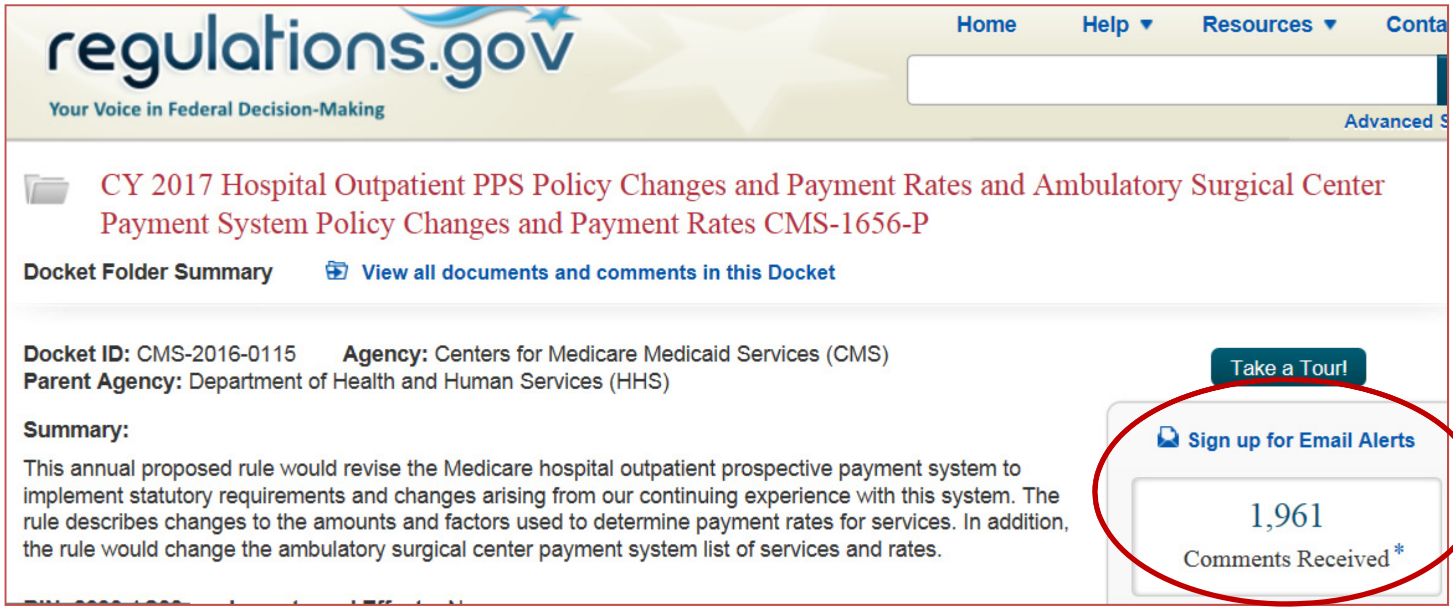
Getting and Analyzing Inconveniently Structured Data in the Internet Era: Making Friends with Python, Webscraping, and APIs



CY 2017 Hospital Outpatient PPS Policy Changes and Payment Rates and Ambulatory Surgical Center Payment System Policy Changes and Payment Rates CMS-1656-P

Docket Folder Summary   View all documents and comments in this Docket

Docket ID: CMS-2016-0115   **Agency:** Centers for Medicare Medicaid Services (CMS)
Parent Agency: Department of Health and Human Services (HHS)

Summary:

This annual proposed rule would revise the Medicare hospital outpatient prospective payment system to implement statutory requirements and changes arising from our continuing experience with this system. The rule describes changes to the amounts and factors used to determine payment rates for services. In addition, the rule would change the ambulatory surgical center payment system list of services and rates.

1,961
Comments Received *

Robert Letzler PhD, Senior Economist

US Government Accountability Office

Goal: Inform decisions about automating repetitive work that is hard in other tools Topics of discussion

- The right tools can tame a deluge of “inconveniently” formatted data
- Example: downloading from regulations.gov
- Quality control tools
- Where to get started and find help

Modern agencies create a deluge of "inconveniently" formatted data

- Government increasingly operates, publishes, discloses, and gets public input electronically
- Data are often fragmented on the web (or a network drive)
- And often stored in a markup language or PDF:
  - Web = HTML
  - Microsoft Office (zipped) XML: DOCX, PPTX, XLSX
  - Data = XML, JSON
- SAS, Stata, Python, R, and Excel can handle "conveniently" formatted data tables

This is an inconvenient package …

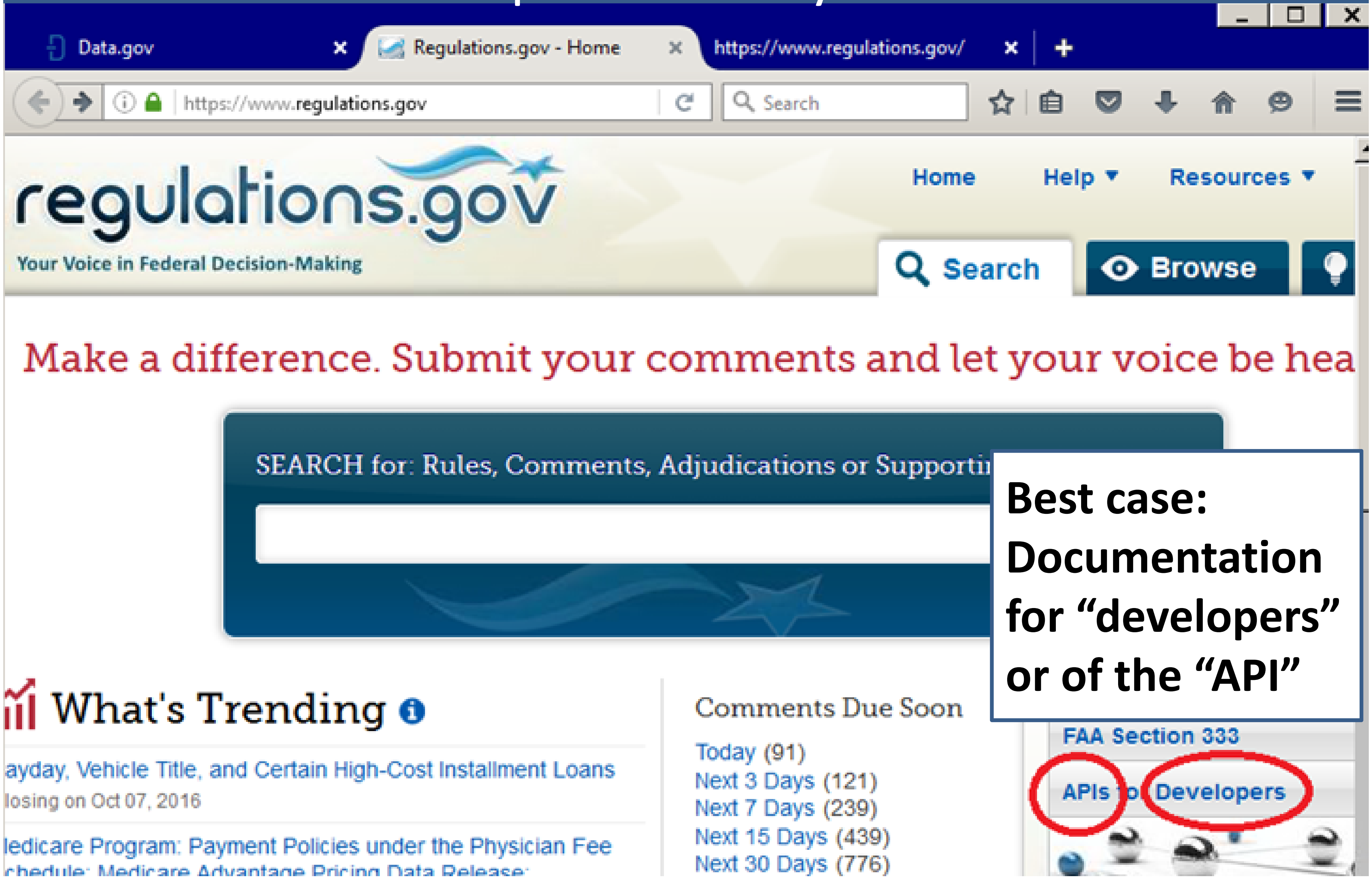
until you get a can opener!

Python is a can opener and a lot more

Free, approachable tools can help collect and extract that "inconveniently" formatted data

- The web, markup languages, and PDFs work in predictable, understandable ways
- A little classroom training and willingness to read, Google, and experiment goes a long way
- These tools shine for the big easy: easily described, very repetitive tasks
- Here are some examples of data types, extraction challenges, and recent GAO projects

Assessing what is involved in automating website interaction: first see if it provides tools for you



Data.gov

Regulations.gov - Home

https://www.regulations.gov/

https://www.regulations.gov

regulations.gov

Your Voice in Federal Decision-Making

Home Help Resources

Search Browse

Make a difference. Submit your comments and let your voice be hea

SEARCH for: Rules, Comments, Adjudications or Supporti

What's Trending

ayday, Vehicle Title, and Certain High-Cost Installment Loans
losing on Oct 07, 2016

ledicare Program: Payment Policies under the Physician Fee
chedule: Medicare Advantage Pricing Data Release:

Comments Due Soon

Today (91)
Next 3 Days (121)
Next 7 Days (239)
Next 15 Days (439)
Next 30 Days (776)

FAA Section 333

APIs for Developers

Best case: Documentation for "developers" or of the "API"

APIs: user interfaces for computers

Websites have lots of formatting for humans

- An API eliminates this clutter
- A typical API offers:
  - Ability to query through a web request: http://api.data.gov:80/regulations/v3/documents.json?api_key=DEMO_KEY&dktid=OCC-2013-0003
  - Results in JSON – key:value pairs, which map to Python dictionaries

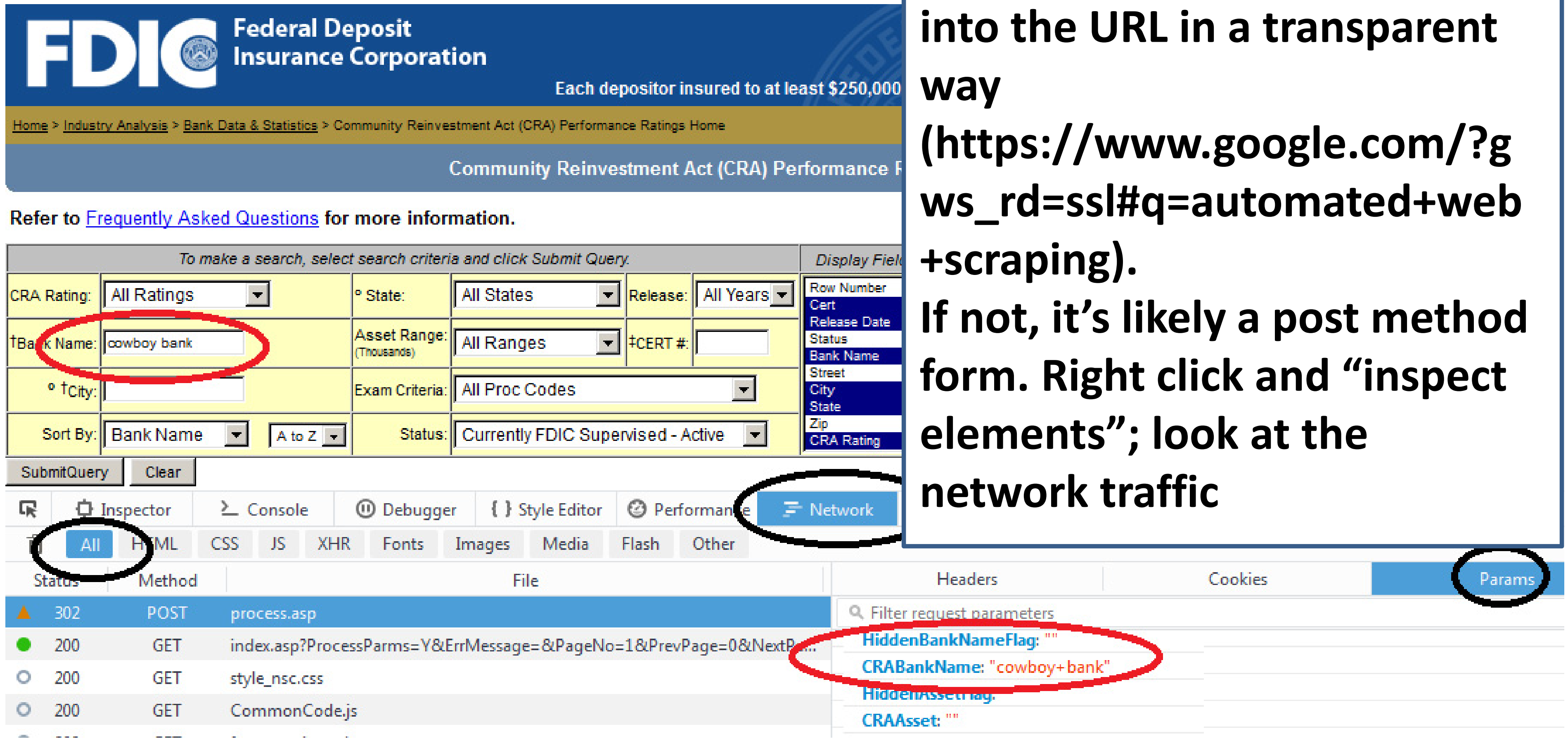
```
{ "documents": [
    { "agencyAcronym": "OCC",
    "allowLateComment": false,
    "attachmentCount": 1,
```

GAO projects using APIs

- Downloaded 190 public comments on the Community Reinvestment Act from regulations.gov
- Generalizing that code to create an in-house web-form to generate a ZIP file containing the documents on any docket
- Downloading hundreds of proposed rules listed on team spreadsheets from FederalRegister.gov
- Building a database of who is coauthoring or citing each other’s work in green chemistry using Elsevier SCOPUS

If there's no API, see how the site requests or presents information

First see if the query is going into the URL in a transparent way (https://www.google.com/?gws_rd=ssl#q=automated+web+scraping).
If not, it's likely a post method form. Right click and "inspect elements"; look at the network traffic


FDIC Federal Deposit Insurance Corporation

Each depositor insured to at least $250,000

Home > Industry Analysis > Bank Data & Statistics > Community Reinvestment Act (CRA) Performance Ratings Home

Community Reinvestment Act (CRA) Performance R

Refer to Frequently Asked Questions for more information.

To make a search, select search criteria and click Submit Query.

CRA Rating:	All Ratings	° State:	All States	Release:	All Years
†Bank Name:	cowboy bank	Asset Range: (Thousands)	All Ranges	‡CERT #:	
° †City:		Exam Criteria:	All Proc Codes		
Sort By:	Bank Name / A to Z	Status:	Currently FDIC Supervised - Active		

Display Fiel: Row Number, Cert, Release Date, Status, Bank Name, Street, City, State, Zip, CRA Rating

SubmitQuery | Clear

Inspector | Console | Debugger | Style Editor | Performance | Network

All | HTML | CSS | JS | XHR | Fonts | Images | Media | Flash | Other

Status	Method	File
302	POST	process.asp
200	GET	index.asp?ProcessParms=Y&ErrMessage=&PageNo=1&PrevPage=0&NextP...
200	GET	style_nsc.css
200	GET	CommonCode.js

Headers | Cookies | Params

Filter request parameters

```
HiddenBankNameFlag: ""
CRABankName: "cowboy+bank"
HiddenAssetFlag:
CRAAsset: ""
```

You can typically automate using just a few relevant pieces – e.g. post-method form submission and page requests; or tags around relevant elements

A GAO project involving post-method forms

- We needed to identify FDIC Community Reinvestment Act exams conducted in 2015
  - This required running ~300 queries (50+ jurisdictions x 2 year x 3 exam types); checking for 2015 in the URL; downloading the 2015 PDFs and creating a comma separated value (CSV) data table. We analyzed the table in SAS
- Website was complicated but limited; the substance complicated; and the audit needs evolving. The project took us about a month

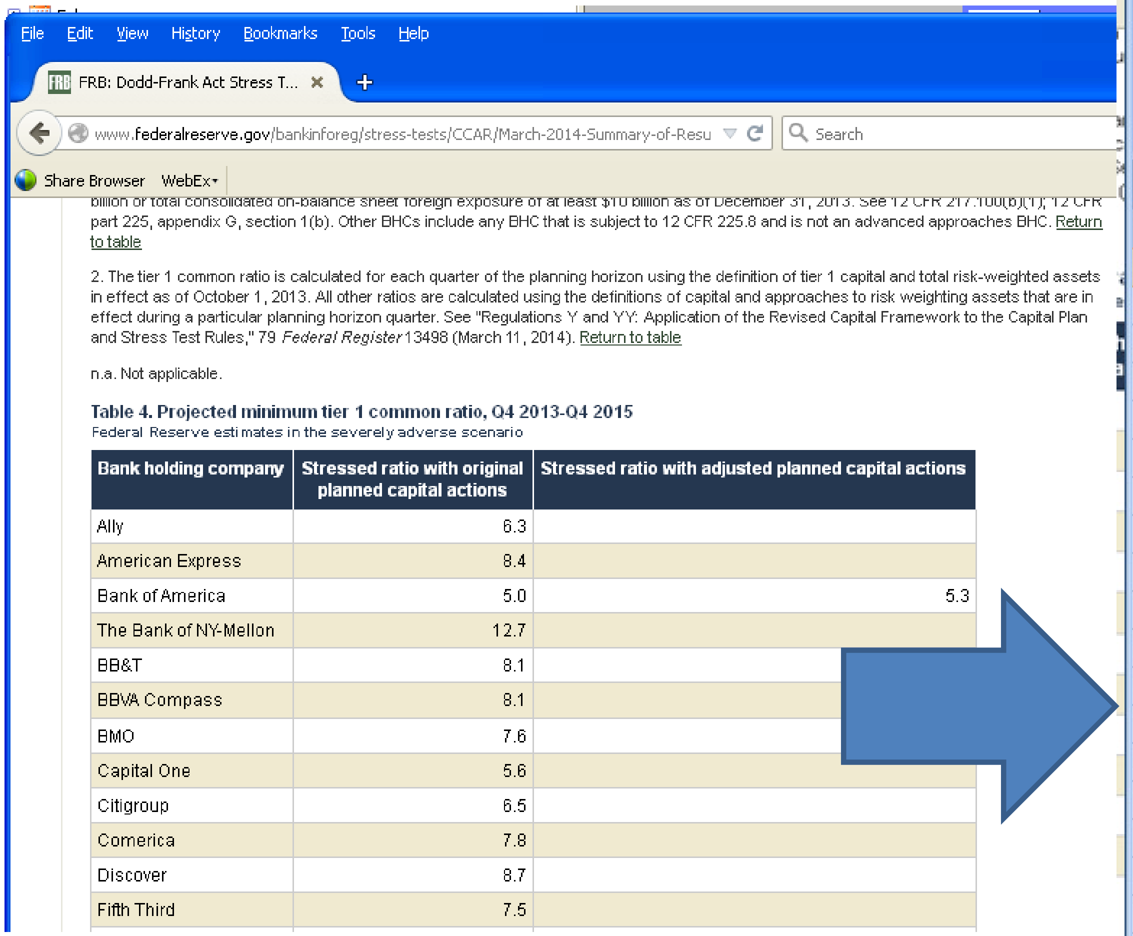
Webpages are written in a markup language, HTML (typically plus JavaScript and Cascading Style Sheets)

- Use “view source” to look at the HTML

East	**$20,000**
West	$49,950

- <TABLE> <TR><TD> East</TD><TD> $20,000</TD></TR>
- <TR><TD>West</TD><TD> $49,950</TD></TR></TABLE>
- If we extact the contents of all the <TD>table cells</TD> and save them in comma separated form, the results will go right into Excel, SAS, or Stata

Extracting data tables from webpages



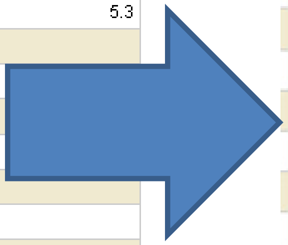
billion or total consolidated on-balance sheet foreign exposure of at least $10 billion as of December 31, 2013. See 12 CFR 217.100(b)(1); 12 CFR part 225, appendix G, section 1(b). Other BHCs include any BHC that is subject to 12 CFR 225.8 and is not an advanced approaches BHC. Return to table

2. The tier 1 common ratio is calculated for each quarter of the planning horizon using the definition of tier 1 capital and total risk-weighted assets in effect as of October 1, 2013. All other ratios are calculated using the definitions of capital and approaches to risk weighting assets that are in effect during a particular planning horizon quarter. See "Regulations Y and YY: Application of the Revised Capital Framework to the Capital Plan and Stress Test Rules," 79 *Federal Register* 13498 (March 11, 2014). Return to table

n.a. Not applicable.

Table 4. Projected minimum tier 1 common ratio, Q4 2013-Q4 2015
Federal Reserve estimates in the severely adverse scenario

Bank holding company	Stressed ratio with original planned capital actions	Stressed ratio with adjusted planned capital actions
Ally	6.3	
American Express	8.4	
Bank of America	5.0	5.3
The Bank of NY-Mellon	12.7	
BB&T	8.1	
BBVA Compass	8.1	
BMO	7.6	
Capital One	5.6	
Citigroup	6.5	
Comerica	7.8	
Discover	8.7	
Fifth Third	7.5	



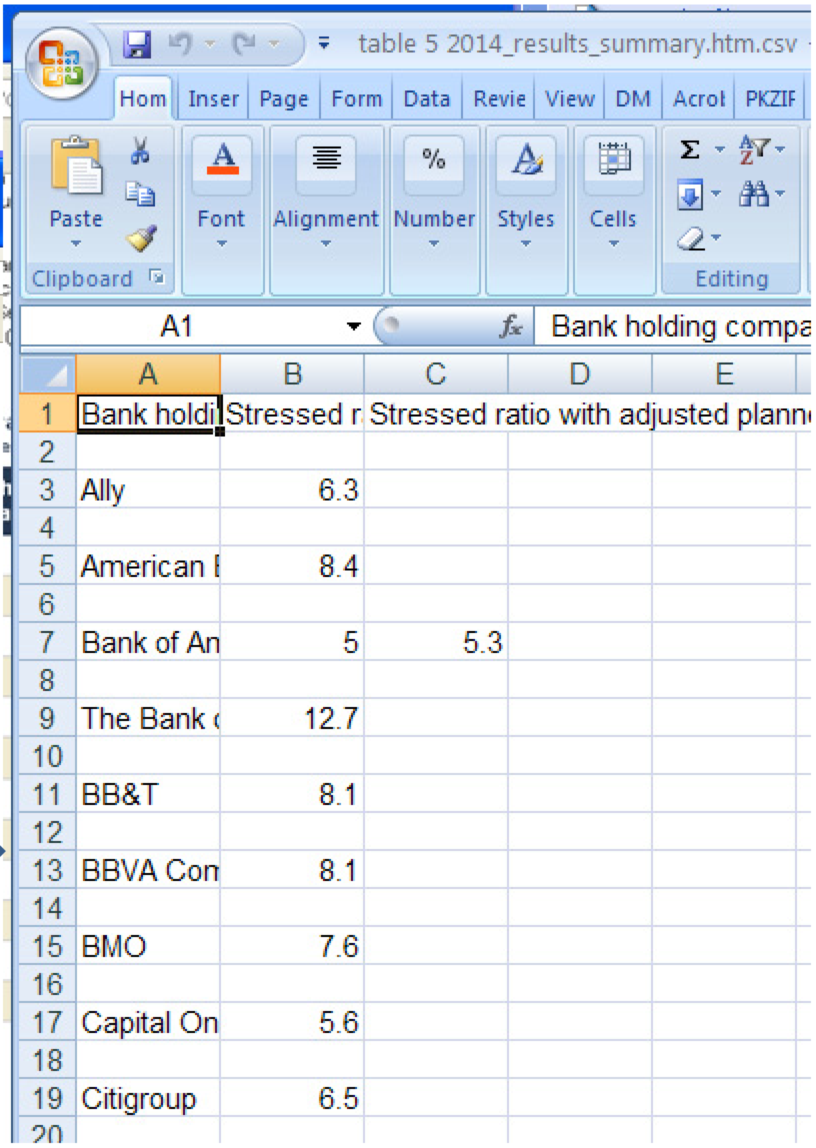


table 5 2014_results_summary.htm.csv

A1 Bank holding compa

	A	B	C	D	E
1	Bank holdi	Stressed r	Stressed ratio with adjusted plann		
2					
3	Ally	6.3			
4					
5	American E	8.4			
6					
7	Bank of An	5	5.3		
8					
9	The Bank o	12.7			
10					
11	BB&T	8.1			
12					
13	BBVA Con	8.1			
14					
15	BMO	7.6			
16					
17	Capital On	5.6			
18					
19	Citigroup	6.5			
20					

Fed data were in a fragmented, inconvenient format
We need to import them in bulk
  (can process data in Python, SAS, Stata, or Excel)
A documented process is desirable

Web site automation difficulty

Difficulty	Type	Strategy
Easiest	Static sites	just grab the relevant webpages (a copier like WinHTTrack may be sufficient)
Easy	Get method forms – everything important is in the URL:	Generate desired HTTP requests (e.g. https://www.google.com/?gws_rd=ssl#q=automating+HTTP+requests) and use results
Easy	API –a user interface tuned for computers	Generate HTTP requests to run desired queries; interpret the conveniently formatted results
Not so hard	Post method forms	Inspect element, watch the network to identify the “payload” to send the form and additional requests to make
Involved	+Javascript, Cookies, CSS, AJAX, poor organization, anti scraping features	Above strategies plus additional steps

PDF is built to describe page layout, not meaning

- APIs are often an ideal data source. PDF is never the ideal data source – but it may be the best available
- It may contain (high quality or garbled) machine readable text; OCR can make images into text.
- Extracting and processing plain text is often straightforward
- Initial attempts to extract tabular data using specialized software yielded mixed results

A GAO audit automated search and documentation of compliant language in PDFs

- Goals:
  - Checking ~200 messy PDF files to see if each contract had 3 required, boilerplate clauses
    - Edit distance was a crucial tool to identify the best match
  - Rapidly find apparent, obvious non compliance for further investigation (e.g. agency sent wrong doc)
  - Extract the PDF pages with interesting language
  - Allow one analyst to efficiently verify rather than one to find and document and one to verify

Automatically filled much of the DCI

Contract Name	DM link to file containing pages with match for terrorism clause	match location(s) for terrorism clause in the file with the best match	OCR text of potential match for terrorism clause	analyst review and notes
Chemtronics	DM#123	Page 2, top	prohibitlon against Sueport for Terronsm: (a) The Contrador/Reclplent …..	
IBEX	DM#124	Page 12 middle	frohibilion against Support for Terrorism: (a) The Contractor/Recipient ….	
Research Octagon Institute	DM#125	Page 4	Prohibition against Support for Terrorism: (a) The Contractor/Recipient …	

Frontiers

- Named Entity Recognition
  - Automatically extracts many of the names, locations, and organizations in plain text
- Text classification
  - Show the computer example documents that are or are not something of interest (a discussion of an IT system intrusion). Then have the computer find more examples of interest

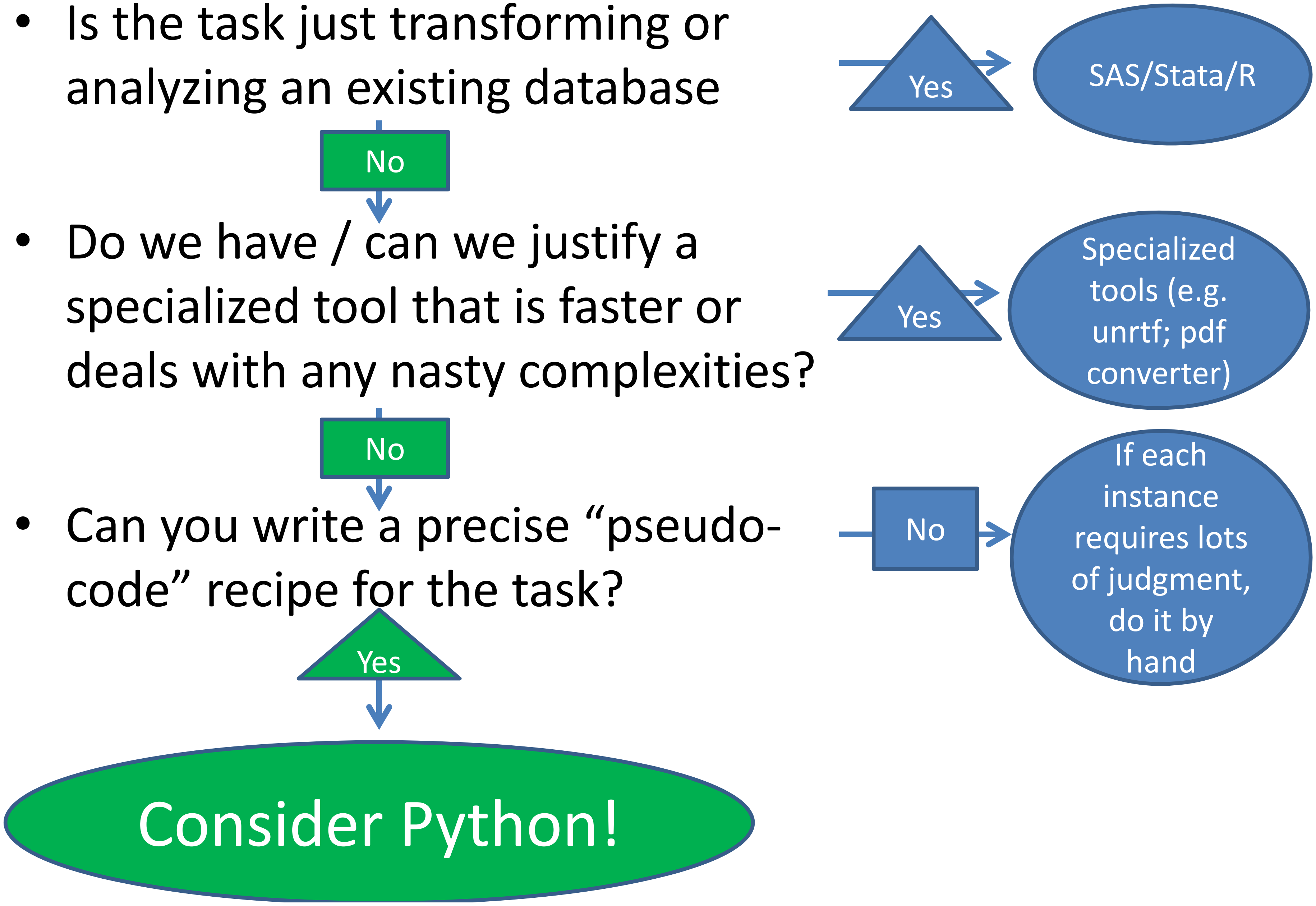
We tackled the projects discussed here in Python

Python is often the right tool

- A popular, high level (i.e. a little code can do a lot) programming language designed to be easy to learn
- Python and hundreds of libraries are free
- Multiplatform (Windows, UNIX, Mac)
- Supports ideas you've learned in SAS, Stata, or Java
- Extremely flexible: Many applications beyond data collection / extraction
- Other languages' capabilities overlap Python's including C/C++/Java/C#, PERL, R, SAS, Stata

Is Python right for you?

- Is the task just transforming or analyzing an existing database
- Do we have / can we justify a specialized tool that is faster or deals with any nasty complexities?
- Can you write a precise "pseudo-code" recipe for the task?


Yes
SAS/Stata/R
No
Yes
Specialized tools (e.g. unrtf; pdf converter)
No
No
If each instance requires lots of judgment, do it by hand
Yes
Consider Python!

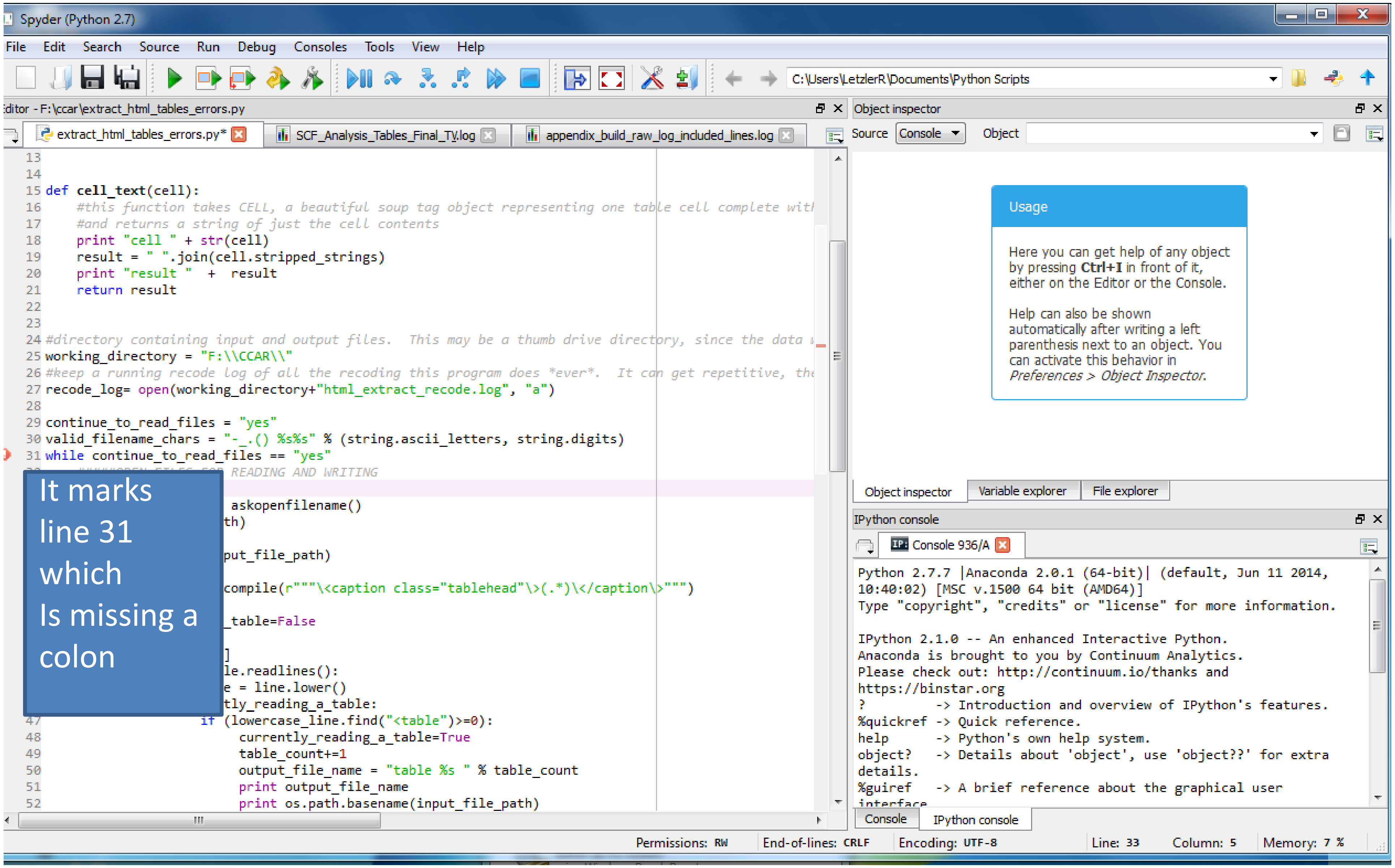
- “Python is still my favorite language for making my computer do things. …C# is my favorite language for building systems.”

  -professional software developer

Putting Python in Context

- SAS and Stata are at their best working with databases
- Python is a general purpose language with database, file, text, math, and internet libraries available
  - Far more flexible; more varied uses
  - It assumes less, so you'll have to write a bit more

Integrated development environments (e.g. Spyder, IDLE) help write, **debug**, and **run** (press F5) Python


Spyder (Python 2.7)
File Edit Search Source Run Debug Consoles Tools View Help
C:\Users\LetzlerR\Documents\Python Scripts
Editor - F:\ccar\extract_html_tables_errors.py
extract_html_tables_errors.py*
SCF_Analysis_Tables_Final_TV.log
appendix_build_raw_log_included_lines.log
Object inspector
Source Console Object
Object inspector Variable explorer File explorer
IPython console
Console 936/A
Console IPython console

```
13
14
15 def cell_text(cell):
16     #this function takes CELL, a beautiful soup tag object representing one table cell complete wit
17     #and returns a string of just the cell contents
18     print "cell " + str(cell)
19     result = " ".join(cell.stripped_strings)
20     print "result "  +  result
21     return result
22
23
24 #directory containing input and output files.  This may be a thumb drive directory, since the data
25 working_directory = "F:\\CCAR\\"
26 #keep a running recode log of all the recoding this program does *ever*.  It can get repetitive, the
27 recode_log= open(working_directory+"html_extract_recode.log", "a")
28
29 continue_to_read_files = "yes"
30 valid_filename_chars = "-_.() %s%s" % (string.ascii_letters, string.digits)
31 while continue_to_read_files == "yes"
                        READING AND WRITING

                         askopenfilename()
                        th)

                        put_file_path)

                        compile(r"""\<caption class="tablehead"\>(.*)\</caption\>""")

                        _table=False

                        ]
                        le.readlines():
                        e = line.lower()
                        tly_reading_a_table:
47             if (lowercase_line.find("<table")>=0):
48                 currently_reading_a_table=True
49                 table_count+=1
50                 output_file_name = "table %s " % table_count
51                 print output_file_name
52                 print os.path.basename(input_file_path)
```

It marks line 31 which Is missing a colon

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Object Inspector.*

```
Python 2.7.7 |Anaconda 2.0.1 (64-bit)| (default, Jun 11 2014,
10:40:02) [MSC v.1500 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 2.1.0 -- An enhanced Interactive Python.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and
https://binstar.org
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra
details.
%guiref   -> A brief reference about the graphical user
interface
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 33 Column: 5 Memory: 7 %

Goal: Inform decisions about automating repetitive work that is hard in other tools
Topics of discussion

- The right tools can tame a deluge of “inconveniently” formatted data
- **Example: downloading from regulations.gov**
- Quality control tools
- Where to get started and find help

FITARA JSON conversion: the ask

- Agencies posted updates to their FITARA implementation plans on April 30, 2016 at https://management.cio.gov/plans/"
- Step 1: convert the plans from JSON to Excel
- Step 2: find instances where "milestoneStatus" = "In progress" and "MilestoneTargetCompletionDate" is before the file date
- As usual, the most time consuming part will be dealing with non standard formats or unexpected entries. That is omitted for brevity here – the demo code leaves out six agencies that use non standard file formats.

Strategy: convert JSON to pipe delimited text, which Excel reads

```
{"milestones": [
{ "milestoneID": 1,
"milestoneDesc": "The Enterprise Information Technology (IT)…. ",
"milestoneTargetCompletionDate": "2015/12/31",
"milestoneStatus": "Complete",
 "milestoneStatusDesc": "Completed on 2015/12/03. …",
"commonBaselineArea": "budgetFormulation", "dcoiArea":
"nonDataCenter" },
```

```
1|The Enterprise Information Technology (IT) …. |2015-12-31
00:00:00|Complete|Completed on 2015/12/03….
|budgetFormulation|nonDataCenter
```

Define some key lists and a function

the requests library facilitates http:// requests

```
#requests has routines for accessing the web through HTTP requests
#we'll use datetime to time stamp this.
import requests, datetime, csv,trace, sys

#this definition is a JSON dictionary lookup that returns a blank if the key does
not exist in the dictionary
#and eliminates stray newlines
def extract(json_dict,key):
    if key in json_dict:
        return str(json_dict[key]).replace("\n"," ")
    else:
        return ""

def main(working_directory):
    #List of links captured from the "Public FITARA April 30th Milestone Updates"
section of https://management.cio.gov/plans/
    #some agency's links were missing:  DoD, Energy and Labor

    fitara_link_list =
["http://www.usda.gov/digitalstrategy/fitaramilestones.json",
"https://www.commerce.gov/sites/commerce.gov/files/fitaramilestones.json", …
```

A python list is denoted [item1,item2,item3]
A python dictionary is denoted:
{“key”:value,”key2”:value}

Request_response.json() maps the JSON to a Python “dictionary.”

Open files for output and put headers on them

```
 all_agency_file = open(working_directory+"Fitara.txt","w",
errors="backslashreplace" )
    all_agency_csv = csv.writer(all_agency_file, lineterminator='\n', delimiter="|")

all_agency_csv.writerow(["URL","file_date","milestoneID","milestoneDesc","miles
toneTargetCompletionDate","milestoneStatus","milestoneStatusDesc","commonB
aselineArea","dcoiArea"])

    overdue_file = open(working_directory+"Fitara_inProgress_overdue.txt","w",
errors="backslashreplace")
    overdue_csv = csv.writer(overdue_file, lineterminator='\n', delimiter="|")

overdue_csv.writerow(["URL","file_date","milestoneID","milestoneDesc","milesto
neTargetCompletionDate","milestoneStatus","milestoneStatusDesc","commonBas
elineArea","dcoiArea"])
```

Visit each agency’s URL and get its update date

```
for agency_URL in fitara_link_list:
    print(agency_URL)
    agency_plan =  requests.get(agency_URL)
    file_date =
datetime.datetime.strptime(agency_plan.json()[
"updatedDate"],"%Y/%m/%d")
```

Take steps for each agency – extract target dates, create a list of fields, write it to the appropriate files

```
for  milestone in agency_plan.json()["milestones"]:
        targetCompletionDate = 
datetime.datetime.strptime(milestone["milestoneTargetCompletionDa
te"],"%Y/%m/%d")
        output_list = [agency_URL, str(file_date), 
extract(milestone,"milestoneID"), extract(milestone,"milestoneDesc"), 
str(targetCompletionDate), extract(milestone,"milestoneStatus"), 
extract(milestone,"milestoneStatusDesc"), 
extract(milestone,"commonBaselineArea"), 
extract(milestone,"dcoiArea")]
        all_agency_csv.writerow( output_list)
        if (milestone["milestoneStatus"]=="InProgress") and 
(targetCompletionDate < file_date):
          overdue_csv.writerow(output_list)
```

Close the files and create an audit trail

```
    all_agency_file.close()
    overdue_file.close()
    #PROBABLY WE SHOULD ADD A LOG THAT RECORDS THE FILE 
DATE AND SIZE OF THIS FILE; THE SIZES OF ALL THE 
DOWNLOADED FILES, THE START TIME, ETC.

 # create a Trace object -- which will create a log file 
that counts the number of executions of each line below.
tracer = trace.Trace(
     #the goal of this line -- which comes straight from 
the sample code -- is to generate trace files only for 
GAO-written code and not more than a dozen trace files for 
Python-supplied code
     ignoredirs=[sys.prefix],
     trace=0,
     count=1)
```

Run everything and write out the trace log file

- working_directory = "R:\\letzlerr\\FITARA\\"

- # run the whole above program while using the tracer object to log which lines got executed. This is separate from "logging," the file I/O log above
- tracer.run('main(working_directory)')
- #now write the trace results to disk
- trace_results = tracer.results()
- trace_results.write_results(show_missing=True, coverdir=working_directory)

Goal: Inform decisions about automating repetitive work that is hard in other tools
Topics of discussion

- The right tools can tame a deluge of “inconveniently” formatted data
- Example: downloading from regulations.gov
- **Quality control tools**
- Where to get started and find help

If you are writing code for a single data set, you don’t care about weird circumstances that don’t arise in it

- Are we running on a static input data set or do we intend to reuse this code with varying inputs that force us to be ready for challenges unseen in the test data?

Asserting that things are as expected lets Python warn you if they’re not

- Assert checks a condition you specify and raises an “exception” if it is not true.
  - SSN length? Thou shall count to 9; 10 is right out!
  - Assert is your friend.
- In Python, an exception can stop the program or jump to an “except:” section of the code

GAO uses internal guidance papers to ensure appropriate planning, audit trail, and review of computer code

- Planning data analysis
  - A process we harness to coordinate between audit-specific subject matter experts and technical experts
- Documenting code
- Review and verification of code
- Same general guidance papers we use for SAS are appropriate for Python

Python facilitates documenting your work

#anything after a pound sign is a comment

- **trace** library creates a log showing how many times each line of code executed.
- Can create your own log files documenting what you did, names, sizes, and dates of files you created, etc.
- Not as automatic as SAS or Stata logging, but you can build exactly the documentation your reviewer needs.

Goal: Inform decisions about automating repetitive work that is hard in other tools
Topics of discussion

- The right tools can tame a deluge of “inconveniently” formatted data
- Example: downloading from regulations.gov
- Quality control tools
- **Where to get started and find help**

Getting Python

Python is free, open source software (by nerds, for nerds)

Python alone: available from Python.org;

Python is likely already installed on Linux/Mac computers, maybe even on Windows

(It’s got plenty of system administration capabilities that make it attractive to IT professionals)

Python distributions make life better

Python distributions with the Numeric/Scientific Python Stack have two advantages:

Technical: The versions of the libraries and Python work together

Bureaucratic: One rather than many installations

GAO uses Anaconda – which is – to our knowledge --the only free, multiplatform Numeric Python stack that supports Python 3.x. It comes with 180+ libraries

Distribution options here: https://www.scipy.org/install.html

The easy way to get Python into your agency may be to install it outside the main network

- Python came to GAO on computers outside of our main network – including some not networked at all.
- Strong internal controls on software installation on the main network are common and reasonable
- We were building the case to roll it out more broadly when our IT department decided to install Python for IT’s own purposes

Python 2.x and Python 3.x are (slightly) incompatible

- GAO ARM/CEA has switched to Python 3; few complaints
- Python 3 reduces pitfalls and confusion
- “*Python 2.x is legacy, Python 3.x is the present and future of the language*”
- **Possible to write code that works in Python 2.7 and Python 3.x.**
- If you have a Linux/UNIX/Mac computer, it likely has Python 2.x installed
- Some training material still in Python 2

Python 3 is better designed than Python 2

- Backstory: Python 2.x included some questionable or dated decisions e.g.:
  - 5.0/4.0 = 1.25 but 5/4 = 1
  - ASCII text (1963 technology) rather than Unicode (supports Chinese characters and emoticons ☺)
  - Python 3 defaults to handling Unicode errors by throwing program-terminating exceptions; you’ll likely want to set it to backslash replacing instead
  - **In 2008, Python 3 fixed them; required updating many libraries to restore compatibility. Python 2.x lives on.**
- Python 2 and Python 3 can coexist on the same system; Anaconda has nice “virtual environment” tools to facilitate this

Don’t be afraid: Plenty of high quality help is available

Start on Google or Python.org; then escalate

- Python tutorial; online courses
- Python help / **docs.python.org/ scipy.org**
- Discussion boards at e.g. stack overflow

Google searches all of these at once

- Colleagues who program in Python or other languages (many ideas, pitfalls, and approaches are the same)
- Post to StackOverflow
- GAO has a support contract

The right role for Python depends on your agency’s needs and existing tools

- GAO has significant investments in SAS and Stata and accompanying skills.
  - Because of that, Python and R are competing to be the go-to tools for challenges that break the SAS and Stata database tables in, statistics or tables out mold
- The need for enough users to do meaningful internal peer review suggests building depth in a few tools

Backup Slides

Python has specific libraries for each example

Dialog boxes:

Tkinter is Python's de-facto standard graphical user interface

Text processing:

string (character strings)

re (regular expressions)

Batch querying the web

urllib (lets you read webpages just like files)

also: *bs4 (*Beautiful soup; an HTML parser)

Deleting outdated files/identifying file management policy violations

os (operating system)

Custom calculations: optimization, simulation

numpy/*scipy* (Numeric Python, Scientific Python)

Surveys/lookup: web access to database

django (Django, “The Web framework for perfectionists with deadlines”)

Analyzing a database

CSV (facilitates work with comma separated value text files)

pandas (data structures and data analysis tools)

Not to mention Pyrex and GrumPy

Technology is increasingly important in our work

- Automate obtaining, extracting, transforming and summarizing information so we can focus on analyzing rather than clicking
- Sometimes the technology is the internal control
- Settle arguments with agencies by doing – in days or weeks -- what they say can’t be done