Files

settings.py

- [ ] **BASE_DIR**
  - [ ] Directory where manage.py exists.
  - [ ] This allows to work relative to directory.
  - [ ] You can try to print BASE_DIR and runserver.
- [ ] **SECRET_KEY**
  - [ ] Should be unique to each project.
  - [ ] Modify few characters if using someone else project.
- [ ] **DEBUG**
  - [ ] Shows details for debugging
  - [ ] Should be changed to False when in production.
- [ ] **ALLOWED_HOSTS**
  - [ ] Allowed domain names and ips.
  - [ ] Used as security measure in production.
- [ ] **INSTALLED_APPS**
  - [ ] Components used in the whole project.
  - [ ] Remember to add all apps you create ans also third-party apps you install in this list.
- [ ] **MIDDLEWARE**
  - [ ] Manages how requests are handled and securities are handled.
- [ ] **ROOT_URLCONF**
  - [ ] Tells django how to manage routes.
- [ ] **TEMPLATES**
  - [ ] How are html templates rendered, where are they stored.
  - [ ] In DIRS list, add os.path.join(BASE_DIR, "templates").
- [ ] **WSGI_APPLICATION**
  - [ ] Tells django how to use servers.
  - [ ] Sometimes we may need to change it.
- [ ] **DATABASES**
  - [ ] Which database engine used and where is database stored.
  - [ ] By default uses sqlite3 database.
    - [ ] **Change database name to create new database.** Eg: change name to db2.sqlite3.
- [ ] **AUTH_PASSWORD_VALIDATORS**
  - [ ] Which password validators are applied.
- [ ] **STATIC_URL**
  - [ ] Talk about later.

models.py

> In docs, arguments given in fields are required arguments. When adding new field, either do null=True or provide some default value(Eg. default="default value").

- [ ] **CharField**
  - [ ] Must have max_length=120 argument.
- [ ] **TextField**

    - [ ] blank=False : Makes field as required while taking input.
    - [ ] null=True : Makes field nullable in database.
- [ ] **DecimalField**
    - [ ] decimal_places=2 is required.
    - [ ] max_digits=1000 is required.
- [ ] **BooleanField**

Commands

manage.py

- [ ] **runserver**
    - [ ] Starts a development server.
    - [ ] You can allow the server to keep running and do all changes in another terminal, including migrations.
- [ ] **makemigrations** and **migrate**
    - [ ] Updates database.
    - [ ] Both commands are run together in sequence.
    - [ ] Run these upon any change in models.py.
    - [ ] To reset database,
        1. Delete all files in migrations folder (except __init__.py)
        2. Delete __pycache__ folder in migrations directory.
        3. Delete db.sqlite3 file.
- [ ] **createsuperuser**
    - [ ] Allows to create a superuser to login into admin page (urls/admin).
- [ ] **startapp appname**
    - [ ] Creates new component.
    - [ ] An app does one thing very good.
    - [ ] You need to add it in INSTALLED_APPS list.
- [ ] **shell**
    - [ ] Allows you to import models and manipulate data to database using the model.
    - [ ] Eg. >>> from products.models import Product >>> Product.objects.all() >>> Product.objects.create(name="Watch", price=22)

views.py

Functional Views

- [ ] Need to add views in urls.py.
- [ ] Takes a request object as argument.
- [ ] Conventionally functions end with _view.
- [ ] Add *args, **kwargs also as arguments in function definitions.
- [ ] Returns either HttpResponse or render(request, template_name, context_dictionary)

request Object

> Request object is also accessible in html templates.

- [ ] .user
    - [ ] Gives username of user logged in.
    - [ ] If no one is logged in, it gives AnonymousUser.
    - [ ] .is_authenticated

urls.py

- [ ] Best practice is to create a urls.py for each app and include it in the main project urls.py.
- [ ] Copy paste main project urls.py to create apps urls.py.
- [ ] Adding urls is given in the starter page.

templates

- [ ] Create a base.html with common headers and other things. Add {% block body %}{% endblock body %} In all other html pages, {% extends 'base.html' %} {% block body %} Then content here will be placed between body block in base.html {% endblock body %}
- [ ] To create components separately, create html documents separately and add {% include 'component.html' %}
- [ ] Context variables can be used inside template with {{ variable }} format.
- [ ] To render a list, use for loop: {% for item in list_of_items %} <li>item</li> {% endfor %}
- [ ] To check for conditions, use {% if variable == "some_value" %} <h3> variable is 'some value'<h3> {% elif variable == "some_other_value" %} <h4>variable is some other value<h4> {% endif %}

> Refer builtin template tags in docs to know about more tags.

- [ ] {% comment "Comment title" %} <tag>Commented text</tag> {% endcomment %}
- [ ] cycle: {% for item in items %} <tr class="{% cycle 'row1' 'row2' %}"></tr>

Filters

- [ ] Filters are used in {{ }} this type of syntax.
- [ ] Filters can be used one on top of other. {{ variable|capfirst|upper }}
- [ ] See docs for builtin filters.
- [ ] Custom filters can be created.
- [ ] Common ones are:
    - [ ] safe : To render text as html (this can be done in view using *mark_safe*).
    - [ ] title : Capitalizes first letter of each word.
    - [ ] striptags : Removes all html tags.
    - [ ] slugify : Replaces spaces with '-'.
    - [ ] add:[number] : Adds a number.