Explaining the blog project.

* First thing was to create the models that we will be using here. Mainly, we needed a Post model and a comment model. Those the two that we need here. And both will be a lot similar to each other.
* For my Post I created an author which because we only have one person to edit or create the posts it is linked to the inbuilt User model. Next will be the title of the post. Then the actual text where the content goes. And there will be a creation date and a published date, so the user will have the ability to create a post and save it. They do not have to post it at that moment. The published date can be left blank and it can be null.
* So, inside the class there is a publish method where if the author publishes the post they get the current time. There is a approve\_comments method that returns just the approved comments. (Basically, the idea is there will be approved and unapproved comments we just want to display the approved ones.) Next after the user clicks the publish button they will be redirected to the post\_details page. They key argument is takes is primary key of the post that links with the post\_details list. The string method gives the title display.
* For the comments model I have to link the comment to the specific post which it can do with a Foreign key. There are author, text and creation date attributes. It will also have an attribute that says if the comment is approved or not.
* Inside the class there are several methods. First one is if the comment is approved or not. Then the get absolute URL method which after the posting of the comments will take the commentator to the post list page. There is also a string method.
* Inside the forms.py file I used the ModelForm to create form from the model fields. Here I am also formatting the CSS styling for specific widgets. As for the fields we are going to use all of them.
* For views.py I started with the homepage which is about.html. That should be pretty simple to create. Then the ListView which was nothing more than just importing stuff and creating the view. Next using the Django ORM, I created a query set that will filter the publication date which is less than or equal to the date right now and will order it in descending order.
* As for the detailed view, it will run when the user clicks on the title of a post it is very simple to create. All we have to do is link the model with the post.

CRUD (Create, Update and Delete)

* In Order to create a post, the user needs to be logged in. Hence, for that we need to check whether the user is logged in or logged out. For the function views we could have used decorator login\_required but for class views we need to import mixins. The required one here is called LoginRequiredMixin. Unlike the decorators it will be used as a parameter and according to Django documentation it needs to have some specific attributes. The specific attributes that we need here are login\_url which basically tells where to find the loginpage and the redirect\_field\_name which tells what to do after the login is done. The form\_class will be the PostForm.
* For the update view it is very similar to the create view in fact everything should be the same.
* For the delete view like I had done before we need the reverselazy function to redirect when the delete is completed.
* Lastly, we need a draft view where all the drafts have been saved. The used needs to be logged in to view the drafts hence the mixin is required. And I have a query set that will fetch the posts that have no published date. If you remember when a post is published it get a published date assigned to it.
* As I am doing this I keep updating my url.py inside my application to make it streamlined.
* After this I ran my server and had a couple of syntax errors and typos but once I fixed those the server is up and running.
* Next I have here is setting up the html and css files.