Contents

1 sketch

1.1 course overview

- May 27 - July 7:
- 10a - 1:50p (3:50/day, 7 hrs/week, 6 weeks)
- each session into blocks of: 1:50, 30 min break, 1:30 hours

1.1.1 Unit 1 (2 weeks): web crawling bots

1.1.2 Unit 2: (1 week): chat bots

1.1.3 Unit 3 (2 weeks): social media bots

1.1.4 Unit 4 (1.5 weeks): project workshops & presentations

1.2 course schedule

1.2.1 unit 1 web crawling (surveillance) - 2 weeks

1. May 29, session 1: intro to Python & web scraping with bs4

   (a) python installations

- install python with anaconda

(b) command line

- lesson: introduction to the command line
  - navigating
  - creating folders and files
  - using nano
  - curl requests

(c) web scraping tools & ethics

- bs4, scrapy, selenium
- what is ethical, what is legal?
- read article on ICE,
  - quickwrite thoughts, discussion.

(d) BREAK

(e) web scraping with bs4

i. Crash course on bs4: American Conservative articles

```
from bs4 import BeautifulSoup

import requests

import lxml

soup = BeautifulSoup(requests.get('https://www.theamericanconservative.com

soup.title

### demonstrate how to use the inspector tool to find different parts of t

# use the find() function to search by element and class
soup.find('', class_ = '')

# use the .text element to pull out just the text from the element
soup.find('', class_ = '').text

#### how would I get all of the headlines? what is the method?

# use the find_all() method to get all of the headlines
headlines = soup.find_all('', class_ = '')
```

```
headlines

for i in headlines:
    print(i.text)

### how would I save this information to a list?

titles = []
for i in headlines:
    titles.append(i.text)

titles

### challenge: use these tools to get the summary blurb for each article.
```

Challenge: write a function that scrapes all of the text from a webapge. Make it so you can pass different URLs into the function, and it will scrape a page for that URL.
Advanced challenge: how would I scrape the article text? hint: look at the URLs.

- make a list of URLs for every page.
- write a loop that goes through each one.
- turns that page into soup, grabs the article data, appends it to the list

Time to explore websites to scrape data from. Check if they are scrapable first.

(f) (if time): freewrite/share

- Brainstorm what kind of data you are interested in working with. Look at some websites that may have that kind of data. Why are you interested in this data?

(g) homework: find 2 scrapable sites Find 2 websites to scrape. Make sure if they are scrapable with bs4. Why are you interested in this data? What could you do with it?

2. June 2, session 2: scrapy & the scrapy shell

   (a) share websites that you found

   (b) introduction to scrapy shell

i. setting up environment Create conda environment for scraping

   Download and install scrapy. Troubleshooting installtions.

```
conda install -c conda-forge scrapy
pip install Scrapy
```

   Introduction to scrapy shell

```
scrapy shell 'https://quotes.toscrape.com/page/1/'
response.css("title::text").get()
```

ii. Extracting data with css selectors

   Syntax:

```
# basic syntax for using element and class to get text
# returns the entire element
response.css("element.class::text")

# getting the title elements
response.css("title::text")

# getting the quote elements
response.css("span.text::text")

# combine with get() to get just the text
response.css("title::text").get()

# and just the first instance
response.css("span.text::text").get()

# or with getall() to get a list
response.css("span.text::text").getall()
```

   Challenge: work to find the rest of the information on the page. Get all the authors and the tags.

iii. writing loops

   - write a loop that saves our information to a loop. I will write the first two lines.
     - looping through a subset of the page.
     - using the print function
     - you will have to expand this loop.

```
# first just looping through to print
for quote in response.css("div.quote"):
     print(quote.css("span.text::text").get())

# now saving it to a dictionary
quotes = {}
for quote in response.css("div.quote"):
    quotes["text"] = quote.css("span.text::text").get()
```

Now, you will expand the loop to include author and tag information.

(c) (if time) explore a website you're interested in to get selectors

(d) BREAK

(e) scrapy project

  i. starting new scrapy project
  Install VS Code
  Following tutorial on scrapy's tutorial in the docs:

```
# create your Scrapy project:
scrapy startproject project_name
cd project_name

# see the directory structure
tree
```

  Create new spider manually (copy/paste code from extracting data in our spider).

```
import scrapy


class QuotesSpider(scrapy.Spider):
    name = "quotes"
    start_urls = [
        "https://quotes.toscrape.com/page/1/",
        "https://quotes.toscrape.com/page/2/",
    ]

    def parse(self, response):
        for quote in response.css("div.quote"):
            yield {
                "text": quote.css("span.text::text").get(),
                "author": quote.css("small.author::text").get(),
                "tags": quote.css("div.tags a.tag::text").getall(),
            }
```

Classes:

- classes are like templates, which you can customize.
- contain properties and functions.
- `QuotesSpider` class builds on the `Spider` class.
- check out the Spider class in the docs.

Run the spider:

```
scrapy crawl quotes
```

Then store the scraped data:

```
scrapy crawl quotes -O quotes.json
```

(f) (if time) individual activity: apply code to website from homework

- change the name, urls, and selectors.
- run the same command

(g) introduce book, read introduction together, discuss

(h) homework: reading response *Compost Engineers* chapters 1 & 2 Joana Varon and Lucía Egaña Rojas. Chapters 1 & 2 from *Compost Engineers and Sus Saberes Lentos: A Manifest for Regenerative Technologies.* Coding Rights, 2024, `https://codingrights.org/docs/compost_engineers.pdf`.

Prompt: Pick an idea from the reading that interests you (either because you agree with it, disagree with it, or are otherwise provoked by it) and explain why.

3. June 5, session 3: blockers & XHR

(a) share homework, discuss reading

(b) scraping XHR Tutorial by Sam Lavigne on scraping Bing and Customs Border Protection.

i. Exploring XHR from the command line

```
from bs4 import BeautifulSoup
import requests

query = "how can i"

url = (
    "https://www.bing.com/AS/Suggestions?pt=page.home&mkt=en-us&qry="
```

```
        + query
        + "&cp=9&csr=1&msbqf=false&pths=1&cvid=6AE710F2D778431589574CB8424EFF7
    )

    response = requests.get(url)

    response
    dir(response)
    response.text
    response.content
    response.json()

    parsed = response.json()

    # what kind of data structure?
    # pull out the completions
    parsed
    parsed['s'][0]
    parsed['s'][0]['q']
    parsed['s'][1]['q']
    parsed['s'][2]['q']

    # write a loop that prints just the completions
    for item in parsed['s']:
        print(item['q'])
```

ii. Script for scraping XHR results.
   - how & why to create a script
   - how & why to write a function

   Run the below. Pipe output through sort -u to sort the output of our script and filter out duplicates.

```
python bing_autocomplete.py | sort -u
```

```
from bs4 import BeautifulSoup
import requests

def auto_complete(query):
  url = (
      "https://www.bing.com/AS/Suggestions?pt=page.home&mkt=en-us&qry="
      + query
      + "&cp=10&cvid=B8D86CB090A240A196E4867715E40B15"
```

```
    )
    response = requests.get(url)
    soup = BeautifulSoup(response.text, "html.parser")
    items = soup.select("li")
    for item in items:
        print(item.text)

base_query = "How can I "
for letter in "abcdefghijklmnopqrstuvwxyz":
    auto_complete(base_query + letter)
    for letter2 in "abcdefghijklmnopqrstuvwxyz":
        auto_complete(base_query + letter + letter2)
```

(c) (if time) guided practice: finding undocumented APIs Yin, Leon. Finding Undocumented APIs. 24 Feb. 2023, `https://inspectelement.org/apis.html#tutorial`.

Uses developer tools to reverse engineer google searches to examine autocomplete results.

(d) BREAK

(e) individual activity: explore how to bypass blockers Try out some of these strategies:

- How To Solve 403 Forbidden Errors When Web Scraping
- How to Bypass Cloudflare in Python
- 4 Methods to Bypass Cloudflare with cURL in 2025

(f) share what we've found

(g) homework: *Compost Engineers* chapters 3 & 4 Joana Varon and Lucía Egaña Rojas. Chapters 3 & 4 from *Compost Engineers and Sus Saberes Lentos: A Manifest for Regenerative Technologies.* Coding Rights, 2024, `https://codingrights.org/docs/compost_engineers.pdf`.

Prompt: From the authors' proposals, what do you find useful or surprising, and what do you have doubts about?

4. June 9, session 4: selenium

(a) share homework, discuss reading

(b) introduction to selenium
Install selenium

`conda install selenium` `pip install selenium`

Install driver

`https://sites.google.com/chromium.org/driver/getting-started?authuser=0`

`https://googlechromelabs.github.io/chrome-for-testing/files`

Open ipython shell

```
# imports: driver, service, by
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By

# variables to scrape site
url = 'https://doge.gov/'
d_path = './chromedriver-mac-arm64/chromedriver'
service = Service(executable_path=d_path)
driver = webdriver.Chrome(service = service)

# scrape site
driver.get(url)

### check inspector for element for each post: div.border-2

# scraping element using "find_element" function, which takes two
# arguments
card = driver.find_element(By.CSS_SELECTOR, "div.border-2")
card
card.text

# multiple elements with find_elements
cards = driver.find_elements(By.CSS_SELECTOR, "div.border-2")
cards

# get just the text
for i in cards:
    print(i.text)
len(cards)

### group challenge: write some code to extract the important
### information from these cards. you'll have to think about strategy:
### are you going to loop through the cards we already haveand take
### out the individual elements from each card, then save them to
### lists? Or will you re-scrape the content, specifically calling
### each item that we want?
```

(c) BREAK

(d) assignment: web scraping Using either scrapy or selenium, scrape some data from a website that you couldn't scrape before. Bring that data to class.

1.2.2 unit 2 chat bots (bias) - 1.5 weeks

1. June 12, session 5: spaCy for processing text
   (a) share scraping assignments
   (b) intro to Python for cleaning text
      - review replace() method and using Regex
      - practice cleaning own dataset
   (c) the spaCy pipeline
   (d) BREAK
   (e) NER in spaCy
   (f) practice NER in dataset
   (g) homework: ACLU tech & privacy analysis write-up Choose a recent topic from this page; write up analysis of what is going on, and your opinion on the issue. How does the issue handle privacy rights and ethical uses of data? `https://www.aclu.org/press-releases?issue=privacy-technology`
2. June 16, session 6: spacy continued, intro to transformers
   (a) share homeworks
   (b) pattern matching in spaCy
   (c) BREAK
   (d) introduction to huggingface
   (e) how to run inference
   (f) individual practice: explore tasks

  (g) homework: run a task on your own data

3. June 19, session 7: transformers and bias

   (a) share homework

   (b) how to fine-tune a model

   (c) introductin to git and github

   (d) individual practice: fine-tune a model with own data

   (e) BREAK

   (f) in class: read and explore COMPAS algorithm

      - "Can You Make AI Fairer than a Judge? Play Our Courtroom Algorithm Game." MIT Technology Review, `https://www.technologyreview.com/2019/10/17/75285/ai-fairer-than-judge-criminal-risk-`
      - recommended:
         - Groves, Lara, et al. "Auditing Work: Exploring the New York City Algorithmic Bias Audit Regime." The 2024 ACM Conference on Fairness, Accountability, and Transparency, ACM, 2024, pp. 1107–20. DOI.org (Crossref), `https://doi.org/10.1145/3630106.3658959`.

   (g) in class: end user algorithmic audit

      - goals:
         - opportunity to discuss what makes something toxic
         - opportunity to examine how models treat toxicity

      i. indie-label
      IndieLabel
      Installations:
         - use conda to create an env with python 3.8
         - then use pip to install the packages
         - if coming up against cython and/or surprise package errors, see:
            - AttributeError: cython_sources [duplicate]
            - Getting errors while installing Surprise package
            - ImportError: cannot import name 'cached_download' from 'huggingface_hub'

```
# create a constraint to avoid cython
```

```
echo "cython<3" > /tmp/constraint.txt
PIP_CONSTRAINT=/tmp/constraint.txt pip install -r requirements.txt

# install scikit-surprise separately with conda
conda install -c conda-forge scikit-surprise

# error when running server.py
pip install huggingface-hub==0.25.2
```

Audit instructions:

- open a blank document for note-taking
- complete the questionnaire to get your model in the "okay" range.
  - as you complete the questionnaire, make notes of deciding factors that made you choose if something is toxic or not toxic.
- explore the "auditing" tab, make notes on your findings.

Report:

- provide 2 examples of choices that were difficult or that made you second guess yourself
- why was the choice difficult?

(h) source code walkthrough: bias evaluation projects

- goals: learn to read complicated python code/projects; adapt pre-existing code to your own purposes/experimentation.

i. biases-llm-reference-letters `https://github.com/uclanlp/biases-llm-reference-let` `tree/main?tab=readme-ov-file`
Useful functions to count how many times certain words appear, male or female words.
Also uses spacy to create lists of male and female nouns and adjectives.

(i) assignment: dataset proposal What is the dataset you'd like to create for your final project? Where would you get the data, and how would you transform it? You can consider tools from this class (like text generation, named entity recognition, pattern matching), or you can consider other possibilities for transforming your data. 1 page, double spaced.

1.2.3 unit 3 social media bots (algorithms) - 1.5 weeks

1. June 23, session 8: twitter bots Twitter bot with Python tutorial

   (a) configuring environments

   ```
   mkdir met_women
   cd met_women

   conda create --name met_women

   # Activate the virtual environment:
   # - MacOS/Linux
   conda activate met_women


   touch .env
   nano .env

   # Consumer Keys > API Key and Secret
   API_KEY=<your-API-key>
   API_SECRET=<your-API-secret>

   # Authentication Tokens > Access Token and Secret
   ACCESS_TOKEN=<your-access-token>
   ACCESS_TOKEN_SECRET=<your-access-token-secret>

   touch .gitignore
   nano .gitignore

   __pycache__
   .env*

   % pip install tweepy, requests, python-dotenv
   ```

   (b) tweet.py

   touch tweet.py code tweet.py

   ```
   import os
   import tweepy
   import requests
   from dotenv import load_dotenv
   ```

```
from random import randint

load_dotenv()

API_KEY = os.getenv("API_KEY")
API_SECRET = os.getenv("API_SECRET")
ACCESS_TOKEN = os.getenv("ACCESS_TOKEN")
ACCESS_TOKEN_SECRET = os.getenv("ACCESS_TOKEN_SECRET")

client = tweepy.Client(
    consumer_key=API_KEY,
    consumer_secret=API_SECRET,
    access_token=ACCESS_TOKEN,
    access_token_secret=ACCESS_TOKEN_SECRET
)

def tweet_women_fact(tweepy_client):

   print('fetching women from the MET...')
   r1 = requests.get("https://collectionapi.metmuseum.org/public/collection/v

   parsed = r1.json()

   number = randint(1, 100)

   obj_id = parsed['objectIDs'][number]

   r2 = requests.get(f"https://collectionapi.metmuseum.org/public/collection/

   parsed = r2.json()

   if parsed['title'] != '':
       text = f"Title: {parsed['title']}"
   else:
       text = f"Title: Unknown"
   if parsed['artistDisplayName'] != '':
       artist = f"Artist: {parsed['artistDisplayName']}"
   else:
       artist = 'Artist: Unknown'
   if parsed['artistGender'] != '':
        gender = parsed['artistGender']
    else:
        gender = 'Gender: Unknown'

    image = parsed['objectURL']

    tweet_text = f"{text}, {artist}, {gender} {image}"
    print('tweeting women from the MET...')

    tweepy_client.create_tweet(text=tweet_text)

tweet_women_fact(client)
```

(c) deploying our bot

Tutorials:

- Adding secrets to github actions
- Automating a Twitter bot with GitHub Actions (github repo)
  - part 1/3
  - part 2/3
  - part 3/3

```
mkdir .github
mkdir .github/workflows
cd .github/workflows
touch actions.yml
code actions.yml
```

```
on:
  schedule:
#    - cron: '0 * * * *' # at top of every hour
    - cron: '0 0 * * *' # At 00:00 every day

  push:

jobs:
  build:

    runs-on: ubuntu-latest
```

```
steps:

  - name: checkout repo content
    uses: actions/checkout@v2 # checkout the repository content

  - name: setup python
    uses: actions/setup-python@v4
    with:
      python-version: '3.10' # install the python version needed

  - name: install python packages
    run: |
      python -m pip install --upgrade pip
      pip install -r requirements.txt

  - name: run scrupt
    run: python tweet.py
    env:
        API_KEY: ${{ secrets.API_KEY }}
        API_SECRET: ${{ secrets.API_SECRET }}
        ACCESS_TOKEN: ${{ secrets.ACCESS_TOKEN }}
        ACCESS_TOKEN_SECRET: ${{ secrets.ACCESS_TOKEN_SECRET }}
```

(d) BREAK

(e) group project: social media bot tutorial

- choose a social media app, like instagram, tiktok, linkedin, or another app of your choice.
- research some tutorials for scraping and/or creating a bot for that app. Make sure the tutorial is recent (in the last year, at minimum).
- with a partner, create a tutorial that you will use to teach your classmates how to scrape or create a bot on that app.
- tutorial should be written in markdown format, with each step described clearly, and code blocks to include code examples.
- you will present the tutorial like a lesson, where you walk your classmates through the process of using the tool.

    - 20-30 minutes lesson.

    Resources:

    - Yin, Piotr Sapiezynski and Leon. Browser Automation. 11 June 2023, `https://inspectelement.org/browser_automation.html`.
    - Instagrapi, instagrapi tutorial
    - Make an Instagram Bot With Python, Geeks for Geeks

   (f) make a plan for actions steps by next class

2. (online) June 26, session 9: group project

   (a) share progress, next steps
   (b) BREAK
   (c) breakout work sessions
   (d) mini-conferences with me

3. (online) June 30, session 10: group projects continued

   (a) tutorial presentations
   (b) BREAK
   (c) introduction to git
   (d) introduce final project assignment
   (e) homework: project proposal
   (f) instagram User: trans_phobia_ pass: supersecure

1.2.4 unit 4 project workshops & presentations - 1 week

1. (online) July 3

   - share progress
   - project workshops
   - mini conferences

2. (online) July 7

   - presentations

1.2.5 recommended readings

1. on data gathering and web scraping

   - Dodge, Jesse, et al. “Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus.” Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, edited by Marie-Francine Moens et al., Association for Computational Linguistics, 2021, pp. 1286–305. ACLWeb, `https://doi.org/10.18653/v1/2021.emnlp-main.98`.
   - Jo, Eun Seo, and Timnit Gebru. “Lessons from Archives: Strategies for Collecting Sociocultural Data in Machine Learning.” Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, Association for Computing Machinery, 2020, pp. 306–16. ACM Digital Library, `https://doi.org/10.1145/3351095.3372829`.
   - Chan, Anita Say. Predatory Data: Eugenics in Big Tech and Our Fight for an Independent Future. University of California Press, 2025. library.oapen.org, `https://doi.org/10.1525/luminos.215`.
   - Métraux, Julia. “Eugenics Isn’t Dead—It’s Thriving in Tech.” Mother Jones, `https://www.motherjones.com/politics/2025/01/eugenics-isnt-dead-its-thriving-in-tech/`. Accessed 14 Feb. 2025.

2. on machine learning

   - Alammar, Jay. The Illustrated BERT, ELMo, and Co. (How NLP Cracked Transfer Learning). `https://jalammar.github.io/illustrated-bert/`. Accessed 14 Apr. 2025.
   - Alammar, Jay. The Illustrated DeepSeek-R1. 10 Feb. 2025, `https://newsletter.languagemodels.co/p/the-illustrated-deepseek-r1`.

3. case studies of algorithmic bias & audits

   - Hada, Rishav, et al. “Akal Badi Ya Bias: An Exploratory Study of Gender Bias in Hindi Language Technology.” The 2024 ACM Conference on Fairness, Accountability, and Transparency, ACM, 2024, pp. 1926–39. DOI.org (Crossref), `https://doi.org/10.1145/3630106.3659017`.

- Gajjala, Radhika, et al. “Get the Hammer out! Breaking Computational Tools for Feminist, Intersectional ‘Small Data’ Research.” Journal of Digital Social Research, vol. 6, no. 2, 2, May 2024, pp. 9–26. jdsr.se, `https://doi.org/10.33621/jdsr.v6i2.193`.
- Tang, Ningjing, et al. “AI Failure Cards: Understanding and Supporting Grassroots Efforts to Mitigate AI Failures in Homeless Services.” The 2024 ACM Conference on Fairness, Accountability, and Transparency, ACM, 2024, pp. 713–32. DOI.org (Crossref), `https://doi.org/10.1145/3630106.3658935`.
- Groves, Lara, et al. “Auditing Work: Exploring the New York City Algorithmic Bias Audit Regime.” The 2024 ACM Conference on Fairness, Accountability, and Transparency, ACM, 2024, pp. 1107–20. DOI.org (Crossref), `https://doi.org/10.1145/3630106.3658959`.
- Costanza-Chock, Sasha, et al. “Who Audits the Auditors? Recommendations from a Field Scan of the Algorithmic Auditing Ecosystem.” Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, Association for Computing Machinery, 2022, pp. 1571–83. ACM Digital Library, `https://doi.org/10.1145/3531146.3533213`.

1.3 assignments

1.3.1 participation (30%)

- includes daily homework

1.3.2 unit assignments (30%)

- includes the assignment at the end of units 1-3

1.3.3 final project: some bot! (40%)

- final project that takes some data from web scraping or APIs, and uses it as the content for a bot.
- bot to be automated and published on github.