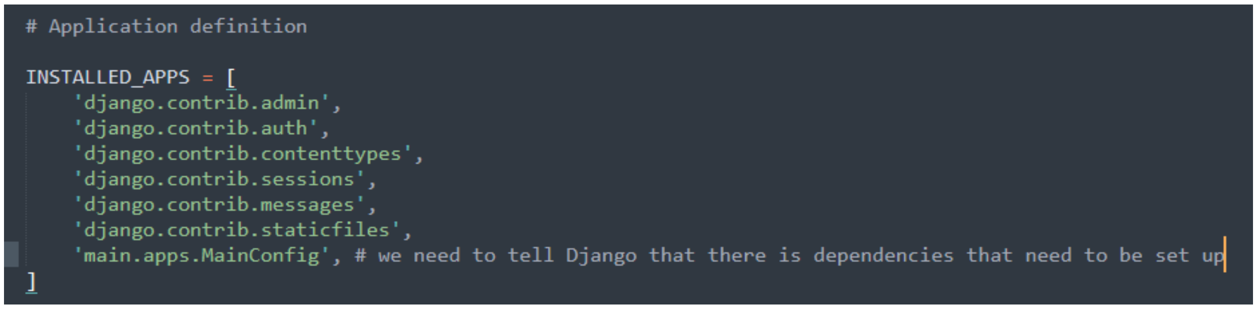
Setting up a database

Database used: SQLite3

1. Navigate to mysite folder and find settings.py file. Add the following line to the code.

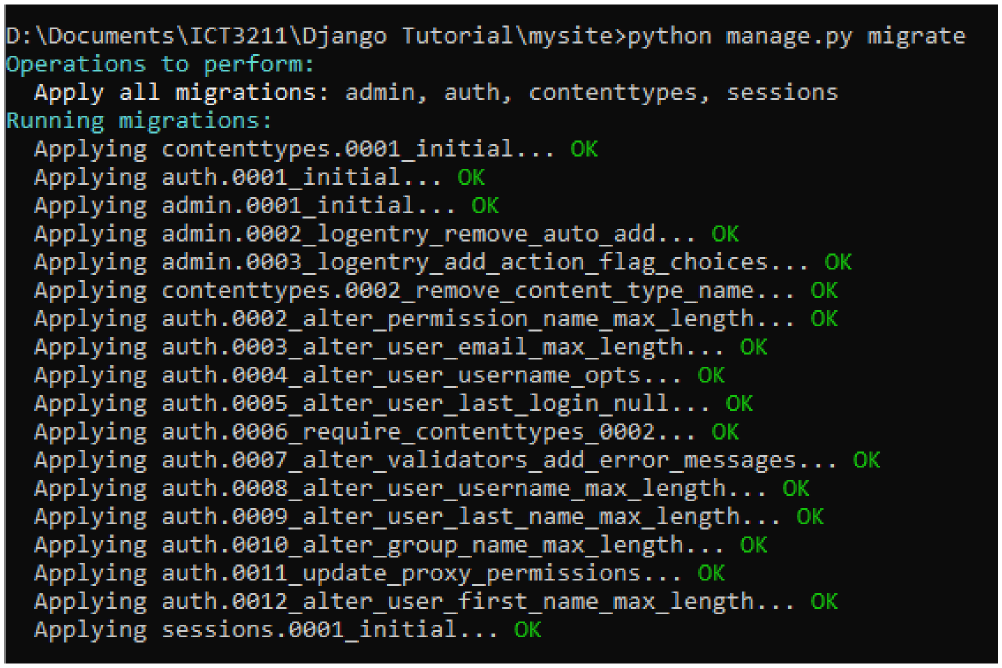
```
'main.apps.MainConfig'
```



```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'main.apps.MainConfig', # we need to tell Django that there is dependencies that need to be set up
]
```

2. Once you're done with that, go to your command terminal and navigate to your folder that you have your app installed in. In my case, it's mysite. Once you have navigated there, type:

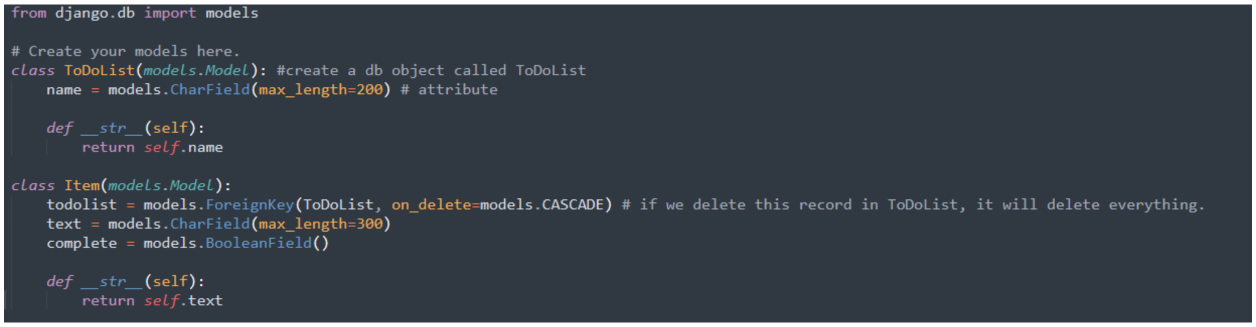
```
python manage.py migrate
```



```
D:\Documents\ICT3211\Django Tutorial\mysite>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

This essentially says that we've made changes to the settings.py file and commits it.

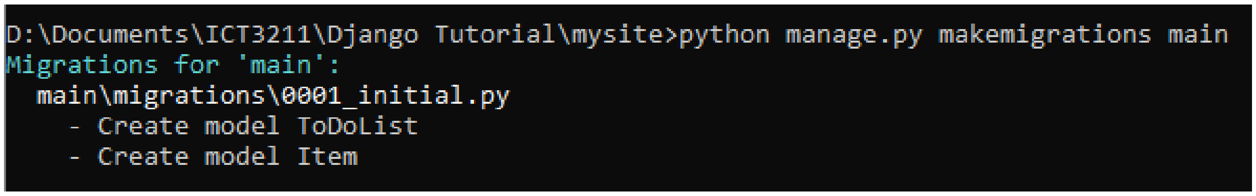
3. Next, we will go to the models.py page. In our application, I want to create a "To-Do List".



```python
from django.db import models

# Create your models here.
class ToDoList(models.Model): #create a db object called ToDoList
    name = models.CharField(max_length=200) # attribute

    def __str__(self):
        return self.name

class Item(models.Model):
    todolist = models.ForeignKey(ToDoList, on_delete=models.CASCADE) # if we delete this record in ToDoList, it will delete everything.
    text = models.CharField(max_length=300)
    complete = models.BooleanField()

    def __str__(self):
        return self.text
```

4. Next, we will need to "tell" Django that we made modifications to our models. To do this, type:

```
python manage.py makemigrations main
```

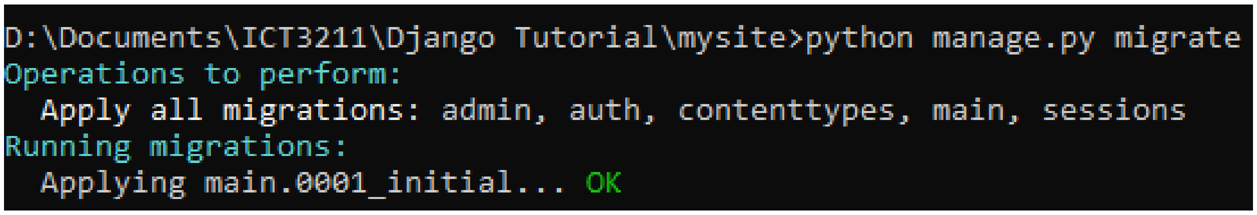


```
D:\Documents\ICT3211\Django Tutorial\mysite>python manage.py makemigrations main
Migrations for 'main':
  main\migrations\0001_initial.py
    - Create model ToDoList
    - Create model Item
```

Here, we see that we have created the models ToDoList and Item, which we have done earlier.

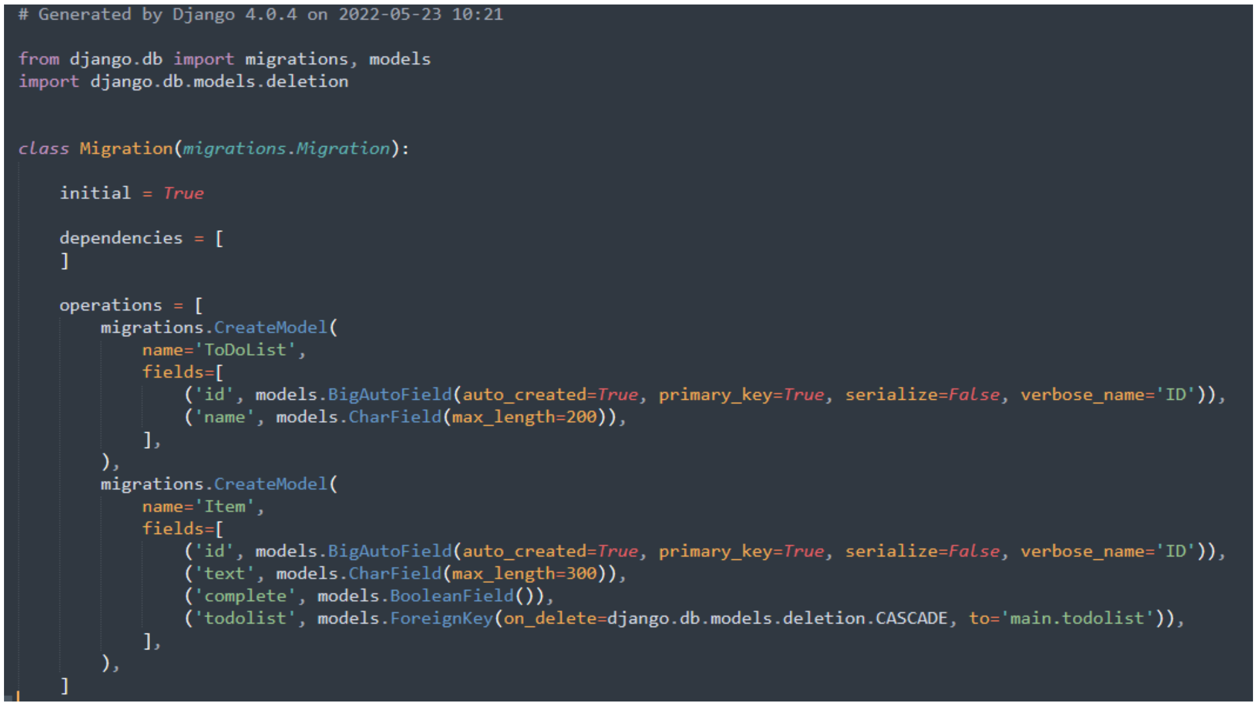
To apply these changes, type:

```
python manage.py migrate
```



```
D:\Documents\ICT3211\Django Tutorial\mysite>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, main, sessions
Running migrations:
  Applying main.0001_initial... OK
```

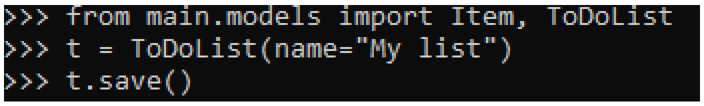
From here, you will notice a file created automatically by Django called “main.0001_intitial”. You can navigate to this file to see the migrations made.



```
# Generated by Django 4.0.4 on 2022-05-23 10:21

from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='ToDoList',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('name', models.CharField(max_length=200)),
            ],
        ),
        migrations.CreateModel(
            name='Item',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('text', models.CharField(max_length=300)),
                ('complete', models.BooleanField()),
                ('todolist', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='main.todolist')),
            ],
        ),
    ]
```

5. Now that we got the creation of models out of our way, the next step is to create our To Do List. For this, we will summon a shell.

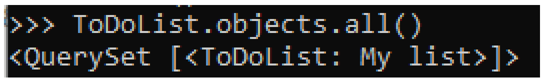
```
python manage.py shell
```



```
>>> from main.models import Item, ToDoList
>>> t = ToDoList(name="My list")
>>> t.save()
```

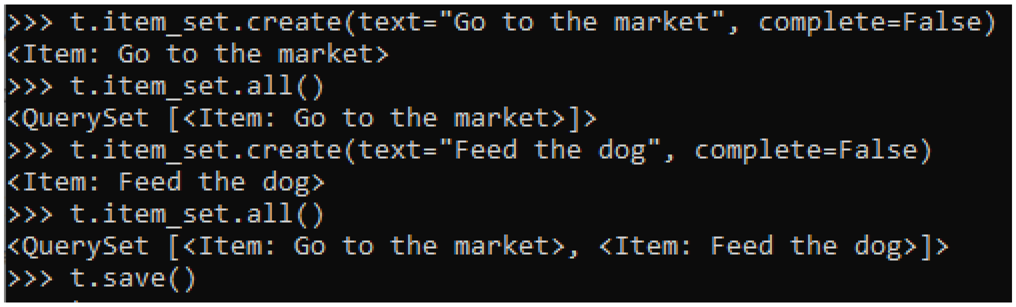
```
from main.models import Item, ToDoList // import our Item & ToDoList from main.models
t = ToDoList(name="My List") // create a ToDoList object called My List
t.save() // save it to db
```

6. Verify that you have created your object correctly.



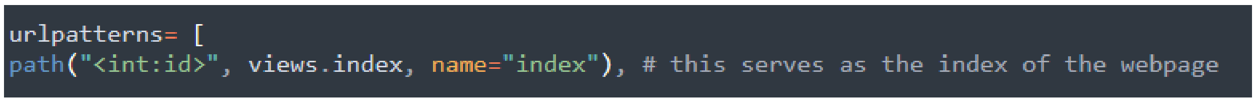
```
>>> ToDoList.objects.all()
<QuerySet [<ToDoList: My list>]>
```

7. Create an item inside of the object, following the format we have created previously in Step 3. Finally, remember to save.



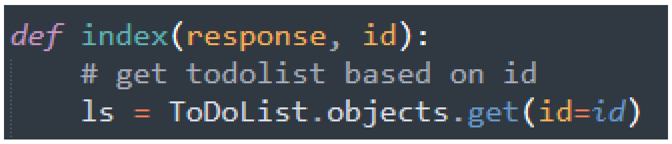
```
>>> t.item_set.create(text="Go to the market", complete=False)
<Item: Go to the market>
>>> t.item_set.all()
<QuerySet [<Item: Go to the market>]>
>>> t.item_set.create(text="Feed the dog", complete=False)
<Item: Feed the dog>
>>> t.item_set.all()
<QuerySet [<Item: Go to the market>, <Item: Feed the dog>]>
>>> t.save()
```

8. Next, we will attempt to display some of our items in the views. In urls.py, modify the path so that when you type an integer in the url (e.g. http://127.0.0.1:8000/1), it will display the appropriate list.



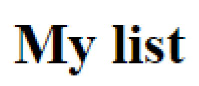
```
urlpatterns= [
path("<int:id>", views.index, name="index"), # this serves as the index of the webpage
```

In the views.py file, modify the index function so that it looks like this:



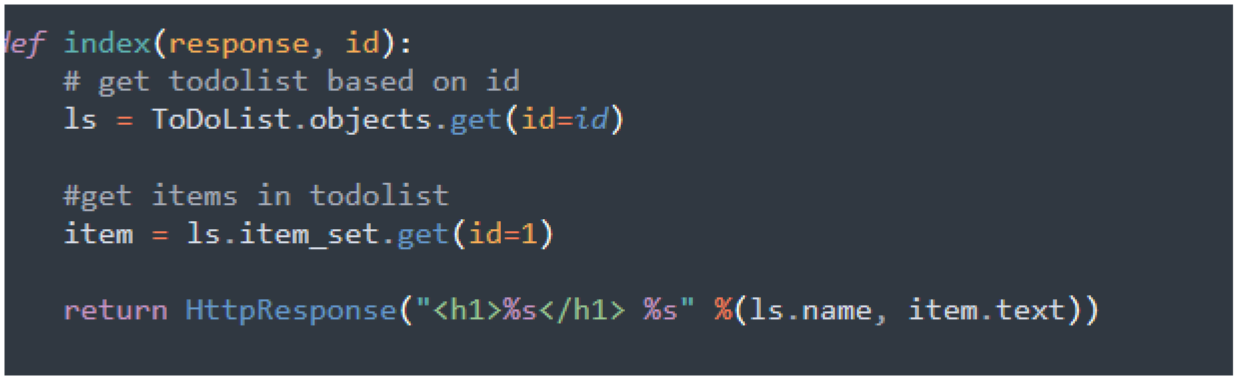
```
def index(response, id):
    # get todolist based on id
    ls = ToDoList.objects.get(id=id)
```

This will get the appropriate list based on their id. In my case, “My list” was id 1. So, if I were to type http://127.0.0.1:8000/1 in the search bar, it would display the relevant list associated with the id.



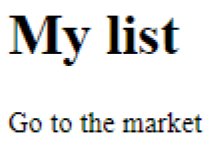
My list

9. In views.py, modify the function so that it looks like this:



```
ef index(response, id):
    # get todolist based on id
    ls = ToDoList.objects.get(id=id)

    #get items in todolist
    item = ls.item_set.get(id=1)

    return HttpResponse("<h1>%s</h1> %s" %(ls.name, item.text))
```

The code will retrieve the items based on the id. In my case, item id=1 is “Go to the market”.



My list

Go to the market