1) first things first we need to create the environment
   do what i say no question:
   run this command in the terminal

   => pip3 install virtualenv
   => virtualenv -p python3 venv

   now activate the environment

   => source venv/bin/activate

   you will see the environment is activated in the shell

2) install the Packages

   => pip3 install django==2.0
   => pip3 install djangorestframework
   => pip3 install django-seed
   => pip3 install django-filter
   => pip3 install django-markdown

3) this is very important step

   => pip3 freeze > requirement.txt

   [it will save the metadata of all the installed packages in this file for future use]

4) create the project
   =>django-admin startproject BLOG

   Go to the directory and run the server

   => cd BLOG
   => python3 manage.py runserver

5) create the app inside the project

   => python3 manage.py startapp blog

6) add the blog app to the projects INSTALLED_APPS

7) migrate the app
   => python3 manage.py migrate

8) Create the admin user for the app

   => python3 manage.py createsuperuser

9) we create a model for the blog
   go to the models.py in the app
   two table

1) Author table
2) Article table

and it will be connected with foreign key

10) now make the migration and migrate

=> python3 manage.py makemigrations
=> python3 manage.py migrate

12) django seeding

this part is difficult we need to change a core library

go to the venv/lib/python3.6/site-packages/django-seed/__init__.py
open the file and go to

function faker function in line 35

change the line from

=> cls.fakers[code].seed(random.randint(1, 10000))

to

=> cls.fakers[code].seed_instance(random.randint(1, 10000))

now type this command in the app folder
=> python3 manage.py shell

```
Python 3.6.9 (default, Nov  7 2019, 10:44:02)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from blog.models import Author,Article
>>> from django_seed import Seed
>>> seeder = Seed.seeder()
>>> seeder.add_entity(Author,100,{'name': lambda x:seeder.faker.name(),'email':lambda
x:seeder.faker.email()})
>>> seeder.add_entity(Article,200)
>>> seeder.execute()
```

no we got the the model and the data

13)

so we see we got all the model working and the data is generated
now we create the url pattern

look this pattern

there will be two "urls.py"

1) in the Project BLOG
2) in the app blog [you have to create it]

your project URL will tells you which app you have to go
and your app URL tell you which logic you have to go

if the project URL is 'api'
and the app url is 'articles'
then your total URL will be

   => localhost:8000/api/articles

     1) go to the urls.py inside the BLOG project

       you will see it tels you to include blog.urls
       that means urls.py inside the blog app
       but there is no urls.py inside a blog
       so we create a urls.py inside the blog

     2) now dont add anything leave it blank
       because we need to create a logic
       then we add the logic with the app url

14) now we create one logic .remember all the logic is written in the
   views.py file

     1) first we add another package 'rest_framework' in the
       project settings.py
     2) now we use the API class from the rest framework
       go to the views.py inside the blog app
       and wite a get method

15) now its time to add the logic with the app url
   so go to the urls.py inside the blog app which was empty

and connect to the logic

go to the blog/urls.py

1) we import the class from views.py and add a route

ok now everything is done

lets run the server and go to the url

localhost:8000/api/articles

and you will see AN ERRRRRRORRRR!!!!!!!

but lets see what is the error

it says the Article is not serializable

dont worry we are close

16) serializer is the most common problem you face in any
api development in any framework

it means the data you fetch from the database
cant be converted to JSON

so lets serialize it

1) make a new file name "serializers.py" inside the blog
and go to that file

2) inside the file we make another serialize schema
using the serializer class
its just like a databse schema

3) now go to the "views.py" and import the serializer
and return the serialize data inside the get method

4) run the server again and go to the url

localhost:8000/api/articles

YESSS !! we got it

4) now make another get request that it for fetching single
post with help of primary key pk

after adding in the views.py

add the corresponding url and this time we pass a parameter

that is the primary key in urls.py inside the blog app

17) now we make a post request
   lets go to the views.py

   this post request will take the posted data and save it

   suppose we decided to post this JSON data

   {
       "article":{
           "title": "tanvir",
           "description" : "tanvir is awosme",
           "body" : "this is post API",
           "author_id":1

       }
   }

   so this JSON data is under the key "article"

   just keep that in mind we need this information

18) you need to understand for every entry
   you need to add

   1) the serializer.py
   2) and the views.py
   3) urls.py in the blog app

   1) add the "create" method in the serializer.py
      and the name have to be "create" otherwise it will not work
      this will change the json to databae objects and save it

   2) go to the views.py and write a post request

   3) now go to urls.py and add a link for it

      remember same path can be used for different HTTP operation
      we use the second path for the POST

   4) now comment out the second get in views.py and its corresponding url. i will explain to you
why later.just do it

   ######

   check the serializer .add another field

   author_id = serializers.IntegerField()

####

5) now open postman and try to post that JSON

19) now we add the put method
   its a little bit difficult because
   1) first we have to fetch the article
   2) then we have to replace the value with the new one
   30 then we have to repost it

   we change this json
   {
      "article":{
         "title": "tanvir",
         "description" : "tanvir is awosme",
         "body" : "this is post API",
         "author_id":1

      }
   }

   to

   {
      "article":{
         "title": "tanvir rahman ornob",
         "description" : "tanvir is awosme and this is edited",
         "body" : "this is a edited post API",
         "author_id":1

      }
   }

   1) make a new method in the serializer.py
      we take the obeject and then change its attribute with the new data
      we use the update() method for that in the serializer

   2) now we add the put method in the views.py

   3) now add the url

      now uncomment the url that are commented we use this for put method

we use this for put because in a single class
you cant use similer method
you have to use in another class
so we use this as a put method and
for fetching single post we use another class later
so we use the url

path('articles/<int:pk>',Articleview.as_view())

for put request

20) last method for delete post
   you know what to do

   but this one is simple no need to add method
   in the serializer
   just add a method delete()
   take the object and delete it

21) now we add another class
   and add the get method for single post fetching