Part 1

- **<u>Create a project in the current directory.</u>**
    - **django-admin startproject** <projectname>

- Inside your <projectname> folder:
    - **manage.py**: command line utility that lets you interact with Django projects in various ways.
    - **Inner <projectname>/ directory:** is the actual Python package for your project.
    - **<project>/__init__.py:** an empty file that tells python that is a directory should be considered a Python package.
    - **<project>/settings.py:** settings/configuration for this Django project.
    - **<project>/urls.py**: The URLs declarations for this Django project, a 'table of contents' of your site.
    - **<project>/wsgi.py**: an entry point for WSGI-compatible web servers to serve your project.

- **<u>Development Server:</u>**
    - cd into <projectname> directory.
    - Run **python manage.py runserver**
    - Should see:

```
Performing system checks...

System check identified no issues (0 silenced).

You have unapplied migrations; your app may not work
properly until they are applied.
Run 'python manage.py migrate' to apply them.

February 01, 2021 - 15:50:53
Django version 2.2, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

    - Extra:
        - Change port: `python manage.py runserver 8080`
        - Change servers IP: `python manage.py runserver 0:8000`

- **<u>Creating app:</u>**
    - Project vs Apps:

      What's the difference between a project and an app? An app is a Web application that does something – e.g., a Weblog system, a database of public records or a simple poll app. A project is a collection of configuration and apps for a particular website. A project can contain multiple apps. An app can be in multiple projects.
    - Make sure you're in the same directory as 'manage.py'.
    - Run **python manage.py startapp <appname>**.

- This will create a <appname> directory that will house the application.

- **Write your first View:**
  - Open ***<app>/views.py*** and insert the code:

```
from django.http import HttpResponse


def index(request):
    return HttpResponse("Hello, world. You're at the
polls index.")
```

  - To call this we need to map it to a URL- we need a **URLconf**.
  - **Create URLconf:**
    - Inside <app> directory.
    - Create file called **urls.py** and insert the code:

```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index, name='index'),

]
```

    - Next, point the root URLconf at the **<app>.urls** module.
    - Go into **<project>/urls.py**, add an import for **django.urls.include** and insert an *include()* in urlpatterns list.

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('polls/', include('polls.urls')),
    path('admin/', admin.site.urls),
]
```

    - Verify it works with **python manage.py runserver**.
    - Open **http://localhost:8000/<project>/**

Part 2

- **Database setup.**
  - Open **<project>/settings.py**, by default it uses SQLite.
  - If you wish to change database:
    - **ENGINE**: ‘django.db.backends.<database name> (eg .sqlite3, .mysql)
    - **NAME**: name of your database, in SQLite the database is a file on your computer, in this case, the **NAME** should be the absolute path of this file.

- ## Migrate:
    - In **<project>/settings.py,** INSTALLED_APPs holds the name of all Django applications that are activated in this Django instance.
    - Default applications:
        - `django.contrib.admin` – The admin site. You'll use it shortly.
        - `django.contrib.auth` – An authentication system.
        - `django.contrib.contenttypes` – A framework for content types.
        - `django.contrib.sessions` – A session framework.
        - `django.contrib.messages` – A messaging framework.
        - `django.contrib.staticfiles` – A framework for managing static files.
    - We need to create tables in the database for these applications before we can use them: **python manage.py migrate**.
    - Migrate looks at INSTALLED_APPS and creates any necessary database tables according to database settings in <project>/settings.py.

- ## Creating Models:
    - Models = essentially your database layout, with metadata.
    - In **<project>/models.py** we can create models as python classes.

```
from django.db import models

class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')

class Choice(models.Model):
    question = models.ForeignKey(Question,
on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)

    votes = models.IntegerField(default=0)
```

    - Each model is represented by a class that inherits **django.db.models.Model**. Each class variable is a database field in the model.

- ## Activation Models:
    - This model code gives Django a lot of information. That django can use to:
        - *Create database schema* (CREATE TABLE statements) *for this app.*
        - *Creates a Python database-access API for accessing* ***Question*** *and* ***Choice*** *objects.*
    - But first we must tell our project that <app> is installed.
    - To include the app in our project, we need to add to **INSTALLED_APPS**.
        - <app>Config is in the **<app>/apps.py**:
            - so its path is **"<app>.apps.<app>Config"**
        - Go to **<project>/settings.py** and add this path.

```
INSTALLED_APPS = [
    'polls.apps.PollsConfig',
        'django.contrib.admin',
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.messages',
        'Django.contrib.staticfiles',

    ]
```

- Now django knows to include the <app>.
- Run **python manage.py make**