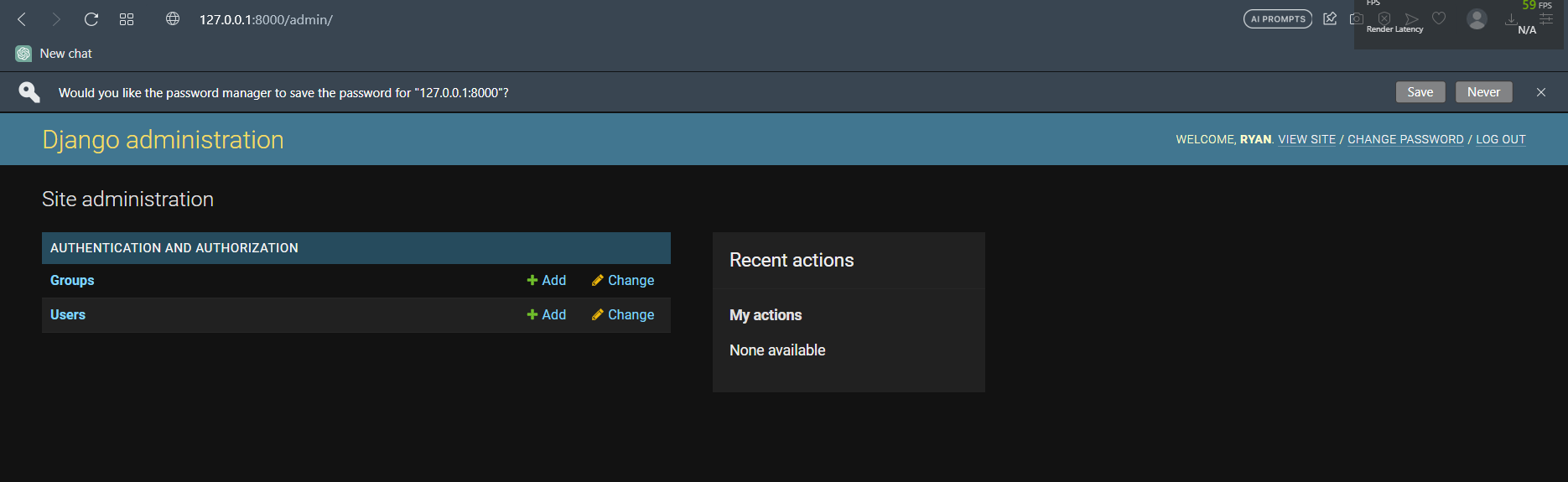
**Tạo 1 tài khoảng admin trên Django**



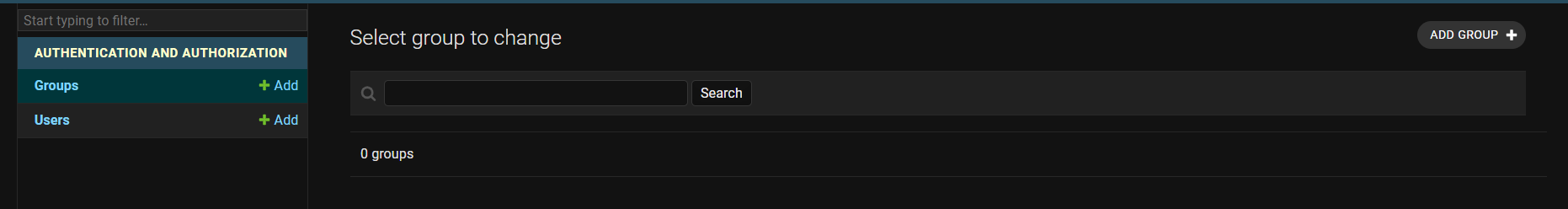
Sau đó cho chạy **runserver** và truy cập đúng đường dẫn …/admin



Nhập tk và mk vừa tạo vào sẽ ra kết quả sau



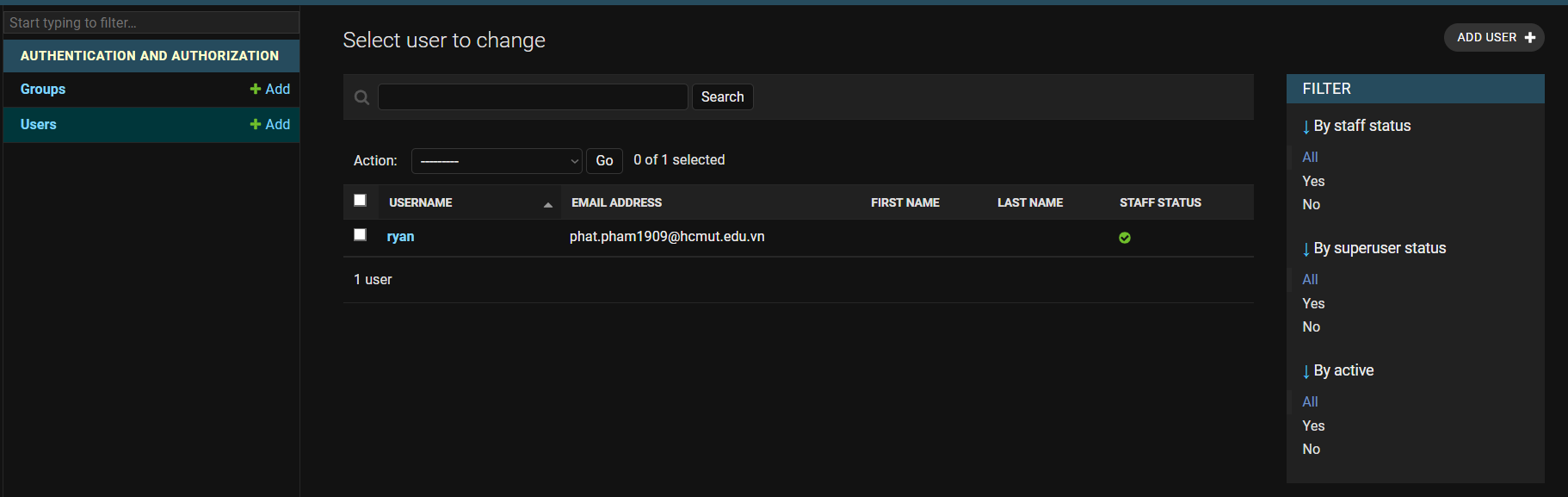
**Group :**



Là nơi tập hợp các tk người dùng được tổ chức lại thành 1 nhóm để quản lý và cấp quyền truy cập

**Users:**

Nơi lưu thông tin tài khoản

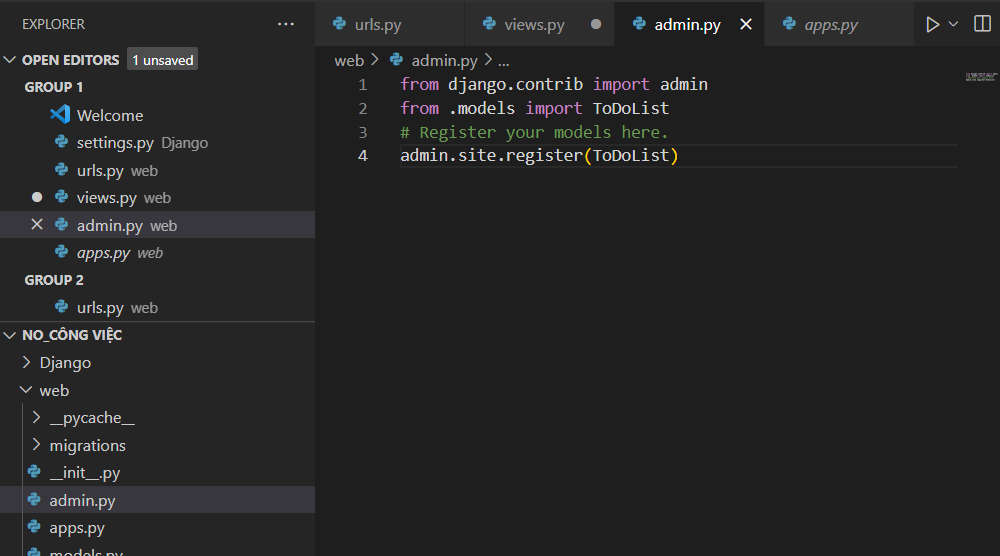


**Để cung cấp bảng database cho trang admin, ta sẽ làm như sau :**

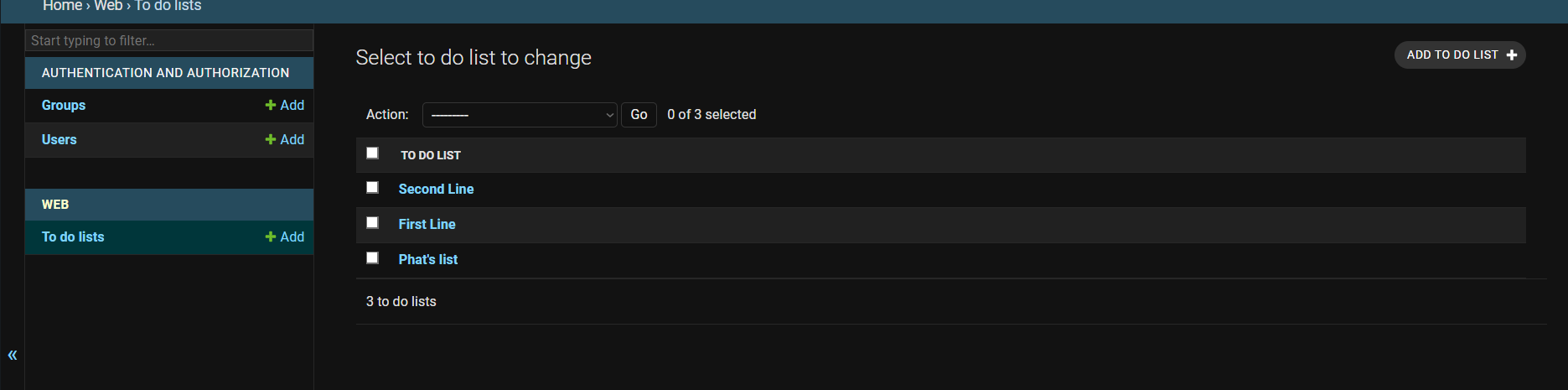
Vào file admin.py trong cái app mình tạo, từ models ta import cái model trong ứng dụng

admin.site.register(ToDoList)

Câu lệnh trên giúp chúng ta khai báo model trong ứng dụng của mình với trang quản trị (admin site) => cho phép xem, xóa, sửa các item trong model đó

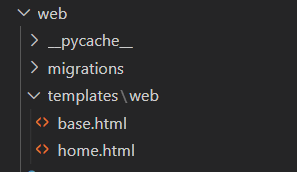


**Kết quả sau khi khia báo với trang admin**



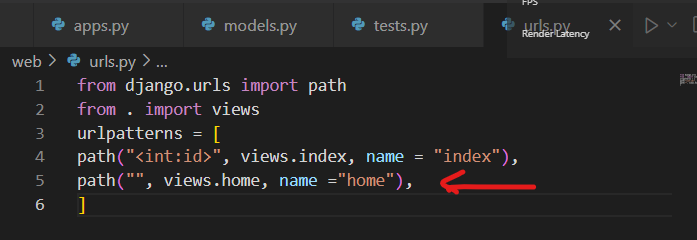
**Templates và custom HTML :**

Đầu tiên tạo thêm 1 folder **templates** trong folder **web** , sau đó trong folder **templates** tạo tiếp folder **web,** trong folder này tạo 2 file .html

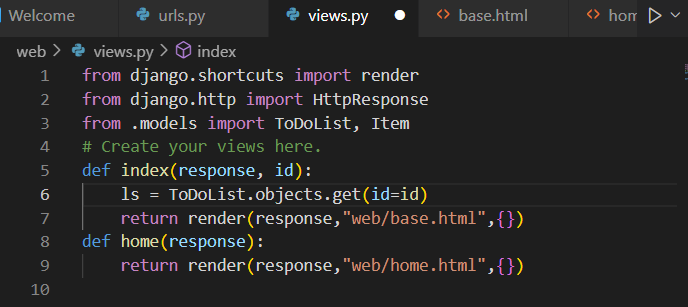


**Yêu cầu là khi bắt đầu truy cập đường dẫn thì sẽ hiện templates của trang home, còn khi thêm …/1 vào đường dẫn thì ra trang base**

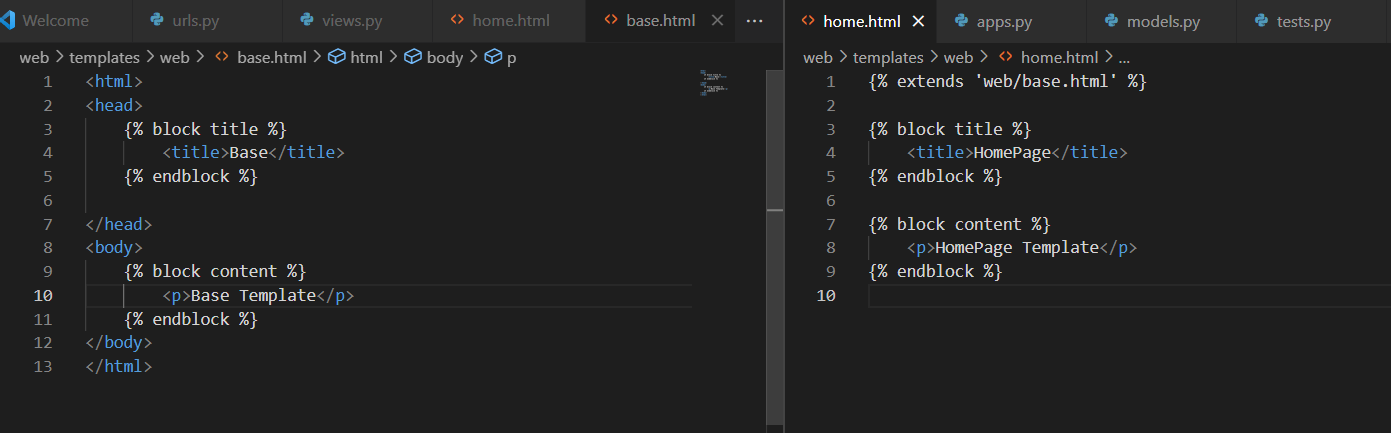
Bước 1 : vào urls.py để chỉnh sửa quyền mặc định khi truy cập đường dẫn mới vào sẽ tới hàm “home” trong file **views**



Bước 2 : tạo hàm con **home** và chỉnh sửa không phải hồi từ Http nữa mà phản hồi từ file .html



Bước 3 : tạo nội dung cho 2 file base.html và home.html



Base.html sẽ là file mẹ, còn home.html sẽ là file con kế thừa file mẹ dựa vào câu lệnh dưới

{% extends 'web/base.html' %}

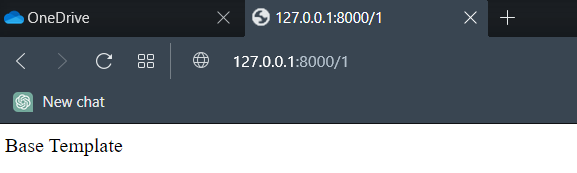
(kế thừa về cấu trúc)

{% block…% } và {%endblock%} là cách để ta ghi đè nội dung từ file con lên file mẹ, nếu file con không ghi đè nội dung lên file mẹ thì mục tương ứng đó sẽ hiện của file mẹ

Vd : file con **<title> </title>** không ghi gì cả thì sẽ hiện cái nội dung trong <title> Nội dung </title> trong file mẹ

**<head>…</head>** là tên trang web ( nhìn xuống hình dưới có Base và HomePage)

Nếu trong <head></head> để trống **<title></title>** thì kết quả tên trang web sẽ là đường dẫn

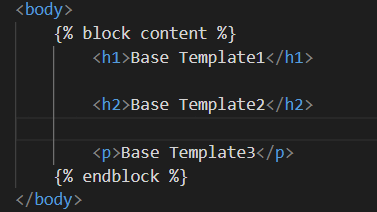


**<body>…</body>** là thông báo phần nội dung trang web

**<p></p>** là paragraph, là nội dung, nội dung trong <p></p> sẽ là chữ thường và nhỏ

Còn có **<h1>…</h1>, <h2>…</h2>**,.. là các header1, header2 ,.. thì chữ sẽ in đậm, có độ lớn giảm dần

Ví dụ :



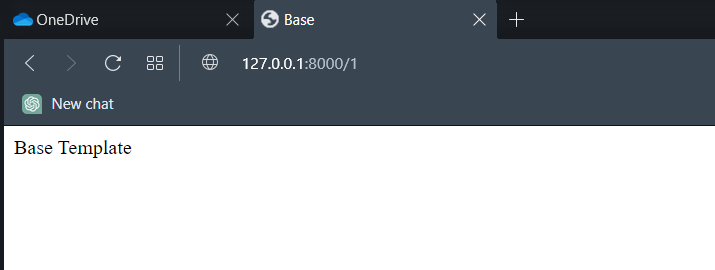
Do cái **ToDoList** của mình chỉ có 3 objects có id lần lượt là 1,3,4, cho nên khi chạy đường dẫn

**<http://127.0.0.1:8000/1>**

**[http://127.0.0.1:8000/3](http://127.0.0.1:8000/1)**

**<http://127.0.0.1:8000/4>**

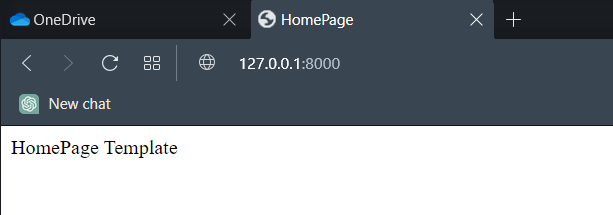
Thì đều cho ra 1 kết quả đó là return lại trang base.html



Còn nếu dùng đường dẫn mặc định là

**<http://127.0.0.1:8000>**

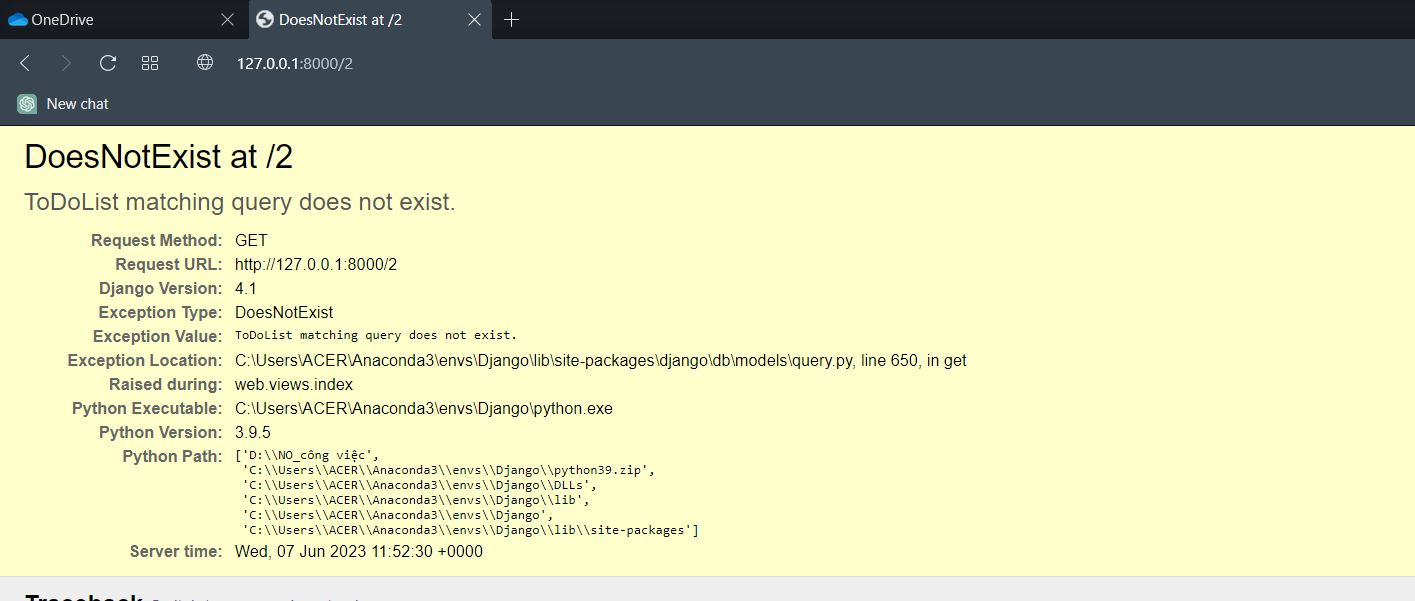
Thì cho ra trang web return lại trang HomePage.html



Còn nếu ta dùng đường dẫn

**<http://127.0.0.1:8000/2>**

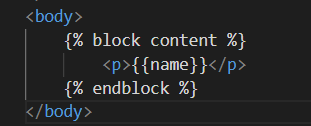
Thì sẽ báo lỗi do cái ToDoList của ta không có object ở id = 2



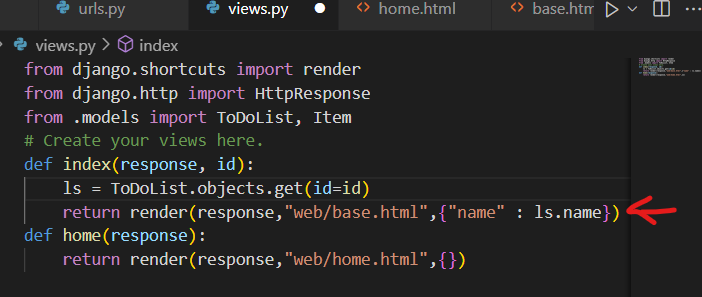
**Yêu cầu : khi nhập đường dẫn tương ứng thì nội dung của trang chính là Item của objects trong ToDoList**

Ví dụ **<http://127.0.0.1:8000/1>** thì nội dung là Phat’s List

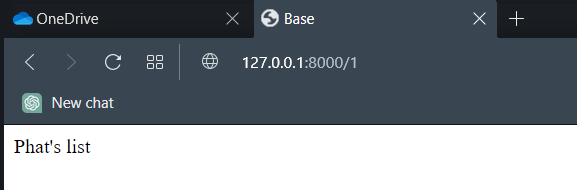
B1 : sửa file base.html



Thì cái nội dung của base.html sẽ được truyền từ biến name vào, mà biến name được gán = ls.name



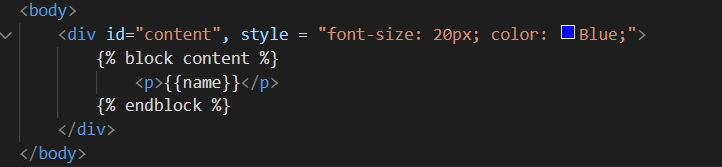
Kết quả :



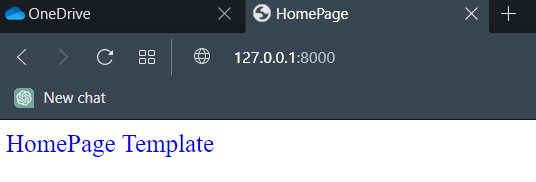
Thẻ **<div></div>** dùng để nhóm các khối lại với nhau trong các khu vực của trang web.

Việc nhóm các cấu trúc lại bằng thẻ **<div></div>** sẽ giúp cho quá trình định dạng, điều khiển,… nguyên cụm dễ dàng hơn rất nhiều. Người dùng chỉ cần thêm hoặc bớt đặc tính bằng một dòng có gắn thẻ div là khối nội dung đó sẽ thay đổi như yêu cầu.

Ví dụ :



Kết quả là các chữ trong content đều màu xanh và có kích thước 20px

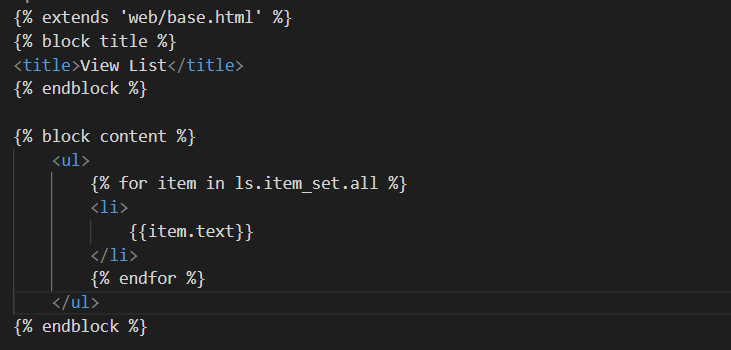


**In ra web các item mà objects đang sở hữu :**

Ví dụ object **Phat’s list** có 3 Item là 

Ta muốn hiện bản danh sách này trên web (không đánh thứ tự)

B1 : tạo file **list.html** và kế thừa file **base.html**



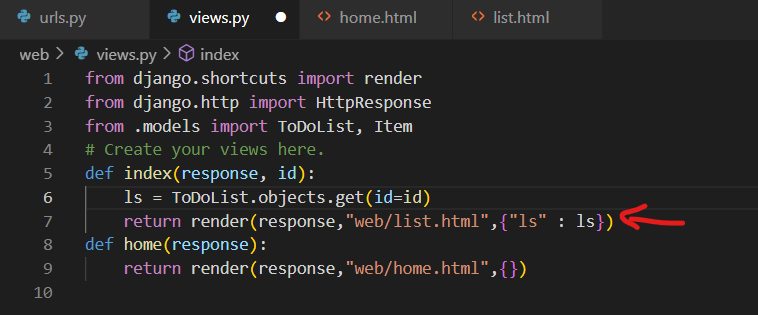
Đặt lại tên **title** là **View List**

Ở phần nội dung

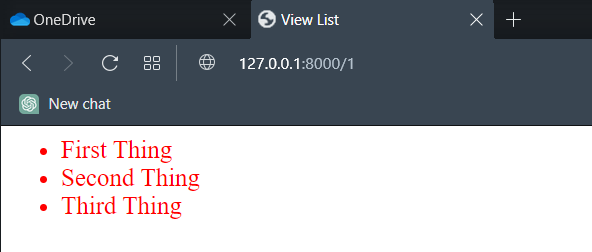
thẻ **<ul></ul>** là thẻ để làm danh sách không thứ tự

Thẻ **<li></li>** là thẻ hiển thị mục của danh sách

ls trong ls.item\_set.all sẽ lấy từ ls = ToDoList.objects.get(id=id)

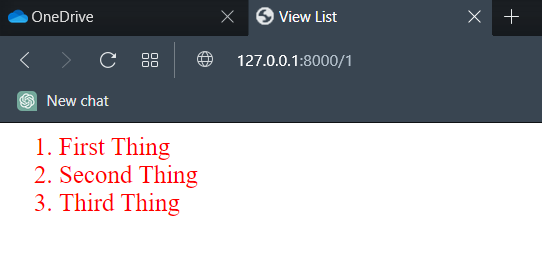


Kết quả :



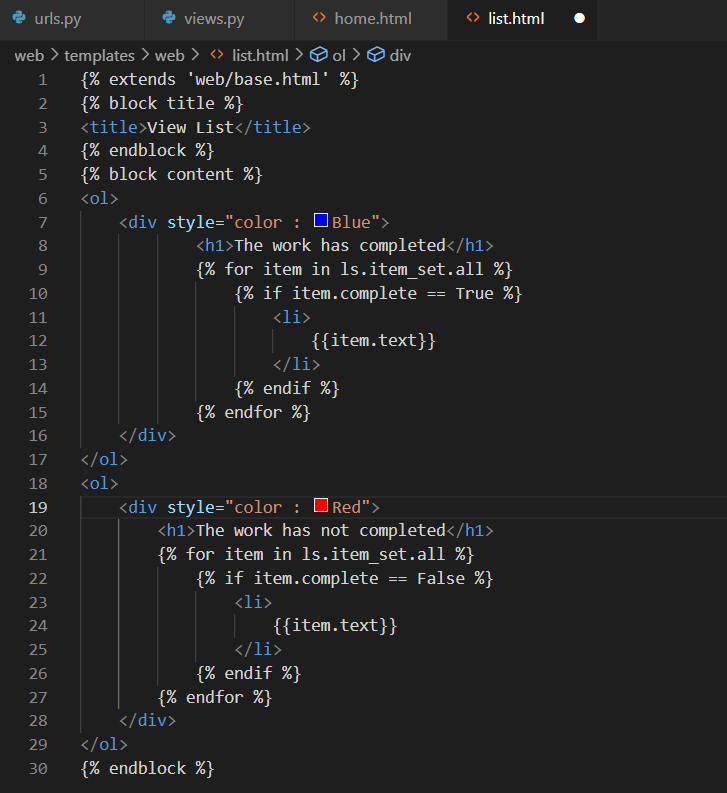
Nếu muốn có số thứ tự thì thay thẻ **<ul>** thành thẻ **<ol>**

Kết quả :

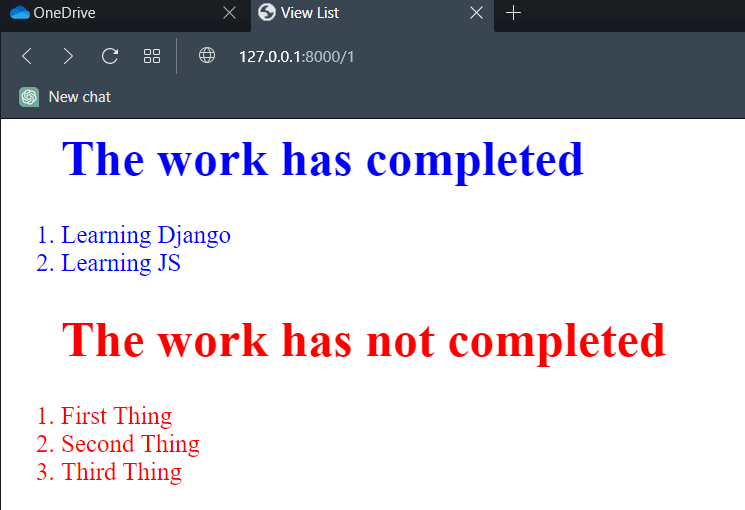


**Yêu cầu : giả sử lúc này trong objects Phat’s list có 3 Item chưa complete, 2 Item đã complete, hãy in nó ra web**

**C1 : chỉ dùng if + for**

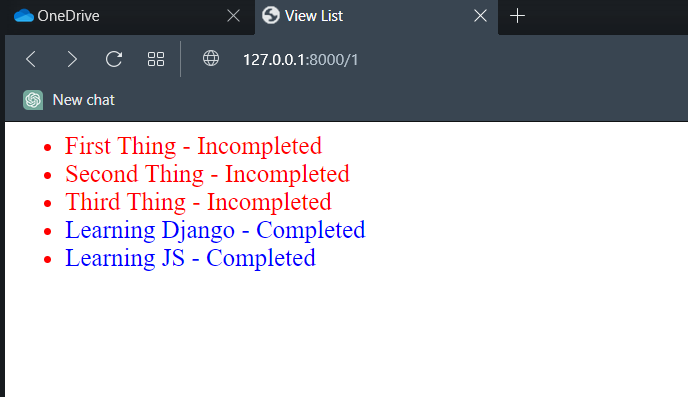


Kết quả :



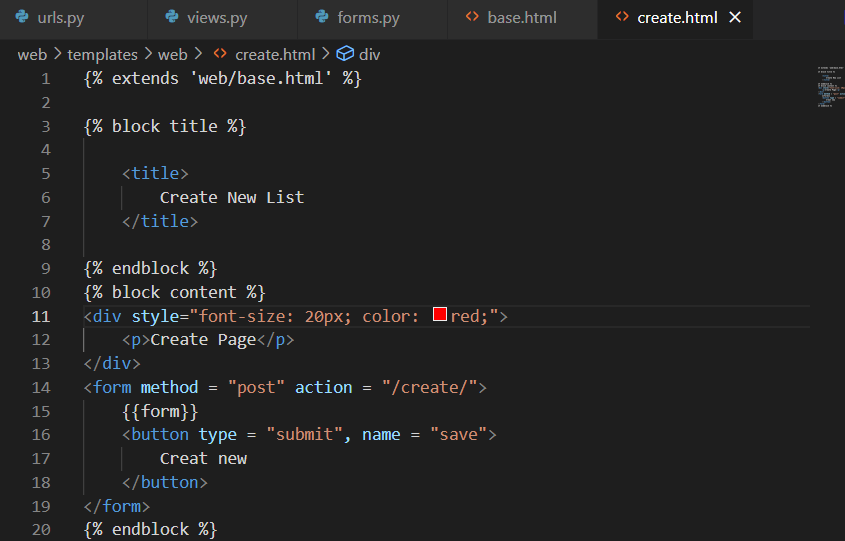
**C2 : sử dụng if - else + for**





**Simple Form:**

Tạo 1 file **create.html** để làm form



Thẻ **<form></form>** dùng để tạo biểu mẫu

**Method** là **post** đểxác định phương thức gửi đi là POST khi ấn nút

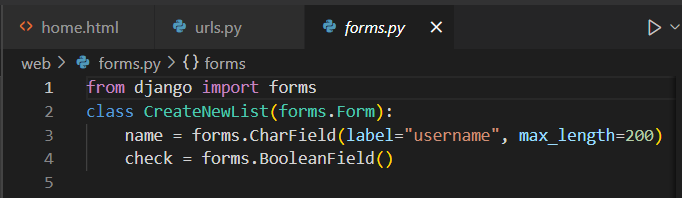
**action** là để xác định đường dẫn đến của biễu mẫu được gửi đi, khi ấn nút **submit** thì biểu mẫu sẽ gửi tới đường dẫn **/create/**

{{form}} là trường được đưa vào từ định nghĩa của class **CreateNewList**

Thẻ **<button></button>** có **type** “submit” nhằm xác định rằng đây là một nút gửi

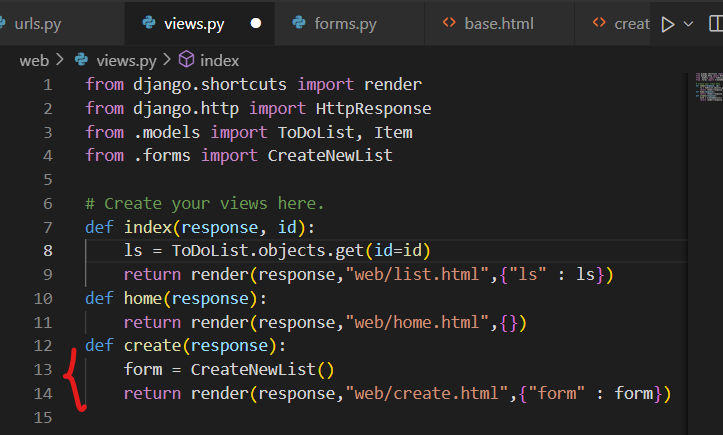
Còn thuộc tính **name** có cũng được, không có cũng không sao

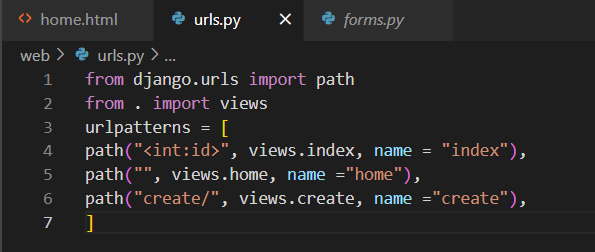
Tạo 1 file **forms.py** để định nghĩa dạng form



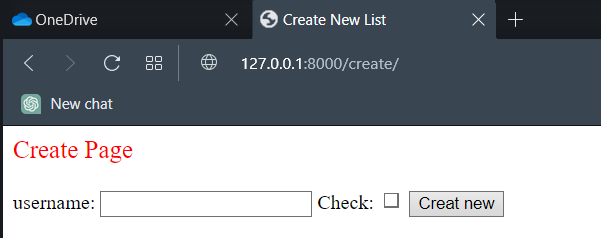
tạo ra 1 khung text để nhập kí tự (max=200) và thêm 1 ô check box để kiểm tra

Khai báo đường dẫn trong file **urls.py** và tạo hàm con **create** để có thể truy cập đường dẫn

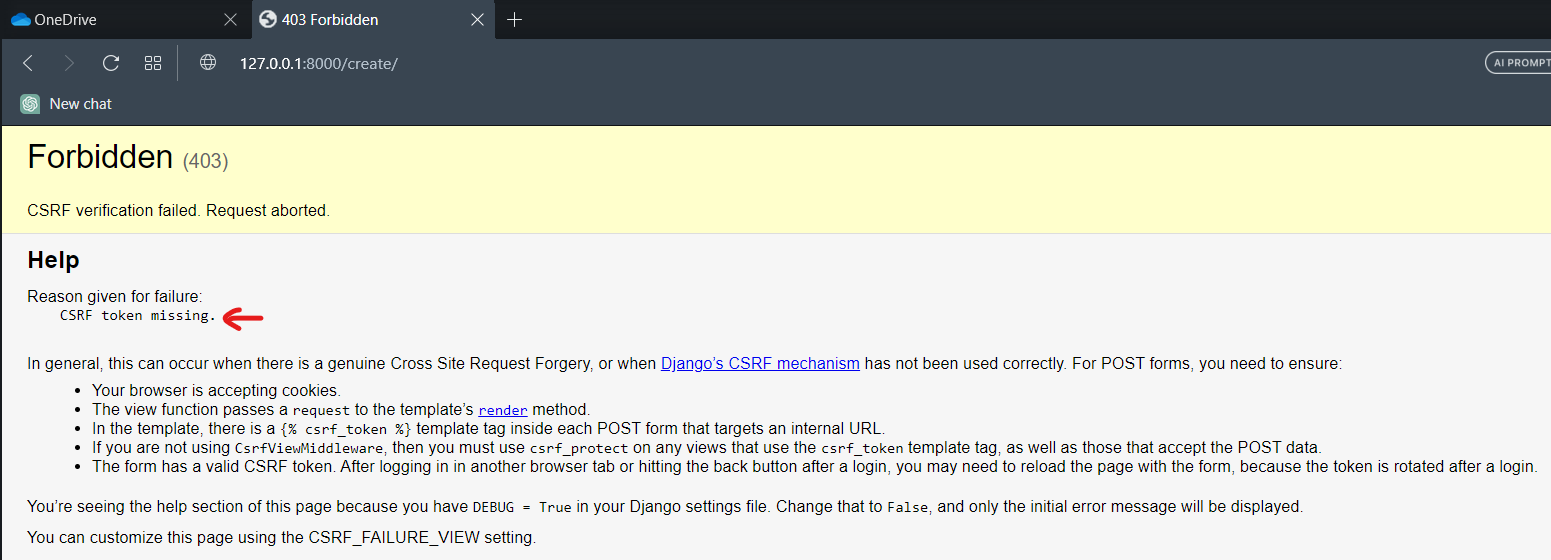




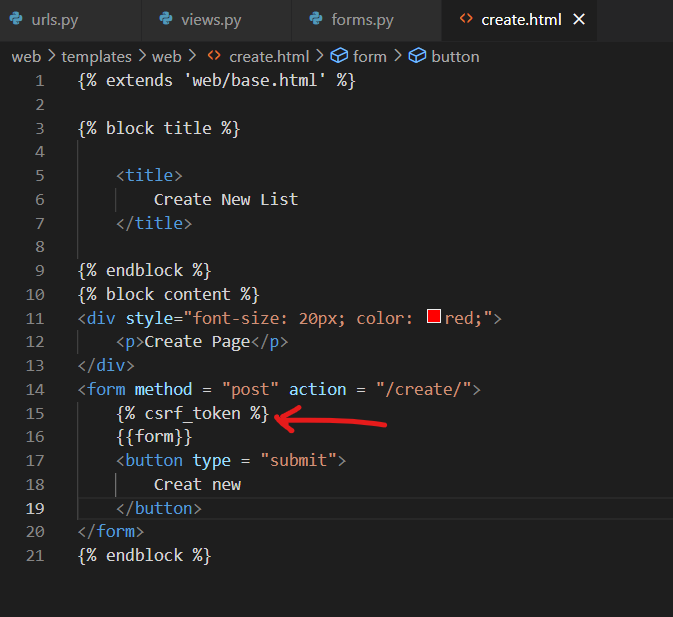
Kết quả :



Tuy nhiên khi ta nhập kí tự , nhấn check và nhấn button, màn hình thông báo lỗi sẽ xuất hiện



Để khắc phục, ta cần phải thêm trường **{% csrf\_token %}** trong file **create.html**



Thì khi ta ấn button, sẽ không còn báo lỗi và sẽ quay về đúng đường dẫn /create/

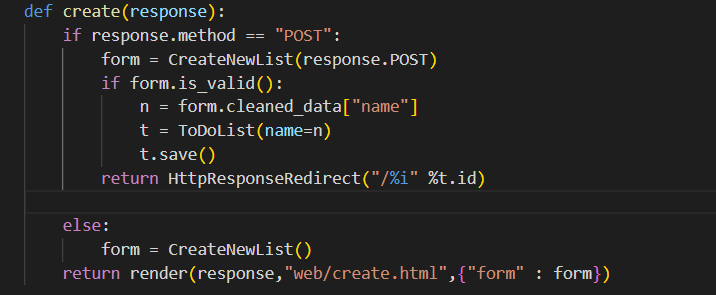
**Phân biệt Post và Get :**

**Post** là gửi thông tin lên cho máy chủ dưới dạng mã hóa ( như password ) để tránh cho ng khác biết, đồng thời sẽ dùng POST khi mà ta muốn chỉnh sửa cơ sở dữ liệu => nên create window cần dùng POST

**Get** chỉ đơn giản là truy xuất dữ liệu từ máy chủ, ai bik cũng kh quan trọng



**Kiểm tra là POST hay GET**



1. **Sự khác nhau giữa CreateNewList() và CreateNewList(response.POST)**  
   CreateNewList() : tạo 1 đối tượng form không có dữ liệu ban đầu và chỉ để hiển thị thông tin người dùng nhập

CreateNewList(response.POST) : tạo 1 đối tượng form mới dựa trên dữ liệu được gửi lên từ biểu mẫu với method POST, và đối tượng form này được sử dụng để kiểm tra tính hợp lệ của dữ liệu được gửi lên từ biếu mẫu

1. Sau khi form hợp lệ (form.is\_valid = true), thì ta sẽ lấy trường đầu vào của form đã qua kiểm duyệt (cleaned\_data là 1 thuộc tính giống như cuốn từ điển, chứa các trường đầu vào được gửi lên từ form hợp lệ)
2. **t = ToDoList(name=n) và t.save()**

Câu lệnh này là tạo 1 objects mới trong class **ToDoList** và sau đó lưu lại

1. return HttpResponseRedirect("/%i" %t.id)

Sẽ điều hướng trình duyệt tới một đường dẫn URL mới được chỉ định trong đối tượng HttpResponseRedirect.

Ý nghĩa code thì khi mà tạo 1 objects ToDoList thành công, sẽ điều hướng ng dùng tới URL có dạng …/id với id là của objects vừa được tạo ra

**Custom Form**

Giả sử ta tạo 1 form có các chức năng là hiện tên của objects và các Item của object đó, nếu Item nào đã hoàn thành (**complete**) thì sẽ có dấu check trong checkbox và ngược lại. Ngoài ra còn có thêm 4 nút chức năng gồm :

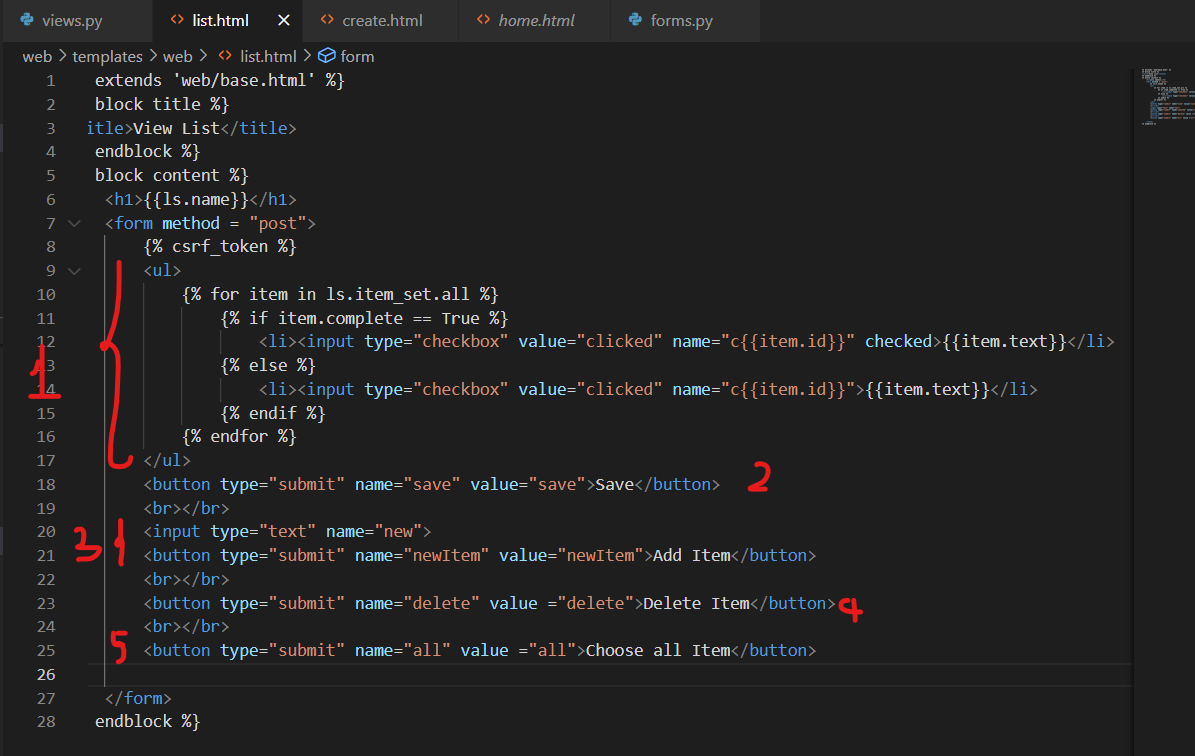
**Save :** nút này dùng để lưu **complete** của Item được tick vào ô checkbox (là như khi checkbox chưa được check, khi ta check vào trong checkbox đó thủ công và nhấn **save** thì Item đó sẽ được chuyển từ **chưa** **complete** sang **complete**)

**Add Item :** dùng để khi ta muốn lưu 1 Item mới vào textbox

**Delete :** sẽ xóa các item đã được check

**All :** chọn toàn bộ các Item (Khi ấn **all** xem như cho tất cả các Item **complete**)

**B1:** sửa file list.html

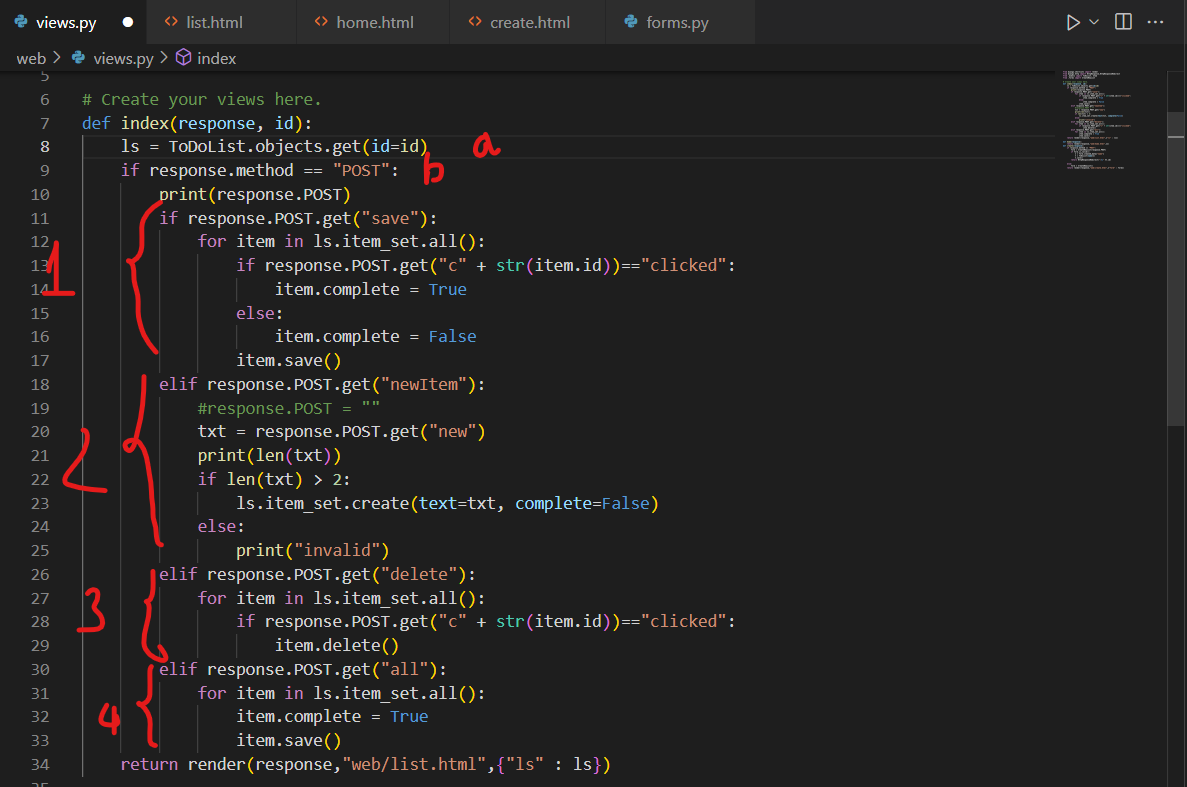


Mục 1 : khi trang web được truy cập từ url, thì các Item đã **complete** sẽ được check check box ( <… **check>**… và ngược lại

Mục 2: tạo button đặt tên **“Save”** , có các thuộc tính là **kiểu “gửi”** , tên nút nhấn là **“save”** dùng để xác định cụ thể nút nhấn nào khi form có nhiều nút nhấn, giá trị nút nhấn là **“value”**  khi nhấn nút thì cái giá trị này được đưa vào trong POST data cùng với tên nút

mục 3 tới mục 5 là tạp các button với các thuộc tính

**B2:** trong file **views.py**



**Mục a** là để lấy objects có id được lấy từ id khi điền URL

**Mục b** là để kiểm tra xem có phải phương thức **POST** được yêu cầu không

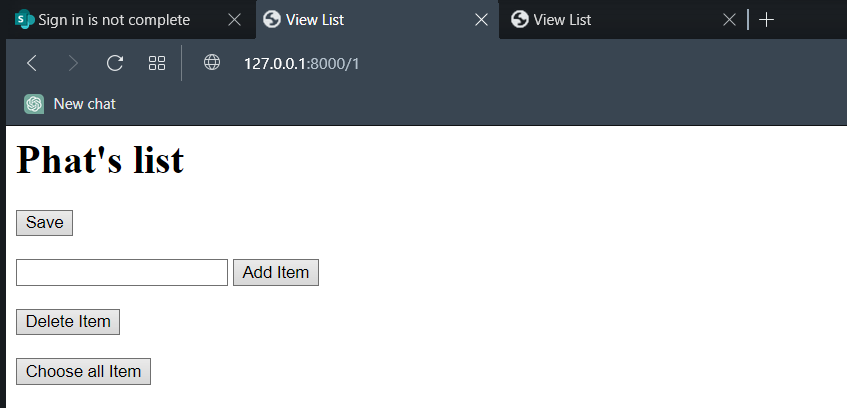
**Mục 1** là kiểm tra xem nút tên **“save”** (nút **Save** ) có được nhấn không, nếu có sẽ cho vòng lặp for lặp qua tất cả các item của objects đó (trong file Html đã có code in các Item đó lên), sau đó kiểm tra xem checkbox của item có được check không, nếu có thì sẽ cho **complete** của item đó là **True** , còn nếu checkbox không có check thì **complete** sẽ là **False**. Sau đó lưu kết quả thay đổi lại.

**Mục 2** là kiểm tra xem nút tên **“newItem”** có được nhấn không, nếu có thì sẽ lấy chuỗi kí tự được nằm trong text có tên **“new”** và kiểm tra xem độ dài của chuỗi kí tự đó, nếu lớn hơn 2 thì sẽ tạo Item đó có tên của chuỗi kí tự trong text, và **complete** sẽ là **False**

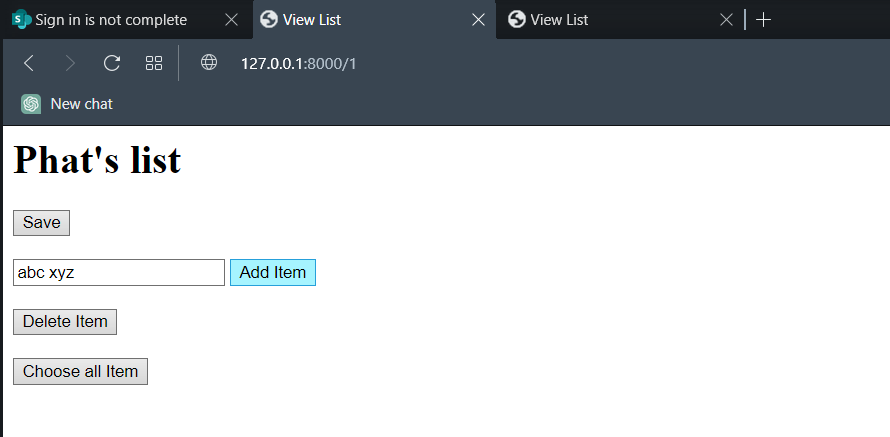
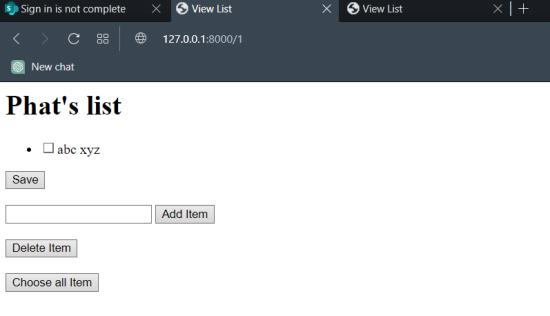
**Mục 3** là kiểm tra nút tên **“delete”** có được nhấn không, nếu được nhấn sẽ xóa các item được checkbox

**Mục 4** là kiểm tra nút tên **“all”** có được nhấn không, nếu có thì tất cả các item sẽ có **complete** là **True** (complete = true thì được check trong checkbox)

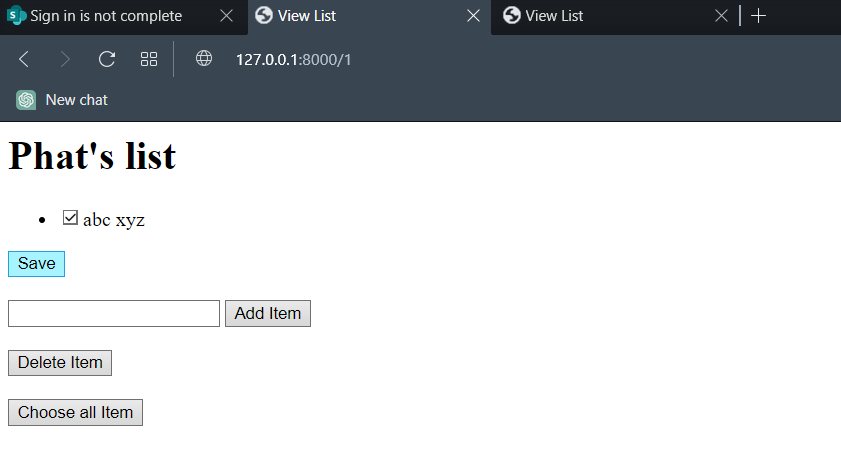
Kết quả : với objects có id = 1 (chưa có item)



Thêm Item vào và nhấn **Add Item**:

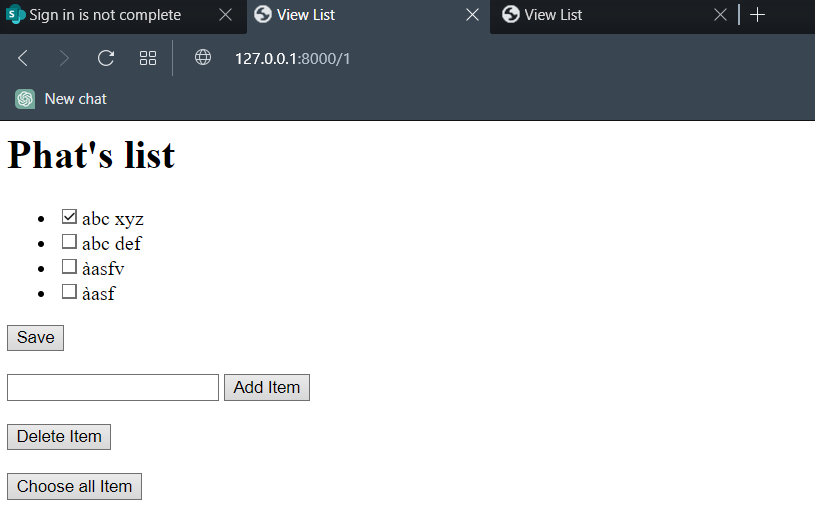
=>

Tick vào checkbox và ấn **Save**

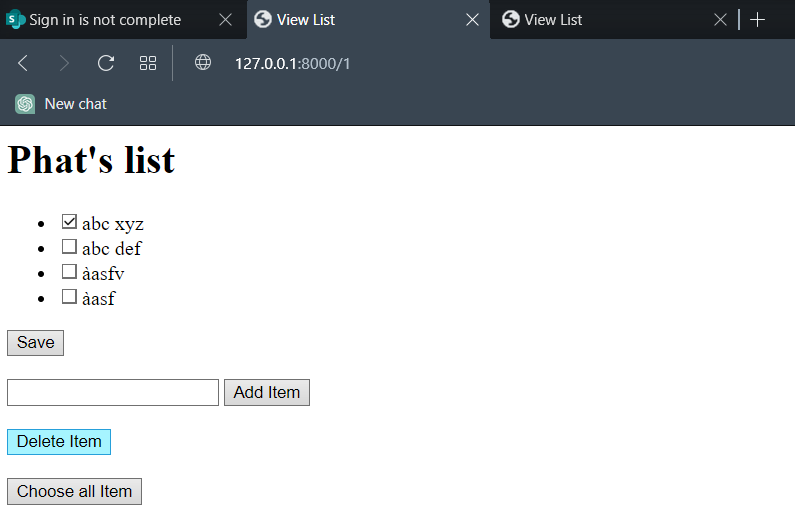
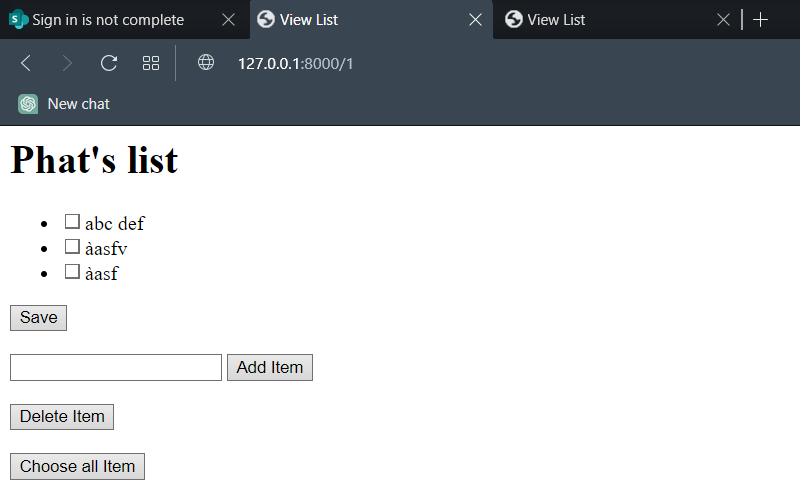


Sau khi ấn **Save** thì reload lại trang sẽ không mất dấu check ( do complete của Item lúc này là **True** )

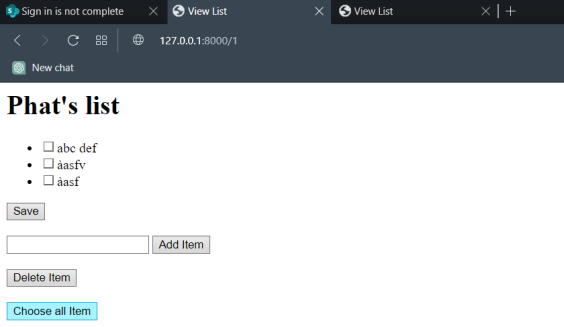
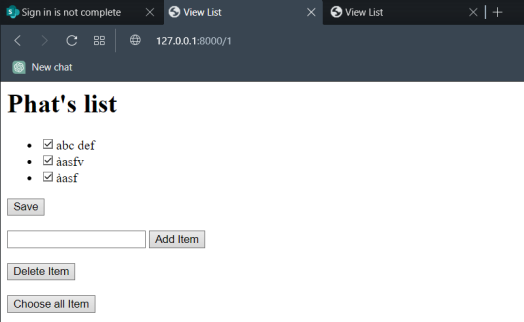
Giả sử ta có các Item và trạng thái Complete sau :



Khi chọn ấn **Delete Item** sẽ làm mất đi item “abc xyz”

=>

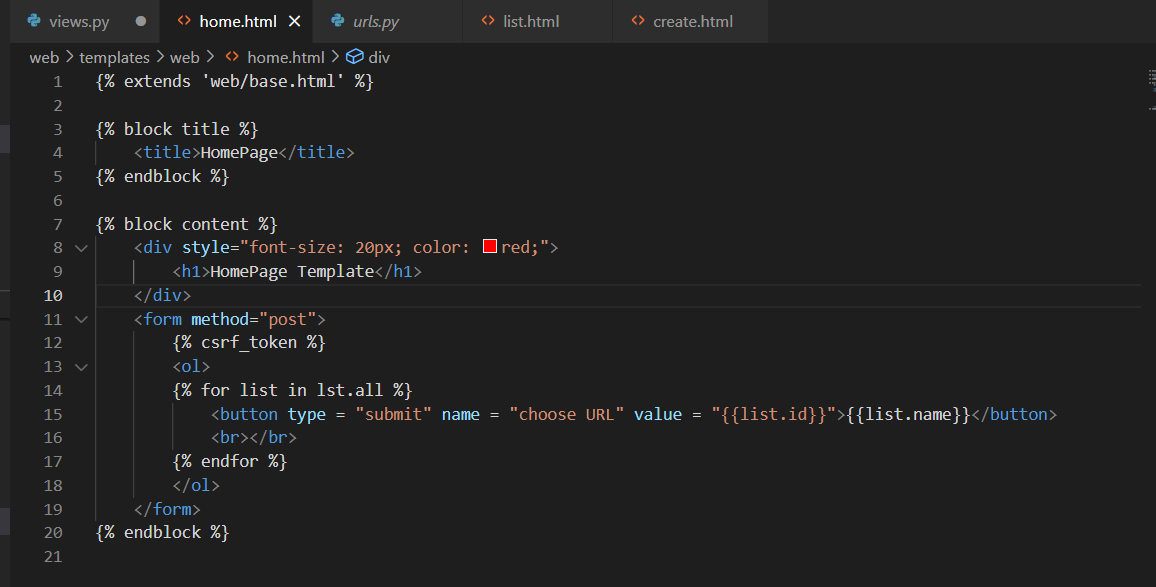
Khi chọn ấn **Choose all Item** thì sẽ check toàn bộ checkbox

=>

**Mở rộng :**

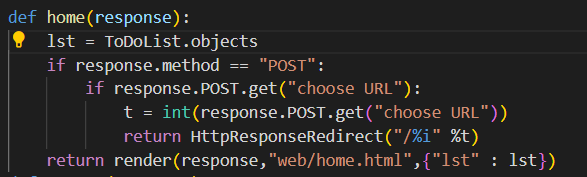
**Khi truy cập vào Home page, thì sẽ có 3 button để lựa chọn vào list, và ở trong các list objects sẽ có 1 button “exit” để thoát khỏi list quay trở lại Home page**

Trong file **home.html** sẽ bổ sung đoạn sau



Truyền lst = ToDoList.objects từ file **views.py** vào, sau đó dùng vòng lặp for để lặp qua hết các objects đó và tạo button cho từng objects list đó.

Trong file **views.py** ,hàm con **home** sẽ bổ sung đoạn sau

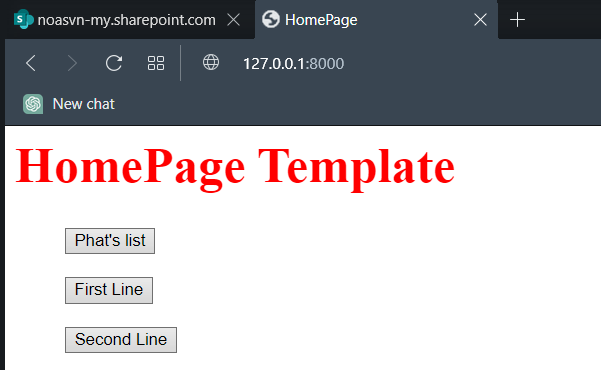
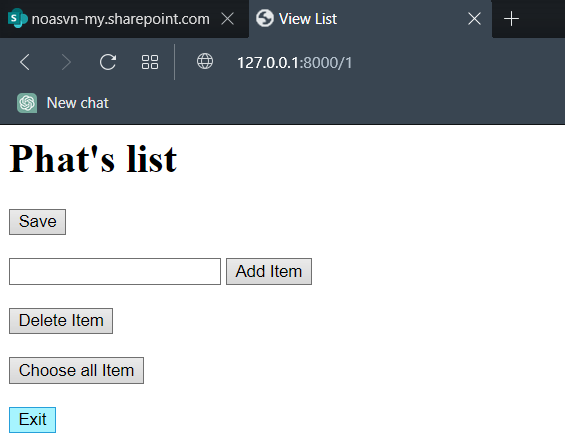


Lúc này khi nhận được request **POST** và khi ấn nút chọn list, ta đặt t là id của objects list ta chọn và sau đó sẽ điều hướng đến URL có địa chỉ id của list được chọn

Trong file **views.py** ,hàm con **index** bổ sung thêm khi nhấn nút **“exit”** thì sẽ đưa về trang chủ

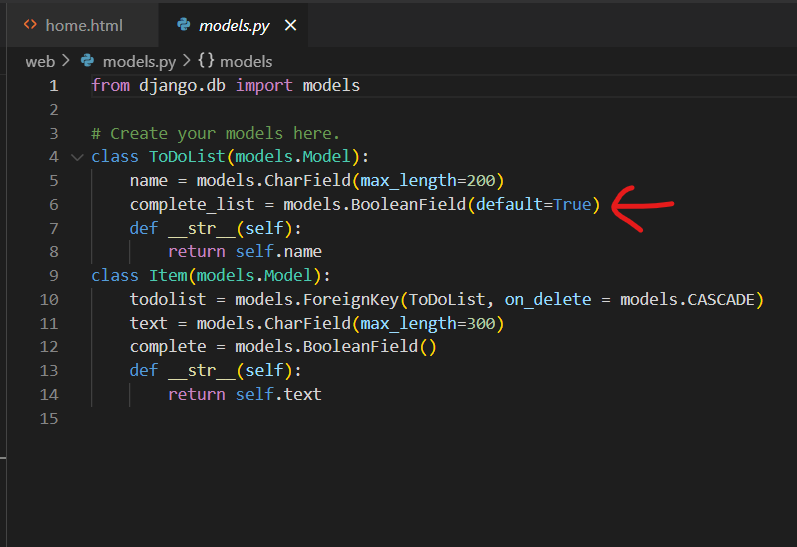


Kết quả :

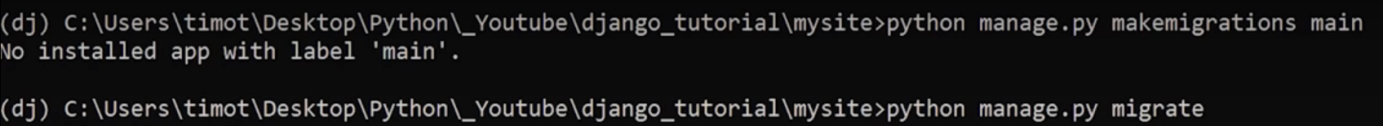
 ===> 

**Mở rộng : trang homepage có mục tạo list mới, nút chọn tất cả list, nút delete list tùy chọn, khi các item trong list đều hoàn thành thì checkbox list đó sẽ đánh dấu check ( và nếu có ít nhất 1 item chưa hoàn thành thì checkbox của list sẽ không được check. Khi list được check đã hoàn thành thì toàn bộ các mục item trong list cũng hoàn thành theo.**

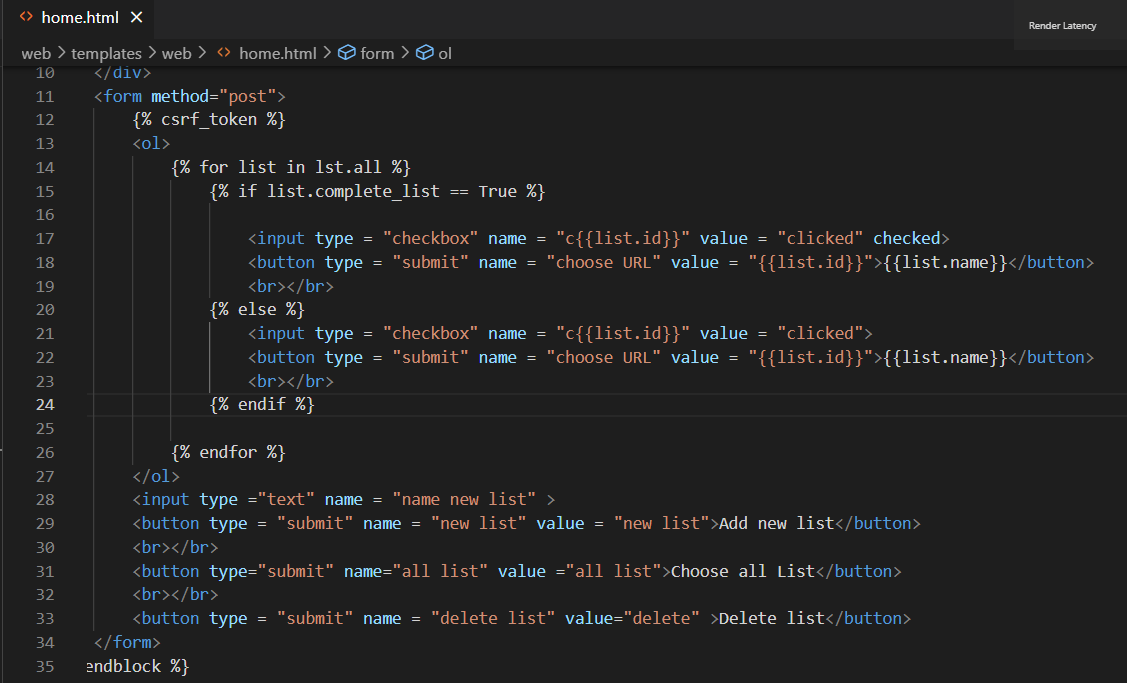
Chỉnh sửa lại thêm thuộc **complete\_list**cho **list** và mặc định một list mới tạo ra sẽ có **complete\_list=true**



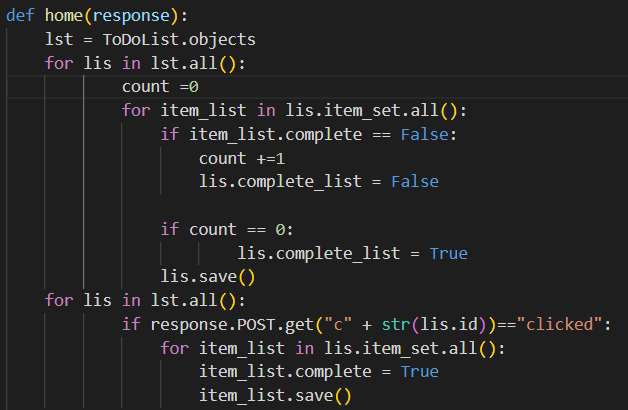
Built lại : (chữ main kế makemigrations thay bằng tên của app mình thiết kế)



Trong file HTML cập nhật các nút mới, trong đó có thêm mục **checkbox** nhằm kiểm tra nếu complete\_list = true thì sẽ check checkbox đó

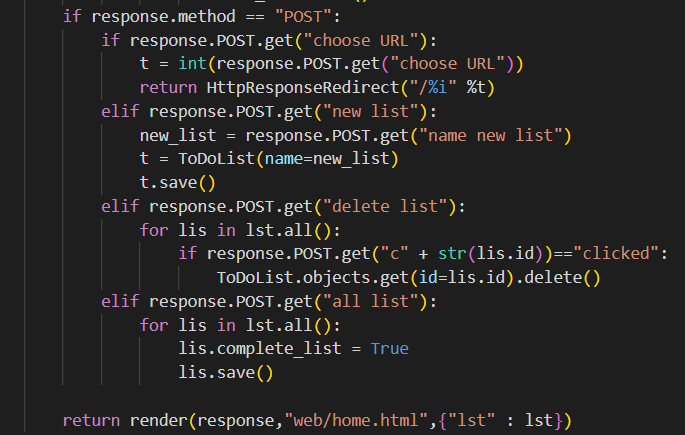


Trong file **views.py**

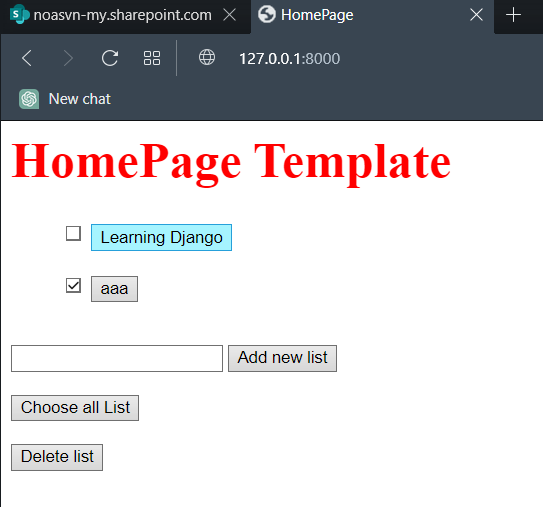
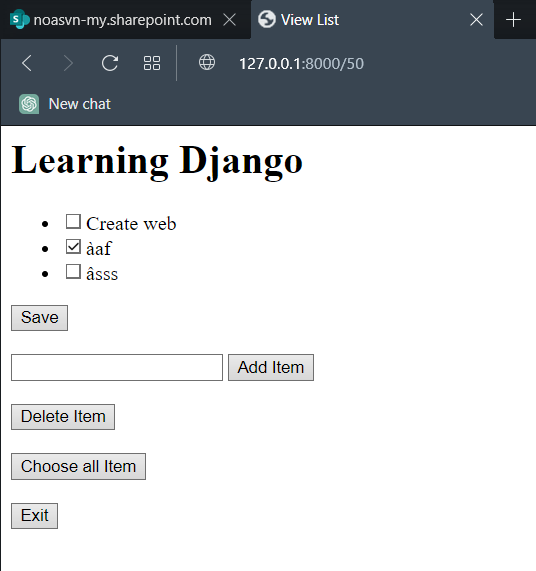


Ở đoạn này, ta sẽ kiểm tra xem là cái item trong list có cái nào chưa complete hay không, nếu có ít nhất 1 cái thì sẽ set complete\_list của list về false sau đó lưu lại, nếu không thì true

Nếu checkbox của list nào được check thì toàn bộ các item trong list đó sẽ đều complete ( vòng for cuối )

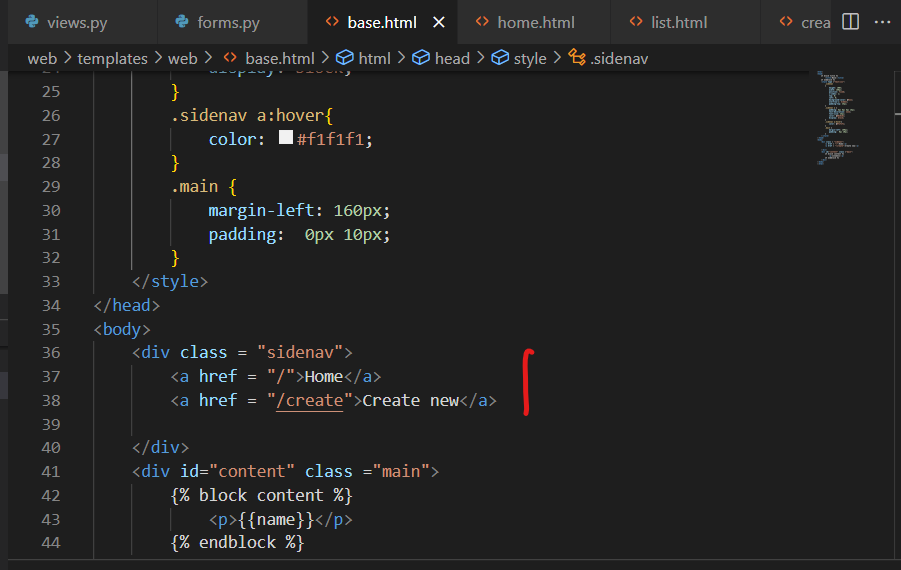


Thiết lập chức năng các nút theo yêu cầu

=>

**Add slidebar**

**< a href =”địa chỉ đường dẫn”>Tên đường dẫn</a>** : là thẻ dùng để tạo ra 1 đường liên kết trên trang web

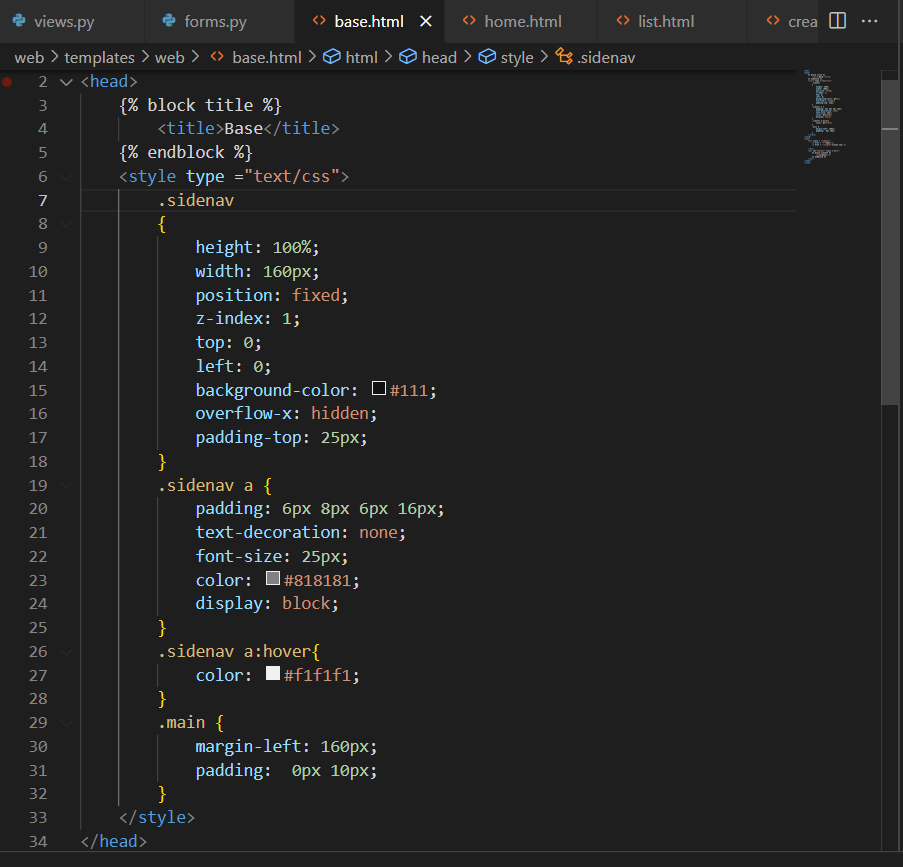


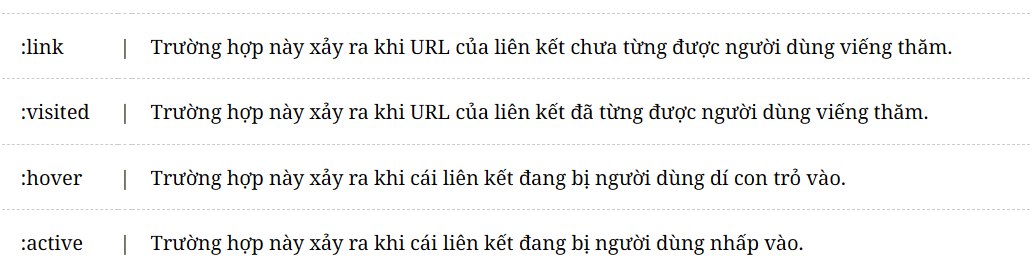
Tạo 2 đường dẫn và 2 đường dẫn đó nằm trong **class “sidenav”**

Tiếp đó là tùy chỉnh giao diện cho class, để có thể tùy chỉnh giao diện của toàn bộ các phần tử con trong class ta muốn, ta sẽ nhập trong thẻ **head** cấu trúc sau: **. Tên class { config }**

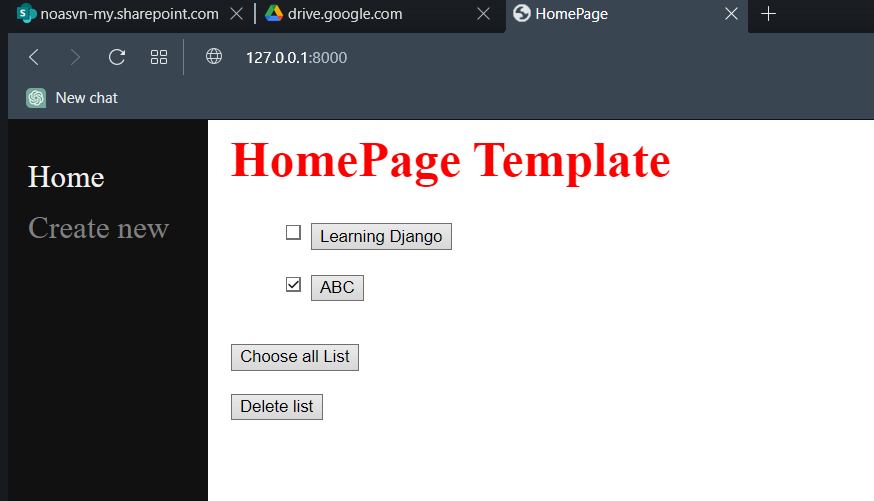
Còn nếu ta chỉ muốn cấu hình giao diện cho các đường dẫn liên kết, ta sẽ sử dụng các cấu trúc sau

**. Tên class a { config }**





Kết quả

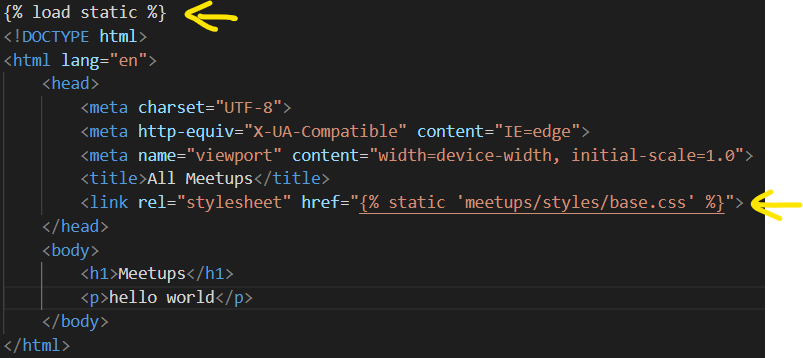


**Static content**

Ảnh, file css, javascript kh thay đổi khi truy cập vào các trang ứng dụng

Cần tạo 1 folder tên static, bên trong là folder tên của apps, sau đó tạo các folder riêng cho từng loại static content

Sau khi ta có nội dung của file style css static muốn truyền vào, ta thực hiện câu lệnh sau



Trong đó địa chị của file nằm trong ‘…’ trong {% … %}