**Django introduction**

v-02

install python and local environment setup

v-03 (virtual environment)

*Declaring environment*

python –m venv myenv(windows)

source myenv/bin/activate(linux)

*Activate it :*

myenv\Scripts\activate

then just install it :   
pip install Django

v-04 (creating project)

first create a folder:

mkdir tutorial

cd tutorial

python -m venv myenv

myenv\Scripts\activate

now install:  
pip install Django

create project:

django-admin startproject first .

python manage.py runserver

v-05 (pycharm trick and installation)

first install pycharm and it will automatically add the virtual environment into it.

After that run the manage.py

It will show inside of the pycharm and then just run by F10

Then configure and set parameter : runserver then save and run

It will run automatically : python manage.py runserver

v-06 (files overview and app)

creating an app command:

django-admin startapp demo

v-07 (migrations)

py manage.py migrate

then we gonna make model in our application:

models.py

set the app to the settings.py file

run: py manage.py makemigrations

now again.: py manage.py migrate (now it ill add inside of our database)

so we have created and learned how to use .gitignore file and just ignore vitual environment to push in git repo:   
open the file and write the name of virtual env:

**myenv/**

That’s it as simple as that

v-08 (user and admin)

so in this video I learned :   
how to create a super user:

py manage.py createsuperuser

and to show the models inside of database:

just go inside of admin.py file:

from .models import Book  
  
# Register your models here.  
admin.site.register(Book)

then just indicate the location and model inside of admin   
after save and reloading all the models will appear here.

v-09 (field option)

Here discussed about how work field options like null, blank, choices , unique,

# STATUSES = (  
# (0, 'Unknown'),  
# (1, 'processed'),  
# (2, 'paid')  
# )  
  
# title = models.CharField(null=True, blank=False, unique=True,  
# default='', choices=STATUSES)

Then just command:

py manage.py makemigrations

finally:

py manage.py migrate

v-10 (field type)

CharField()

TextField()

Blank= True

title = models.CharField(max\_length=36, blank=False, unique=True)  
description = models.TextField(max\_length=256, blank=True)  
  
# price = models.FloatField(default=0, max\_digits=0, decimal\_places=2)  
price = models.DecimalField(default=0, decimal\_places=2)  
# price = models.BigIntegerField(default=0)  
  
# published = models.DateField(auto\_now=True, auto\_now\_add=True)  
# published = models.TimeField(auto\_now=True, auto\_now\_add=True)  
# published = models.DateTimeField(auto\_now=True, auto\_now\_add=True)  
published = models.DateField()  
is\_published = models.BooleanField(default=False)  
  
cover = models.FileField(upload\_to='covers')  
# cover = models.ImageField(upload\_to='covers', height\_field='')

saw all the types of field such as Decimal, Date, etc

then:   
py manage.py makemigrations

py manage.py migrate

v-11 (urls)

so basically the I am changing the urls from project first/urls.py

from django.contrib import admin  
from django.urls import path, include  
  
urlpatterns = [  
 path('demo/', include('demo.urls')),  
 path('admin/', admin.site.urls),  
]

then changing demo/urls.py

from django.urls import path  
from . import views  
  
urlpatterns = {  
 path('', views.first),  
}

finally changing the view file:

from django.shortcuts import render  
from django.http import HttpResponse  
# Create your views here.  
def first(request):  
 return HttpResponse('First message!')

v-12 (class based view)

so class based view we use for so many functionality builtin here and easy to work.

To work with class view you have to go views.py:

Then import

from django.views import View

after that write the class based view:

class Another(View):  
 def get(self, request):  
 return HttpResponse('another method in class!')

then edit the demo/urls.py:

from .views import Another

urlpatterns = {  
 path('first', views.first),  
 path('another', Another.as\_view()),  
  
}

v-13 (Model objects method)