* Exploring Web Scraping with …
  + Python web scraping packages
  + Automated http to download data
  + Ways to select exact data to extract
  + Chrome developer tools and Jupyter lab
  + Build a spider
* How Web Scarping is Useful
  + Profit: web scraping business model
  + Data is locked inside the web
  + If there is an api to access data it is the better choice
    - But if there isn’t, web scraping is useful
* Web Scraping Examples
* Scraping for a Car
  + Web Scraping Python Libraries
    - Beautiful soup
    - Scrapy
    - Selenium
* Human Browsing Versus Web..
  + Human/Broswer
    - Enter a url or click a bookmark
    - Download html
    - Parse html and render
    - Review for useful information
    - Interpret
    - Remember the information
    - Click a link-enter another url
  + Web Scraper
    - Set a start\_url
    - Download html
    - Parse html
    - Extract useful information
    - Transform or Aggregate
    - Save the data
    - Go to the next url
* HTTP Overview
  + Request message and response message
  + Hyper-Text Transfer Protocol(HTTP) is the protocol that powers the web
    - HTTPS is encrypted http
  + HTTP Request
    - Web address or url
    - A verb
    - User agent
  + HTTP Request: Verb
    - GET: retrieves data
    - POST: sends data to the server
  + HTTP Request: User Agent
    - Identifies the browser or web scraper
* URL Hacking
  + Scheme
    - https://
  + Host
    - [www.iseecars.com](http://www.iseecars.com)
  + Port
    - :443
  + Path
    - /used-cars/used-tesla-for-sale
  + Query String(?) or URL Fragment(#)
    - #Location=66952&Radius=all…
  + Python f strings
    - f’Location={location}’
  + example of scraping
    - import request
    - start\_url = ‘https://www.iseecars.com/used-cars/used-tesla-for-sale’
    - download\_page = requests.get(start\_url)
    - print(downloaded\_page.text)
* HTML & CSS Selectors
  + Html & css selectors
  + XPath
  + Chrome developer tools
* XPath
  + //: look anywhere
  + Ex: ‘//ul[@class=”listings”]/li[@id=”vin3827”]’
  + XPath or CSS Selector
    - Either or
* Chrome Developer Tools
  + Control-Shift-C
* Is Web Scraping Legal? Is it Et..
* Web Scraping Legal Risks
  + Depends on what you do with data after scraping
  + Beware of private website(logging in, etc)
  + Relatively risky scraping
    - Large scale scraping for profit
    - Scraping to create a commercial product
    - Scraping large company web sites for profit
    - Creating and selling derivative works
    - Scraping personally identifiable data
* hiQ Labs Legal Case
* Python Scraping Environment
  + Lupyter lab
  + Pyenv and pipenv for python packages
  + Install required programs
    - ‘pip install requests’
    - ‘pip install beautifulsoup4’
    - ‘pip install pandas’
    - ‘pip install jupyterlab’
    - ‘pip install pyenv’
  + Install packages
    - pyenv install 3.7.4
    - pyenv local 3.7.4
    - pipenv --python 3.7.4
    - pipenv install requests
    - pipenv install beautifulsoup5
    - pipenv install pandas
    - pipenv install jupyterlab
  + pyenv lets you manage different python versions
  + pipenv exists to manage python packages in virtual environments
* Download a Page to Scrape
  + Ex)
    - import requests
    - from bs4 import BeautifulSoup
    - import pandas as pd
    - start\_url = ‘https://en.wikipedia.org/wiki/Tesla,\_Inc.’
    - download\_html = requests.get(start\_url)
    - soup = BeautifulSoup(download\_html.text)
    - with open(‘downloadt.html’, ‘w’) as file:
    - file.write(soup.prettify())
* Extract Data to Pandas
  + Pandas is the go to tool to work with tables
  + Ex)
    - full\_table = soup.select(‘table.wikitable tbody’)[0]
* Scrapy Framework
  + Application framework for crawling web sites and extracting structured data
  + CLI for creating a scrapy project
  + Template to quickly build a spider
  + Library
    - Set of related functions
    - You call it
    - Code structure is up to you
  + Framework
    - A complete application
    - It calls you
    - Structure specified by the framework
  + Development environment
    - Make a directory
    - Cd into it
    - pyenv install 3.7.5
    - pyenv local 3.7.4
    - pipenv --python 3.7.4
    - pipenv install scrapy
    - pipenv shell
  + Initialize scrapy project
    - scrapy startproject [projectname]
    - scrapy crawl [projectname] -o [output]
* Scrapy Shell
  + scrapy shell ‘url’
* Truecar Spider
  + import scarpy
  + class TruecarSpider(scrapy.Spider):
  + name = ‘truecar’
  + def start\_requests(self):
  + urls = [‘urls1’, …]
  + for url in urls:
  + yield scrapy.Request(url=url, callback=self.parse)
  + Yield in python
    - Yield helps with processes that have a delay - like waiting on a web page to load
    - Pause run to completion - creates a series of value over time
    - More memory efficient and faster
    - Think of yield as a lazy return
  + Real World Scraping tips
    - Use private or incognito mode, closer to what scrapy see(no cookies)
    - Save the downloaded page locally
    - Use an IDE and a debugger
    - Break down the problem one HTML chunk at a time
    - Web scraping is brittle and prone to break
* JavaScript Scraping Problem
* Requests-HTML
  + Uses pypeteer to fully render javascript
  + Headless browsers
* Selenium
  + Requests-HTML
    - Pyppeteer: based on chrome
    - Easier to get started
  + Selenium
    - Firefox, safari, opera, chrome, edge
    - Very powerful
  + Install selenium
  + Also need to install web driver
    - Specific to browser
    - Download, unzip & add to path
* Summary