CSCI E-33a (Web50)
Section 3

Ref: Lecture 3 (Django)

Vlad Popil

Feb 8, 2022

About me

Vlad Popil

- Master's (ALM) in Software Engineering
- Software Engineer at Google

Email: vlad@cs50.harvard.edu

Sections: Tue 8:30-10:00 pm ET
Office Hours: Thu 9:00-10:30 pm ET

Agenda

- Logistics
- HTTP
- Django
- Markdown
- Regex
- Project 1
- Anaconda/Miniconda and/or venv distribution
- Chrome developer tools (Network again)
- IDEs
- Grading criteria (not exhaustive)
- Pycodestyle (module)
- Tips
- Q&A

Logistics

Intro

- Refer to website: https://cs50.harvard.edu/extension/web/2022/spring/
- Sections and office hours schedule on website sections
- Get comfortable with command line
- Text editor is usually sufficient to write code, BUT IDEs is faster!
- Zoom:
  - Use zoom features like raise hand, chat and other
  - Video presence is STRONGLY encouraged
  - Mute your line when not speaking (enable temporary unmute
- 6 Projects
  - Start early (or even better RIGHT AWAY!!!)
  - Post **and answer** questions on Ed platform
  - Remember: bugs can take time to fix
  - Grade -> 3 × Correctness (5/5) + 2 × Design [code] (5/5) + 1 × Style [code] (5/5) (Project 0 is an exception)
    E.g. 15+10+5=30/30 | e.g. Correctness can be 15, 12, 9, 6, 3, 0
  - Lateness policy - 0.1per minute => **16hrs 40 min**, plus one time 3-day extension
  - Set a reminder to submit the Google Form for each project
  - Project 1 - Due Sunday, Feb 20th at 11:59pm ET **<< ONLY 12 FULL DAYS LEFT >>**

Reminders

- Sections/Office Hours:
  - Sections are recorded (published 72hrs), office hours are not
  - Real-time attendance is required of at least one section
  - 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

Are sections a good use of my time?

- First section and Project 0 may seem a bit introductory, but beware!!!...
- Tons of tips and hints that ~~can~~ **will** save hours or even days
- Debugging approaches not covered in lectures
- Supplemental material which complement lectures well
- And more...

YES!!!

10,000 foot overview

- *Section 0 - SKIPPED*
- *Section 1+2 (Git + Python)* - Chrome Dev Tools (Inspector), CDT (Network), Project 0, Grading aspects
- *Section 3 (Django)* - Env Config, Markdown, RegEx, IDEs, pycodestyle, Debugging, Project 1
- *Section 4 (SQL, Models, Migrations)* - IDE's, linting, DB modeling, Project 2
- *Section 5 (JavaScript)* - cURL/Postman, jshint, CDT + IDE's Debugging, Project 3
- *Section 6 (User Interfaces)* - Animations, DB modeling, Pagination, Project 4
- *Section 7 (Testing, CI/CD)* - Test Driven Development, DevOps, Final Project
- *Section 8 (Scalability and Security)* - Cryptography, CAs, Attacks, App Deployment (Heroku)

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

Burning Questions?

Please ask questions, or topics to cover today!

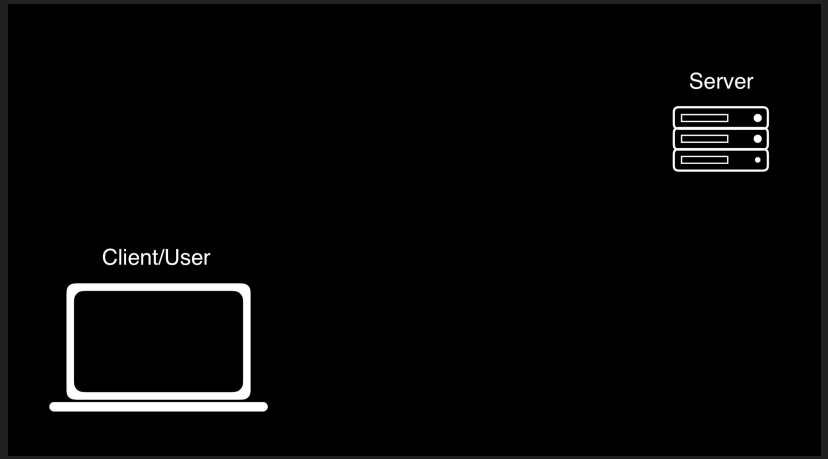
Topics:

- *Regex processing optional*

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 (/ˈdʒæŋgoʊ/ 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. `python -m pip 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
`django-admin startproject [PROJECT_NAME]`	Creates a new Django project
`python[3] manage.py runserver`	Runs the project on a locally hosted server
`ctrl-C`	Stops running the server
`python manage.py startapp [APP_NAME]`	Creates a new Django app within a project
`python manage.py migrate`	Creates a table to store session data
`python manage.py createsuperuser`	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
})
```

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`):

```python
from django import forms

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

Handling Form Submission and Rendering

```python
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>
```

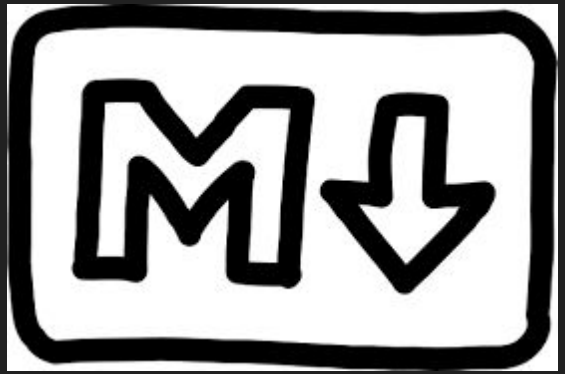
Django

Demo `budget` …

Other Helpful Material

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*

** Since Fall 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]"	Try it »
\	Signals a special sequence (can also be used to escape special characters)	"\d"	Try it »
.	Any character (except newline character)	"he..o"	Try it »
^	Starts with	"^hello"	Try it »
$	Ends with	"world$"	Try it »
*	Zero or more occurrences	"aix*"	Try it »
+	One or more occurrences	"aix+"	Try it »
{}	Exactly the specified number of occurrences	"al{2}"	Try it »
|	Either or	"falls|stays"	Try it »
()	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
\A	Returns a match if the specified characters are at the beginning of the string	"\AThe"	Try it »
\b	Returns a match where the specified characters are at the beginning or at the end of a word	r"\bain" r"ain\b"	Try it » Try it »
\B	Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word	r"\Bain" r"ain\B"	Try it » Try it »
\d	Returns a match where the string contains digits (numbers from 0-9)	"\d"	Try it »
\D	Returns a match where the string DOES NOT contain digits	"\D"	Try it »
\s	Returns a match where the string contains a white space character	"\s"	Try it »
\S	Returns a match where the string DOES NOT contain a white space character	"\S"	Try it »
\w	Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)	"\w"	Try it »
\W	Returns a match where the string DOES NOT contain any word characters	"\W"	Try it »
\Z	Returns a match if the specified characters are at the end of the string	"Spain\Z"	Try it »

Project

Project 1

- Start early!!!
- Make a checklist of requirement and check all before submission
- Make sure there’s no bugs
- Google Form (alarm for 11:45pm)
- 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.py**
- How to get a page? **utils.py**
- How to get all pages? **utils.py**
- markdown2.markdown()

IDEs and Debugging

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
- alias code="/Applications/Visual\ Studio\ Code.app/Contents/Resources/app/bin/code"

PyCharm

- No more Demo

Chrome Developer Tools (Network)

In Chrome:

1. Right click
2. Inspect
3. → Demo

Extremely powerful! Let’s try...

pycodestyle (formerly pep8)

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

Mac or Windows or Linux?

1. Mac
2. Linux
3. Windows with WSL
4. Remote IDEs e.g. codespaces
5. Windows
6. Chromebook? ¯_(ツ)_/¯

Running Python

1. Native installation:
    a. which python
    b. which python3
    c. python --version
    d. python3 --version
    e. python3 -m pip install requests
2. Virtual / Anaconda Environments

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)

venv

- Lightweight Virtual Environment (https://docs.python.org/3/library/venv.html)
- Commands
  a. Create new environment: `
     - `` `python3 -m venv venv_e33a` ``
  b. Deactivate/Activate environment:
     - `` `deactivate` ``
     - `` `. ./venv_e33a/bin/activate` ``
  c. Install more packages:
     - `` `pip install django` `` BUT:
       It is better to `` `python -m pip install django` `` (to assure proper pip)

HTML beautifiers/prettify

- Automatically formats your HTML (except line breaks)
- Most IDEs supports integration of marketplace beautifiers
- Demo...

Grading criteria generic suggestions (not limited to)

- Correctness:
  - All requirements + bugs
- Design (not limited to):
  - Simplest solution
  - Avoiding repetition (refactoring)
  - Structure (e.g separate files vs inline styling)
- Style (not limited to):
  - File naming/structure
  - Line breaks
  - Spacing / Indentation
  - Naming
  - Comments

Both Design and Style consider readability but from different perspective.

Random Tips

- Video Speed Controller (Chrome Extension)
  - Spotify + Hulu + Showtime => $5
  - GitHub Education Pack
  - Windows licence (https://harvard.onthehub.com)
  - Chrome Tabs
  - Code in Place 2022 (https://codeinplace.stanford.edu/)
  - DSA:
    - Video by CS50: https://www.youtube.com/watch?v=0DC1lk-SeOI&ab_channel=CS50
    - Leetcode / AlgoExpert / Etc.
    - Stanford Algorithms Specialization (EdX link /Coursera) - more theory (time consuming)
    - e22 seems good!
    - e20 + e124 (combo) - HARD!
  - System Design:
    - Grokking System Design
    - Alex Xu - System Design

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-spring-2022

CSCI E-33a (Web50)
Section 3

Ref: Lecture 3 (Django)

Vlad Popil

Feb 8, 2022