**BRING GLASSES TO NEXT CLASS**

**HW-make flashcards of all the commands (models, etc. on docs.djangoproject.com**

**go to official django documentation (on django website)**

**have info on how to solve problems**

**read/ get familiar with “model field reference”**

**look at charfiled (know how to do this for next class)**

**ask art to add you to #febwepapp (slack channel)**

**LOOK AT DJANGO DOCUMENTATION DURING THE WEEK !!**

**Need to know how to do everything**

**All the model fields**

**Save everything from coding class to hard drive**

**HOMEWORK:**

1. **create folder called “template” within “posts” folder/app**
2. **within folder template create another folder, “posts”**
3. **then create static folder within the posts (on the same level and posts^)**
4. **create 10 posts**

Big O quantitation is a way to measure how fast algorithms are

Chart in presentation

Use python3 whenever teach uses py3

How to start virtual environment – insturctions for conda on slack

^C(env)

py3 manage.py startapp posts

CRUD

Create

Retrieve

Update

Delete

Good practice to keep everything in house (a global URL and local URL file)

Functions within a *class* are called *methods*

**Whenever you name a class it MUST start with a capital letter**

Models – generic class

User= person who creates the post

User, title, slug, image, content, draft, etc = properities of the class

Title is going to contain characters

Have to input all fields

To create a database you have to create the names, go there, pick the right field

Slug acts like a primary key because it has to be unique

You can use slug instead of a primary key

Content is probably going to be a text field

Drafts need Boolean data (boolean= is it true or false)

*from django.db import models*

*# Create your models here.*

*class Post(models.Model):*

*user*

*title = models.CharField(max\_length=120)*

*slug =*

*image*

*content*

*draft*

*updated*

*published*

user is different- uses different table, has to be connected somehow

from django.db import models

from django.conf import settings

# Create your models here.

class Post(models.Model):

user

title = models.CharField(max\_length=120)

slug = models.SlugField(max\_length=120, unique=True)

image = models.ImageField(upload\_to='posts')

content = models.TextField()

draft = models.BooleanField(default=False)

updated = models.DateField(auto\_now=True, auto\_now\_add=False)

published = mdoels.DateField(auto\_now=False, auot\_now\_add=True

class Meta:

ordering = ['-published']

^^^ class within a class

self is a

to create a database with django you need to

1. make migrations
2. migrate

We went to the folder, found settings in project & added it to installed apps

*Stay after class to learn how to use sublime*

*Look @ code online (slack)*

First you do make migrations

Then you do migrate

Posts= name of app

Post= name of class

^don’t forget!^

for taking the admin (model) and trying to add a built in template

posts vs models

we are trying to import some model/ class

so we are saying go to models, where you will find this post

how to create folders

we have posts folders (also an app)

need to create folder within this^ folder called “template”

then create another folder called “posts” within the template folder

^supposed to be referring to the name of the app

\*we are going to create an html template, pages

django is going to grab the html page and render it (show it to us)

its going to be saved in a different location

which is why we have to specify the html file (in posts)

\*so django goes first to template, then to posts

another\_app

then a new template within that app

then create another folder called another\_app within the app

\*then you will have to create “static” folder (must be in posts, on same level as template)

and another folder called post (app)

also have files (view, model)

URLS.py

URL (i.e. 127.00.0)/List\_post/

URLS.py

postsURLS.py (or rg?)

views.py (this renders template (i.e. Lists.html))

in order to do a post you need some kind of command

like create a button, providing some kind of url (for a web app)

then django is going to check url files (will look in global and local urls)

then it finds that url which tells it to go to views and create a new class

something to trigger – something to manage – all right in database