

Django tutorial

Contents

1	Prerequisite	2
2	Installation	2
3	Project creation	3
4	Launch the webpage	4
5	Let's create an Application	5
6	Django_neomodel	8
7	Creating a simple model	11

1 Prerequisite

1. Python ≥ 3.5
2. Pip ≥ 3
3. Neo4j ≥ 4.2

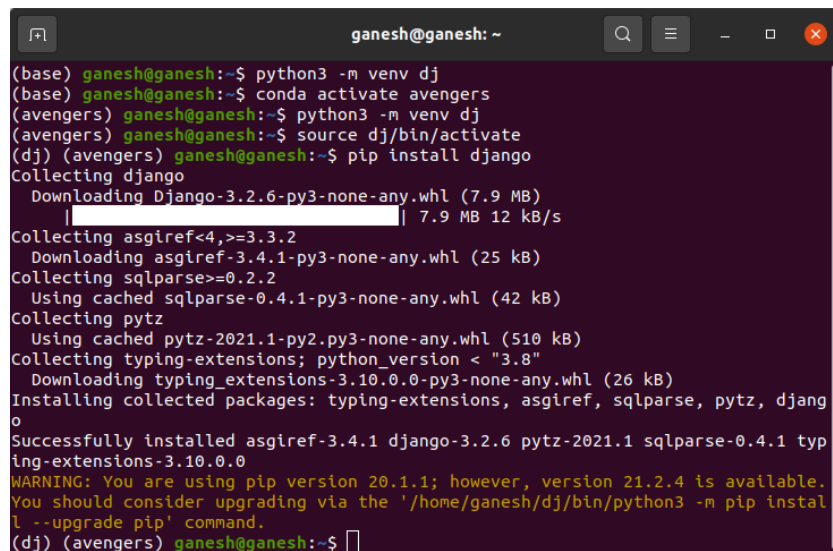
2 Installation

- Its better to create a virtual environment before installing django.
- Create the virtual environment with the following command

```
$ python3 -m venv env  
$ source env/bin/activate
```

- Virtual environment env (in the screenshot it has named as dj) is created and activated, now install django as shown in Figure 1

```
$ pip install django
```



```
ganesh@ganesh: ~  
(base) ganesh@ganesh:~$ python3 -m venv dj  
(base) ganesh@ganesh:~$ conda activate avengers  
(avengers) ganesh@ganesh:~$ python3 -m venv dj  
(avengers) ganesh@ganesh:~$ source dj/bin/activate  
(dj) (avengers) ganesh@ganesh:~$ pip install django  
Collecting django  
  Downloading Django-3.2.6-py3-none-any.whl (7.9 MB)  
    | 7.9 MB 12 kB/s  
Collecting asgiref<4,>=3.3.2  
  Downloading asgiref-3.4.1-py3-none-any.whl (25 kB)  
Collecting sqlparse>=0.2.2  
  Using cached sqlparse-0.4.1-py3-none-any.whl (42 kB)  
Collecting pytz  
  Using cached pytz-2021.1-py2.py3-none-any.whl (510 kB)  
Collecting typing-extensions; python_version < "3.8"  
  Downloading typing_extensions-3.10.0.0-py3-none-any.whl (26 kB)  
Installing collected packages: typing-extensions, asgiref, sqlparse, pytz, django  
Successfully installed asgiref-3.4.1 django-3.2.6 pytz-2021.1 sqlparse-0.4.1 typ  
ing-extensions-3.10.0.0  
WARNING: You are using pip version 20.1.1; however, version 21.2.4 is available.  
You should consider upgrading via the '/home/ganesh/dj/bin/python3 -m pip instal  
l --upgrade pip' command.  
(dj) (avengers) ganesh@ganesh:~$
```

Figure 1: *Django installation using pip*

- To verify the installation type the following command this should display the version of the django

```
$ python -m django --version
```

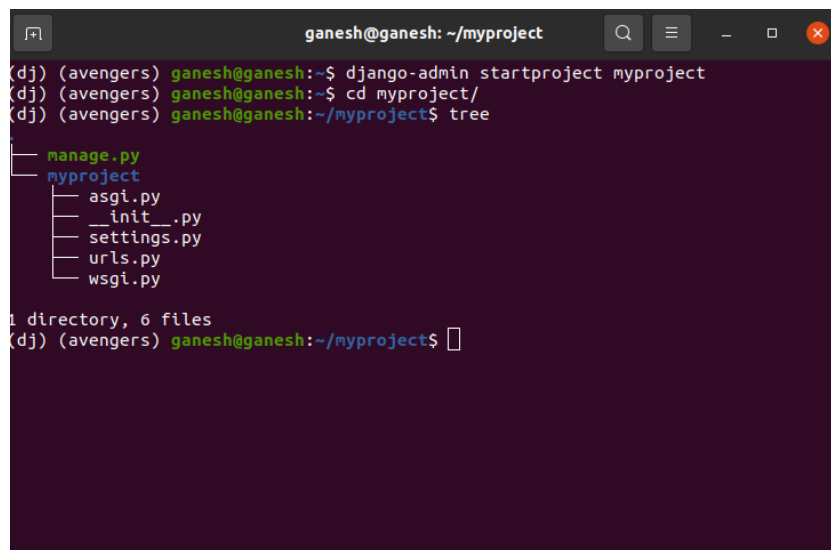
3 Project creation

- Create a project in django using the following command

```
$ django-admin startproject myproject
```

- After executing the command a folder is created that can be navigated and visualized using tree (should have been installed [not necessary]). Refer Figure 2

```
$ cd myproject  
$ tree
```



A terminal window titled 'ganesh@ganesh: ~/myproject' showing the steps to create a Django project. The commands executed are: `django-admin startproject myproject`, `cd myproject/`, and `tree`. The output of `tree` shows a directory structure with `manage.py` at the root and a subdirectory `myproject` containing `asgi.py`, `__init__.py`, `settings.py`, `urls.py`, and `wsgi.py`. The terminal also shows the prompt `(dj) (avengers) ganesh@ganesh:~/myproject$`.

Figure 2: *Django project creation*

- The folder contains `manage.py` and another folder of the project name itself.

- **Manage.py** used to execute the command line commands which is unedited majorly.
- The myproject folder consist of 5 files namely asgi.py, __init__.py, settings.py, urls.py, wsgi.py.
- **asgi.py** supply an application callable which the application server uses to communicate with your code
- **__init__.py** is an empty file that tells the python that the project is a python package.
- **settings.py** used to do configuration for the project that contains security key, debug mode, database settings.
- **urls.py** used to map certain urls with the project that consists of default url to the admin.
- **wsgi.py** is python web application where the web-server communicate.

4 Launch the webpage

- To launch the webpage run the following command as shown in Figure 3

```
$ python3 manage.py runserver
```

- Do not terminate the command, the server should run to visualize the webpage in the browser.
- Open the browser and paste the url "http://127.0.0.1:8000/" to open the webpage.
- The initial webpage looks as presented in Figure 4.

```
ganesh@ganesh: ~/myproject
(dj) (avengers) ganesh@ganesh:~/myproject$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you
apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
August 13, 2021 - 00:48:39
Django version 3.2.6, using settings 'myproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
[13/Aug/2021 00:48:43] "GET / HTTP/1.1" 200 10697
[13/Aug/2021 00:48:43] "GET /static/admin/css/fonts.css HTTP/1.1" 200 423
[13/Aug/2021 00:48:43] "GET /static/admin/fonts/Roboto-Light-webfont.woff HTTP/1.1" 200 85692
[13/Aug/2021 00:48:43] "GET /static/admin/fonts/Roboto-Regular-webfont.woff HTTP/1.1" 200 85876
[13/Aug/2021 00:48:43] "GET /static/admin/fonts/Roboto-Bold-webfont.woff HTTP/1.1" 200 86184
Not Found: /favicon.ico
[13/Aug/2021 00:48:44] "GET /favicon.ico HTTP/1.1" 404 2113
```

Figure 3: *Launching the webpage*

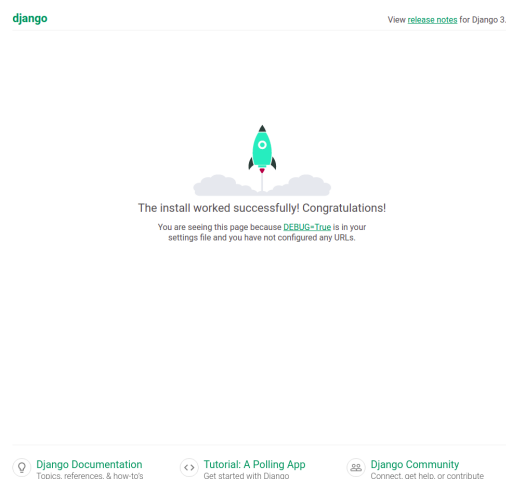
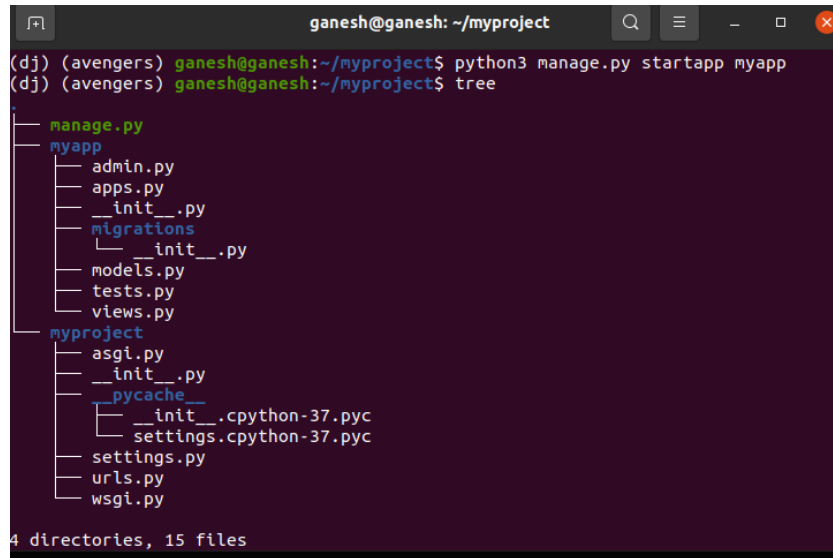


Figure 4: *Viewing the webpage*

5 Let's create an Application

- To create application the following command is used (myapp is the name for the application and the preceding is the command to create the app)

```
$ python3 manage.py startapp myapp
```



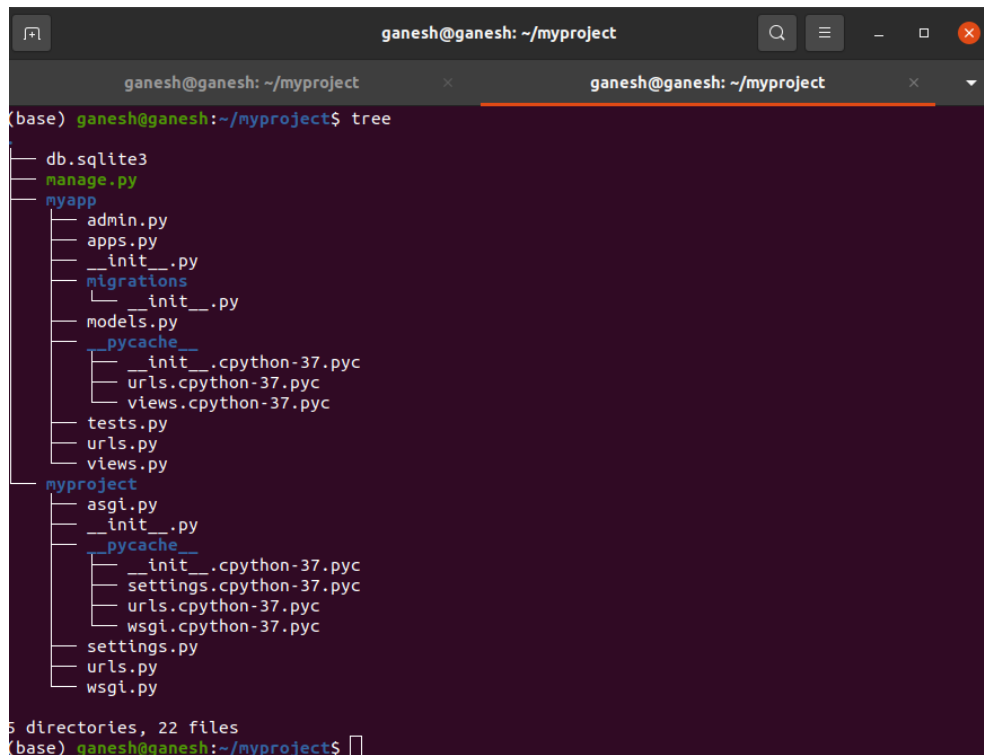
```
ganesh@ganesh: ~/myproject
(dj) (avengers) ganesh@ganesh:~/myproject$ python3 manage.py startapp myapp
(dj) (avengers) ganesh@ganesh:~/myproject$ tree
.
├── manage.py
├── myapp
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
└── myproject
    ├── asgi.py
    ├── __init__.py
    ├── pycache
    │   ├── __init__.cpython-37.pyc
    │   └── settings.cpython-37.pyc
    ├── settings.py
    ├── urls.py
    └── wsgi.py
4 directories, 15 files
```

Figure 5: *Creation of the app*

- Figure 5 shows the tree structure after creating the app where the app is shown along with the predefined created files.
- The application is to create a page that displays a heading of the page. To do this open the project in Vs-code or other code editor.
 - Open the views.py from the myapp directory. Type the following

```
1 from django.shortcuts import render
2 from django.http import HttpResponse
3
4 def home(request):
5     return HttpResponse('<h1> Myapp Home </h1>')
6
7 def about(request):
8     return HttpResponse('<h1> Myapp About </h1>')
```

- The above program consists of the content that has to be displayed in the page.
- Now create a urls.py under myapp (Figure 6) directory similar to the one in myproject (Figure 5)



```
ganesh@ganesh: ~/myproject
(base) ganesh@ganesh:~/myproject$ tree
.
├── db.sqlite3
├── manage.py
├── myapp
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── __pycache__
│   │   ├── __init__.cpython-37.pyc
│   │   ├── urls.cpython-37.pyc
│   │   └── views.cpython-37.pyc
│   ├── tests.py
│   ├── urls.py
│   └── views.py
└── myproject
    ├── asgi.py
    ├── __init__.py
    ├── __pycache__
    │   ├── __init__.cpython-37.pyc
    │   ├── settings.cpython-37.pyc
    │   ├── urls.cpython-37.pyc
    │   └── wsgi.cpython-37.pyc
    ├── settings.py
    ├── urls.py
    └── wsgi.py

5 directories, 22 files
(base) ganesh@ganesh:~/myproject$
```

Figure 6: *Urls.py is created under myapp*

- In the new urls.py type the following

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.home, name='Myapp-Home'),
6     path('about/', views.about, name='Myapp-about'),
7
8 ]
```

- Modify the old urls.py as shown below

```
1 from django.contrib import admin
2 from django.urls import path, include
3
4 urlpatterns = [
5     path('admin/', admin.site.urls),
6     path('Myapp/', include('myapp.urls')),
7 ]
```

- When the server is ran using the below command

```
$ python3 manage.py runserver
```

- It displays an error page having admin and Myapp as written in the old urls.py file. The following URL "http://localhost:8000/Myapp/" should make the page as shown in Figure 7



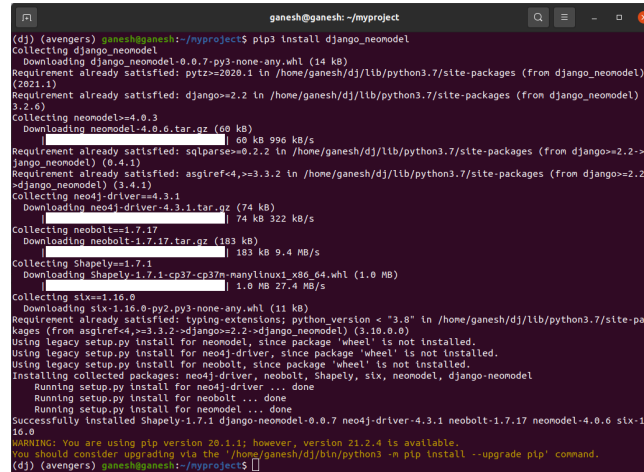
Figure 7: *Output of myapp*

- Therefore once the serve is ran, it looks for the URL to direct since there is no main URL it waits for the trigger either admin/ or Myapp/ should be navigated manually. Once Myapp is included in the URL along with the localhost this redirects to url.py in the app which calls the home function from the view.

6 Django_neomodel

- Install django_neomodel using the following command


```
$ pip3 install django_neomodel
```



```
ganesh@ganesh: ~/myproject
(dj) (avengers) ganesh@ganesh:~/myproject$ pip3 install django_neomodel
Collecting django_neomodel
  Downloading django_neomodel-0.0.7-py3-none-any.whl (14 kB)
Requirement already satisfied: pytz>=2020.1 in /home/ganesh/dj/lib/python3.7/site-packages (from django_neomodel) (2021.1)
Requirement already satisfied: django>=2.2 in /home/ganesh/dj/lib/python3.7/site-packages (from django_neomodel) (3.2.0)
Collecting neomodel>=4.0.3
  Downloading neomodel-4.0.6.tar.gz (60 kB)
    60 kB 996 kB/s
Requirement already satisfied: sqlparse>=0.2.2 in /home/ganesh/dj/lib/python3.7/site-packages (from django>=2.2->django_neomodel) (0.4.1)
Requirement already satisfied: asgiref<4,>=3.3.2 in /home/ganesh/dj/lib/python3.7/site-packages (from django>=2.2->django_neomodel) (3.4.1)
Collecting neo4j-driver==4.3.1
  Downloading neo4j-driver-4.3.1.tar.gz (74 kB)
    74 kB 322 kB/s
Collecting neobolt==1.7.17
  Downloading neobolt-1.7.17.tar.gz (183 kB)
    183 kB 9.4 MB/s
Collecting Shapely==1.7.1
  Downloading Shapely-1.7.1-cp37-cp37m-manylinux1_x86_64.whl (1.0 MB)
    1.0 MB 27.4 MB/s
Collecting six==1.16.0
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Requirement already satisfied: typing_extensions; python_version < "3.8" in /home/ganesh/dj/lib/python3.7/site-packages (from asgiref<4,>=3.3.2->django>=2.2->django_neomodel) (3.10.0.0)
Using legacy setup.py install for neomodel, since package 'wheel' is not installed.
Using legacy setup.py install for neo4j-driver, since package 'wheel' is not installed.
Using legacy setup.py install for neobolt, since package 'wheel' is not installed.
Installing collected packages: neo4j-driver, neobolt, Shapely, six, neomodel, django_neomodel
  Running setup.py install for neo4j-driver ... done
  Running setup.py install for neobolt ... done
  Running setup.py install for neomodel ... done
Successfully installed Shapely-1.7.1 django_neomodel-0.0.7 neo4j-driver-4.3.1 neobolt-1.7.17 neomodel-4.0.6 six-1.16.0
WARNING: You are using pip version 20.1.1; however, version 21.2.4 is available.
You should consider upgrading via the '/home/ganesh/dj/bin/python3 -m pip install --upgrade pip' command.
(dj) (avengers) ganesh@ganesh:~/myproject$
```

Figure 8: *Django_neomodel installation using pip*

- The project and application is already being created. Therefore, need to register our application in myproject/settings.py file so Django can recognize this new application. We must also register django_neomodel and set up the connection to our neo4j database.
- Open myproject/settings.py and add the following
 - Under the application definition in the installed_apps add the following lines

```
1 # django.contrib.auth etc
2 'myapp.apps.MyappConfig',
3 'django_neomodel'
```

```

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-n5x18na0y-6=jg#s!#g3c3omj*^g0!ek-jfy%8y'(#9

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'django.contrib.auth etc',
    'myapp.apps.MyapiConfig',
    'django_neomodel'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

```

Figure 9: *Application definition*

- Under the database add the following lines and comment the existing lines mentioned with sqlite. If "os" is not imported, import it at the start

```

1 NEOMODEL_NEO4J_BOLT_URL = os.environ.get(
2     'NEO4J_BOLT_URL', 'bolt://neo4j:NEO4J@localhost:7687'
3 )
4
5 # you are free to add this configurations
6 NEOMODEL_SIGNALS = True
7 NEOMODEL_FORCE_TIMEZONE = False
8 NEOMODEL_ENCRYPTED_CONNECTION = True
9 NEOMODEL_MAX_POOL_SIZE = 50

```

```

7 # Database
8 # https://docs.djangoproject.com/en/3.2/ref/settings/#databases
9
10 NEOMODEL_NEO4J_BOLT_URL = os.environ.get(
11     'NEO4J_BOLT_URL', 'bolt://neo4j:neo4j@localhost:7687'
12 )
13
14 # you are free to add this configurations
15 NEOMODEL_SIGNALS = True
16 NEOMODEL_FORCE_TIMEZONE = False
17 NEOMODEL_ENCRYPTED_CONNECTION = True
18 NEOMODEL_MAX_POOL_SIZE = 50
19
20 # DATABASES = {
21 #     'default': {
22 #         'ENGINE': 'django.db.backends.sqlite3',
23 #         'NAME': BASE_DIR / 'db.sqlite3',
24 #     }
25 # }
26
27 # Password validation
28 # https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators
29
30 AUTH_PASSWORD_VALIDATORS = [
31     {
32         'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
33     },
34     {
35         'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
36     },
37     {
38         'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
39     },
40 ]

```

Figure 10: *Database*

7 Creating a simple model

- Will have two entities Person and City and two relationships between them(LivesIn and Friend)
- Open myapp/models.py and type the following

```

1 from neomodel import (StructuredNode, StringProperty,
2 IntegerProperty, UniqueIdProperty, RelationshipTo)
3
4 # Create your models here.
5
6
7 class City(StructuredNode):
8     code = StringProperty(unique_index=True, required=True)
9     name = StringProperty(index=True, default="city")
10
11 class Person(StructuredNode):
12     uid = UniqueIdProperty()
13     name = StringProperty(unique_index=True)
14     age = IntegerProperty(index=True, default=0)
15
16 # Relations :
17 city = RelationshipTo(City, 'LIVES_IN')
18 friends = RelationshipTo('Person', 'FRIEND')

```

- To verify that everything works correctly create constraints and indexes by typing the following command

```
$ python manage.py install_labels
```

```
(dj) (avengers) ganesh@ganesh:~/myproject$ python manage.py install_labels
Setting up indexes and constraints...

Found django_neomodel.DjangoNode
! Skipping class django_neomodel.DjangoNode is abstract
Found myapp.models.City
+ Creating unique constraint for code on label City for class myapp.models.City
+ Creating index name on label City for class myapp.models.City
Found myapp.models.Person
+ Creating unique constraint for uid on label Person for class myapp.models.Person
+ Creating unique constraint for name on label Person for class myapp.models.Person
+ Creating index age on label Person for class myapp.models.Person

Finished 3 classes.
(dj) (avengers) ganesh@ganesh:~/myproject$
```

Figure 11: *Creating constraints*

- After creating constraints now its possible to create nodes from django shell
- To activate the shell type the following commands in the terminal

```
$ python manage.py shell
```

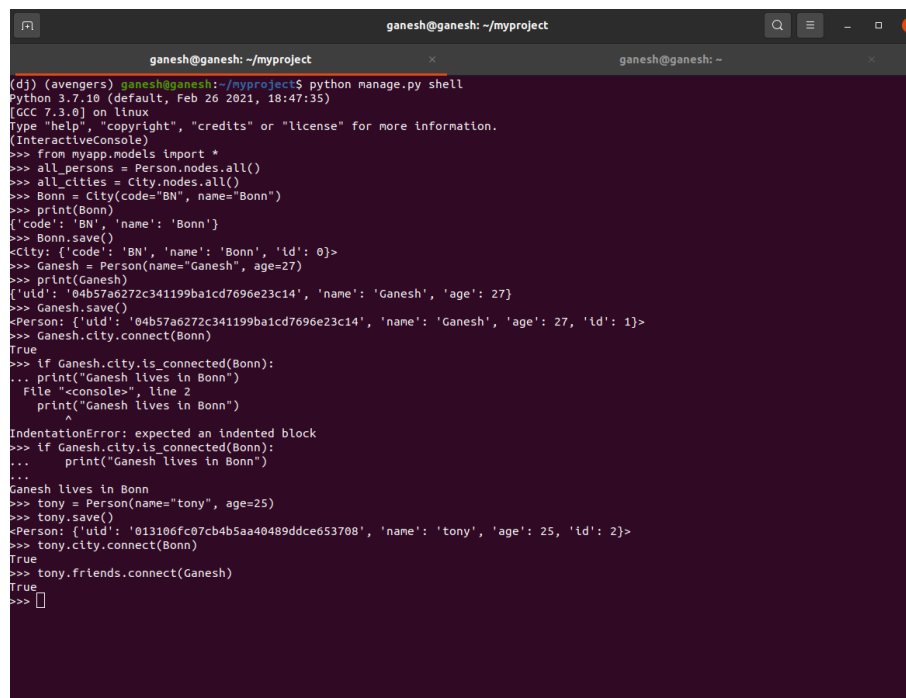
- The above commands open the shell, type the following code line by line as shown in the Figure 12

```
1 from myapp.models import *
2
3 all_persons = Person.nodes.all()
4 all_cities = City.nodes.all()
5 Bonn = City(code="BN", name="Bonn")
6 print(Bonn)
7 Bonn.save()
8 Ganesh = Person(name="Ganesh", age=27)
9 print(Ganesh)
10 Ganesh.save()
```

```

11 Ganesh.city.connect(Bonn)
12 if Ganesh.city.is_connected(Bonn):
13     print("Ganesh lives in Bonn")
14 tony = Person(name="tony", age=25)
15 tony.save()
16 tony.city.connect(Bonn)
17 tony.friends.connect(Ganesh)

```



```

ganesh@ganesh: ~/myproject
(dj) (avengers) ganesh@ganesh:~/myproject$ python manage.py shell
Python 3.7.10 (default, Feb 26 2021, 18:47:35)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from myapp.models import *
>>> all_persons = Person.nodes.all()
>>> all_cities = City.nodes.all()
>>> Bonn = City(code="BN", name="Bonn")
>>> print(Bonn)
{'code': 'BN', 'name': 'Bonn'}
>>> Bonn.save()
<City: {'code': 'BN', 'name': 'Bonn', 'id': 0}>
>>> Ganesh = Person(name="Ganesh", age=27)
>>> print(Ganesh)
{'uid': '04b57a6272c341199ba1cd7696e23c14', 'name': 'Ganesh', 'age': 27}
>>> Ganesh.save()
<Person: {'uid': '04b57a6272c341199ba1cd7696e23c14', 'name': 'Ganesh', 'age': 27, 'id': 1}>
>>> Ganesh.city.connect(Bonn)
True
>>> if Ganesh.city.is_connected(Bonn):
...     print("Ganesh lives in Bonn")
...     File "<console>", line 2
...         print("Ganesh lives in Bonn")
...         ^
IndentationError: expected an indented block
>>> if Ganesh.city.is_connected(Bonn):
...     print("Ganesh lives in Bonn")
...
Ganesh lives in Bonn
>>> tony = Person(name="tony", age=25)
>>> tony.save()
<Person: {'uid': '013106fc07cb4b5aa40489ddce653708', 'name': 'tony', 'age': 25, 'id': 2}>
>>> tony.city.connect(Bonn)
True
>>> tony.friends.connect(Ganesh)
True
>>> 

```

Figure 12: *Creating nodes from django shell*

- In parallel the nodes are created in the Neo4j database that has been linked to django that is shown in the below Figure 13

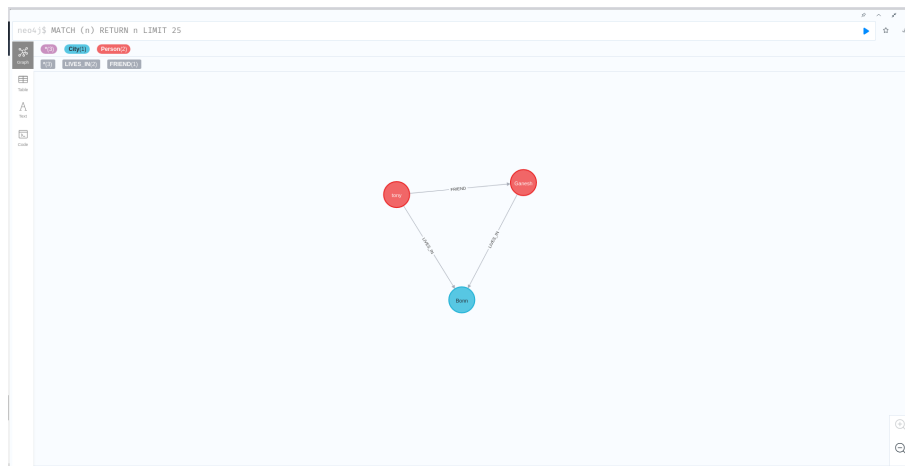


Figure 13: *Created nodes in Neo4j*