

# CSCI S-33a (Web50)

## Section 2

*Ref: Lectures 3 (Django)*

Vlad Popil

*Jul 1, 2020*

# My Info

## About me:

### Vlad Popil

Master's (ALM) in Software Engineering

Principal Data Analyst at Capital One

Email: [volodymyr.popil@gmail.com](mailto:volodymyr.popil@gmail.com)

Sections: Wed 8:00-9:30 pm EDT + 1st week only on Thu 8:00-9:30 pm

Office Hours: Sat 2:00-3:30pm EDT

# Agenda

- Logistics
- HTTP
- Django
- Anaconda/Miniconda distribution
- Markdown
- Regex
- Chrome developer tools (Network)
- IDEs
- cURL/Postman
- Project 1
- Grading criteria (not exhaustive)
- pycodestyle
- Tips
- Q&A

# Logistics

# Reminders

- Zoom:
  - Use zoom features like raise hand, chat and other
  - Video presence is STRONGLY encouraged
  - Mute your line when not speaking (enable temporary unmute)
- Projects:
  - Start early (or even better RIGHT AWAY!!!)
  - Post questions on Ed platform
  - Remember: bugs can take time to fix
  - Grade  $\rightarrow 3 \times \text{correctness} + 2 \times \text{design} + 1 \times \text{style}$
  - Lateness policy -  $0.01 \text{ per minute} \times 60 \times 24 \times 7 \text{ days} \sim 100$
  - Set a reminder to submit the form for each project
  - Online search, Ed platform, etc.
  - Documentation
  - Project 0 - Due Sunday, Jul 5 at 11:59pm EDT (**4 DAYS LEFT**)

# Reminders

- Sections/Office Hours:
  - Sections are recorded, office hours are not
  - Real-time attendance encouraged
  - Video and participation encouraged even more
- Section prep:
  - Watch lecture
  - Review project requirements
- Office hours prep:
  - Write down your questions as you go, TODO, etc.
  - Come with particular questions

# 10,000 foot overview

- *Section 0 (HTML, CSS)* - Chrome Dev Tools (Inspector), Grading aspects, Overviews
- *Section 1 (Git + Python)* - Python Installation, IDEs, CDT (Network), Project 0
- *Section 2 (Django)* - Markdown, *RegEx*, IDEs extra, pycodestyle, Debugging, Project 1
- *Section 3 (SQL, Models, Migrations)* - cURL/Postman, Models, DB modeling, Project 2
- *Section 4 (JavaScript)* - AJAX, linting, jshint, CDT Debugging, Project 3
- *Section 5 (User Interfaces)* - Animations, DB modeling, Pagination, Project 4
- *Section 6 (Testing, CI/CD)* - Test Driven Development, DevOps, Final Project
- *Section 7 (Scalability and Security)* - Cryptography, CAs, Attacks, App Deployment

Most sections: material review, logistics, project criteria review, reminders, hints, etc.

# Burning Questions?

Please ask questions, or topics to cover today!

Topics:

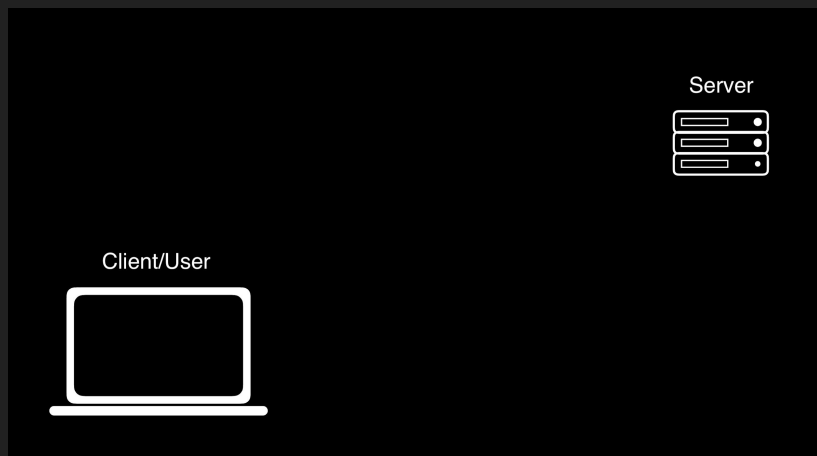
- *Markdown verification*
- *File path ``/Users/Ayush/Desktop/img1.png`` Relative path: ``../img1.png``*
- *Using submit50 and Anaconda navigator. ``pip install submit50``*
- *Feedback for Project 0 (~ Saturday Jul 4th)*
- *Project 1 - css you can change, and appearance doesn't matter much.*
- *`<>` block content - templates*



# Lecture Recap

# HTTP

- Hypertext Transfer Protocol
- Set of standard protocols for how clients and servers interact on the web
- A Web Browser (Chrome, Safari, etc) will get a response and decide what to display.



Status Code	Description
200	OK
301	Moved Permanently
403	Forbidden
404	Not Found
500	Internal Server Error

# HTTP Request (most common)

## 1. GET

- The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.

## 2. POST

- A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.

## 3. PUT

- Replaces all current representations of the target resource with the uploaded content.

## 4. DELETE

- Removes all current representations of the target resource given by a URI.

# Django

Django (/ˈdʒæŋɡoʊ/ JANG-goh; stylised as django) is a Python-based free and open-source web framework.

The framework was named after guitarist Django Reinhardt.

# Django - The D is silent

<https://www.youtube.com/watch?v=ci4g8D5wSww>



# Django Setup

1. `pip3 install django`
2. `django-admin startproject project_name`
  - a. navigate to the new folder (`cd project_name`)
3. Check that it works so far: `python(3) manage.py runserver`
  - a. visit <http://127.0.0.1:8000/> which is your local server
  - b. Ctrl-C exits
4. Make a new app: `python3 manage.py startapp app_name`
5. Navigate to `settings.py` and add `app_name` to `INSTALLED_APPS = [ ... ]`
6. In `urls.py` Add `path("url_extension/", include("app_name.urls"))`
  - a. `import include from django.urls`
7. Create a new file `urls.py` in your app folder
  - a. with a list `urlpatterns = [ path("", views.function_name, name="some_name")]`
  - b. `from . import views` AND `from django.urls import path`
8. Navigate to `views.py` in your app and define a function that takes in a request and returns an HTTP Response or a rendered template.

# Important Django Commands:

Command	What it Does
<code>django-admin startproject [PROJECT_NAME]</code>	Creates a new Django project
<code>python[3] manage.py runserver</code>	Runs the project on a locally hosted server
<code>ctrl-C</code>	Stops running the server
<code>python manage.py startapp [APP_NAME]</code>	Creates a new Django app within a project
<code>python manage.py migrate</code>	Creates a table to store session data
<code>python manage.py createsuperuser</code>	Create admin to login into admin view.

# Organization of Files

PROJECT\_FOLDER

manage.py

PROJECT\_FOLDER

settings.py ... urls.py ... etc.

APP\_1\_FOLDER

views.py ... urls.py ... models.py ... forms.py ... etc.

templates

APP\_1\_FOLDER

html1.html ... etc.

static

APP\_1\_FOLDER

styles.css



# Rendering HTML Templates

```
from django.shortcuts import render
```

```
render(request, "app_name/file_name.html")
```

```
render(request, "app_name/file_name.html", {
```

```
    "var_name_1": value1,
```

```
    "var_name_2": value2
```

← **Context**

```
})
```

# Django Templating Language

- Include programming logic in our HTML files (if/elif/else, for loops, variables)
- Link static pages to our HTML file
- Link to routes within our application
- Extend layout templates to cut down on repeated code
- Documentation [here!](#)

# Using the Templating Language

- Include logic between `{%` and `%}`
- Include variables between `{{` and `}}`
- Link to static files by adding `{% load static %}` to top of page, and then replacing a hard-coded link with `{% static 'app_name/file_name' %}`
- Link to routes using `href="{% url 'route_name' %}"` where `route_name` is the name we assigned a route in `urls.py`

# Extend HTML Template

- In parent.html:

```
...code...
```

```
{% block block_name %}
```

```
{% endblock %}
```

```
...code...
```

- In child.html:

```
{% extends "path_to_parent" %}
```

```
{% block block_name %}
```

```
... HTML to be inserted in block of the parent file...
```

```
{% endblock %}
```

# Forms in Django

- CSRF Verification Required
- Determine request time using `request.method`
- Create forms either in HTML or using the Django Form Class (recommended)
- Example Django Form (in `forms.py`):

```
from django import forms

class NameForm(forms.Form):
    your_name = forms.CharField(label='Your name', max_length=100)
```

# Handling Form Submission and Rendering

```
from django.http import HttpResponseRedirect
from django.shortcuts import render

from .forms import NameForm

def get_name(request):
    # if this is a POST request we need to process the form data
    if request.method == 'POST':
        # create a form instance and populate it with data from the request:
        form = NameForm(request.POST)
        # check whether it's valid:
        if form.is_valid():
            # process the data in form.cleaned_data as required
            # ...
            # redirect to a new URL:
            return HttpResponseRedirect('/thanks/')

    # if a GET (or any other method) we'll create a blank form
    else:
        form = NameForm()

    return render(request, 'name.html', {'form': form})
```

- `NameForm()` creates new Form Object
- `NameForm(request.POST)` creates new Form Object based on submitted info
- `is_valid` checks validity of a form

# Including Form in HTML

```
<form action="/your-name/" method="post">  
  {% csrf_token %}  
  {{ form }}  
  <input type="submit" value="Submit">  
</form>
```

# Django

Demo `budget` ...



Main material

# Anaconda Distribution

- Anaconda - World's Most Popular Python/R Data Science Platform
- Miniconda (lighter version):
  - a. Download <https://docs.conda.io/en/latest/miniconda.html>
  - b. Run in terminal in Downloads: `zsh Miniconda3-latest-MacOSX-x86_64.sh`
  - c. Run `conda init` ONLY if not prompted during installation
  - d. Create new environment:
    - `conda create -n s33a python=3.7`
  - e. See environments:
    - `conda env list`
  - f. Deactivate/Activate environment:
    - `conda deactivate`
    - `conda activate s33a`
  - g. Install more packages:
    - `conda install django` (preferred)
    - `pip install django` (if conda doesn't find), although  
It is better to `python -m pip install django` (to assure proper pip)

# Markdown

- Very readable markup language
- Used to write a README on GitHub, also to write the lecture notes
- Many options for editors:
  - Typoora
  - Visual Studios Code Markdown Extensions
  - MacDown
  - Dillinger
- Easy (comparatively) to convert into many output formats
- Let's write a simple Markdown File!



# Regular Expressions\*

*\* Summer 2020 - not required for Project1*

# Regular Expressions

- A sequence of characters that define a search pattern
- Very efficiently searched for by computers
- Can be expanded to be used for search and replace problems
- [Regex Cheat Sheet](#)
- [Cheat Sheet Specific to Python](#)

# Metacharacters

Metacharacters are characters with a special meaning:

Character	Description	Example	Try it
[]	A set of characters	"[a-m]"	<a href="#">Try it »</a>
\	Signals a special sequence (can also be used to escape special characters)	"\d"	<a href="#">Try it »</a>
.	Any character (except newline character)	"he..o"	<a href="#">Try it »</a>
^	Starts with	"^hello"	<a href="#">Try it »</a>
\$	Ends with	"world\$"	<a href="#">Try it »</a>
*	Zero or more occurrences	"aix*"	<a href="#">Try it »</a>
+	One or more occurrences	"aix+"	<a href="#">Try it »</a>
{}	Exactly the specified number of occurrences	"a1{2}"	<a href="#">Try it »</a>
	Either or	"falls stays"	<a href="#">Try it »</a>
()	Capture and group		

# Special Sequences

A special sequence is a `\` followed by one of the characters in the list below, and has a special meaning:

Character	Description	Example	Try it
<code>\A</code>	Returns a match if the specified characters are at the beginning of the string	<code>"\AThe"</code>	<a href="#">Try it »</a>
<code>\b</code>	Returns a match where the specified characters are at the beginning or at the end of a word	<code>r"\bain"</code> <code>r"ain\b"</code>	<a href="#">Try it »</a> <a href="#">Try it »</a>
<code>\B</code>	Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word	<code>r"\Bain"</code> <code>r"ain\B"</code>	<a href="#">Try it »</a> <a href="#">Try it »</a>
<code>\d</code>	Returns a match where the string contains digits (numbers from 0-9)	<code>"\d"</code>	<a href="#">Try it »</a>
<code>\D</code>	Returns a match where the string DOES NOT contain digits	<code>"\D"</code>	<a href="#">Try it »</a>
<code>\s</code>	Returns a match where the string contains a white space character	<code>"\s"</code>	<a href="#">Try it »</a>
<code>\S</code>	Returns a match where the string DOES NOT contain a white space character	<code>"\S"</code>	<a href="#">Try it »</a>
<code>\w</code>	Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore <code>_</code> character)	<code>"\w"</code>	<a href="#">Try it »</a>
<code>\W</code>	Returns a match where the string DOES NOT contain any word characters	<code>"\W"</code>	<a href="#">Try it »</a>
<code>\Z</code>	Returns a match if the specified characters are at the end of the string	<code>"Spain\Z"</code>	<a href="#">Try it »</a>

# IDEs and Debugging



# Chrome Developer Tools (Network)

In Chrome:

1. Right click
2. Inspect
3. → Demo

Extremely powerful! Let's try...

# cURL / Postman

Allows to call API endpoints directly.

Demo...

# Integrated Development Environments (Intro)

- Text Editor or Heavy IDE?
- Options:
  - VS Code
  - PyCharm (Pro)
  - Atom
  - Sublime
  - vim/Emacs
  - And dozens more, including Notepad :)
- My suggestion: VS Code or PyCharm
- Benefits: Debugging, Autocomplete, Navigation, Find Usages, Linting, Refactoring, Running App and much more.

# VS Code

- Demo

# PyCharm

- Demo

# pycodestyle (formerly pep8)

- `pip install pycodestyle`
- `pycodestyle app.py --max-line-length=120`

# Project 1

- Start early!!!
- Make a checklist of requirement and check all before submission
- Make sure there's no bugs
- Google Form
- CaSe InSeNsItIvE SeArCh (try?)
- `random` has a function to choose from list without prior indexing (try?)
- Use { \_\_\_\_ | safe } to render
- `return HttpResponseRedirect(reverse("entry", args=(title,)))`
- How to store the new page? utils.
- How to get a page? utils.
- How to get all pages? utils.
- `markdown2.markdown()`

# Grading criteria generic suggestions (not limited to)

- Correctness:
  - All requirements + no bugs
- Design (not limited to):
  - Responsive
  - Simplest solution
  - Avoiding repetition (refactoring)
  - Structure (e.g separate files vs inline styling)
- Style (not limited to):
  - File structure
  - Line breaks
  - Spacing
  - Naming
  - **Comments**

Both Design and Style consider readability but from different perspective.



# Random Tips

- HTML beautifiers/prettify
- Windows licence (<https://harvard.onthehub.com/>)
- Video Speed Controller
- The Great Suspender
- GitHub Education Pack
- Spotify

# Fruit of the day

<<< If you are watching this recorded >>

Please email the word: **PLUM**

To: [volodymyr.popil@gmail.com](mailto:volodymyr.popil@gmail.com)

Thank you.

# Q&A

Please ask any questions. Ideas:

- Anything discussed today
- Anything from lecture material
- About the project
- Logistics
- *Random*

# Resources

- <https://github.com/vpopil/e33a-sections-summer-2020>

# CSCI S-33a (Web50)

## Section 2

*Ref: Lectures 3 (Django)*

Vlad Popil

*Jul 1, 2020*