**Useful Commands:**

Pip freeze

Python –version

Python manage.py help

Make a virtual environment

python -m venv ./venv

Activate Environment:

venv\Scripts\activate.bat

Install:

Pip install django

Start a project with name btre (don’t forgot the fullstop!)

django-admin startproject btre .

Should set up my Git repo here:

python manage.py runserver

Create Django app called pages:

Python manage.py startapp pages

Next step is to add this app to the settings.py file of the project (btre)

Under the “Installed Apps” Section add ‘pages.apps.PagesConfig’ this ‘PagesConfig’ can be found in the pages ‘apps.py’ file

Now in my pages app create a file called ‘urls.py’:

from django.urls import path

from . import views

urlpatterns = [

    path('', views.index, name='index')

]

We don’t have this ‘index’ member in the above code and it is flagging as an error because we do not have this method inside our views file. We must now add this to our views file.

Pages > views.py

from django.shortcuts import render

from django.http import HttpResponse

# Create your views here.

def index(request):

    return HttpResponse('<h1>Hello Django</h1>')

Now if we look at our webpage it is still not showing this “Hello Django” message as we still have to take this ‘urls.py’ file that we created and added it to the main ‘urls.py’ in the main project.

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

    path('', include('pages.urls')),

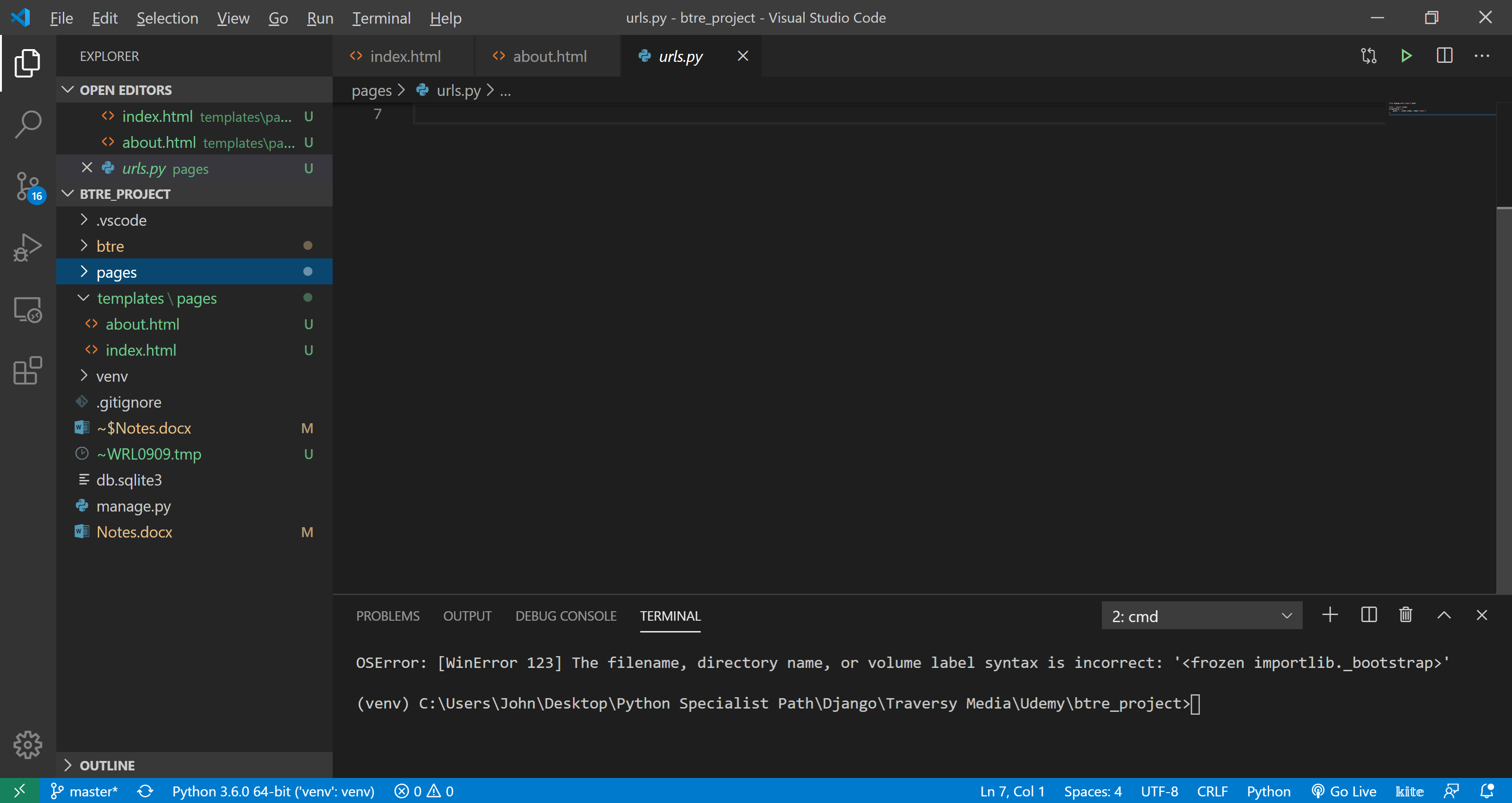
    path('admin/', admin.site.urls),

]

OK so now we have a basic webpage displaying the “Hello Django” message, we don’t really want to just insert html straight into our code like that. Instead we will make a template folder in our root directory to house these (front end) files. If we use this new file structure we have to tell the Django project where to find these templates, we can do that by updating the ‘DIRS’: line with the following in the projects ‘settings.py’ file under the templates section

'DIRS': [os.path.join(BASE\_DIR, 'templates')],

OK so now I have added a templates folder to the root, inside that folder I have another folder ‘pages’ inside pages I have ‘index.html’ and about.html’



Now that we have added these two new html files we need to add them to the pages ‘url.py’ file (note we already have the index added from before).

from django.urls import path

from . import views

urlpatterns = [

    path('', views.index, name='index'),

    path('about', views.about, name='about')

]

Now we will get an error because we don’t have an about method in our views file so that line ‘views.about’ is a problem.

Pages ‘views.py’

from django.shortcuts import render

from django.http import HttpResponse

# Create your views here.

def index(request):

    return render(request, 'pages/index.html')

def about(request):

    return render(request, 'pages/about.html')

[Django documentation to render() method above](https://docs.djangoproject.com/en/3.0/topics/http/shortcuts/#django.shortcuts.render)

In the code above we are using the built in Django shortcut function ‘Render()’ to render the html page, it will return a HTMLResponse object with the rendered content. The request is a method such as ‘POST’, ‘GET’ etc …. In this case if we print request we get <WSGIRequest: GET '/'> . The second field here is the template name which we need to pass the location of the file. This is slightly confusing at first as you would think that you would pass ‘templates/pages/index.html’ but this would be incorrect as we have already told django to look for templates in the templates folder in one of the steps above.

Creating a ‘base.html’ template

There are some html components that we want in all of our pages, instead of copying and pasting the same code to multiple locations we can use a ‘base.html’ file. In our template folder we can all this ‘base.html’ file directly here.

Base.html

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>BT Real Estate</title>

  </head>

  <body>

    {% block content %} {% endblock %}

  </body>

</html>

Index.html

{% extends 'base.html' %} {% block content %}

<h1>Home</h1>

{% endblock %}

Note:

This uses jinja (this is a kind of programming language on its own?? )

{% extends 'base.html' %} {% block content %}

**Dealing with Static Files and paths:**

In the main project folder ‘btre’ create a new folder called static. In this static folder we will add any static files such as bootstrap files, css, webfonts. In this example I am following the Traversity media udemy course so he has given a list of files to add.

Now that we have added the static folder containing the static files we need to tell Django where to find them. We do this by modifying the main project ‘settings.py’ file with the following:

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/3.0/howto/static-files/

STATIC\_ROOT = os.path.join(BASE\_DIR, 'static')

STATIC\_URL = '/static/'

STATICFILES\_DIRS = [

    os.path.join(BASE\_DIR, 'btre/static')

]

Now that we have set up the static structure we need to run:

python manage.py collectstatic

The above command goes into all of the applications and if it has a static folder it takes everything from that folder and puts it into a root static folder. This root static folder is automatically created whrn the collectstatic command is ran.

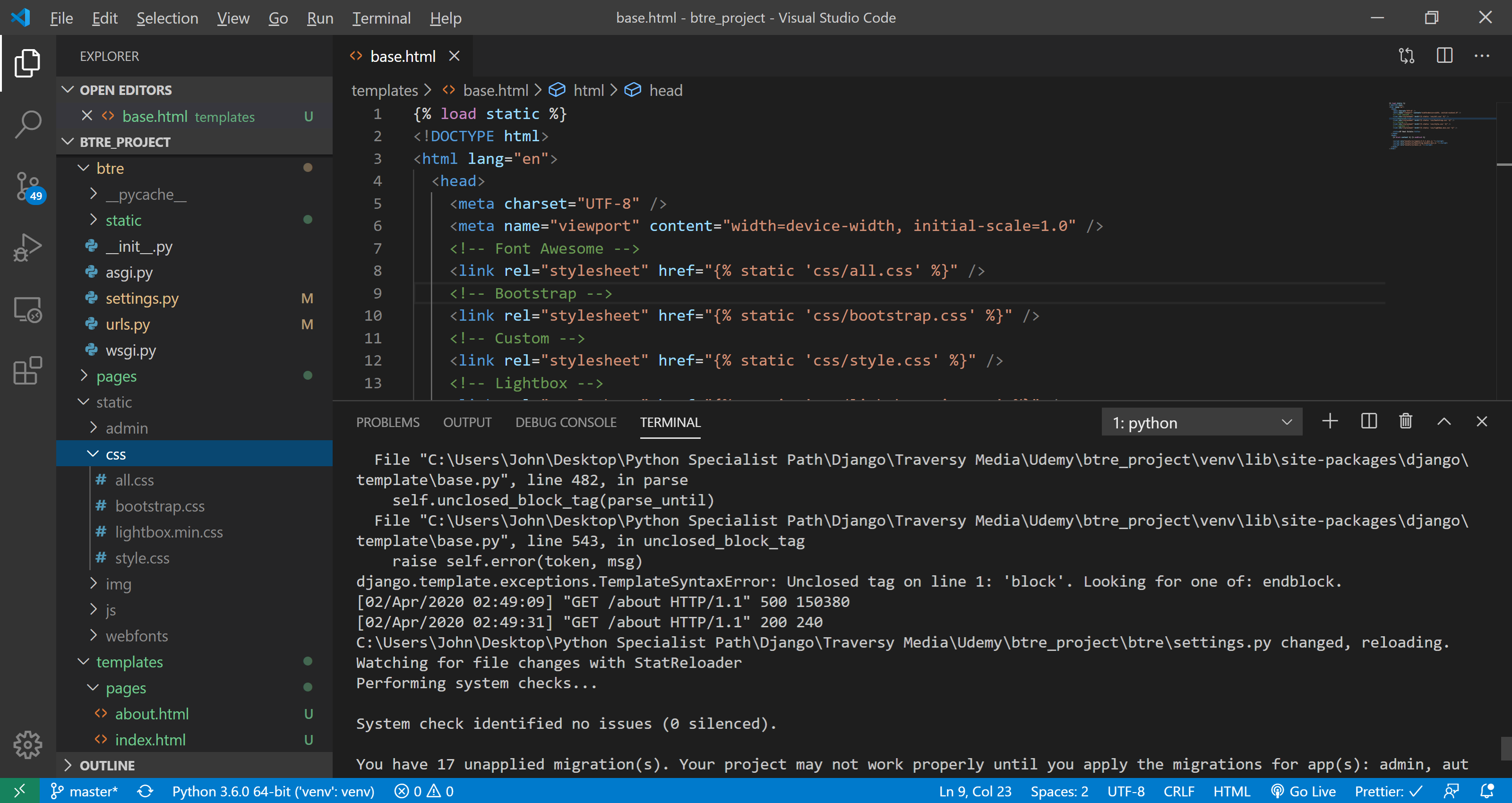
After this we want to update our .gitignore with:

/static

Now in our base.html file we need to modifiy a lot of things, first we need to load static at the very top of our file.

{% load static %}

Then we need to connect all of our links (note: this are actually in our static root folder)in the correct way for example



<!-- Font Awesome -->

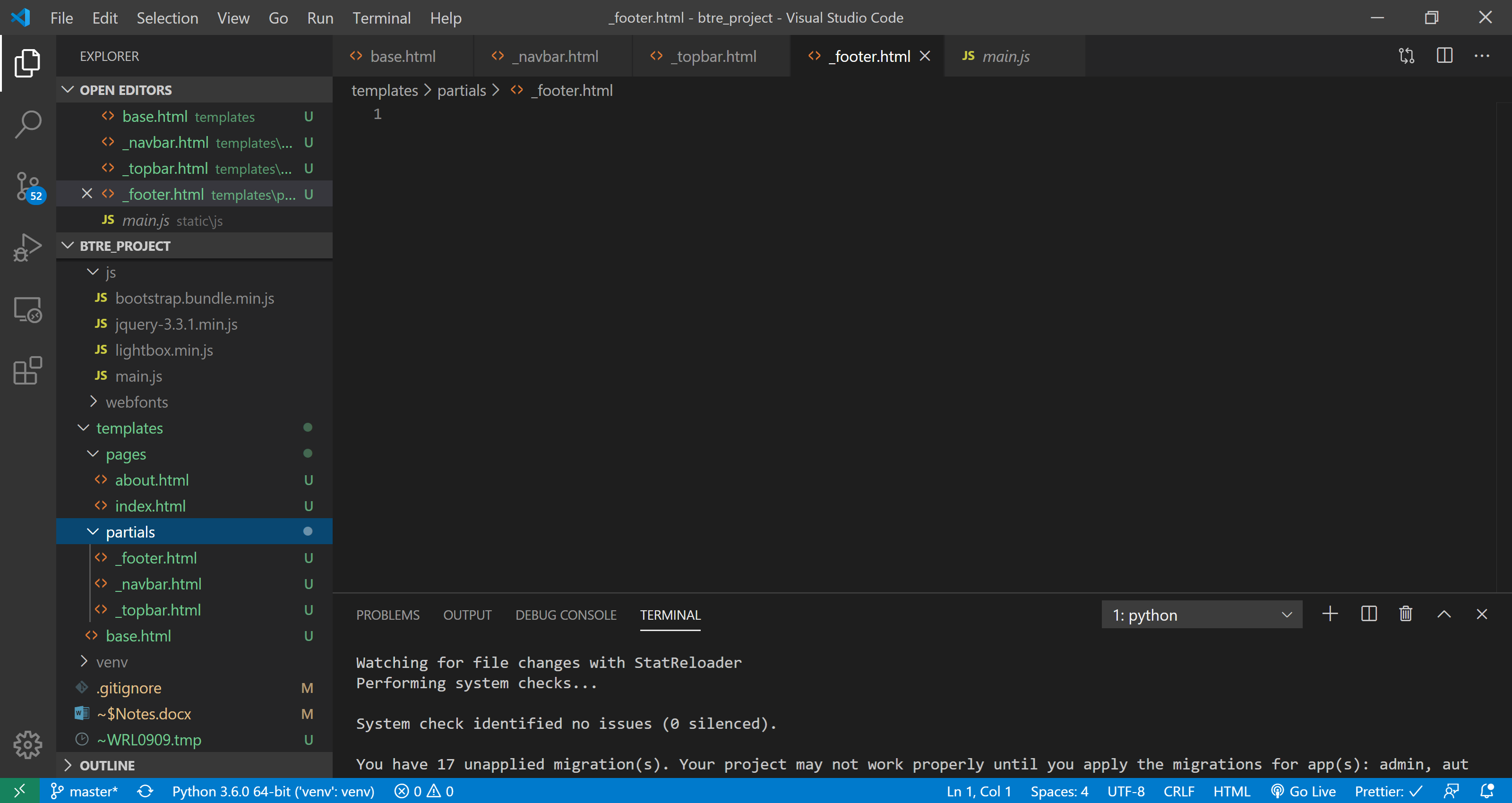
    <link rel="stylesheet" href="{% static 'css/all.css' %}" />

We need to do the same with the script tags:

<script src="{% static 'js/jquery-3.3.1.min.js' %} "></script>

Now we have the option of putting all of our html markup that is used on multiple pages into our ‘base.html’ this is ok to do but it is better if we use partials.

To use partials we need to create a ‘partial’ folder in our ‘templates’ folder. In this folder we can put in components of our pages such as the navbar, header, footer ect. The naming convention for partial files is ‘\_name.html’ the underscore is used.



To include these in the ‘base.html’ we just have to use

<!-- Nav Bar -->

    {% include 'partials/\_navbar.html' %}

The next step is to add our links within the site, so for example if we click ‘home’ on the navbar we want to be taken to the home page. To achieve this we use the following

<a class="nav-link" href="{% url 'index' %}">Home</a>

Note: we can use just ‘index’ instead of ‘index.html’ because in the pages app ‘urls.py’ file we have called it this:

path('', views.index, name='index'),