

# Coursera: Python for Everybody Specialization

The screenshot shows the Coursera platform interface for the 'Python for Everybody' specialization. At the top, there's a navigation bar with a menu icon, the Coursera logo, and a search bar. Below the header, the specialization title 'Python for Everybody Specialization Certificate' is displayed in large white text. Underneath the title, a subtitle reads 'Learn to Program and Analyze Data with Python. Develop programs to gather, clean, analyze, and visualize data.' To the left, there are two buttons: 'Enrolled' (in a yellow box) and 'Already enrolled' (with 'Financial aid available' below it). A large statistic '241,470 already enrolled!' is prominently shown. On the right, the 'Offered By' section features the University of Michigan logo, which consists of a yellow 'M' and the text 'UNIVERSITY OF MICHIGAN'. The main content area has a dark blue background with a collage of Python-related images, including code snippets and gears. The book cover for 'PYTHON FOR EVERYBODY EXPLORING DATA IN PYTHON 3' by Charles Severance is also visible.

## A How-to-Guide

Alex Huerta

BUS 464-12: Barry D. Floyd

4 Sep. 2019

The specialization is broken down into 5 courses, each of them containing weeks of information/lessons, quizzes, and assignments. The Python for Everybody Specialization course layout is detailed as follows:

#### Course 1: Programming for Everybody

- Week 1 - Ch.1: Why We Program
- Week 2 - Installing and Using Python
- Week 3 - Ch.1: Why We Program (cont.)
- Week 4 - Ch.2: Variables and Expressions
- Week 5 - Ch.3: Conditional Code
- Week 6 - Ch.4: Functions
- Week 7 - Ch.5: Loops and Iteration

#### Course 2: Python Data Structures

- Week 1 - Ch.6: Strings
- Week 2 - Unit: Installing and Using Python
- Week 3 - Ch.7: Files
- Week 4 - Ch.8: Lists
- Week 5 - Ch.9: Dictionaries
- Week 6 - Ch.10: Tuples
- Week 7 - Graduation

#### Course 3: Using Python to Access Web Data

- Week 1 - Getting Started
- Week 2 - Ch.11: Regular Expressions
- Week 3 - Ch.12: Networks and Sockets
- Week 4 - Ch.12: Programs that Surf the Web
- Week 5 - Ch.13: Web Services and XML
- Week 6 - Ch.13: JSON and REST Architecture

#### Course 4: Using Databases with Python

- Week 1 - Object Oriented Python
- Week 2 - Basic Structured Query Language
- Week 3 - Data Models and Relational SQL
- Week 4 - Many-to-Many Relationships in SQL
- Week 5 - Databases and Visualization

#### Course 5: Capstone: Retrieving, Processing and Visualizing data with Python

- Week 1 - Welcome to the Capstone
- Week 2 - Building a Search Engine
- Week 3 - Exploring Data Sources (Project)
- Week 4 - Spidering and Modeling Email Data
- Week 5 - Accessing New Data Sources (Project)

Week 6 - Visualizing Email Data

Week 7 - Visualizing New Data Sources (Project)

## Recommendations

While this may not be true for every one, one of the reasons why I was able to grasp the knowledge in the python courses without too much difficulty was because I had already learned how to code in Java prior to taking the courses. Learning Java before taking Python helped immensely. Both of the languages are similar; so a lot of the key concepts are the same such as: strings, variables, classes, objects, methods, inheritance, arrays, if statements, loops, web scraping, etc. I will admit however, that Java is a little harder to learn than Python. Syntactically, Java is way more formal and requires extra code for similar commands in python, like printing, getting user input, declaring strings & variables, etc. For example, take the print statement. In Java it's "system.out.print("x")", whereas in python it's just "print("x")". By learning the harder syntax first, it makes learning python much easier as it feels simpler to use and more straight forward in a sense.

Around the time I got to Course 3, I also started learning SQL, which came in handy as Course 4 is all SQL based. While SQL and Python are significantly different, by knowing both you are able to accomplish so much like: pull information from the web, read it, build a database, store the database with the info pulled from the internet and then visualize it using javascript. Overall, because of Python's simplicity and compatibility, the more programming languages you know, the better.

Another option that aids in ensuring success in the Python course is to watch a beginners tutorial ahead of time. Watching a tutorial will give you a brief overlook of the whole language, making it familiar and easier to understand when going through the course. Personally, I watched a four and a half hour Python tutorial offered by freecodecamp.org on YouTube (<https://www.youtube.com/watch?v=rfscVS0vtbw>). The tutorial recommended to download Python 3 instead of version 2, as it's the newer supported version, and more organizations are using version 3. Another recommendation was to use the IDE Pycharm for coding, which I personally like. Pycharm is easy to use and highlights certain terms to make the code easier to read. The tutorial covers the following topics:

Installing Python, Printing to the Screen (Hello World, Drawing Shapes), Variables & Data types, Working with Strings & Numbers, Getting input from users, Building a Basic Calculator, Mad Libs Game, Lists, List functions, Tuples, Functions, Return Statements, If Statements and Comparisons, Building a Better Calculator, Dictionaries, Loops, Build a Guessing Game, Exponent Functions, 2D

Lists & Nested Loops, Building a Translator, Try/Except, Reading/Writing Files, Modules & Pip, Classes & Objects, Building a Multiple Choice Quiz, Object Functions, and the Python Interpreter.

Watching the tutorial gave me a glimpse of what to expect and actually helped me speed through some of the early sections in the courses.

Another time saving recommendation, if you plan on applying for financial aid, you must fill out an individual application for each of the five courses. Filling out the application under the Python for Everybody Specialization (all 5 courses) will only submit an application for the first course. It takes two weeks for Coursera to review financial aid applications, so if you wish to avoid any unwanted time delays, ensure that you submit an application for each course.

Plan your time accordingly, the time estimates that Coursera gives doesn't factor in the extra time it will take to pause and repeat videos. In my case, it usually took me double the time that the time estimates predicted to get past the material. Lastly, as it will be thoroughly mentioned throughout the courses, it is very beneficial to download the "Sample code" folder from <https://www.py4e.com/materials> and the lecture slides from <https://www.py4e.com/lessons>, so you can follow along.

## **Course 1: Programming for Everybody (Getting Started with Python)**

Course 1 is broken down into 7 weeks, has 5 quizzes, and 7 assignments. If you happened to watch the Python tutorial on YouTube prior to starting the course, you should be able to breeze through it fairly quickly. All of the topics were covered in the tutorial, and as it's the first course in the specialization, it only covers the easier introductory topics. Week 1, "Ch.1: Why We Program?," is all about welcoming you to the course. There's no coding to be done, quizzes or assignments, just videos and instructions on Python 3, how to download course texts, submit assignments, and a hardware/software overview (CPU, RAM, input/output devices, Main & secondary memory). Week 2, "Installing and Using Python", is very similar to week 1, there's no quizzes or assignments, and there's only one line of code you learn. For the most part week 2 shows you how to actually install python and then write your first program, which is just printing out "hello world" to the screen, and how to use the "Python Playground", the interface used to submit coding assignments.

Week 3, "Ch.1: Why We Program (cont.)," is where the course starts lecturing more in-depth regarding python coding, so you want to make sure that Python has been installed on your computer. There's a lecture video that gives an overview of writing paragraphs of code, the logic of how the code is read/ran, and gives peaks of the structure one's code needs follow for topics that will be learned later like: if statements,

loops, dictionaries, etc. One major takeaway is that Python reads code, top-down. In addition there's a Quiz on the Ch.1 Material (Terms, and basic logic of code structure), as well as the first assignment, "Hello World." See appendix B.1 for more information.

Week 4, "Ch.2: Variables and Expression," is pretty self-descriptive as you watch three lecture videos on variables, expressions, constants, reserved words, type, conversion between types, operators, operator precedence, user input, integer division, and comments. One of the key concepts to understanding week 4 is to think of variables as containers for storing data and expressions as the ways to represent that data. There's a quiz on the concepts described, as well as two assignments that require utilization of those concepts. See appendix B.2 for more information.

Week 5, "Ch.3: Conditional Code," explains logic based coding by introducing topics such as: If, Else, and Elif statements, which are used to make one-way, two-way, or multi-way decisions with certain conditions. Some of the other topics to focus on are the comparison operators: == (equal to), <=, >=, <, >, != (not equal), and the "try/except" which is used to catch errors. You will then be quizzed on the topics and assigned two assignments to put them in practice. See appendix B.3 for more information.

Week 6, "Ch.4: Functions," is pretty straight-forward and covers functions, which are easiest to think of as reusable pieces of code or "stored" code that we use. They are very useful for saving time and code. The important thing to remember is that functions take some input (parameters) and processes an output (return value). It's important to pay attention to the lectures on how to build and use functions as you'll be quizzed on them and be given an assignment on them. See appendix B.4 for more information.

Week 7, "Ch.5: Loops," covers everything regarding loops. You can think of loops as repeated steps that have iteration variables that change each time through a loop. To understand this section it's important to focus on these topics in the lecture: While-loops (indefinite), For-loops (definite), Infinite loops, iteration variables, breaks, continues, and idioms. You'll be quizzed on the material and given an assignment on it as well. See appendix B.5 for more information.

## **Course 2: Python Data Structures**

Course 2 is broken down into 7 weeks, 5 quizzes, and 7 assignments, just like course 1. Similar to the previous course, if you watched the recommended Python tutorial on Youtube then this course should also be a breeze as all the topics in the course were covered in the tutorial. Week 1, "Ch.6: Strings," introduces the concepts of strings and string manipulation with functions. It's easiest to understand strings as a sequence of characters (letters and integers) that are defined by quotes. To understand this chapter it's important to focus on these topics during the lecture videos: String type, Read/Converting, Indexing Strings [], Slicing Strings [2:4], looping through strings (for and while), concatenation (+), string operations, string library, comparisons,

search/replace text, and stripping whitespace, as you'll be quizzed and given an assignment. See appendix C.1 for more information. As for Week 2, "Unit: Installing and Using Python," feel free to skip this material as it's the exact same thing that was shown in week 2 of course 1.

Week 3, "Ch.7: Files," has to do with Python and external files and how to process them, such as opening them, reading them, or writing to them. To understand how to use python with files you must focus on these topics: Opening a file - file handle, file structure - newline character, reading files line by line (for loop), searching for lines, reading file names, and dealing with bad files. As always you'll be quizzed material, and this week will be given two assignments on the concepts. See appendix C.2 for more information.

Week 4, "Ch.8: Lists," goes into everything you need to know about lists, specifically how to manipulate them, and the relations between lists and strings. Lists are very similar to Arrays in Java. Lists are best thought of as a collection, they allow us to put many values into a single variable. Pay attention to the following topics during the lecture videos: Lists and definite loops (for), indexing and lookup, list mutability (elements can be changed with an index operator), function (len, min, max, sum), slicing and sorting lists, parsing, appending/removing, and splitting strings into lists. You will be quizzed on the concepts above (syntax and definition) as well given to assignments on them. See appendix C.3 for more information.

Week 5, "Ch.9: Dictionaries," it's no coincidence that the course introduces dictionaries next , given that they're very similar. Like lists, dictionaries are collections. However, whereas a list is a linear collection of values that stay in order, dictionaries are like a bag of values, each with its own label or key. Dictionaries are Python's most powerful data collection, if you understand them they can prove useful for fast database-like operations. This week covers dictionaries as a whole with emphasis on counting dictionaries, and the relation between dictionaries and files. Focus on these topics from the lecture videos as you'll be quizzed and given an assignment: Lists vs Dictionaries, Dictionary constants, the get()" method, the most common word, hashing & lack of order, writing dictionary loops, sorting dictionaries, and sneak peek: tuples. See appendix C.4 for more information.

Week 6, "Ch.10: Tuples," another concept similar to lists, tuples are used to store multiple values although tuples are immutable; meaning once you create one, you can't alter its contents (like a string). Tuples, unlike lists are also comparable (>, <, =). Tuples are useful when working with data that needs to be immutable and comparable, like coordinates for example. To understand tuples focus on these topics: tuple syntax, immutability, comparability, sorting, tuples in assignment statements, and sorting dictionaries by key or value. Be prepared for a quiz and an assignment on the topics above. See appendix C.5 for more information.

Week 7, “Graduation,” marks the end of the Python for Everybody Specialization’s “halfway mark.” Feel free to skip this section if necessary. There’s no quizzes or assignments, just a “Graduation” video, and a video that briefly explains what’s to come in the second half of the specialization.

### Course 3: Using Python to Access Web Data

Course 3 is broken down into 6 weeks, 5 quizzes, and 8 assignments. Unlike the two previous courses, expect to start spending more time on the last three courses as little of the content was covered in the Python Youtube tutorial. Week 1, “Getting Started,” is the only section in this course that you should skip through. The content is the same as week 2 from the previous courses, the only video worth watching is the one on the new “Peer-graded” way to submit assignments, instead of the python playground. Week 2, “Ch.11: Regular Expressions,” thoroughly covers regular expressions, also referred to “Regex” or “Regexp,” they provide a concise and flexible means for matching strings of text, such as particular characters, words, or patterns of characters.<sup>1</sup> To understand regular expressions it important to focus on these topics during lecture: Regular Expression quick guide, the regular expression module (import re, re.search(), re.findall()), wild-card characters, matching and extracting data, Double Split pattern vs Regex version, and the escape character. As always you’ll be quizzed and given an assignment on the topics above. See appendix D.1 for more information.

Week 3, “Ch.12: Networks and Sockets,” introduces web data and network architecture by focusing on networked technology, Hypertext Transfer Protocol (HTTP), and sockets (an endpoint of a bidirectional inter-process communication flow across and Internet protocol-based network). To understand this week’s material focus on these lecture topics: TCP and Application Protocols (HTTP), TCP connections/Sockets, TCP port numbers, Sockets in Python, and Getting Data From the Server. Expect to be quizzed on the material above, as well as an assignment that puts it to use. See appendix D.2 for more information.

Week 4, “Ch.12: Programs that Surf the Web,” goes further into web exploration and explains more efficient ways to receive web data. This week goes into Unicode Characters & Strings, Retrieving/Parsing web pages, and using **urllib** & BeautifulSoup. Pay attention to these topics during lecture: ASCII (characters and strings), HTTP request, **urllib** (Python library that does all the socket work for us, web pages look like files), following links, Parsing HTML (web scraping), and BeautifulSoup (Python package for parsing HTML and XML documents, and extracting data from HTML). After the

---

<sup>1</sup> [http://en.wikipedia.org/wiki/Regular\\_expression](http://en.wikipedia.org/wiki/Regular_expression)

lecture you'll have a quiz and two assignments to complete regarding the material. See appendix D.3 for more information.

Week 5, "Ch.13: Web Services and XML," focuses on the first half of chapter 13 which mainly focuses on XML, one of the two commonly used formats for data going between applications across networks (JSON is the other). Extensible Markup Language's (XML) main purpose is to help information systems share structured data. The lecture videos focus on the type of data on the web and XML, specifically the XML schema and parsing XML. To understand this half of the chapter focus on these topics: XML elements (nodes) and basics, terminology (tags, attributes, serialize/deserialize), XML trees and paths, XML Schema (contract), XSD XML Schema (W3C spec), XSD structure, XSD data types, XSD constraints, and the ISO 8601 date/time format. You will have a quiz on those topics and an assignment as well. See appendix D.4 for more information.

Week 6, "Ch.13: JSON and REST Architecture," covers the second half of chapter 13 and focuses on JavaScript Object Notation (JSON), the Service Oriented Approach, Service Oriented Architectures, using Application Program Interfaces (APIs), and securing API requests. JSON's syntax is very similar to Dictionaries, to understand the structure I'd recommend reviewing Dictionaries. It will also help you to think of APIs as largely abstract in that it specifies an interface and controls the behavior of the objects specified in that interface.<sup>2</sup> Focus on the topics mentioned above during the lecture videos as you'll have a quiz to complete as well as two assignments. See appendix D.5 for more information.

## **Course 4: Using Databases with Python**

Course 4 is broken down into 5 weeks, 5 quizzes, and 5 assignments. Week 1, "Object Oriented Python," goes over Object Oriented definitions/terminology, Classes and Objects, Object Life Cycle, and Object Inheritance (the ability to extend a class to make a new class). As a quick tip, it helps to think of a Class as a template, think of a Method/message as a defined capability of a class, think of a Field/attribute as a bit of data in a class, and think of an Object-instance as a particular instance of a class. You will be assigned a quiz so make sure to pay close attention to the topics mentioned above during lecture. There's no assignment this week so I referenced notes from the YouTube Python tutorial to help you understand the material, see appendix E.1 for more information.

Week 2, "Basic Structured Query Language," marks the introduction into another coding language, SQL. The introduction this week focuses on installing the SQLite

---

<sup>2</sup> <http://en.wikipedia.org/wiki/API>

Browser and the first concepts from chapter 15: Relational Databases, Using Databases, and Single Table CRUD (Create, Read, Update, Delete). To understand this week's material focus on these concepts from the lecture: Terminology (Database, Relation, Tuple, Attribute), CRUD, Database Administrator and Model, and SQL Syntax (Create, Insert, Delete, Update, Select, Order By). Prepare to be quizzed on the concepts and given two assignments. See appendix E.2 for more information.

Week 3, "Data Models and Relational SQL," we are diving further into chapter 15 where the focus is on designing data models, representing a data model in tables, inserting relational data, and reconstructing data with the JOIN operator. I recommend to focus on these topics during lecture: Database Design, Building a Data Model, Database Normalization (3NF), Integer Reference Pattern, Keys (Primary, Logical, Foreign), Relationship Building (in tables), and Using Join Across Tables. These concepts are important, they will teach you how to make programs and databases faster and efficient, as well as prepare you for the quiz and assignment. See appendix E.3 for more information.

Week 4, "Many-to-Many Relationships in SQL," covers the last portion of chapter 15. Many-to-Many relationships are what we need to model a connection where each side can have many values. There's not that much to them, focus on learning the syntax, and how to properly insert values into a many-to-many relationship, and you'll be set for the quiz and the assignment. See appendix E.4 for more information. It's important that you understand the material from chapter 15 because if you do, it will allow you to scale to large amounts of data.

Week 5, "Databases and Visualization," covers chapter 16 and focuses on GeoCoding, GeoCoding Visualization, and web crawling. This is the last material to learn in the course and the bulk of it is conceptual. There's not that much to learn as the material is mostly just combining everything you have learned (APIs, SQL, reading files, pulling web data, `urllib` to read HTML files, try and except, JSON, loops, dictionaries, parsing, if-statements, stripping/slicing, functions, and objects/inheritance). You are provided with all of the code for the lecture and the assignment, the lecture videos are really just code walkthroughs done by the instructor, Dr. Chuck, so all you have to do is pay attention and follow along. As for the assignment, all you do is one small edit then run the code, and submit screenshots of your results. See appendix E.5 for more information.

## **Course 5: Capstone: Retrieving, Processing and Visualizing data with Python**

Course 5 is broken down into 7 weeks, 2 assignments (projects), and only 1 quiz. This course is very similar to week 5, "Databases and Visualization," from the last course. There's no new data to learn, you are covering the same concepts on

Visualization from Chapter 16. The course walks you through the two assignments, which are similar to the one in week 5 from course 4. You'll be provided with all of the code, and the lecture videos are Dr. Chuck walking you through all of the code for the assignments, all you have to do is pay attention, run the code, and submit the requested screenshots. This course is all about combining all of the concepts you have learned throughout the course (APIs, SQL, reading files, pulling web data, **urllib** to read HTML files, try and except, JSON, loops, dictionaries, lists, parsing, web-scraping & parsing with BeautifulSoup, if-statements, stripping/slicing, functions, Regular expressions, encoding/decoding, and objects/inheritance), and since Dr. Chuck explains the assignments in the lecture, there's really no more advice I can give that I haven't already.

Week 1, "Welcome to the Capstone," introduces you to the last course and briefly explains the two assignments that will be lectured on. One of the differences between the last visualization assignment we did and these projects is that instead of reading data from files we will be reading data directly from web pages (url's). So, since these projects will involve more web-scraping, this week you are having a review quiz (already taken in course 4) to refresh you on internet topics like: UTF-8, ASCII, unicode, and encoding/decoding data. Aside from the review material, feel to skip through the rest of week 1.

Week 2, "Building a Search Engine," introduces us to the first project, a search engine. All the code required for the search engine has already been completed and is included in the "pagerank" folder within the "Sample code" folder that was downloaded at the beginning of the specialization. The "pagerank" folder should contain the following files: spider.py, sprank.py, spreset.py, spjson.py, spdump.py, spider.js, force.js, force.html, and d3.v2.js. The lecture videos break up the Pagerank assignment code (search engine) into three parts; Spidering, Computation, and Visualization. The Spidering part uses the spider.py code to create the spider.sqlite database, connects to a url web page (user-decides), then "spiders" or web crawls for a number (user decides) of subpages, and inserts the data into the database. The Computation part uses the sprank.py code, which contains the "Page Rank" algorithm, the code reads the spider.sqlite database, retrieves web page ID's, and ranks them in terms of visitation frequency, essentially looking for the most popular/useful pages. The database records two ranks, (New & Old Rank), because let's say we chose to "spider" 10 pages and then later wanted to "spider" 30 pages, the previous ranks will need to be recalculated with the algorithm to account for the new pages. The Visualization part uses the spdump.py code to read the spider.sqlite database and uses a Select statement to retrieve the web page counts (frequencies), old rank, new rank, web page ID, and the url for all of the web pages, then prints it to the screen. The Visualization part also uses the spjson.py code to look at all that data and produce a JavaScript file. It's going to write a JavaScript

file, that is fed into the visualization, using d3 (Javascript library used for visualization, this is provided as this is not covered in the Python for Everybody Specialization). Spjson.py joins the links with the webpages, where HTML is not null, or error is not null, in order by the number of inbound links. So, we're looking at the webpages (user decides how many nodes) that have the highest number of inbound links. Spjson.py creates the spider.js file which gives all the information on the website nodes as well as the dimensions (circle weight and line thickness) which will be fed to the force.html file that will link us to the visualization. The force.html file also uses the force.js file (which contains the d3 javascript code that draws the circles and defines the circle colors, and makes the circles bigger and smaller, and connects all the lines in between the circles) as its visualization source. So spider.js is also feeding the force.js file. So when we're all done running the code, we simply open force.html and select a browser to view the visualization of the data. See appendix F.1 for screenshots of the project.

Week 3, “Exploring Data Sources (Project)”, provides a break in between assignments. This week you are simply provided with a list of different common data sources that you could use on the Page Rank code. There’s also an optional choice to post to a discussion post on one of the provided databases. Feel free to skip through this week if desired.

Week 4, “Spidering and Modeling Email Data,” introduces us to the first half of the second project, the second half of the project is done during week 6. The second project is very similar to the previous one, except instead of spidering web pages, this time we will spider email data, about 1gb of it. Just like the Page Rank project/assignment, all the code required for this project has already been completed and is included in the “gmane” folder within the “Sample code” folder that was downloaded at the beginning of the specialization. The “gmane” folder should contain the following files: gmane.py, gmodel.py, mapping.sqlite, gbasic.py, gline.py, gline.js, gline.htm, gwrd.py, gwrd.js, gwrd.htm, d3.v2.js, and d3.layout.cloud.js. The lectures for the whole Gmane assignment code are also broken into three parts: Retrieval (similar to Spidering), Modeling (similar to Computation), and Visualization. This week we only focus on Retrieval and Modeling. The Retrieval part uses the gmane.py file to create the content.sqlite database, connects to mbox.dr-chuck.net, then “spiders” the email data (gmane.py uses re.findall and Regex to parse the data they need), then inserts the data into the content.sqlite database. The Modeling part uses the gmodel.py file to clean up the data (as it’s still fairly raw email data) that was retrieved by using the gmane.py code. The gmodel.py code creates a new database called index.sqlite, connects to the mapping.sqlite database (created by Dr. Chuck to track people who changed email addresses, maps the old emails to the new ones, don’t need to worry about this so much) to create a dictionary of the data. The gmodel.py code then reads the content.sqlite database and continues to use Regex and stripping to parse the data

as well as using zlib to compress some of the data, then inserts the new data into the index.sqlite database. This concludes the week 4 material, see appendix F.2 for screenshots of the code.

Week 5, “Accessing New Data Sources (Project),” provides you with another break. This week there’s another optional discussion board on using data sources, as well as one of Dr.Chuck’s non-Python related videos. Feel free to skip through this week if desired.

Week 6, “Visualizing Email Data,” we are picking up right back where we left off from week 4, visualization. The Visualization part uses the gbasic.py code to read the index.sqlite database, creates dictionaries (senders, messages, sendcounts, sendorgs) as it’s faster than using a Select statement in SQL, then uses a for-loop to go through all the messages, sorts them (sorted function), and then prints out the top email list participants and organizations (amount depends user). The Visualization part also uses the gword.py code to read the index.sqlite database and use the data to produce the gword.js (Java Script) file which will contain all the dimension and design information for the d3 visualization files, which will then be fed to the gword.htm file. Once we’re done running all the code, we just open the gword.htm file and select a browser to be linked to the “Word Cloud” visualization. Similarly, we will follow the same steps, but with different code to create a different visualization. This time we will use the gline.py (breaks the data into months for the top 10 organizations) code to produce the gline.js file which contains the dimension and design information for the d3 visualization files, which will then be fed to the gline.htm file. Then after running the code we can open the gline.htm file and select a browser to be linked to the “Time-Line” visualization which represents developer participation by month. See appendix F.3 for screenshots of the code.

Week 7, “Visualizing New Data Sources (Project)”, congratulations, you made it to the end of the course, you can now relax. This week’s material is just another optional discussion board asking you to reflect on the assignment and what experience you think was gained by taking the Python for Everybody Specialization.

Upon the successful completion of each course in the specialization you will receive your certificate. Once obtaining all five certificates you will have earned the Python for Everybody Specialization certification. See appendix to view my certifications, hopefully seeing mine will encourage you to get your own. Python is a very sought out skill in the tech industry, those who have a thorough understanding of the language are only benefiting themselves, and setting themselves up for success.

## Appendix

### A. Work Log:

Date	Start Time	End Time	Hrs	Activity
1/8/19	8pm	12am	4	Research: creating WinPE USB, how to load it on a server
1/10/19	6pm	7pm	1	Met with Best Buy, ultimately rejected initial project idea
1/10/19	5pm	6:30pm	1.5	Met with OfficeMax, ultimately rejected initial project idea
1/17/19	8am	8:30am	0.5	Research: Found out about the Coursera Python Specialization
1/18/19	8pm	9pm	1	Submitted financial aid application for Coursera
2/7/19	3pm	9pm	6	Watched Python Youtube tutorial, had to pause/replay for notes
2/11/19	10am	11am	1	Coursera – Chapter 1: Why We Program
2/20/19	4pm	6:30pm	2.5	Reviewed Youtube tutorial, quick review to refresh
2/21/19	7pm	12am	5	Course 1: Week 1, Week 2, Week 3, Week 4
2/23/19	6pm	11am	5	Course 1: Week 5, Week 6, Week 7
3/14/19	4pm	6:10pm	2.17	Course 2: Week 1
3/18/19	5pm	9pm	4	Course 2: Week 2, Week 3. Week 4
3/23/19	8:30pm	12:30am	4	Course 2: Week 5, Week 6, Week 7
4/26/19	5pm	10pm	5	Course 3: Week 1, Week 2
4/27/19	8am	12:30pm	4.5	Course 3: Week 3
5/5/19	12pm	6pm	5	Course 3: Week 4
5/19/19	5pm	8pm	3	Course 3: Week 5
5/20/19	8am	10am	2	Course 3: Week 5
8/5/19	12pm	6:20pm	4	Took a break, needed to review Youtube tutorial and Course 3
8/6/19	8am	1pm	5	Course 3: Week 6
8/8/19	10am	5pm	5	Course 4: Week 1, Week 2
8/10/19	12pm	5pm	4	Course 4: Week 3
8/13/19	6pm	10pm	4	Course 4: Week 4
8/17/19	8pm	11:30	3.5	Course 4: Week 5

8/19/19	8am	12pm	3	Course 5: Week 1, Week 2, Week 3
8/23/19	8pm	1:30pm	4	Course 5: Week 4, Week 5, Week 6, Week 7
8/28/19	9pm	11am	2	Started Final Report
8/29/19	7pm	11am	2	Final Report: Had to re-review early Python
8/30	10pm	12am	2	Final Report: Had to re-review early Python
8/31	9am	11pm	2	Final Report: Had to re-review Python
9/1	10am	2pm	4	Final Report
9/2	4pm	8pm	4	Final Report
9/3	12pm	2pm	2	Final Report
9/3	11pm	10:30am	11.5	Finishing Final Report
<b>Total Amount of Hours = 121.17</b>				

## B. Course 1:

**B.1:** Week 3 - Hello World Assignment: How to print text out to the screen, for integers, no quotes required.

The screenshot shows a programming assignment interface. On the left, a yellow box contains the assignment instructions: "Write a program that uses a `print` statement to say 'hello world' as shown in 'Desired Output'." Below this are buttons for "Check Code", "Reset Code", and a status message "Grade updated on server." The code editor on the left contains the following Python code:

```
1
2 print("hello world")
```

On the right, under "Your Output", it shows the result of running the code: "hello world". Below this is a "Desired Output" box containing "hello world".

**B.2:** Week 4 - Assignment 2.2: Variables (name), how to get user input (input), and combining a string and variable with an expression.

The screenshot shows a programming assignment interface. On the left, a yellow box contains the assignment instructions: "2.2 Write a program that uses `input` to prompt a user for their name and then welcomes them. Note that `input` will pop up a dialog box. Enter Sarah in the pop-up box when you are prompted so your output will match the desired output." Below this are buttons for "Check Code", "Reset Code", and a status message "Grade updated on server." The code editor on the left contains the following Python code:

```
1
2
3 name = input("Enter your name")
4 print("Hello " + name)
```

On the right, under "Your Output", it shows the result of running the code: "Hello Sarah". Below this is a "Desired Output" box containing "Hello Sarah".

Week 4 - Assignment 2.3: In addition to the use of variables and expressions, requires conversion between types, thus the use of `float(input())` to change a string into an integer, and `str(pay)` to do the opposite. Also introduces operators for multiplication to calculate pay.

**2.3** Write a program to prompt the user for hours and rate per hour using input to compute gross pay. Use 35 hours and a rate of 2.75 per hour to test the program (the pay should be 96.25). You should use `input` to read a string and `float()` to convert the string to a number. Do not worry about error checking or bad user data.

Check Code    Reset Code    Grade updated on server.

```

1 # This first line is provided for you
2
3 hrs = float(input("Enter Hours:"))
4 pay_rate = float(input("Enter Pay rate: "))
5 pay = hrs * pay_rate
6
7 print("Pay: " + str(pay))

```

Your Output  
Pay: 96.25

Desired Output  
Pay: 96.25

**B.3:** Week 5 - Assignment 3.1: Introduces basic If statements (If- elif) to improve the code above by taking into account the condition of overtime to get an accurate calculation. Two-way decision, also includes the use of comparison operators (<,>).

**3.1** Write a program to prompt the user for hours and rate per hour using input to compute gross pay. Pay the hourly rate for the hours up to 40 and 1.5 times the hourly rate for all hours worked above 40 hours. Use 45 hours and a rate of 10.50 per hour to test the program (the pay should be 498.75). You should use `input` to read a string and `float()` to convert the string to a number. Do not worry about error checking the user input - assume the user types numbers properly.

Check Code    Reset Code    Grade updated on server.

```

1 hrs = float(input("Enter Hours:"))
2
3 base_pay = float(input("Enter Pay rate: "))
4
5 if hrs < 40:
6     total_pay = hrs * base_pay
7 elif hrs > 40:
8     total_pay = (40 * base_pay) + (hrs - 40) * (1.5 * base_pay)
9
10 print(total_pay)

```

Your Output  
498.75

Desired Output  
498.75

Week 5 - Assignment 3.3: Multi-way decisions and nested If-statements used to calculate grade.

3.3 Write a program to prompt for a score between 0.0 and 1.0. If the score is out of range, print an error. If the score is between 0.0 and 1.0, print a grade using the following table:

Score Grade

$\geq 0.9$  A  
 $\geq 0.8$  B  
 $\geq 0.7$  C  
 $\geq 0.6$  D  
 $< 0.6$  F

If the user enters a value out of range, print a suitable error message and exit. For the test, enter a score of 0.85.

Your Output

B

[Check Code](#)

[Reset Code](#)

 Grade updated on server.

```
1 score = float(input("Enter Score: "))
2
3 if score <= 1.0 and score >= 0.0:
4     if score >= 0.9:
5         print("A")
6     elif score >= 0.8:
7         print("B")
8     elif score >= 0.7:
9         print("C")
10    elif score >= 0.6:
11        print("D")
12    else:
13        print("F")
14 elif score > 1.0 or score < 0.0:
15     print("Value out of range")
16     exit()
```

Desired Output

B

**B.4: Week 6 - Assignment 4.6:** Functions, how to define them (def), set parameters (h, r/ hrs, rate) and how to call it (computePay(hrs,rate)). Now instead of repeating code to compute the pay, can just call the function.

4.6 Write a program to prompt the user for hours and rate per hour using input to compute gross pay. Pay should be the normal rate for hours up to 40 and time-and-a-half for the hourly rate for all hours worked above 40 hours. Put the logic to do the computation of pay in a function called `computePay()` and use the function to do the computation. The function should return a value. Use 45 hours and a rate of 10.50 per hour to test the program (the pay should be 498.75). You should use `input` to read a string and `float()` to convert the string to a number. Do not worry about error checking the user input unless you want to - you can assume the user types numbers properly. Do not name your variable sum or use the `sum()` function.

Your Output

498.75

[Check Code](#)

[Reset Code](#)

 Grade updated on server.

```
1 def computePay(h, r):
2     if h > 40.0:
3         pay = (r * 40.0) + (h - 40) * 1.5 * r
4     else:
5         pay = h * r
6     return pay
7
8 hrs = float(input("Enter Hours: "))
9 rate = float(input("Enter Pay rate: "))
10 p = computePay(hrs, rate)
11 print(p)
```

Desired Output

498.75

**B.5:** Week 7 - Assignment 5.2: Use of loops (while-loop), breaks, try/except, and a boolean to create the described program.

5.2 Write a program that repeatedly prompts a user for integer numbers until the user enters 'done'. Once 'done' is entered, print out the largest and smallest of the numbers. If the user enters anything other than a valid number catch it with a try/except and put out an appropriate message and ignore the number. Enter 7, 2, bob, 10, and 4 and match the output below.

Check Code    Reset Code    Grade updated on server.

```
1 largest = None
2 smallest = None
3 while True:
4     num = input("Enter a number: ")
5     if num == "done":
6         break
7
8     try:
9         number = int(num)
10    except ValueError:
11        print("Invalid input")
12    else:
13        if smallest is None:
14            smallest = number
15            largest = number
16        elif number > largest:
17            largest = number
18        elif smallest > number:
19            smallest = number
20
21 print("Maximum is", largest)
22 print("Minimum is", smallest)
```

Your Output  
Invalid input  
Maximum is 10  
Minimum is 2

Desired Output  
Invalid input  
Maximum is 10  
Minimum is 2

**C. Course: 2**

**C.1:** Week 1 - Assignment 6.2: Focusing on strings, the assignment has us use the find() function to any white spaces in the text, indexing the white space, and then stripping away the whitespace from the left side by using lstrip().

6.5 Write code using find() and string slicing (see section 6.10) to extract the number at the end of the line below. Convert the extracted value to a floating point number and print it out.

Check Code    Reset Code    Grade updated on server.

```
1 text = "X-DSPAM-Confidence:      0.8475";
2
3 find_space = text.find(" ")
4 number = text[find_space::1]
5 stripNumber = number.lstrip();
6 result = float(stripNumber)
7
8 def print_again(printed):
9     print(printed)
10
11 print_again(result)
```

Your Output  
0.8475

Desired Output  
0.8475

**C.2:** Week 3 - Assignment 7.1: Writing a simple program to open and read a file (`open(fname, "r")`), and then print the file line by line (`.read()` function), in all caps (`.upper()` function), and rid of any white space after the text (`.rstrip()` function). Sample data is provided in the sample code download.

**7.1** Write a program that prompts for a file name, then opens that file and reads through the file, and print the contents of the file in upper case. Use the file `words.txt` to produce the output below.  
You can download the sample data at  
<http://www.py4e.com/code3/words.txt>

**Check Code**   **Reset Code**   **Grade**

updated on server.

```
1 # Use words.txt as the file name
2 fname = input("Enter file name: ")
3 fh = open(fname, "r")
4
5 print(fh.read().upper().rstrip())
6
```

**Your Output**

```
WRITING PROGRAMS OR PROGRAMMING IS A VERY C
REATIVE
AND REWARDING ACTIVITY YOU CAN WRITE PROGR
AMS FOR
MANY REASONS RANGING FROM MAKING YOUR LIVIN
G TO SOLVING
A DIFFICULT DATA ANALYSIS PROBLEM TO HAVING
FUN TO HELPING
SOMEONE ELSE SOLVE A PROBLEM THIS BOOK ASS
UMES THAT
{\EM EVERYONE} NEEDS TO KNOW HOW TO PROGRAM
AND THAT ONCE
YOU KNOW HOW TO PROGRAM, YOU WILL FIGURE OU
T WHAT YOU WANT
TO DO WITH YOUR NEWFOUND SKILLS

WE ARE SURROUNDED IN OUR DAILY LIVES WITH C
OMPUTERS RANGING
```

**Desired Output**

```
WRITING PROGRAMS OR PROGRAMMING IS A VERY C
REATIVE
AND REWARDING ACTIVITY YOU CAN WRITE PROGR
AMS FOR
MANY REASONS RANGING FROM MAKING YOUR LIVIN
G TO SOLVING
A DIFFICULT DATA ANALYSIS PROBLEM TO HAVING
FUN TO HELPING
SOMEONE ELSE SOLVE A PROBLEM THIS BOOK ASS
UMES THAT
{\EM EVERYONE} NEEDS TO KNOW HOW TO PROGRAM
AND THAT ONCE
YOU KNOW HOW TO PROGRAM, YOU WILL FIGURE OU
T WHAT YOU WANT
TO DO WITH YOUR NEWFOUND SKILLS

WE ARE SURROUNDED IN OUR DAILY LIVES WITH C
OMPUTERS RANGING
```

Week 3 - Assignment 7.2: Requires the use of a for loop to go through each line in the file looking for the specified text, while counting the number of lines (cnt) and calculating the average of those values (ans = total / cnt). Sample data is provided in the sample code download.

7.2 Write a program that prompts for a file name, then opens that file and reads through the file, looking for lines of the form:

X-DSPAM-Confidence: 0.8475

Count these lines and extract the floating point values from each of the lines and compute the average of those values and produce an output as shown below. Do not use the sum() function or a variable named sum in your solution.

You can download the sample data at <http://www.py4e.com/code3/mbox-short.txt> when you are testing below enter **mbox-short.txt** as the file name.

**Check Code**

**Reset Code**



Grade updated on server.

```
1 # Use the file name mbox-short.txt as the file name
2 fname = input("Enter file name: ")
3 fh = open(fname)
4 cnt = 0
5 total = 0
6 answer = 0
7 for line in fh:
8     if not line.startswith("X-DSPAM-Confidence:") : continue
9     cnt = cnt +1
10    number = float(line[21:])
11    total = number + total
12 ans = total/cnt
13
14 print("Average spam confidence:", ans)
15
16
```

#### Your Output

Average spam confidence: 0.750718518519

#### Desired Output

Average spam confidence: 0.750718518519

**C.3: Week 4 - Assignment 8.4:** Creating a list (lst), using a for loop to go through the file and separate strings into a list of words (palabra = line.rstrip().split()), adding those words (element) into the list we created (lst.append(element)) and alphabetizing it (lst.sort()).

8.4 Open the file **romeo.txt** and read it line by line. For each line, split the line into a list of words using the **split()** method. The program should build a list of words. For each word on each line check to see if the word is already in the list and if not append it to the list. When the program completes, sort and print the resulting words in alphabetical order.

You can download the sample data at <http://www.py4e.com/code3/romeo.txt>

**Check Code**

**Reset Code**



Grade

updated on server.

```
1 fname = input("Enter file name: ")
2 fh = open(fname)
3 lst = list()
4 for line in fh:
5     palabra = line.rstrip().split()
6     for element in palabra:
7         if element in lst:
8             continue
9         else:
10            lst.append(element)
11 lst.sort()
12 print(lst)
13
```

#### Your Output

```
['Arise', 'But', 'It', 'Juliet', 'Who', 'already',
 'and', 'breaks', 'east', 'envious',
 'fair', 'grief', 'is', 'kill', 'light', 'mon',
 'pale', 'sick', 'soft', 'sun', 'the',
 'through', 'what', 'window', 'with', 'yonder']
```

#### Desired Output

```
['Arise', 'But', 'It', 'Juliet', 'Who', 'already',
 'and', 'breaks', 'east', 'envious',
 'fair', 'grief', 'is', 'kill', 'light', 'mon',
 'pale', 'sick', 'soft', 'sun', 'the',
 'through', 'what', 'window', 'with', 'yonder']
```

Week 4 - Assignment 8.5: Using a for loop with an if statement to read through the file looking for lines (strings) that start with "From:" (if not line.startswith("From: "): continue), creating a list of them (words = line.split()), and printing out the strings with the "From:" removed by slicing them (print(words[1])).

8.5 Open the file mbox-short.txt and read it line by line. When you find a line that starts with 'From ' like the following line:

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

You will parse the From line using split() and print out the second word in the line (i.e. the entire address of the person who sent the message). Then print out a count at the end.

Hint: make sure not to include the lines that start with 'From:'.

You can download the sample data at <http://www.py4e.com/code3/mbox-short.txt>

**Check Code** **Reset Code** **Grade updated on server.**

```
1 fname = input("Enter file name: ")
2 if len(fname) < 1 : fname = "mbox-short.txt"
3
4 fh = open(fname)
5 count = 0
6
7 for line in fh:
8     line = line.rstrip()
9     if not line.startswith('From '): continue
10    words = line.split()
11    print(words[1])
12    count +=1
13
14
15 print("There were", count, "lines in the file with From as the first word")
16
```

**Your Output**

```
stephen.marquard@uct.ac.za
louis@media.berkeley.edu
zqian@umich.edu
rjlowe@iupui.edu
zqian@umich.edu
rjlowe@iupui.edu
cwen@iupui.edu
cwen@iupui.edu
gsilver@umich.edu
gsilver@umich.edu
zqian@umich.edu
gsilver@umich.edu
wagnermr@iupui.edu
zqian@umich.edu
antranig@caret.cam.ac.uk
gopal.ramasammycook@gmail.com
david.horwitz@uct.ac.za
david.horwitz@uct.ac.za
```

**Desired Output**

```
stephen.marquard@uct.ac.za
louis@media.berkeley.edu
zqian@umich.edu
rjlowe@iupui.edu
zqian@umich.edu
rjlowe@iupui.edu
cwen@iupui.edu
cwen@iupui.edu
gsilver@umich.edu
gsilver@umich.edu
zqian@umich.edu
gsilver@umich.edu
wagnermr@iupui.edu
zqian@umich.edu
antranig@caret.cam.ac.uk
gopal.ramasammycook@gmail.com
david.horwitz@uct.ac.za
david.horwitz@uct.ac.za
```

**C.4:** Week 5 - Assignment 9.4: Similar to the previous code, after creating the list (palabras) of the second word of those lines, we create a dictionary (cnt = dict()) that maps the sender's email to a count of the number of times they appear (2nd for loop). The last for loop reads through the dictionary to find the most common sender.

9.4 Write a program to read through the `mbox-short.txt` and figure out who has sent the greatest number of mail messages. The program looks for 'From ' lines and takes the second word of those lines as the person who sent the mail. The program creates a Python dictionary that maps the sender's mail address to a count of the number of times they appear in the file. After the dictionary is produced, the program reads through the dictionary using a maximum loop to find the most prolific committer.

**Your Output**  
cwen@iupui.edu 5

**Check Code**   **Reset Code**   **Grade updated on server.**

```
1 name = input("Enter file:")
2 if len(name) < 1 : name = "mbox-short.txt"
3 handle = open(name)
4 txt = handle.read()
5
6 palabras = list()
7
8 for line in handle:
9     if not line.startswith("From"): continue
10    line = line.split()
11    palabras.append(line[1])
12
13 cnt = dict()
14
15 for word in palabras:
16     cnt[word] = cnt.get(word, 0) + 1
17
18 maximumvalue = None
19 maximumkey = None
20
21 for key,val in cnt.items():
22     if val > maximumvalue:
23         maximumvalue = val
24         maximumkey = key
25
26 print(maximumkey, maximumvalue)
```

**Desired Output**  
cwen@iupui.edu 5

**C.5:** Week 6 - Assignment 10.2: Tuples come into play after the first for loop, when the list (lista) is made. Tuples were utilized to input the key and value into the new list (lista.append((key, value))) that was created.

10.2 Write a program to read through the **mbox-short.txt** and figure out the distribution by hour of the day for each of the messages. You can pull the hour out from the 'From ' line by finding the time and then splitting the string a second time using a colon.

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

Once you have accumulated the counts for each hour, print out the counts, sorted by hour as shown below.

[Check Code](#)[Reset Code](#) Grade updated on server.

```
1 name = input("Enter file:")
2 if len(name) < 1 : name = "mbox-short.txt"
3 handle = open(name)
4
5 cnt = dict()
6
7 for line in handle:
8     if line.startswith("From "):
9         tiempo = line.split()[5].split(":")
10        cnt [tiempo[0]] = cnt.get(tiempo[0], 0) + 1
11
12 lista = list()
13
14 for key, value in cnt.items():
15     lista.append((key, value))
16 lista.sort()
17
18 for hour, cnt in lista:
19     print(hour, cnt)
20
```

**Your Output**

```
04 3
06 1
07 1
09 2
10 3
11 6
14 1
15 2
16 4
17 2
18 1
19 1
```

**Desired Output**

```
04 3
06 1
07 1
09 2
10 3
11 6
14 1
15 2
16 4
17 2
18 1
19 1
```

## D. Course 3:

**D.1:** Week 2 - Extracting Data with Regular Expressions Assignment: Screenshot of the Regex quick guide from the lecture slides, may be difficult to memorize, so it's always a good idea to keep one handy. As the prompt says, you need to save the data files in the same folder where you code is saved or it will not work. After opening the file (hdl = open) and creating a list (numbers), Regex is used in the first for-loop, using re.findall() to extract portions of a string (in our case, integers) that match the regular expression, [0-9]+, which means one or more digits. Then the second loop sums it up.

# Regular Expression Quick Guide

^	Matches the <b>beginning</b> of a line
\$	Matches the <b>end</b> of the line
.	Matches <b>any</b> character
\s	Matches <b>whitespace</b>
\S	Matches <b>any non-whitespace</b> character
*	Repeats a character zero or more times
*?	Repeats a character zero or more times ( <b>non-greedy</b> )
+	Repeats a character one or more times
+?	Repeats a character one or more times ( <b>non-greedy</b> )
[aeiou]	Matches a single character in the listed <b>set</b>
[^XYZ]	Matches a single character <b>not</b> in the listed <b>set</b>
[a-z0-9]	The set of characters can include a <b>range</b>
(	Indicates where string <b>extraction</b> is to start
)	Indicates where string <b>extraction</b> is to end

## Prompt for the assignment:

Welcome Alex Huerta from Using Python to Access Web Data

Your answer is correct, score saved. ×

### Finding Numbers in a Haystack

In this assignment you will read through and parse a file with text and numbers. You will extract all the numbers in the file and compute the sum of the numbers.

#### Data Files

We provide two files for this assignment. One is a sample file where we give you the sum for your testing and the other is the actual data you need to process for the assignment.

- Sample data: [http://py4e-data.dr-chuck.net/regex\\_sum\\_42.txt](http://py4e-data.dr-chuck.net/regex_sum_42.txt) (There are 90 values with a sum=445833)
- Actual data: [http://py4e-data.dr-chuck.net/regex\\_sum\\_189368.txt](http://py4e-data.dr-chuck.net/regex_sum_189368.txt) (There are 99 values and the sum ends with 821)

These links open in a new window. Make sure to save the file into the same folder as you will be writing your Python program. **Note:** Each student will have a distinct data file for the assignment - so only use your own data file for analysis.

#### Data Format

The file contains much of the text from the introduction of the textbook except that random numbers are inserted throughout the text. Here is a sample of the output you might see:

```
Why should you learn to write programs? 7746
12 1929 8827
Writing programs (or programming) is a very creative
7 and rewarding activity. You can write programs for
many reasons, ranging from making your living to solving
8837 a difficult data analysis problem to having fun to helping 128
someone else solve a problem. This book assumes that
everyone needs to know how to program ...
```

The sum for the sample text above is **27486**. The numbers can appear anywhere in the line. There can be any number of numbers in each line (including none).

#### Handling The Data

The basic outline of this problem is to read the file, look for integers using the `re.findall()`, looking for a regular expression of `'[0-9]+'` and then converting the extracted strings to integers and summing up the integers.

#### Turn in Assignment

Enter the sum from the actual data and your Python code below:

Sum: 478821 (ends with 821)

Python code:

```
import re

hdl = open("regex_sum_189368.txt", "r")
rst = 0
number = list()
for line in hdl:
    strnumber = re.findall("[0-9]+", line)
    if len(strnumber) > 0:
        for number in strnumber:
            rst = rst + int(number)
print(rst)
```

Select Language

IDE running my code on the sample data file (left) and on the actual data file (right) with the terminal displaying the output (highlighted):

The screenshot shows a PyCharm interface with two code editors and a terminal window.

**Code Editor 1:**

```
import re

hdl = open("regex_sum_42.txt", "r")
rst = 0
number = list()
for line in hdl:
    strnumber = re.findall("[0-9]+", line)
    if len(strnumber) > 0:
        for number in strnumber:
            rst = rst + int(number)
print(rst)
```

**Code Editor 2:**

```
import re

hdl = open("regex_sum_189368.txt", "r")
rst = 0
number = list()
for line in hdl:
    strnumber = re.findall("[0-9]+", line)
    if len(strnumber) > 0:
        for number in strnumber:
            rst = rst + int(number)
print(rst)
```

**Terminal:**

```
ch 11 assignment x
/Users/alex/PycharmProjects/PY4E/venv/bin/python "/Users/alex/PycharmProjects/PY4E/regex_sum_189368.py"
445833
```

Process finished with exit code 0

**D.2:** Week 3 - Understanding the Request/Response Cycle Assignment: First screenshot shows sample code that is provided to us during lecture. Sockets must be imported, and then defined (mysock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)), then you establish the connection to the host (data.pr4e.org) and port (80, web-server) you desire (mysock.connect(('data.pr4e.org', 80))). Then you create a variable (cmd) that stores a GET command for the exact webpage you want to get data on, as well as encoding it (.encode()) in UTF-8 before sending the command to the host (mysock.send(cmd)). Use of a while-loop to retrieve the data (mysock.recv()), now the data retrieved will be in UTF-8, so it will need to be decoded to understand it (print(data.decode())). To retrieve the information needed for this week's assignment all that's needed is to change the url address in the cmd variable to "<http://data.pr4e.org/intro-short.txt>".

Sample lecture code with output from the web page in the console terminal:

The screenshot shows the PyCharm IDE interface. The top navigation bar has tabs for "12-NetworkedTech.py", "Ch.12 assignment.py", "12-surfingTheWebAssignment.py", and "12-s". The main editor window contains the following Python code:

```
1 # TCP ports
2 # port 80 = web-server
3 # port 443 = https
4
5
6 import socket
7 mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8 mysock.connect(('data.pr4e.org', 80))
9
10 cmd = "GET http://data.pr4e.org/romeo.txt HTTP/1.0\r\n\r\n".encode()
11 mysock.send(cmd)
12
13 while True:
14     data = mysock.recv(512)
15     if len(data) < 1:
16         break
17     print(data.decode())
18 mysock.close()
19
20 while True
```

The "Run" tool window is open, showing the output of the run named "12-NetworkedTech". The output is as follows:

```
Run: 12-NetworkedTech ×
▶ /Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects
HTTP/1.1 200 OK
Date: Sun, 01 Sep 2019 23:51:36 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Sat, 13 May 2017 11:22:22 GMT
ETag: "a7-54f6609245537"
Accept-Ranges: bytes
Content-Length: 167
Cache-Control: max-age=0, no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Connection: close
Content-Type: text/plain

But soft what light through yonder window breaks
It is the east and Juliet is the sun
Arise fair sun and kill the envious moon
Who is already sick and pale with grief

Process finished with exit code 0
```

## Prompt for the assignment with required information entered:

Done

Welcome Alex Huerta from Using Python to Access Web Data

Your answer is correct, score saved.

Exploring the HyperText Transport Protocol

You are to retrieve the following document using the HTTP protocol in a way that you can examine the HTTP Response headers.

- <http://data.pr4e.org/intro-short.txt>

There are three ways that you might retrieve this web page and look at the response headers:

- Preferred: Modify the [socket1.py](#) program to retrieve the above URL and print out the headers and data. Make sure to change the code to retrieve the above URL - the values are different for each URL.
- Open the URL in a web browser with a developer console or FireBug and manually examine the headers that are returned.
- Use the [telnet](#) program as shown in lecture to retrieve the headers and content.

Enter the header values in each of the fields below and press "Submit".

Last-Modified: Sat, 13 May 2017 11:22:22 GMT  
ETag: "1d3-54f6609240717"  
Content-Length: 467  
Cache-Control: max-age=0, no-cache, no-store, must-revalidate  
Content-Type: text/plain

Submit

Code to receive information for the assignment, information needed is in the console terminal:

```
import socket
mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect(('data.pr4e.org', 80))

cmd = "GET http://data.pr4e.org/intro-short.txt HTTP/1.0\r\n\r\n".encode()
mysock.send(cmd)

while True:
    data = mysock.recv(512)
    if len(data) < 1:
        break
    print(data.decode())

mysock.close()

while True > if (len(data) < 1)
```

Run: Ch.12 assignment

```
/Users/alex/PycharmProjects/PY4E/venv/bin/python "/Users/alex/PycharmProject
HTTP/1.1 200 OK
Date: Mon, 02 Sep 2019 00:31:28 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Sat, 13 May 2017 11:22:22 GMT
ETag: "1d3-54f6609240717"
Accept-Ranges: bytes
Content-Length: 467
Cache-Control: max-age=0, no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Connection: close
Content-Type: text/plain

Why should you learn to write programs?

Writing programs (or programming) is a very creative
and rewarding activity. You can write programs
for
many reasons, ranging from making your living to solving
a difficult data analysis problem to having fun to helping
someone else solve a problem. This book assumes that
everyone needs to know how to program, and that once
you know how to program you will figure out what you want
to do with your newfound skills.

Process finished with exit code 0
```

**D.3: Week 4 - Scraping HTML Data with BeautifulSoup:** The Python file, `urllink2.py`, is provided in the Sample code folder that was downloaded at the beginning of the specialization as a reference. Use of `urllib` to read the html from the data file link entered by the user (`html = request.urlopen(input("Enter - ")).read()`), and use of BeautifulSoup to parse the data and extract the numbers (`sopa = BeautifulSoup(html, "html.parser")`). Computing the sum of numbers in the file with a for-loop.

### Prompt for the assignment:

Welcome Alex Huerta from Using Python to Access Web Data

Your current grade on this assignment is: 100%

**Scraping Numbers from HTML using BeautifulSoup** In this assignment you will write a Python program similar to <http://www.py4e.com/code3/urllink2.py>. The program will use `urllib` to read the HTML from the data files below, and parse the data, extracting numbers and compute the sum of the numbers in the file.

We provide two files for this assignment. One is a sample file where we give you the sum for your testing and the other is the actual data you need to process for the assignment.

- Sample data: [http://py4e-data.dr-chuck.net/comments\\_42.html](http://py4e-data.dr-chuck.net/comments_42.html) (Sum=2553)
- Actual data: [http://py4e-data.dr-chuck.net/comments\\_189370.html](http://py4e-data.dr-chuck.net/comments_189370.html) (Sum ends with 20)

You do not need to save these files to your folder since your program will read the data directly from the URL. **Note:** Each student will have a distinct data url for the assignment - so only use your own data url for analysis.

#### Data Format

The file is a table of names and comment counts. You can ignore most of the data in the file except for lines like the following:

```
<tr><td>Modu</td><td><span class="comments">90</span></td></tr>
<tr><td>Kenzie</td><td><span class="comments">88</span></td></tr>
<tr><td>Hubert</td><td><span class="comments">87</span></td></tr>
```

You are to find all the `<span>` tags in the file and pull out the numbers from the tag and sum the numbers.

Look at the [sample code](#) provided. It shows how to find all of a certain kind of tag, loop through the tags and extract the various aspects of the tags.

```
...
# Retrieve all of the anchor tags
tags = soup('a')
for tag in tags:
    # Look at the parts of a tag
    print 'TAG:',tag
    print 'URL:',tag.get('href', None)
    print 'Contents:',tag.contents[0]
    print 'Attrs:',tag.attrs
```

You need to adjust this code to look for `span` tags and pull out the text content of the span tag, convert them to integers and add them up to complete the assignment.

#### Sample Execution

```
$ python3 solution.py
Enter - http://py4e-data.dr-chuck.net/comments_42.html
Count 50
Sum 2...
```

#### Turning in the Assignment

Enter the sum from the actual data and your Python code below:

Sum:  (ends with 20)

Python code:

Modified code to look for span tags and get the total amount of numbers in the those span tags, derived from urllink2.py. Answer in console terminal:

```
3  from urllib import request
4  from bs4 import BeautifulSoup
5  html = request.urlopen(input("Enter - ")).read()
6  sopa = BeautifulSoup(html, "html.parser")
7  tags = sopa('span')
8  sumFin = 0
9  for tag in tags:
10     sumFin = sumFin+int(tag.contents[0])
11 print(sumFin)

Run: 12-surfingTheWebAssignment ×
> /Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/12-surfingTheWebAssignment.py
Enter - http://py4e-data.dr-chuck.net/comments_189370.html
2620
Process finished with exit code 0
```

**Week 4 - Following Links in HTML using BeautifulSoup:** The Python file, `urllinks.py`, is provided in the Sample code folder that was downloaded at the beginning of the specialization as a reference. Use of `urllib` to read the HTML from the data file link entered ("html" variable). Use of BeautifulSoup to parse and extract the "href=" values from the anchor tags ("sopa" variable). For-loop to scan for a tag that is in a particular position ("pos" variable) relative to the first name in the list, follow that link and repeat the process a number of times ("cnt" variable) and report the last name you find.

## Prompt for the assignment:

Welcome Alex Huerta from Using Python to Access Web Data

Your current grade on this assignment is: 100%

### Following Links in Python

In this assignment you will write a Python program that expands on <http://www.py4e.com/code3/urllinks.py>. The program will use `urllib` to read the HTML from the data files below, extract the href= values from the anchor tags, scan for a tag that is in a particular position relative to the first name in the list, follow that link and repeat the process a number of times and report the last name you find.

We provide two files for this assignment. One is a sample file where we give you the name for your testing and the other is the actual data you need to process for the assignment

- Sample problem: Start at [http://py4e-data.dr-chuck.net/known\\_by\\_Fikret.html](http://py4e-data.dr-chuck.net/known_by_Fikret.html)

Find the link at position 3 (the first name is 1). Follow that link. Repeat this process 4 times. The answer is the last name that you retrieve.

Sequence of names: Fikret Montgomery Mhairade Butchi Anayah

Last name in sequence: Anayah

- Actual problem: Start at: [http://py4e-data.dr-chuck.net/known\\_by\\_Alexx.html](http://py4e-data.dr-chuck.net/known_by_Alexx.html)

Find the link at position 18 (the first name is 1). Follow that link. Repeat this process 7 times. The answer is the last name that you retrieve.

Hint: The first character of the name of the last page that you will load is: M

### Strategy

The web pages tweak the height between the links and hide the page after a few seconds to make it difficult for you to do the assignment without writing a Python program. But frankly with a little effort and patience you can overcome these attempts to make it a little harder to complete the assignment without writing a Python program. But that is not the point. The point is to write a clever Python program to solve the program.

### Sample execution

Here is a sample execution of a solution:

```
$ python3 solution.py
Enter URL: http://py4e-data.dr-chuck.net/known_by_Fikret.html
Enter count: 4
Enter position: 3
Retrieving: http://py4e-data.dr-chuck.net/known_by_Fikret.html
Retrieving: http://py4e-data.dr-chuck.net/known_by_Montgomery.html
Retrieving: http://py4e-data.dr-chuck.net/known_by_Mhairade.html
Retrieving: http://py4e-data.dr-chuck.net/known_by_Butchi.html
Retrieving: http://py4e-data.dr-chuck.net/known_by_Anayah.html
```

The answer to the assignment for this execution is "Anayah".

### Turning in the Assignment

Enter the last name retrieved and your Python code below:

Name:  (name starts with M)

Python code:

Expanded version of urllinks.py code to retrieve the last names from the tags. Output in the console terminal:

**D.4:** Week 5 - Extracting Data from XML Assignment: Included my notes from the XML lecture so you can visualize an XML tree and get a better understanding of how XML data works with Python. The assignment requires the use of **urllib** to read the XML data from the URL entered (exmls = request.urlopen(address).read()), then uses the XML element tree to parse and extract the comment counts from the data (tree = ET.fromstring(exmls)). The program finds the counts by using the findall() function and sums them up with the for-loop.

My notes, displays XML data (XML tree) interacting with Python, console terminal shows output:

The screenshot shows the PyCharm IDE interface. The code editor window contains a Python script named Ch.13-w5Notes.py. The script demonstrates how to parse XML using the ElementTree module. It first creates a string of XML data representing a person with a name, phone number, and email address. It then parses this XML into a tree, prints the name, and prints the value of the 'hide' attribute of the email element. A horizontal line follows, after which it prints the XML input string. The script then creates another XML string representing multiple users, parses it, and prints the count of users. Finally, it loops through each user element, printing the name, ID, and the value of the 'x' attribute.

```
1 # Web services and XML
2 import xml.etree.ElementTree as ET
3 data = '''<person>
4     <name>Chuck</name>
5     <phone type="intl">
6         +1 734 303 4456
7     </phone>
8     <email hide="yes"/>
9 </person>'''
10
11 tree = ET.fromstring(data)
12 print('Name:', tree.find('name').text)
13 print('Attr:', tree.find('email').get('hide'))# Attr = attribute
14
15 print("-----")
16
17 input = '''<stuff>
18     <users>
19         <user x="2">
20             <id>001</id>
21             <name>Chuck</name>
22         </user>
23     </users>
24 </stuff>'''
25
26 stuff = ET.fromstring(input)
27 lst = stuff.findall('users/user')
28 print('User count:', len(lst))
29
30 for item in lst:
31     print('Name', item.find('name').text)
32     print('ID', item.find('id').text)
33     print('Attribute', item.get("x"))
```

Run: Ch.13-w5Notes

```
/Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/Ch.13-w5Note
Name: Chuck
Attr: yes
-----
User count: 1
Name Chuck
ID 001
Attribute 2
Process finished with exit code 0
```

## Prompt for the assignment:

Welcome Alex Huerta from Using Python to Access Web Data

Your current grade on this assignment is: 100%

### Extracting Data from XML

In this assignment you will write a Python program somewhat similar to <http://www.py4e.com/code3/geoxml.py>. The program will prompt for a URL, read the XML data from that URL using `urllib` and then parse and extract the comment counts from the XML data, compute the sum of the numbers in the file.

We provide two files for this assignment. One is a sample file where we give you the sum for your testing and the other is the actual data you need to process for the assignment.

- Sample data: [http://py4e-data.dr-chuck.net/comments\\_42.xml](http://py4e-data.dr-chuck.net/comments_42.xml) (Sum=2553)
- Actual data: [http://py4e-data.dr-chuck.net/comments\\_189372.xml](http://py4e-data.dr-chuck.net/comments_189372.xml) (Sum ends with 87)

You do not need to save these files to your folder since your program will read the data directly from the URL. **Note:** Each student will have a distinct data url for the assignment - so only use your own data url for analysis.

### Data Format and Approach

The data consists of a number of names and comment counts in XML as follows:

```
<comment>
  <name>Matthias</name>
  <count>97</count>
</comment>
```

You are to look through all the `<comment>` tags and find the `<count>` values sum the numbers. The closest sample code that shows how to parse XML is [geoxml.py](http://www.py4e.com/code3/geoxml.py). But since the nesting of the elements in our data is different than the data we are parsing in that sample code you will have to make real changes to the code.

To make the code a little simpler, you can use an XPath selector string to look through the entire tree of XML for any tag named 'count' with the following line of code:

```
counts = tree.findall('.//count')
```

Take a look at the Python ElementTree documentation and look for the supported XPath syntax for details. You could also work from the top of the XML down to the comments node and then loop through the child nodes of the comments node.

### Sample Execution

```
$ python3 solution.py
Enter location: http://py4e-data.dr-chuck.net/comments_42.xml
Retrieving http://py4e-data.dr-chuck.net/comments_42.xml
Retrieved 4189 characters
Count: 50
Sum: 2...
```

### Turning in the Assignment

Enter the sum from the actual data and your Python code below:

Sum:  (ends with 87)

Python code:

Code for assignment that receives the sum of the counts, info needed is in the console terminal:

The screenshot shows the PyCharm IDE interface. At the top, there are four tabs: Ch.13-w5Notes.py, Ch.13-w5Assignment.py (which is the active tab), 12-surfingTheWebAssignment.py, and 12-surfingTheWeb.py. The Ch.13-w5Assignment.py code window contains the following Python script:

```
15 from urllib import request
16 import xml.etree.ElementTree as ET
17
18 address = input("Enter location: ")
19 if len(address) < 1:
20     address = "http://python-data.dr-chuck.net/comments_189372.xml"
21 print("Retrieving " + address)
22
23 exmls = request.urlopen(address).read()
24 print("Retrieved: " + str(len(exmls)) + " characters")
25
26 tree = ET.fromstring(exmls)
27
28 cnt = tree.findall('.//count')
29 print("Count: " + str(len(cnt)))
30
31 total = 0
32
33 for count in cnt:
34     total += int(count.text)
35
36 print("Sum: " + str(total))
37
38
```

Below the code editor is the 'Run' tool window. It shows the command run: /Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/Ch.13-w5Assignment.py. The output pane displays the following text:

```
Enter location:
Retrieving http://python-data.dr-chuck.net/comments_189372.xml
Retrieved: 4227 characters
Count: 50
Sum: 2387
Process finished with exit code 0
```

**D.5: Week 6 - Extracting Data from JSON Assignment:** Included my notes from the JSON lecture to help you visualize the JSON architecture and see how Python interacts with JSON and pulls data. The assignment will require you to read the JSON data from the URL entered (info = addy.read()) using `urllib`. Then you will need to use JSON to parse and extract comment counts from the JSON data (data = json.loads(info) and data = data["comments"]). Finally we use the for-loop to sum all of the counts from the file.

My JSON notes from lecture, output from the code is shown in the console terminal:

The screenshot shows the PyCharm IDE interface. The top navigation bar has tabs for "Ch.13 - JSON.py", "ch.13w6JSONAssignment.py", "13.7 GeoJSON.py", and "Ch.13w6G". The main code editor contains Python code for working with JSON data. The code defines a variable `data` containing a nested dictionary, prints it to the console, and then defines another variable `input` containing a list of two dictionaries. It loads this list into `info`, prints the length, and then iterates through each item in `info` to print its name, id, and x attribute. Finally, it prints a summary message and ends with a blank line.

```
1 import json
2 data = '''
3     "name" : "Chuck",
4     "phone" : {
5         "type" : "intl",
6         "number" : "+1 734 303 4456"
7     },
8     "email" : {
9         "hide" : "yes"
10    }
11  }'''
12 # JSON represents data as nested "lists" and "dictionaries"
13 info = json.loads(data)
14 print('Name:', info["name"])
15 print('Hide:', info["email"]["hide"])
16
17 # New example
18 print("-----Done w/Example-----")
19 print(" ")
20
21 input = '''
22     { "id" : "001",
23       "x" :"2",
24       "name" : "Chuck"
25     },
26     { "id" : "009",
27       "x" : "7",
28       "name" : "Chuck"
29   }
30 '''
31
32 info = json.loads(input)
33 print(info)
34 print('User count:', len(info))
35 for item in info:
36     print('Name', item['name'])
37     print('Id', item['id'])
38     print('Attribute', item['x'])
39
40 print("-----Done w/Example-----")
41 print(" ")
```

Run: Ch.13 - JSON

```
/Users/alex/PycharmProjects/PY4E/venv/bin/python "/Users/alex/PycharmProjects/PY4E/Ch.13 - JSON.py
Name: Chuck
Hide: yes
-----Done w/Example-----
[{"id": "001", "x": "2", "name": "Chuck"}, {"id": "009", "x": "7", "name": "Chuck"}]
User count: 2
Name Chuck
Id 001
Attribute 2
Name Chuck
Id 009
Attribute 7
-----Done w/Example-----

Process finished with exit code 0
```

## Prompt for the assignment:

Welcome Alex Huerta from Using Python to Access Web Data

Your current grade on this assignment is: 100%

### Extracting Data from JSON

In this assignment you will write a Python program somewhat similar to <http://www.py4e.com/code3/json2.py>. The program will prompt for a URL, read the JSON data from that URL using `urllib` and then parse and extract the comment counts from the JSON data, compute the sum of the numbers in the file and enter the sum below:

We provide two files for this assignment. One is a sample file where we give you the sum for your testing and the other is the actual data you need to process for the assignment.

- Sample data: [http://py4e-data.dr-chuck.net/comments\\_42.json](http://py4e-data.dr-chuck.net/comments_42.json) (Sum=2553)
- Actual data: [http://py4e-data.dr-chuck.net/comments\\_189373.json](http://py4e-data.dr-chuck.net/comments_189373.json) (Sum ends with 58)

You do not need to save these files to your folder since your program will read the data directly from the URL. **Note:** Each student will have a distinct data url for the assignment - so only use your own data url for analysis.

### Data Format

The data consists of a number of names and comment counts in JSON as follows:

```
{  
    "comments": [  
        {  
            "name": "Matthias",  
            "count": 97  
        },  
        {  
            "name": "Geomer",  
            "count": 97  
        }  
        ...  
    ]  
}
```

The closest sample code that shows how to parse JSON and extract a list is [json2.py](http://www.py4e.com/code3/json2.py). You might also want to look at [geoxml.py](http://www.py4e.com/code3/geoxml.py) to see how to prompt for a URL and retrieve data from a URL.

### Sample Execution

```
$ python3 solution.py  
Enter location: http://py4e-data.dr-chuck.net/comments_42.json  
Retrieving http://py4e-data.dr-chuck.net/comments_42.json  
Retrieved 2733 characters  
Count: 50  
Sum: 2...
```

### Turning in the Assignment

Enter the sum from the actual data and your Python code below:

Sum:  (ends with 58)

Python code:

Code for assignment that calculates the sum of all the numbers in the JSON data, sum is provided in the console terminal:

The screenshot shows the PyCharm IDE interface. At the top, there are four tabs: 'Ch.13 - JSON.py', 'ch.13w6JSONAssignment.py' (which is the active tab), '13.7 GeoJSON.py', and 'Ch.13w6GeoJSONassig'. The main code editor window displays the following Python script:

```
15 import urllib.request, urllib.parse, urllib.error
16 import json
17
18 url = input("Enter location: ")
19 addy = urllib.request.urlopen(url)
20 info = addy.read()
21
22 amount = 0
23
24 while True:
25     if len(url) < 1: break
26
27     print("Retrieving: ", addy)
28     print("Retrieved", len(info), " characters")
29
30     data = json.loads(info)
31     data = data["comments"]
32     # go through each record in the dictionary
33     for item in data:
34         # print("Count: ", item["count"]) # to see each sum/count, un-grey this
35         amount += int(item["count"])
36         # print("Sum: ", amount)
37     print("Total Summation of Counts: ", amount)
38 break
39
```

Below the code editor is the 'Run' tool window. It shows the command run: '/Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/ch.13w6JSONAssignment.py'. The output pane contains the following text:

```
Enter location: http://py4e-data.dr-chuck.net/comments_189373.json
Retrieving: <http.client.HTTPResponse object at 0x108ba6f60>
Retrieved 2716 characters
Total Summation of Counts: 2658
Process finished with exit code 0
```

**Week 6 - Using the GeoJSON API Assignment:** The Python file, geojson.py, is provided in the Sample code folder that was downloaded at the beginning of the specialization as a reference. Use of the API endpoint (<http://py4e-data.dr.chuck.net/json?>) which has a static subset of Google's GeoCode data (lines 34-48 in the code for the assignment). The program then contacts a web service and retrieves JSON data for the web service and parses it (lines 57-62 in the code for the assignment), then receives the first "place\_id" from the JSON data (js = json.loads(data)) and print("Place id: ", js["results"][0]["place\_id"]).

## Prompt for the assignment:

Welcome Alex Huerta from Using Python to Access Web Data

Your current grade on this assignment is: 100%

**Calling a JSON API**

In this assignment you will write a Python program somewhat similar to <http://www.py4e.com/code3/geojson.py>. The program will prompt for a location, contact a web service and retrieve JSON for the web service and parse that data, and retrieve the first place\_id from the JSON. A place ID is a textual identifier that uniquely identifies a place as within Google Maps.

**API End Points**

To complete this assignment, you should use this API endpoint that has a static subset of the Google Data:

<http://py4e-data.dr-chuck.net/json?>

This API uses the same parameter (address) as the Google API. This API also has no rate limit so you can test as often as you like. If you visit the URL with no parameters, you get "No address..." response.

To call the API, you need to provide the address that you are requesting as the **address=** parameter that is properly URL encoded using the `urllib.parse.urlencode()` function as shown in <http://www.py4e.com/code3/geojson.py>

**Test Data / Sample Execution**

You can test to see if your program is working with a location of "South Federal University" which will have a **place\_id** of "ChIJLRAfow6StocRyuUGlcx9\_m8".

```
$ python3 solution.py
Enter location: South Federal University
Retrieving http://...
Retrieved 2418 characters
Place id ChIJLRAfow6StocRyuUGlcx9_m8
```

**Turn In**

Please run your program to find the **place\_id** for this location:

IU

Make sure to enter the name and case exactly as above and enter the **place\_id** and your Python code below. Hint: The first seven characters of the **place\_id** are "ChIJM62 ..."

Make sure to retrieve the data from the URL specified above and **not** the normal Google API. Your program should work with the Google API - but the **place\_id** may not match for this assignment.

place\_id:

Python code:

Modified code to retrieve the first “place\_id” from the “IU” location, derived from “geojson.py”. Outputs from test location (South Federal University) and actual location (IU) displayed in console terminal:

The screenshot shows the PyCharm IDE interface. The top navigation bar has tabs for "13.7 GeoJSON.py", "Ch.13w6GeoJSONAssignment.py", "13.7-UsingAppProgramInterfaces.py", and "Ch.13 - JSON.py". The main area displays the Python code for "Ch.13w6GeoJSONAssignment.py". The code uses the `urllib` and `json` modules to interact with Google's Places API. It prompts the user for a location, constructs a URL with parameters, sends a request, and prints the retrieved place ID if successful or an error message if failed.

```

30 import urllib.request, urllib.parse, urllib.error
31 import json
32 import ssl
33
34 api_key = False
35 # If you have a Google Places API key, enter it here
36 # api_key = 'AIzaSy_IDByT70'
37 # https://developers.google.com/maps/documentation/geocoding/intro
38
39 if api_key is False:
40     api_key = 42
41     serviceurl = 'http://py4e-data.dr-chuck.net/json?'
42 else:
43     serviceurl = 'https://maps.googleapis.com/maps/api/geocode/json?'
44
45 # Ignore SSL certificate errors
46 ctx = ssl.create_default_context()
47 ctx.check_hostname = False
48 ctx.verify_mode = ssl.CERT_NONE
49
50 while True:
51     address = input('Enter location: ')
52     if len(address) < 1: break
53
54     parms = dict()
55     parms['address'] = address
56     if api_key is not False: parms['key'] = api_key
57     url = serviceurl + urllib.parse.urlencode(parms)
58
59     print('Retrieving', url)
60     uh = urllib.request.urlopen(url, context=ctx)
61     data = uh.read().decode()
62     print('Retrieved', len(data), 'characters')
63
64     try:
65         js = json.loads(data)
66         print("Place id: ", js["results"][0]["place_id"])
67     except:
68         js = None
69         print("----- Unable To Retrieve -----")
70
71

```

The bottom panel shows the "Run" tab selected, with the output window displaying the execution of the script and its results. The output shows two runs: one for "South Federal University" and one for "IU".

```

Run: Ch.13w6GeoJSONAssignment
▶ ↑ /Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/Ch.13w6GeoJSONAssignment.py
Enter location: South Federal University
Retrieving http://py4e-data.dr-chuck.net/json?address=South+Federal+University&key=42
Retrieved 2418 characters
Place id: ChIJLRAfow6StocRyuUGlcx9_m8
Enter location: IU
Retrieving http://py4e-data.dr-chuck.net/json?address=IU&key=42
Retrieved 2314 characters
Place id: ChIJM62JZ8JmbIgRNtQVs4ugmTQ

```

## E. Course 4:

**E.1:** Week 1 - Notes on Object Oriented programming (Objects, Classes, Inheritance), from YouTube Python tutorial.

Here we are simply defining (creating) the “Chef” class and assigning it 3 functions/methods: `make_chicken`, `make_salad`, `make_special_dish`.

```
class Chef:  
    def make_chicken(self):  
        print("The chef makes a chicken")  
  
    def make_salad(self):  
        print("The chef makes a salad")  
  
    def make_special_dish(self):  
        print("The chef makes bbq ribs")
```

Next, to introduce inheritance, we are creating another class (Chinese\_Chef) and we are importing the Chef class from the Chef.py file and passing it to the Chinese\_Chef class. This means that the Chinese\_Chef class will have all the functions from the Chef class, plus the new ones (make\_fried\_rice) and by defining the make\_special\_dish again this will overwrite what the function does, for the Chinese\_Chef only.

```
from Chef import Chef  
  
class Chinese_Chef(Chef):  
    def make_fried_rice(self):  
        print("The chef makes fried rice")  
  
    def make_special_dish(self):  
        print("The chef makes orange chicken ")
```

To show how inheritance works, in a separate file we'll import both classes and call on functions from both classes. Notice how the Chinese\_Chef class is able to run the make\_chicken function, and the make\_special\_dish function is different from Chef.

```
from Chef import Chef  
from Chinese_Chef import Chinese_Chef  
  
myChef = Chef()  
myChef.make_chicken()  
myChef.make_special_dish()  
print("-----")  
myChinese_Chef = Chinese_Chef()  
myChinese_Chef.make_chicken()  
myChinese_Chef.make_special_dish()  
  
  
Inheritance x  
/Users/alex/PycharmProjects/untitled/venv/bin/pyt  
The chef makes a chicken  
The chef makes bbq ribs  
  
The chef makes a chicken  
The chef makes orange chicken  
  
Process finished with exit code 0
```

**E.2: Week 2 - Our First Database Assignment:** This is solely executed in SQLite, no Python. After copying and pasting the Create statement and the Delete/Insert statement from the prompt into the “Execute SQL” option in SQLite we are left with an “Ages” table. Then after running the Select statement from the prompt in SQLite we are provided with the information needed to answer the prompt.

Prompt for the assignment:

Welcome Alex Huerta from Using Databases with Python

To get credit for this assignment, perform the instructions below and enter the code you get here:

4B65747369613338

(Hint: starts with 4B6)

## Instructions

If you don't already have it, install the SQLite Browser from <http://sqlitebrowser.org/>.

Then, create a SQLITE database or use an existing database and create a table in the database called "Ages":

```
CREATE TABLE Ages (
    name VARCHAR(128),
    age INTEGER
)
```

Then make sure the table is empty by deleting any rows that you previously inserted, and insert these rows and only these rows with the following commands:

```
DELETE FROM Ages;
INSERT INTO Ages (name, age) VALUES ('Ruan', 19);
INSERT INTO Ages (name, age) VALUES ('Saim', 27);
INSERT INTO Ages (name, age) VALUES ('Ketsia', 38);
INSERT INTO Ages (name, age) VALUES ('Robin', 15);
INSERT INTO Ages (name, age) VALUES ('Mishkat', 15);
```

Once the inserts are done, run the following SQL command:

```
SELECT hex(name || age) AS X FROM Ages ORDER BY X
```

Find the **first** row in the resulting record set and enter the long string that looks like **53656C696E613333**.

**Note:** This assignment must be done using SQLite - in particular, the **SELECT** query above will not work in any other database. So you cannot use MySQL or Oracle for this assignment.

“Ages” table resulting from Create and Delete/Insert statements:

The screenshot shows the DB Browser for SQLite interface. The title bar says "DB Browser for SQLite - /Users/alex/". The toolbar has buttons for New Database, Open Database, Write Changes, Revert Changes, Database Structure, Browse Data (which is selected), Edit Pragmas, and Execute SQL. Below the toolbar, it says "Table: Ages". There are buttons for Filter, Refresh, Print, New Record, and Delete Record. The main area shows a table with two columns: "name" and "age". The data rows are:

	name	age
1	Ruan	19
2	Saim	27
3	Ketsia	38
4	Robin	15
5	Mishkat	15

Output provided from the Select statement, first row is the data needed:

The screenshot shows the DB Browser for SQLite interface. The title bar says "DB Browser for SQLite - /Users/alex/". The toolbar has buttons for New Database, Open Database, Write Changes, Revert Changes, Database Structure, Browse Data, Edit Pragmas, and Execute SQL (which is selected). Below the toolbar, there are several icons for file operations. The main area has three tabs: "SQL 1" (selected), "SQL 2", and "SQL 3". The SQL tab contains the query:

```
1 SELECT hex(name || age) AS X FROM Ages ORDER BY X
```

Below the SQL tab, the results are shown in a table with one column "X". The data rows are:

X
1 4B65747369613338
2 4D6973686B61743135
3 526F62696E3135
4 5275616E3139
5 5361696D3237

At the bottom, the message "Result: 5 rows returned in 16ms" is displayed, followed by "At line 1:" and the original SQL query.

Week 2 - Counting Email in a Database Assignment: The Python file, emaildb.py, is provided in the Sample code folder that was downloaded at the beginning of the specialization as a reference. Use of Python and SQL to create a new database (conn = sqlite3.connect('CountingOrgs.sqlite')). Lines 16-17 create the table “Counts”, lines 27-28 check if the received domains exist in the database. Use of the If-Else statement to insert the domains received from the mbox.txt file (lines 20-25). Lines 38-40 run the Select statement through a for-loop to print the results to the screen in Python. Submit the database that was created to complete the assignment.

### Prompt for the assignment:

Welcome Alex Huerta from Using Databases with Python

To get credit for this assignment, perform the instructions below and upload your SQLite3 database here:

CountingOrgs.sqlite

(Must have a .sqlite suffix)

Hint: The top organizational count is 536.

You do not need to export or convert the database - simply upload the **.sqlite** file that your program creates. See the example code for the use of the **connect()** statement.

## Counting Organizations

This application will read the mailbox data (mbox.txt) and count the number of email messages per organization (i.e. domain name of the email address) using a database with the following schema to maintain the counts.

```
CREATE TABLE Counts (org TEXT, count INTEGER)
```

When you have run the program on **mbox.txt** upload the resulting database file above for grading.

If you run the program multiple times in testing or with different files, make sure to empty out the data before each run.

You can use this code as a starting point for your application: <http://www.py4e.com/code3/emaildb.py>.

The data file for this application is the same as in previous assignments: <http://www.py4e.com/code3/mbox.txt>.

Because the sample code is using an **UPDATE** statement and committing the results to the database as each record is read in the loop, it might take as long as a few minutes to process all the data. The commit insists on completely writing all the data to disk every time it is called.

The program can be speeded up greatly by moving the commit operation outside of the loop. In any database program, there is a balance between the number of operations you execute between commits and the importance of not losing the results of operations that have not yet been committed.

Code for the assignment that produces the sum of the domains in descending order:

```
1 import sqlite3
2
3 # Read the filename or if null use the standard name
4 name = input("Enter file: ")
5 if len(name) < 1: name = "mbox.txt"
6 handle = open(name)
7
8 templist = list()
9 count = 0
10
11 # Open database connection
12 conn = sqlite3.connect('CountingOrgs.sqlite')
13 cur = conn.cursor()
14
15 # Create table or delete its content
16 cur.execute('''CREATE TABLE IF NOT EXISTS Counts (org TEXT, count INTEGER)''')
17 cur.execute('''DELETE FROM Counts''')
18
19 # Find all emails in the From lines
20 for line in handle:
21     line = line.rstrip()
22     if not line.startswith('From '): continue
23     # Get the domains
24     words = line.split()
25     domain = words[1].split("@")
26     # Check if domain exist in the database
27     cur.execute('''SELECT count FROM Counts where org = ?''', (domain[1],))
28     count = cur.fetchone()
29     # If not create an entry
30     if count is None:
31         cur.execute('''INSERT INTO Counts (org, count) VALUES (?,1)''', (domain[1],))
32     else:
33         # If it does update the record
34         cur.execute('''UPDATE Counts SET count = count + 1 WHERE org = ?''', (domain[1],))
35
36 conn.commit()
37
38 table = 'SELECT org, count FROM Counts ORDER BY count DESC'
39 for line in cur.execute(table):
40     print(str(line[0]), ":", str(line[1]))
41
42 conn.close()
```

Run: CH15assignment2

```
/Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/CH15assignment2.py
Enter file: mbox.txt
iupui.edu : 536
umich.edu : 491
indiana.edu : 178
caret.cam.ac.uk : 157
vt.edu : 110
uct.ac.za : 96
media.berkeley.edu : 56
ufp.pt : 28
gmail.com : 25
et.gatech.edu : 17
txstate.edu : 17
whitman.edu : 17
lancaster.ac.uk : 14
bu.edu : 14
stanford.edu : 12
unicon.net : 9
loi.nl : 9
rsmart.com : 8
ucdavis.edu : 1
fhda.edu : 1
utoronto.ca : 1
```

Process finished with exit code 0

E.3: Week 3 - “Multi-Table Database - Tracks” Assignment: Tracks.py and Library.xml are provided in the Sample code folder that was downloaded at the beginning of the specialization, as a reference. I have commented (#) notes on the modified version of the tracks.py file, to explain the code. The gist of it is, lines 15-42 is where we put the

Create statement provided in the prompt (creates the multiple tables and deletes their content), lines 86-109 is where we insert the data retrieved from reading the XML file into the tables, and lines 112-120 is where we put the Select statement (uses JOINS to pull data from multiple tables), provided in the prompt, into a variable, which we can then print out by using a for-loop. Submit the database that has been created (trackdb.sqlite) to complete the assignment.

### Prompt for the assignment:

Welcome Alex Huerta from Using Databases with Python

Your current grade on this assignment is: 100%

To get credit for this assignment, perform the instructions below and upload your SQLite3 database here:

Choose File No file chosen  
 (Must have a .sqlite suffix)  
 Submit

You do not need to export or convert the database - simply upload the **.sqlite** file that your program creates. See the example code for the use of the `connect()` statement.

## Musical Track Database

This application will read an iTunes export file in XML and produce a properly normalized database with this structure:

```
CREATE TABLE Artist (
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    name TEXT UNIQUE
);

CREATE TABLE Genre (
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    name TEXT UNIQUE
);

CREATE TABLE Album (
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    artist_id INTEGER,
    title TEXT UNIQUE
);

CREATE TABLE Track (
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    title TEXT UNIQUE,
    album_id INTEGER,
    genre_id INTEGER,
    len INTEGER, rating INTEGER, count INTEGER
);
```

If you run the program multiple times in testing or with different files, make sure to empty out the data before each run.

You can use this code as a starting point for your application: <http://www.py4e.com/code3/tracks.zip>. The ZIP file contains the `Library.xml` file to be used for this assignment. You can export your own tracks from iTunes and create a database, but for the database that you turn in for this assignment, only use the `Library.xml` data that is provided.

To grade this assignment, the program will run a query like this on your uploaded database and look for the data it expects to see:

```
SELECT Track.title, Artist.name, Album.title, Genre.name
    FROM Track JOIN Genre JOIN Album JOIN Artist
    ON Track.genre_id = Genre.ID AND Track.album_id = Album.id
        AND Album.artist_id = Artist.id
    ORDER BY Artist.name LIMIT 3
```

The expected result of the modified query on your database is: (shown here as a simple HTML table with titles)

Track	Artist	Album	Genre
Chase the Ace	AC/DC	Who Made Who	Rock
D.T.	AC/DC	Who Made Who	Rock
For Those About To Rock (We Salute You)	AC/DC	Who Made Who	Rock

Modified code, derived from track.py, with the output (track names, band, album, and genre) in the console terminal:



```
1 import sqlite3
2 import urllib.request, urllib.parse, urllib.error
3 import xml.etree.ElementTree as ET
4
5 # Get file
6 fname = input('Enter file name: ')
7 if len(fname) < 1: fname = 'Library.xml'
8 uh = open(fname)
9
10 # Open database connection
11 conn = sqlite3.connect('musicLibrary.sqlite')
12 cur = conn.cursor()
13
14 # Create tables or delete their content
15 cur.executescript("""
16     CREATE TABLE IF NOT EXISTS Artist (
17         id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
18         name TEXT UNIQUE);
19
20     CREATE TABLE IF NOT EXISTS Genre (
21         id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
22         name TEXT UNIQUE);
23
24     CREATE TABLE IF NOT EXISTS Album (
25         id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
26         artist_id INTEGER,
27         title TEXT UNIQUE);
28
29     CREATE TABLE IF NOT EXISTS Track (
30         id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
31         title TEXT UNIQUE,
32         album_id INTEGER,
33         genre_id INTEGER,
34         len INTEGER,
35         rating INTEGER,
36         count INTEGER);
37
38     DELETE FROM Artist;
39     DELETE FROM Genre;
40     DELETE FROM Album;
41     DELETE FROM Track
42 """)
43
44
45 # Function that reconstruct the dictionaries in a standard format, from:
46 # <key>Track ID</key><integer>369</integer>
47 # <key>Name</key><string>Another One Bites The Dust</string>
48 # <key>Artist</key><string>Queen</string>
49 to = {"Track ID" : 369} , {"Name" : "Another One Bites The Dust"} , {"Artist" : "Queen"}
50 def lookup(d, key):
51     found = False
52     for child in d:
53         if found: return child.text
54         if child.tag == 'key' and child.text == key:
55             found = True
56     return None
57
58
59 # Parsing xml
60 stuff = ET.parse(fname)
61 # Finding all dict (3rd levels)
62 all = stuff.findall('dict/dict/dict')
63 print('Dict count:', len(all))
```

## Code pt.2 :

```
64
65     # For each dict (track) insert values in the database
66     for entry in all:
67
68         # Cleaning up some incomplete records
69         if (lookup(entry, 'Track ID') is None): continue
70         if (lookup(entry, 'Genre') is None): continue
71
72         # Collecting all fields from the track
73         name = lookup(entry, 'Name')
74         artist = lookup(entry, 'Artist')
75         album = lookup(entry, 'Album')
76         count = lookup(entry, 'Play Count')
77         rating = lookup(entry, 'Rating')
78         length = lookup(entry, 'Total Time')
79         genre = lookup(entry, 'Genre')
80
81         if name is None or artist is None or album is None: continue
82
83         # print(name, artist, album, count, rating, length, genre)
84
85         # Insert or ignore if already exist this record in Artist
86         cur.execute(''INSERT OR IGNORE INTO Artist (name)
87             VALUES ( ? )'', (artist,))
88         cur.execute('SELECT id FROM Artist WHERE name = ? ', (artist,))
89         artist_id = cur.fetchone()[0]
90
91         # Insert or ignore if already exist this record in Album
92         cur.execute(''INSERT OR IGNORE INTO Album (title, artist_id)
93             VALUES ( ?, ? )'', (album, artist_id))
94         cur.execute('SELECT id FROM Album WHERE title = ? ', (album,))
95         album_id = cur.fetchone()[0]
96
97         # Insert or ignore if already exist this record in Genre
98         cur.execute(''INSERT OR IGNORE INTO Genre (name)
99             VALUES ( ? )'', (genre,))
100        cur.execute('SELECT id FROM Genre WHERE name = ? ', (genre,))
101        genre_id = cur.fetchone()[0]
102
103        # Insert or update values in Track
104        cur.execute(''INSERT OR REPLACE INTO Track
105            (title, album_id, len, rating, count, genre_id)
106            VALUES ( ?, ?, ?, ?, ?, ? )'',
107            (name, album_id, length, rating, count, genre_id))
108
109        conn.commit()
110
111        # Result query
112        table = '''SELECT Track.title, Artist.name, Album.title, Genre.name
113            FROM Track JOIN Genre JOIN Album JOIN Artist
114            ON Track.genre_id = Genre.ID and Track.album_id = Album.id
115            AND Album.artist_id = Artist.id
116            ORDER BY Artist.name LIMIT 3'''
117        for line in cur.execute(table):
118            print(str(line[0]), "|", str(line[1]), "|", str(line[2]), "|", str(line[3]))
119
120    conn.close()
```

Run: Ch15w3Assignment

```
/Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/Ch15w3
Enter file name:
dict count: 404
For Those About To Rock (We Salute You) | AC/DC | Who Made Who | Rock
Hells Bells | AC/DC | Who Made Who | Rock
Shake Your Foundations | AC/DC | Who Made Who | Rock
Process finished with exit code 0
```

“musicLibrary.sqlite” database structure, resulting from the Create statement:

The screenshot shows the "Database Structure" tab of the SQLite browser. It displays a tree view of tables and their columns. The tables are:

- Album**: Contains columns id (INTEGER), artist\_id (INTEGER), and title (TEXT). The schema is: CREATE TABLE Album ("id" INTEGER NOT NULL, "artist\_id" INTEGER, "title" TEXT UNIQUE);
- Artist**: Contains columns id (INTEGER) and name (TEXT). The schema is: CREATE TABLE Artist ("id" INTEGER NOT NULL, "name" TEXT UNIQUE);
- Genre**: Contains columns id (INTEGER) and name (TEXT). The schema is: CREATE TABLE Genre ("id" INTEGER NOT NULL, "name" TEXT UNIQUE);
- Track**: Contains columns id (INTEGER), title (TEXT), album\_id (INTEGER), genre\_id (INTEGER), len (INTEGER), rating (INTEGER), and count (INTEGER). The schema is: CREATE TABLE Track ("id" INTEGER NOT NULL, "title" TEXT UNIQUE, "album\_id" INTEGER, "genre\_id" INTEGER, "len" INTEGER, "rating" INTEGER, "count" INTEGER);
- sqlite\_sequence**: Contains columns name (TEXT) and seq (TEXT). The schema is: CREATE TABLE "sqlite\_sequence" ("name" TEXT, "seq" TEXT);

The Select statement from the prompt running in the SQLite browser:

The screenshot shows the "Execute SQL" tab of the SQLite browser. The SQL query is:

```
1 SELECT Track.title, Artist.name, Album.title, Genre.name
2   FROM Track JOIN Genre JOIN Album JOIN Artist
3     ON Track.genre_id = Genre.ID and Track.album_id = Album.id
4       AND Album.artist_id = Artist.id
5 ORDER BY Artist.name LIMIT 3
```

The results are displayed in a table:

	title	name	title	name
1	For Those About To Rock (We Salute You)	AC/DC	Who Made Who	Rock
2	Hells Bells	AC/DC	Who Made Who	Rock
3	Shake Your Foundations	AC/DC	Who Made Who	Rock

At the bottom, the results are summarized as:

Result: 3 rows returned in 20ms  
At line 1:  
SELECT Track.title, Artist.name, Album.title, Genre.name  
FROM Track JOIN Genre JOIN Album JOIN Artist  
ON Track.genre\_id = Genre.ID and Track.album\_id = Album.id  
AND Album.artist\_id = Artist.id  
ORDER BY Artist.name LIMIT 3

**E.4:** Week 4 - “Many Students in Many Courses” Assignment: The python file, roster.py, is provided in the “Sample code” folder that was downloaded at the beginning of the specialization, as a reference. This assignment is fairly simple, all that needs to be done is to modify the code to include a “role” column in the SQL code (lines 44 and 79), as well as the Python code (lines 64, 66, 80). Then after running the modified python file to create the database, you run the Select statement (uses JOIN) provided in the prompt to get the data needed to complete the assignment.

Prompt for the assignment:

---

Your answer is correct, score saved. ×

To get credit for this assignment, perform the instructions below and enter the code you get here:

416164616D736931313030

(Hint: starts with 416)

## Instructions

This application will read roster data in JSON format, parse the file, and then produce an SQLite database that contains a User, Course, and Member table and populate the tables from the data file.

You can base your solution on this code: <http://www.py4e.com/code3/roster/roster.py> - this code is incomplete as you need to modify the program to store the **role** column in the **Member** table to complete the assignment.

Each student gets their own file for the assignment. Download [this file](#) and save it as `roster_data.json`. Move the downloaded file into the same folder as your `roster.py` program.

Once you have made the necessary changes to the program and it has been run successfully reading the above JSON data, run the following SQL command:

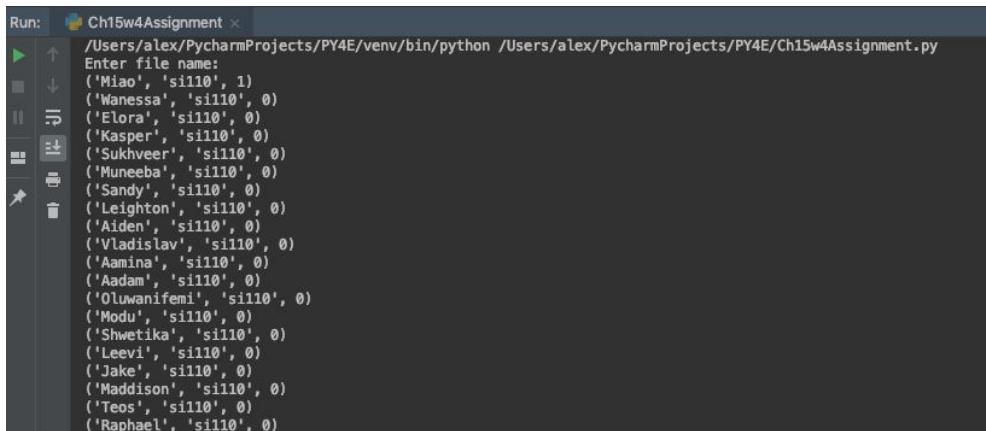
```
SELECT hex(User.name || Course.title || Member.role ) AS X FROM
    User JOIN Member JOIN Course
    ON User.id = Member.user_id AND Member.course_id = Course.id
    ORDER BY X
```

Find the **first** row in the resulting record set and enter the long string that looks like **53656C696E613333**.

Modified version of roster.py code to include the “role column”:

```
19 import json
20 import sqlite3
21
22 conn = sqlite3.connect('rosterdb.sqlite')
23 cur = conn.cursor()
24
25 # Do some setup
26 cur.executescript('''
27 DROP TABLE IF EXISTS User;
28 DROP TABLE IF EXISTS Member;
29 DROP TABLE IF EXISTS Course;
30
31 CREATE TABLE User (
32     id      INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
33     name    TEXT UNIQUE
34 );
35
36 CREATE TABLE Course (
37     id      INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
38     title   TEXT UNIQUE
39 );
40
41 CREATE TABLE Member (
42     user_id    INTEGER,
43     course_id  INTEGER,
44     role       INTEGER,
45     PRIMARY KEY (user_id, course_id)
46 )
47 ''')
48
49 fname = input('Enter file name: ')
50 if len(fname) < 1:
51     fname = 'roster_data.json'
52
53 [
54     [ "Charley", "si110", 1 ],
55     [ "Mea", "si110", 0 ],
56 ]
57 str_data = open(fname).read()
58 json_data = json.loads(str_data)
59
60 for entry in json_data:
61
62     name = entry[0]
63     title = entry[1]
64     role = entry[2]
65
66     print((name, title, role))
67
68     cur.execute('''INSERT OR IGNORE INTO User (name)
69                 VALUES ( ? )''', (name, ))
70     cur.execute('SELECT id FROM User WHERE name = ? ', (name, ))
71     user_id = cur.fetchone()[0]
72
73     cur.execute('''INSERT OR IGNORE INTO Course (title)
74                 VALUES ( ? )''', (title, ))
75     cur.execute('SELECT id FROM Course WHERE title = ? ', (title, ))
76     course_id = cur.fetchone()[0]
77
78     cur.execute('''INSERT OR REPLACE INTO Member
79                  (user_id, course_id, role) VALUES ( ?, ?, ? )''',
80                  (user_id, course_id, role))
81
82 conn.commit()
```

Output from the code above in the console terminal:



The screenshot shows the PyCharm IDE's Run tab with the following details:

- Run configuration: Ch15w4Assignment
- Command: /Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/Ch15w4Assignment.py
- Output window:

```
Enter file name:
('Miao', 'si110', 1)
('Wanessa', 'si110', 0)
('Elora', 'si110', 0)
('Kasper', 'si110', 0)
('Sukheer', 'si110', 0)
('Muneeba', 'si110', 0)
('Sandy', 'si110', 0)
('Leighton', 'si110', 0)
('Aiden', 'si110', 0)
('Vladislav', 'si110', 0)
('Aamina', 'si110', 0)
('Adam', 'si110', 0)
('Oluwanifemi', 'si110', 0)
('Modu', 'si110', 0)
('Shwetika', 'si110', 0)
('Leevi', 'si110', 0)
('Jake', 'si110', 0)
('Maddison', 'si110', 0)
('Teos', 'si110', 0)
('Raphael', 'si110', 0)
```

Results from running the Select statement provided in the prompt, in the SQLite browser, the correct answer is the top row:

The screenshot shows the SQLite browser interface. At the top, there are tabs for "Database Structure", "Browse Data", "Edit Pragmas", and a highlighted "Execute SQL". Below the tabs is a toolbar with icons for file operations like Open, Save, Print, and Database Management. A navigation bar below the toolbar has four entries: "SQL 1" (with an X), "SQL 1" (with an X), "SQL 1" (with an X), and "SQL 1" (highlighted with a blue background). The main area contains the SQL query:

```
1 SELECT hex(User.name || Course.title || Member.role ) AS X FROM
2     User JOIN Member JOIN Course
3     ON User.id = Member.user_id AND Member.course_id = Course.id
4 ORDER BY X
```

Below the query is a table titled "X" with three rows of data:

	X
1	416164616D736931313030
2	416168726F6E736933363430
3	41616D696E61736931313030

At the bottom of the interface, the results are summarized as:

Result: 403 rows returned in 67ms  
At line 1:  
SELECT hex(User.name || Course.title || Member.role ) AS X FROM  
User JOIN Member JOIN Course  
ON User.id = Member.user\_id AND Member.course\_id = Course.id  
ORDER BY X

**E.5: Week 5 - “Retrieving GeoData” Assignment:** The “geodata.zip” file containing geoload.py, geodump.py, where.data, where.js, and where.html is provided in the “Sample code” folder that was downloaded at the beginning of the specialization. For this assignment you need to add a location (school) to the where.data file, then run the geoload.py file to lookup all of the entries in the where.data (including the new one) and produce the geodata.sqlite database. After, run the geodump.py file to read the database and produce the where.js file. Finally, you can open where.html to visualize the map.

## Prompt for the assignment:

Welcome Alex Huerta from Using Databases with Python

### Retrieving GEOData

Download the code from <http://www.py4e.com/code3/geodata.zip> - then unzip the file and edit where.data to add an address nearby where you live - don't reveal where you live. Then run the geoload.py to lookup all of the entries in where.data (including the new one) and produce the geodata.sqlite. Then run geodump.py to read the database and produce where.js. You can run the programs and then scroll back to take your screen shots when the program finishes. Then open where.html to visualize the map. Take screen shots as described below. Make sure that your added location shows in all three of your screen shots.

This is a relatively simple assignment. Don't take off points for little mistakes. If they seem to have done the assignment give them full credit. Feel free to make suggestions if there are small mistakes. Please keep your comments positive and useful. If you do not take grading seriously, the instructors may delete your response and you will lose points.

#### Please Upload Your Submission:

Screen shot (JPG or PNG - Maximum 1MB) of the geoload.py program output in the command line showing your added location

No file chosen

(Please use PNG or JPG files)

Screen shot (JPG or PNG - Maximum 1MB) of the geodump.py program output in the command line showing your added location

No file chosen

(Please use PNG or JPG files)

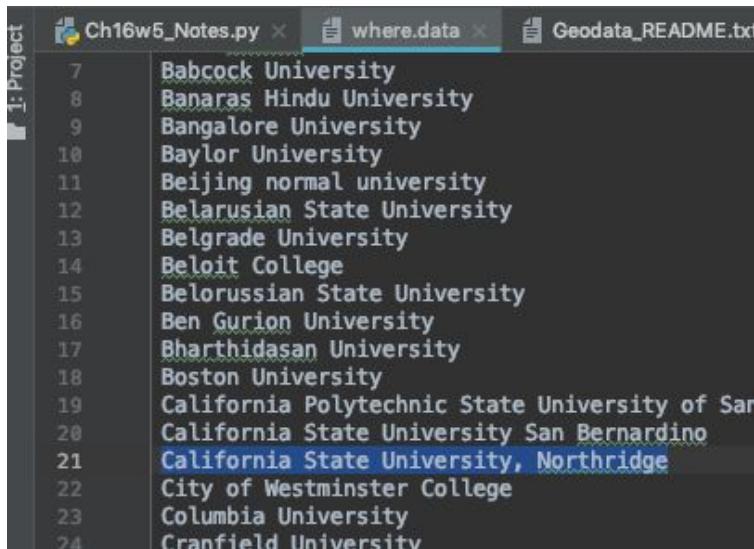
Screen shot (JPG or PNG - Maximum 1MB) of the map zoomed into the location that you added.) of your home page.

No file chosen

(Please use PNG or JPG files)

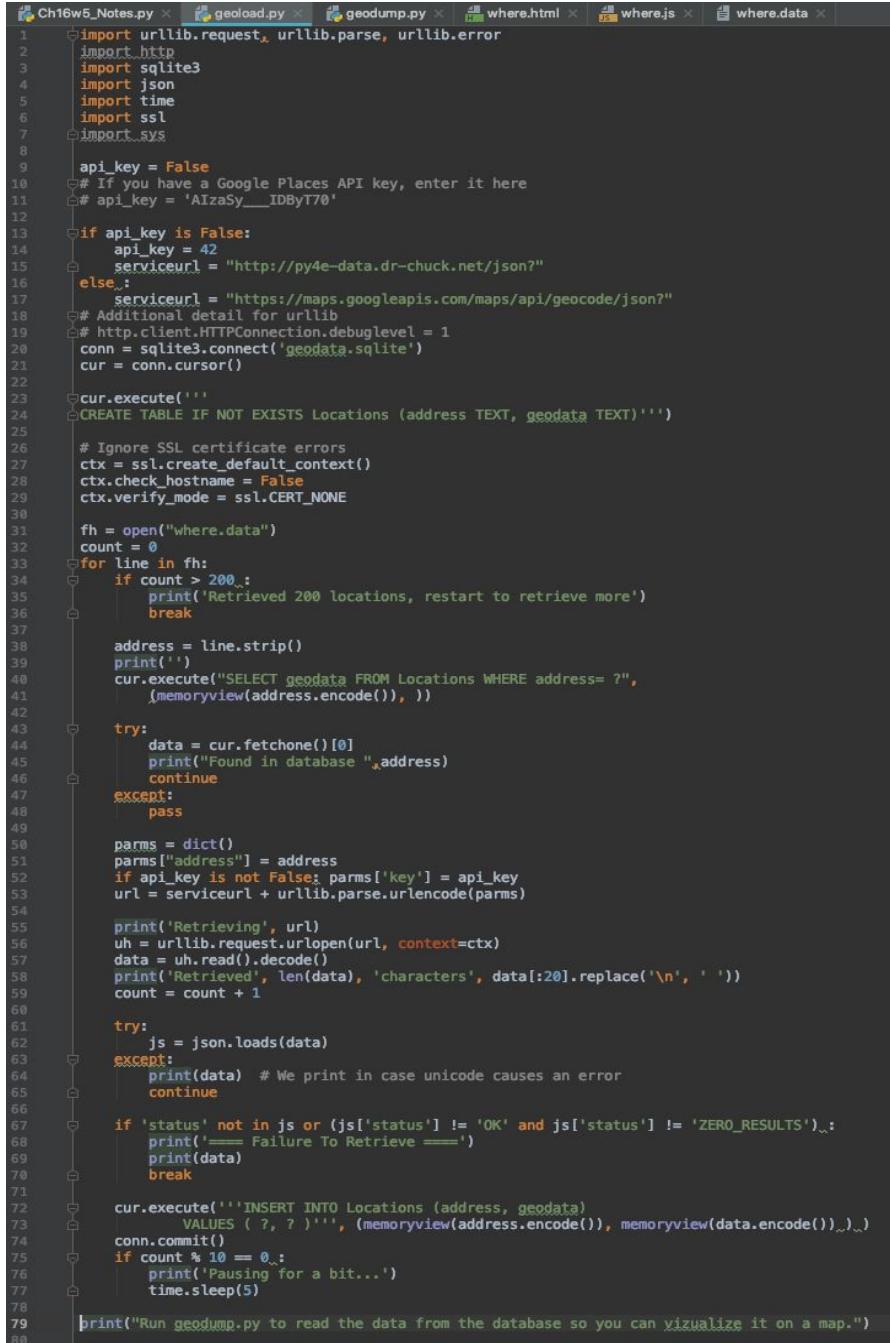
Enter optional comments below

The location I chose to add (Cal State University, Northridge)  
to the where.data file:



```
Project: Ch16w5_Notes.py x where.data x Geodata README.txt
1: 7 Babcock University
2: 8 Banaras Hindu University
3: 9 Bangalore University
4: 10 Baylor University
5: 11 Beijing normal university
6: 12 Belarusian State University
7: 13 Belgrade University
8: 14 Beloit College
9: 15 Belorussian State University
10: 16 Ben Gurion University
11: 17 Bharthidasan University
12: 18 Boston University
13: 19 California Polytechnic State University of San
14: 20 California State University San Bernardino
15: 21 California State University, Northridge
16: 22 City of Westminster College
17: 23 Columbia University
18: 24 Cranfield University
```

Geoload.py creates the geodata.sqlite database, uses the API endpoint (<http://py4e-data.dr.chuck.net/json?>) which has a static subset of Google's GeoCode, then reads the where.data file to lookup all of the entries (including Cal State University Northridge), the program ultimately retrieves the locations from the JSON data:



```
1 import urllib.request, urllib.parse, urllib.error
2 import http
3 import sqlite3
4 import json
5 import time
6 import ssl
7 import sys
8
9 api_key = False
10 # If you have a Google Places API key, enter it here
11 # api_key = 'AIzaSy__IDByT70'
12
13 if api_key is False:
14     api_key = 42
15     serviceurl = "http://py4e-data.dr-chuck.net/json?"
16 else:
17     serviceurl = "https://maps.googleapis.com/maps/api/geocode/json?"
18 # Additional detail for urllib
19 # http.client.HTTPConnection.debuglevel = 1
20 conn = sqlite3.connect('geodata.sqlite')
21 cur = conn.cursor()
22
23 cur.execute('')
24 CREATE TABLE IF NOT EXISTS Locations (address TEXT, geodata TEXT)
25
26 # Ignore SSL certificate errors
27 ctx = ssl.create_default_context()
28 ctx.check_hostname = False
29 ctx.verify_mode = ssl.CERT_NONE
30
31 fh = open("where.data")
32 count = 0
33 for line in fh:
34     if count > 200:
35         print('Retrieved 200 locations, restart to retrieve more')
36         break
37
38     address = line.strip()
39     print('')
40     cur.execute("SELECT geodata FROM Locations WHERE address= ?",
41                 (memoryview(address.encode()), ))
42
43     try:
44         data = cur.fetchone()[0]
45         print("Found in database ", address)
46         continue
47     except:
48         pass
49
50     parms = dict()
51     parms["address"] = address
52     if api_key is not False: parms['key'] = api_key
53     url = serviceurl + urllib.parse.urlencode(parms)
54
55     print('Retrieving', url)
56     uh = urllib.request.urlopen(url, context=ctx)
57     data = uh.read().decode()
58     print('Retrieved', len(data), 'characters', data[:20].replace('\n', ' '))
59     count = count + 1
60
61     try:
62         js = json.loads(data)
63     except:
64         print(data) # We print in case unicode causes an error
65         continue
66
67     if 'status' not in js or (js['status']) != 'OK' and js['status'] != 'ZERO_RESULTS':
68         print('==== Failure To Retrieve ====')
69         print(data)
70         break
71
72     cur.execute('''INSERT INTO Locations (address, geodata)
73                           VALUES ( ?, ? )''', (memoryview(address.encode()), memoryview(data.encode())))
74     conn.commit()
75     if count % 10 == 0:
76         print('Pausing for a bit...')
77         time.sleep(5)
78
79 print("Run geodump.py to read the data from the database so you can visualize it on a map.")
```

Output from geoload.py showing my added location (screenshot required for submission):

```
Run: geoload
↑ Found in database Bharthidasan University
↓ Found in database Boston University
☰ Found in database California Polytechnic State University of San Luis Obispo
☰ Found in database California State University San Bernardino
☰ Retrieving http://py4e-data.dr-chuck.net/json?address=California+State+University+Northridge&key=42
Retrieved 2486 characters { "results" : [
    Found in database City of Westminster College
    Found in database Columbia University
    Found in database Cranfield University
```

Geodata.sqlite database created by geoload.py:

Database Structure    Browse Data

Table: Locations

	address	geodata
1	AGH University ...	{ "results" : [
2	Academy of Fin...	{ "results" : [
3	American Unive...	{ "results" : [
4	Arizona State U...	{ "results" : [
5	Athens Informat...	{ "results" : [
6	BITS Pilani	{ "results" : [
7	Babcock Univers...	{ "results" : [
8	Banaras Hindu ...	{ "results" : [
9	Bangalore Unive...	{ "results" : [
10	Baylor University	{ "results" : [
11	Beijing normal ...	{ "results" : [
12	Belarusian State...	{ "results" : [
13	Belgrade Univer...	{ "results" : [
14	Beloit College	{ "results" : [
15	Belorussian Stat...	{ "results" : [
16	Ben Gurion Univ...	{ "results" : [
17	Bharthidasan Un...	{ "results" : [
18	Boston University	{ "results" : [
19	California Polute...	{ "results" : [

1 - 19 of 300

Geodump.py reads the database and produces the where.js file. Geodump.py uses SQL to Select the locations from the database and JSON to retrieve the locations and parse them with their coordinates (screenshot required for submission) :

The screenshot shows a code editor with multiple tabs and a run output window.

**Code Editor Tabs:**

- Ch16w5\_Notes.py
- where.data
- Geodata\_README.txt
- geoload.py
- geodump.py
- where.html

**Geodump.py Script Content (Lines 1-40):**

```

1 import sqlite3
2 import json
3 import codecs
4
5 conn = sqlite3.connect('geodata.sqlite')
6 cur = conn.cursor()
7
8 cur.execute('SELECT * FROM Locations')
9 fhand = codecs.open('where.js', 'w', "utf-8")
10 fhand.write("myData = [\n")
11 count = 0
12 for row in cur:
13     data = str(row[1].decode())
14     try: js = json.loads(str(data))
15     except: continue
16
17     if not('status' in js and js['status'] == 'OK'): continue
18
19     lat = js["results"][0]["geometry"]["location"]["lat"]
20     lng = js["results"][0]["geometry"]["location"]["lng"]
21     if lat == 0 or lng == 0: continue
22     where = js["results"][0]["formatted_address"]
23     where = where.replace('"', "")
24     try:
25         print(where, lat, lng)
26
27         count = count + 1
28         if count > 1: fhand.write(",\n")
29         output = "[ "+str(lat)+" , "+str(lng)+" , "+where+" ]"
30         fhand.write(output)
31     except:
32         continue
33
34 fhand.write("\n];\n")
35 cur.close()
36 fhand.close()
37 print(count, "records written to where.js")
38 print("Open where.html to view the data in a browser")
39
40

```

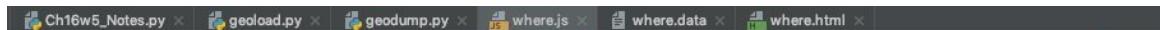
**Run Output (Output of geodump.py):**

```

Run: geodump
5, Vrienerlaan, 7522 NB Enschede, Netherlands 52.23979/7.40000001 6.8499883
201 Presidents Cir, Salt Lake City, UT 84112, USA 40.7649368 -111.8421021
Universitätsring 1, 1010 Wien, Austria 48.21318549999999 16.3600504
Krakowskie Przedmieście 26/28, 00-927 Warszawa, Poland 52.2403463 21.0186012
Seattle, WA 98195, USA 47.65533509999999 -122.3035199
18115 Campus Way NE, Bothell, WA 98011, USA 47.7589 -122.1906495
200 University Ave W, Waterloo, ON N2L 3G1, Canada 43.4722854 -80.5448576
Building 21, 11000 University Pkwy, Pensacola, FL 32514, USA 30.5474687 -87.216141
Madison, WI 53706, USA 43.076592 -89.4124875
Canal Rd, Quaid-i-Azam Campus, Lahore, Punjab, Pakistan 31.47898409999999 74.2661627
1 Jan Smuts Ave, Johannesburg, 2000, South Africa -26.1916888 28.0327614
Saulėtekio al. 11, Vilnius 10221, Lithuania 54.7224678 25.3376688
Universiteto g. 3, Vilnius 01513, Lithuania 54.6825757 25.2876469
907 Floyd Ave, Richmond, VA 23284, USA 37.5483122 -77.4526805
Blacksburg, VA 24061, USA 37.22838429999999 -80.42341669999999
666, Upper Indira Nagar, Bibwewadi, Pune, Maharashtra 411037, India 18.464077 73.867619
Krakowskie Przedmieście 26/28, 00-927 Warszawa, Poland 52.2403463 21.0186012
Pullman, WA, USA 46.7319225 -117.1542121
Wayne State, Detroit, MI, USA 42.3590346 -83.07087369999999
3848 Harrison Blvd, Ogden, UT 84408, USA 41.1918555 -111.9449141
Herzl St 234, Rehovot, Israel 31.9045055 34.8083407
4001 700 E #700, Salt Lake City, UT 84107, USA 40.6848059 -111.8698679
3800 Victory Pkwy, Cincinnati, OH 45207, USA 39.149448 -84.47363740000002
18111 Nordhoff St, Northridge, CA 91330, USA 34.2410366 -118.5276745
Shaibet an Nakareyah, Zagazig, Ash Sharqia Governorate 44519, Egypt 30.5883084 31.4831937
Ashfaq Ahmed Road, H-8/2. H 8/2 H-8, Islamabad, Islamabad Capital Territory 44000, Pakistan 33.68
Tempe, AZ 85281, USA 33.4242399 -111.9280527
Gerais, Paineiras - MG, 35622-000, Brazil -18.9494274 -45.4934107

```

## Where.js file, the result from running geodump.py:

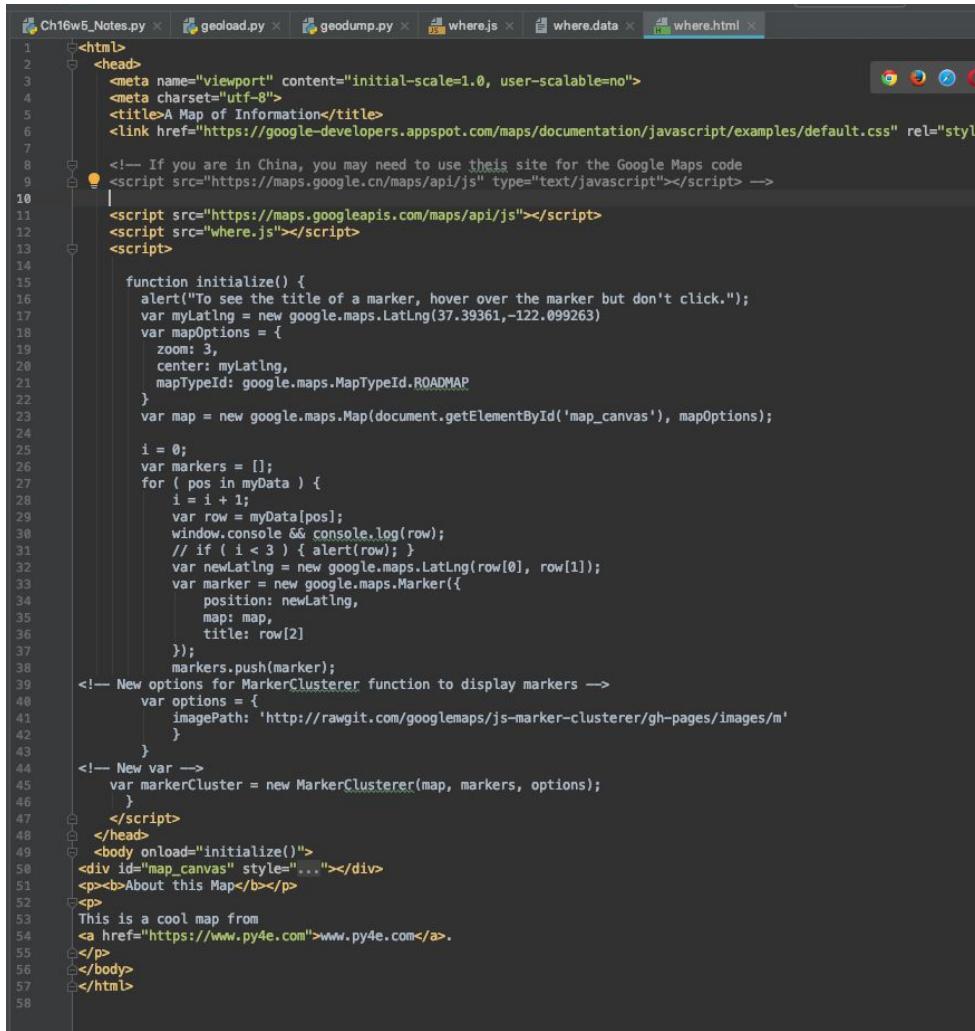


\*.js files are supported by IntelliJ IDEA Ultimate Edition

Check I

```
1 myData = [
2     [50.06688579999999, 19.9136192, 'aleja Adama Mickiewicza 30, 30-059 Kraków, Poland'],
3     [52.2394019, 21.0150792, 'Krakowskie Przedmieście 5, 00-068 Warszawa, Poland'],
4     [30.018923, 31.499674, 'AUC Ave, Cairo Governorate 11835, Egypt'],
5     [33.4242399, -111.9280527, 'Tempe, AZ 85281, USA'],
6     [33.9094132, -83.4603953, '300, 2500 Daniels Bridge Rd, Athens, GA 30606, USA'],
7     [28.3588163, 75.58802039999999, 'Vidya Vihar, Pilani, Rajasthan 333031, India'],
8     [6.8939569, 3.7187158, 'Ilishan-Remo, Nigeria'],
9     [25.2677203, 82.99125819999999, 'Ajjagara, Varanasi, Uttar Pradesh 221005, India'],
10    [12.9504371, 77.5022085, 'Gnana Bharathi Main Rd, Gnana Bharathi, Bengaluru, Karnataka 560056, India'],
11    [31.5497007, -97.1143046, '1301 S University Parks Dr, Waco, TX 76706, USA'],
12    [39.9619537, 116.3662615, '19 Xinjiekou Outer St, Bei Tai Ping Zhuang, Haidian Qu, Beijing Shi, China, 100875'],
13    [53.8938988, 27.5460609, 'Prospekt Nezavisimosti 4, Minsk, Belarus'],
14    [44.8184339, 20.4575676, 'Studentski trg 1, Beograd, Serbia'],
15    [42.5030333, -89.0309048, '700 College St, Beloit, WI 53511, USA'],
16    [53.8938988, 27.5460609, 'Prospekt Nezavisimosti 4, Minsk, Belarus'],
17    ,['Israel', 'נַסְעָה', 'שָׁדֶן בְּגִדְעוֹן 1, בָּרוּשְׁבָּעָה', '31.262218,34.801461'],
18    [10.6779085, 78.74454879999999, 'Palkalaiperur, Tiruchirappalli, Tamil Nadu 620024, India'],
19    [42.3504997, -71.1053991, 'Boston, MA 02215, USA'],
20    [35.3050053, -120.6624942, 'San Luis Obispo, CA 93407, USA'],
21    [34.1821786, -117.3235324, '5500 University Pkwy, San Bernardino, CA 92407, USA'],
22    [51.5210038, -0.1746353, '25 Paddington Green, London W2 1NB, UK'],
23    [40.8075355, -73.9625727, '116th St & Broadway, New York, NY 10027, United States'],
24    [52.0746136, -0.6282833, 'College Rd, Cranfield, Bedford MK43 0AL, UK'],
25    [50.1030364, 14.3912841, 'Zikova 1903/4, 166 36 Praha 6, Czechia'],
26    [43.7044406, -72.2886935, 'Hanover, NH 03755, USA'],
27    [37.3201481, -122.0453953, '21250 Stevens Creek Blvd, Cupertino, CA 95014, USA'],
28    [51.377409, 7.4921548, 'Universitätsstraße 47, 58097 Hagen, Germany'],
29    [48.4331922, 35.0427966, 'Haharina Ave, 72, Dnipropetrovsk, Dnipropetrovsk oblast, Ukraine, 49000'],
30    [38.4306911, 27.1369201, 'Kültür Mahallesi, Cumhuriyet Blv No:144, 35220 Konak/Izmir, Turkey'],
31    [39.9566127, -75.18994409999999, '3141 Chestnut St, Philadelphia, PA 19104, USA'],
32    [30.2863279, -97.7386071, '116 Inner Campus Drive, Austin, TX 78712, USA'],
33    [36.0014258, -78.9382286, 'Durham, NC 27708, USA'],
34    [45.786447, 4.764139000000001, '23 Avenue Guy de Collongue, 69130 Écully, France'],
35    [48.708759, 2.164006, '3 Rue Jolijet Curie, 91190 Gif-sur-Yvette, France'],
36    [36.1660867, -79.5053531, '50 Campus Drive, Elon, NC 27244, USA'],
37    [55.48843069999999, 8.4467103, 'Spangshjerg Kirkevej 103, 6700 Esbjerg, Denmark'],
38    [-2.1481458, -79.9644885, 'Via Perimetral 5, Guayaquil, Ecuador'],
39    [51.2468622, 6.791686899999999, 'Münsterstraße 156, 40476 Düsseldorf, Germany'],
40    [47.722336, 13.0871409, 'Urstein Süd 1, 5412 Puch bei Hallein, Austria'],
41    [-23.6958721, -46.54702839999999, 'Av. Pereira Barreto, 400 – Baeta Neves, São Bernardo do Campo – SP, 09751-000, Brazil'],
42    [45.2461012, 19.8516968, 'Trg Dositeja Obradovića 6, Novi Sad 200314, Serbia'],
43    [40.7529512, -73.4267093, '2350 NY-110, Farmingdale, NY 11735, USA'],
44    [-19.870682, -43.9677359, 'Av. Pres. Antônio Carlos, 6627 – Pampulha, Belo Horizonte – MG, 31270-901, Brazil'],
45    [26.3749876, -80.10106329999999, '777 Glades Rd, Boca Raton, FL 33431, USA'],
46    [42.7702667, -72.0560856, '40 University Dr, Bldg 1, Rm 23451, Nashua, NH 03063, USA']]
```

Where.html file used to open a browser to link you to the visualization of the location data:



```
1  <html>
2  <head>
3      <meta name="viewport" content="initial-scale=1.0, user-scalable=no">
4      <meta charset="utf-8">
5      <title>A Map of Information</title>
6      <link href="https://google-developers.appspot.com/maps/documentation/javascript/examples/default.css" rel="stylesheet">
7
8      <!-- If you are in China, you may need to use this site for the Google Maps code
9      |>
10     <script src="https://maps.google.cn/maps/api/js" type="text/javascript"></script> -->
11
12     <script src="https://maps.googleapis.com/maps/api/js"></script>
13     <script src="where.js"></script>
14     <script>
15
16         function initialize() {
17             alert("To see the title of a marker, hover over the marker but don't click.");
18             var myLatlng = new google.maps.LatLng(37.39361,-122.099263);
19             var mapOptions = {
20                 zoom: 3,
21                 center: myLatlng,
22                 mapTypeId: google.maps.MapTypeId.ROADMAP
23             }
24             var map = new google.maps.Map(document.getElementById('map_canvas'), mapOptions);
25
26             i = 0;
27             var markers = [];
28             for ( pos in myData ) {
29                 i = i + 1;
30                 var row = myData[pos];
31                 // if ( i < 3 ) { alert(row); }
32                 var newLatlng = new google.maps.LatLng(row[0], row[1]);
33                 var marker = new google.maps.Marker({
34                     position: newLatlng,
35                     map: map,
36                     title: row[2]
37                 });
38                 markers.push(marker);
39             <!-- New options for MarkerClusterer function to display markers -->
40             var options = {
41                 imagePath: 'http://rawgit.com/googlemaps/js-marker-clusterer/gh-pages/images/m'
42             }
43         <!-- New var -->
44         var markerCluster = new MarkerClusterer(map, markers, options);
45     }
46     </script>
47 </head>
48 <body onload="initialize()">
49 <div id="map_canvas" style="... "></div>
50 <p><b>About this Map</b></p>
51 <p>
52     This is a cool map from
53     <a href="https://www.py4e.com">www.py4e.com</a>.
54 </p>
55 </body>
56 </html>
```

Link that you are lead to after running the where.html file (in chrome), shows the added location's address (retrieved using the API), visualization of data (screenshot required for submission):

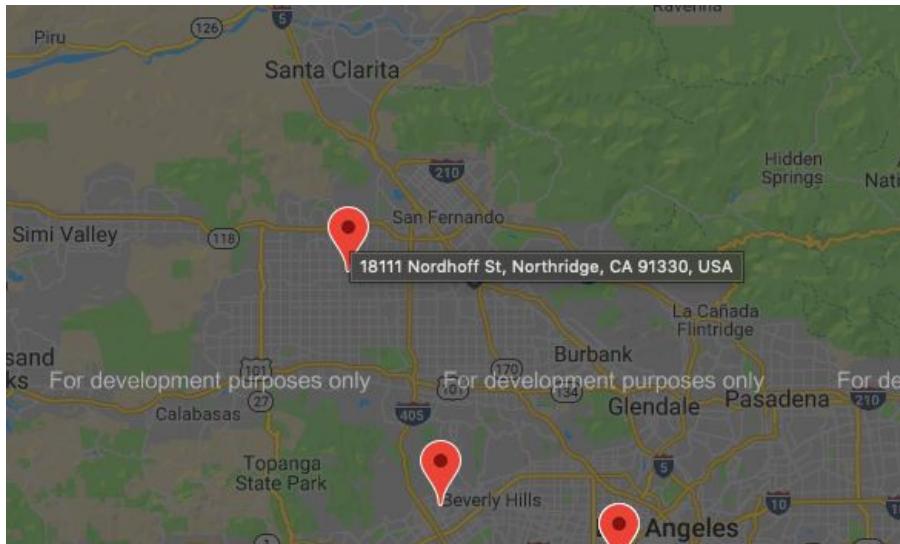
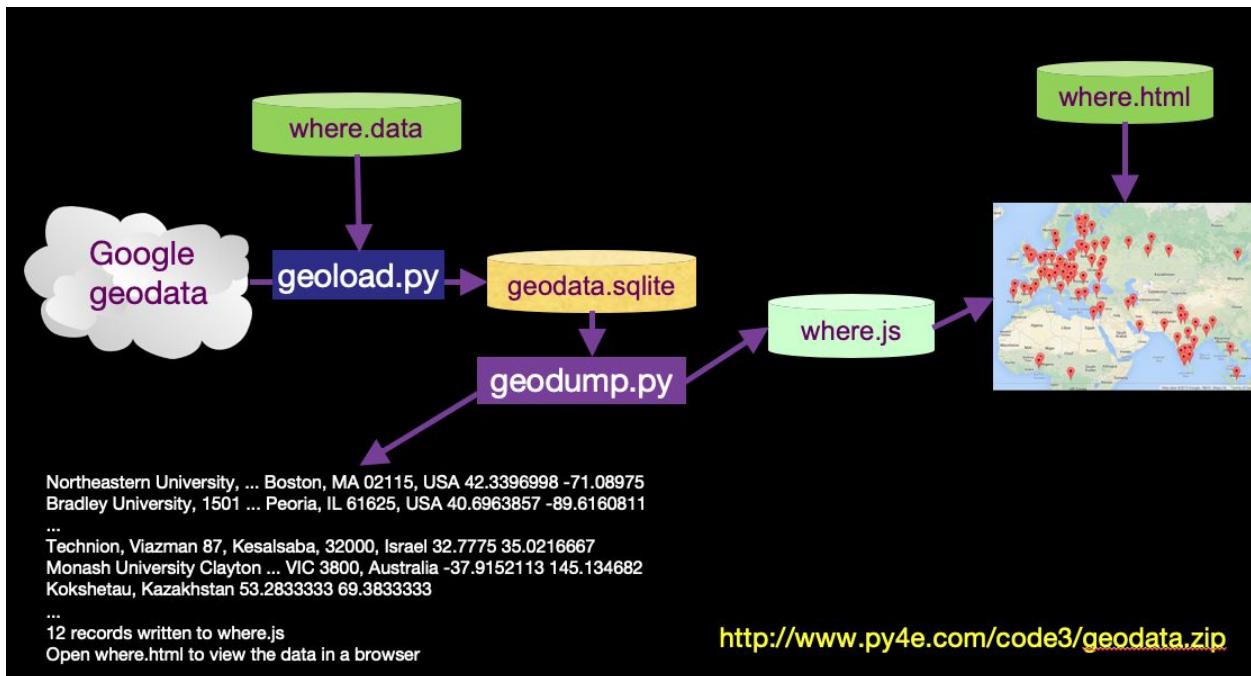


Diagram from the Ch.16 lecture slides, shows how each of the pages of code provided relate to each other, helps to understand what they do:



## F. Course 5:

### F.1: Week 2 - Page Rank Assignment:

## Prompt for the assignment:

Welcome Alex Huerta from Capstone: Retrieving, Processing, and Visualizing Data with Python

### Page Rank

First you will spider 100 pages from <http://python-data.dr-chuck.net/> run the page rank algorithm and take some screen shots. Then you will reset the spider process and spider 100 pages from any other site on the Internet, run the page rank algorithm, and take some screen shots.

Don't take off points for little mistakes. If they seem to have done the assignment give them full credit. Feel free to make suggestions if there are small mistakes. Please keep your comments positive and useful. The course staff will assign the last 30% of the grade and it may take a few days to get graded.

#### Please Upload Your Submission:

Screenshot of spdump.py crawling 100 pages of python-data.dr-chuck.net including rows retrieved. You must run sprank.py first in order to show the page rankings.

Choose File | No file chosen

(Please use PNG or JPG files)

Screenshot showing force.html restricted to 25 nodes for python-data.dr-chuck.net. You must show the starting url.

Choose File | No file chosen

(Please use PNG or JPG files)

Screenshot of spdump.py crawling 100 pages of a website of your choosing (other than dr. Chuck's website) including rows retrieved. You must run sprank.py first in order to show the page rankings.

Choose File | No file chosen

(Please use PNG or JPG files)

Screenshot of force.html restricted to 25 nodes for your chosen website. You must show the starting url.

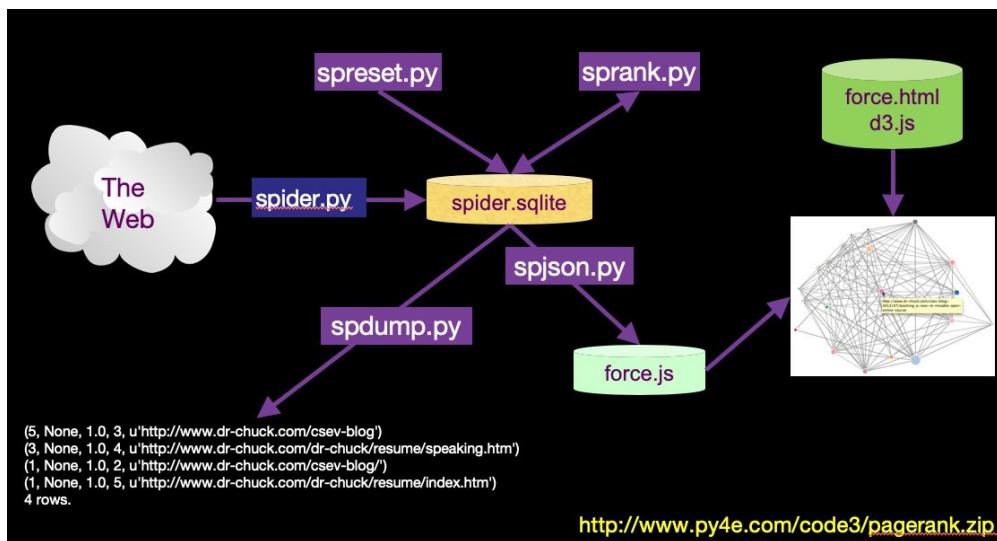
Choose File | No file chosen

(Please use PNG or JPG files)

Enter optional comments below

Submit

Diagram from the Ch.16 lecture slides, shows how each of the pages of code provided relate to each other, helps to understand what they do:



## Spider.py (pt. 1):

```
1 import sqlite3
2 import urllib.error
3 import ssl
4 from urllib.parse import urljoin
5 from urllib.parse import urlparse
6 from urllib.request import urlopen
7 from bs4 import BeautifulSoup
8
9 # Building databases, reading urls, parsing html w/beautiful soup
10 # Ignore SSL certificate errors
11 ctx = ssl.create_default_context()
12 ctx.check_hostname = False
13 ctx.verify_mode = ssl.CERT_NONE
14
15 conn = sqlite3.connect('spider.sqlite')
16 cur = conn.cursor()
17
18 cur.execute('CREATE TABLE IF NOT EXISTS Pages
19     (id INTEGER PRIMARY KEY, url TEXT UNIQUE, html TEXT,
20      error INTEGER, old_rank REAL, new_rank REAL)')
21
22 cur.execute('CREATE TABLE IF NOT EXISTS Links
23     (from_id INTEGER, to_id INTEGER)')
24
25 cur.execute('CREATE TABLE IF NOT EXISTS Webs (url TEXT UNIQUE)')
26
27 # Check to see if we are already in progress...
28 cur.execute('SELECT id,url FROM Pages WHERE html is NULL and error is NULL ORDER BY RANDOM() LIMIT 1')
29 row = cur.fetchone()
30 if row is not None:
31     print("Restarting existing crawl. Remove spider.sqlite to start a fresh crawl.")
32 else:
33     starturl = input('Enter web url or enter: ')
34     if (len(starturl) < 1): starturl = 'http://www.dr-chuck.com/'
35     if (starturl.endswith('/')): starturl = starturl[:-1]
36     web = starturl
37     if (starturl.endswith('.htm') or starturl.endswith('.html')):
38         pos = starturl.rfind('/')
39         web = starturl[:pos]
40
41 if (len(web) > 1):
42     cur.execute('INSERT OR IGNORE INTO Webs (url) VALUES ( ? )', (web,))
43     cur.execute('INSERT OR IGNORE INTO Pages (url,html,new_rank) VALUES ( ?,NULL,1.0 )', (starturl,))
44     conn.commit()
45
46 # Get the current webs
47 cur.execute('SELECT url FROM Webs')
48 webs = list()
49 for row in cur:
50     webs.append(str(row[0]))
51
52 print(webs)
53
54 many = 0
55 while True:
56     if (many < 1):
57         sval = input('How many pages: ')
58         if (len(sval) < 1): break
59         many = int(sval)
60     many = many - 1
61
62 cur.execute('SELECT id,url FROM Pages WHERE html is NULL and error is NULL ORDER BY RANDOM() LIMIT 1')
63 try:
64     row = cur.fetchone()
65     # print row
66     fromid = row[0]
67     url = row[1]
68 except:
69     print('No unretrieved HTML pages found')
70     many = 0
71     break
72
73 print(fromid, url, end=' ')
74
75 # If we are retrieving this page, there should be no links from it
76 cur.execute('DELETE from Links WHERE from_id=?', (fromid,))
77 try:
78     document = urlopen(url, context=ctx)
```

## Spider.py (pt.2):

```
77     try:
78         document = urlopen(url, context=ctx)
79
80         html = document.read()
81         if document.getcode() != 200:
82             print("Error on page: ", document.getcode())
83             cur.execute('UPDATE Pages SET error=? WHERE url=?', (document.getcode(), url))
84
85         if 'text/html' != document.info().get_content_type():
86             print("Ignore non text/html page")
87             cur.execute('DELETE FROM Pages WHERE url=?', (url,))
88             cur.execute('UPDATE Pages SET error=0 WHERE url=?', (url,))
89             conn.commit()
90             continue
91
92         print('('+str(len(html))+')', end=' ')
93
94         soup = BeautifulSoup(html, "html.parser")
95     except KeyboardInterrupt:
96         print('')
97         print('Program interrupted by user...')
98         break
99     except:
100         print("Unable to retrieve or parse page")
101         cur.execute('UPDATE Pages SET error=-1 WHERE url=?', (url,))
102         conn.commit()
103         continue
104
105         cur.execute('INSERT OR IGNORE INTO Pages (url, html, new_rank) VALUES ( ?, NULL, 1.0 )', (url,))
106         cur.execute('UPDATE Pages SET html=? WHERE url=?', (memoryview(html), url))
107         conn.commit()
108
109         # Retrieve all of the anchor tags
110         tags = soup('a')
111         count = 0
112         for tag in tags:
113             href = tag.get('href', None)
114             if ( href is None ): continue
115             # Resolve relative references like href="/contact"
116             up = urlparse(href)
117             if ( len(up.scheme) < 1 ):
118                 href = urljoin(url, href)
119             ipos = href.find('#')
120             if ( ipos > 1 ): href = href[:ipos]
121             if ( href.endswith('.png') or href.endswith('.jpg') or href.endswith('.gif') ): continue
122             if ( href.endswith('/') ): href = href[:-1]
123             # print href
124             if ( len(href) < 1 ): continue
125
126             # Check if the URL is in any of the webs
127             found = False
128             for web in webs:
129                 if ( href.startswith(web) ):
130                     found = True
131                     break
132             if not found: continue
133
134             cur.execute('INSERT OR IGNORE INTO Pages (url, html, new_rank) VALUES ( ?, NULL, 1.0 )', (href,))
135             count = count + 1
136             conn.commit()
137
138             cur.execute('SELECT id FROM Pages WHERE url=? LIMIT 1', (href,))
139             try:
140                 row = cur.fetchone()
141                 toid = row[0]
142             except:
143                 print('Could not retrieve id')
144                 continue
145             # print fromid, toid
146             cur.execute('INSERT OR IGNORE INTO Links (from_id, to_id) VALUES ( ?, ? )', (fromid, toid))
147
148             print(count)
149
150             cur.close()
```

Results from running spider.py, “spidering” 100 pages from <http://python-data.dr-chuck.net>:

```
Run: spider
      /Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/pagerank/spider.py
      Enter web url or enter: http://python-data.dr-chuck.net
      ['python-data.dr-chuck.net', 'http://python-data.dr-chuck.net']
      How many pages: 1
      2 http://python-data.dr-chuck.net (1394) 6
      How many pages: 10
      5 http://python-data.dr-chuck.net/comments\_42.html (3521) 0
      3 http://python-data.dr-chuck.net/geojson Ignore non text/html page
      6 http://python-data.dr-chuck.net/comments\_42.xml Ignore non text/html page
      8 http://python-data.dr-chuck.net/known\_by\_42.html (12020) 100
      74 http://python-data.dr-chuck.net/known\_by\_Elyssa.html (12117) 100
      61 http://python-data.dr-chuck.net/known\_by\_Fruin.html (12054) 100
      179 http://python-data.dr-chuck.net/known\_by\_Aaminah.html (12092) 100
      7 http://python-data.dr-chuck.net/comments\_42.json Ignore non text/html page
      384 http://python-data.dr-chuck.net/known\_by\_Diaz.html (12022) 100
      59 http://python-data.dr-chuck.net/known\_by\_Amaja.html (12022) 100
      How many pages: 100
      119 http://python-data.dr-chuck.net/known\_by\_Elizabeth.html (12090) 100
      4 http://python-data.dr-chuck.net/regex\_sum\_42.txt Ignore non text/html page
      624 http://python-data.dr-chuck.net/known\_by\_Leland.html (12054) 100
      122 http://python-data.dr-chuck.net/known\_by\_Aimee.html (12054) 100
      833 http://python-data.dr-chuck.net/known\_by\_Pushkar.html (12071) 100
      185 http://python-data.dr-chuck.net/known\_by\_Abiha.html (12052) 100
      157 http://python-data.dr-chuck.net/known\_by\_Kianna.html (12017) 100
      1040 http://python-data.dr-chuck.net/known\_by\_Arella.html (12100) 100
      1172 http://python-data.dr-chuck.net/known\_by\_Mikee.html (12064) 100
      1102 http://python-data.dr-chuck.net/known\_by\_Pavit.html (12057) 100
      1227 http://python-data.dr-chuck.net/known\_by\_Nikoleta.html (12055) 100
      1371 http://python-data.dr-chuck.net/known\_by\_Ekaterina.html (12046) 100
      610 http://python-data.dr-chuck.net/known\_by\_Saoirse.html (12076) 100
      1534 http://python-data.dr-chuck.net/known\_by\_Sinem.html (12004) 100
      1232 http://python-data.dr-chuck.net/known\_by\_Nikos.html (12042) 100
      627 http://python-data.dr-chuck.net/known\_by\_Anneroy.html (12040) 100
      15 http://python-data.dr-chuck.net/known\_by\_Ryden.html (12101) 100
      480 http://python-data.dr-chuck.net/known\_by\_Emilly.html (12031) 100
      256 http://python-data.dr-chuck.net/known\_by\_Jaye.html (12055) 100
      1686 http://python-data.dr-chuck.net/known\_by\_Ralfs.html (12040) 100
      360 http://python-data.dr-chuck.net/known\_by\_Andra.html (12019) 100
      2056 http://python-data.dr-chuck.net/known\_by\_Katum.html (12057) 100
      760 http://python-data.dr-chuck.net/known\_by\_Lyall.html (12044) 100
      318 http://python-data.dr-chuck.net/known\_by\_Darach.html (12059) 100
      1177 http://python-data.dr-chuck.net/known\_by\_Tyelor.html (12073) 100
      579 http://python-data.dr-chuck.net/known\_by\_Loki.html (12059) 100
      254 http://python-data.dr-chuck.net/known\_by\_Mahnoor.html (12051) 100
      1954 http://python-data.dr-chuck.net/known\_by\_Leanne.html (12020) 100
      323 http://python-data.dr-chuck.net/known\_by\_Anna.html (12037) 100
      412 http://python-data.dr-chuck.net/known\_by\_Cathal.html (12085) 100
      2105 http://python-data.dr-chuck.net/known\_by\_Lock.html (12044) 100
      1457 http://python-data.dr-chuck.net/known\_by\_Ryleigh.html (12017) 100
      2394 http://python-data.dr-chuck.net/known\_by\_Ryleigh.html (12005) 100
      632 http://python-data.dr-chuck.net/known\_by\_Demira.html (12046) 100
      228 http://python-data.dr-chuck.net/known\_by\_Amara.html (12071) 100
      714 http://python-data.dr-chuck.net/known\_by\_Mali.html (12078) 100
      1152 http://python-data.dr-chuck.net/known\_by\_Pawlo.html (12023) 100
      1767 http://python-data.dr-chuck.net/known\_by\_Carrick.html (12088) 100
      1587 http://python-data.dr-chuck.net/known\_by\_Paul.html (12051) 100
      1477 http://python-data.dr-chuck.net/known\_by\_Nawel.html (12015) 100
      1344 http://python-data.dr-chuck.net/known\_by\_Veronica.html (12010) 100
      2079 http://python-data.dr-chuck.net/known\_by\_Lida.html (12009) 100
      2638 http://python-data.dr-chuck.net/known\_by\_Xiong.html (12079) 100
      2497 http://python-data.dr-chuck.net/known\_by\_Michee.html (12052) 100
      2072 http://python-data.dr-chuck.net/known\_by\_Tansy.html (12055) 100
      685 http://python-data.dr-chuck.net/known\_by\_Masood.html (12025) 100
      351 http://python-data.dr-chuck.net/known\_by\_Youssef.html (12071) 100
      65 http://python-data.dr-chuck.net/known\_by\_Aadan.html (12026) 100
      2080 http://python-data.dr-chuck.net/known\_by\_Remi.html (12020) 100
      148 http://python-data.dr-chuck.net/known\_by\_Stella.html (12111) 100
      1310 http://python-data.dr-chuck.net/known\_by\_Vanni.html (12005) 100
      161 http://python-data.dr-chuck.net/known\_by\_Aimee.html (12052) 100
      2163 http://python-data.dr-chuck.net/known\_by\_Menna.html (12018) 100
      2471 http://python-data.dr-chuck.net/known\_by\_Teighen.html (12075) 100
      2324 http://python-data.dr-chuck.net/known\_by\_Ruta.html (12076) 100
      2783 http://python-data.dr-chuck.net/known\_by\_Alasdair.html (12075) 100
      1444 http://python-data.dr-chuck.net/known\_by\_Elias.html (12044) 100
      1057 http://python-data.dr-chuck.net/known\_by\_Madelyn.html (12015) 100
      268 http://python-data.dr-chuck.net/known\_by\_Kaid.html (12081) 100
      948 http://python-data.dr-chuck.net/known\_by\_Nicole.html (12101) 100
```

Spider.sqlite database created by running spider.py:

Database StructureBrowse DataEdit PragmasExecute SQL

Table: Pages

	<b>id</b>	<b>url</b>	<b>html</b>	<b>error</b>	<b>old_rank</b>	<b>new_rank</b>
1	1	python-data.dr...	NULL	-1	NULL	1.0
2	2	http://python-...	<html><head>	NULL	NULL	1.0
3	5	http://python-...	<html><head>	NULL	NULL	1.0
4	8	http://python-...	<html><head>	NULL	NULL	1.0
5	9	http://python-...	NULL	NULL	NULL	1.0
6	10	http://python-...	NULL	NULL	NULL	1.0
7	11	http://python-...	NULL	NULL	NULL	1.0
8	12	http://python-...	NULL	NULL	NULL	1.0
9	13	http://python-...	NULL	NULL	NULL	1.0
10	14	http://python-...	NULL	NULL	NULL	1.0
11	15	http://python-...	<html><head>	NULL	NULL	1.0
12	16	http://python-...	NULL	NULL	NULL	1.0
13	17	http://python-...	NULL	NULL	NULL	1.0
14	18	http://python-...	NULL	NULL	NULL	1.0
15	19	http://python-...	NULL	NULL	NULL	1.0
16	20	http://python-...	NULL	NULL	NULL	1.0
17	21	http://python-...	NULL	NULL	NULL	1.0
18	22	http://python-...	NULL	NULL	NULL	1.0
19	23	http://python-...	NULL	NULL	NULL	1.0

1 - 19 of 4809

Go to:

## Sprank.py (pt.1):

```
Course5w2_Notes.py x spider.py x sprank.py x spdump.py x spjson.py x spider.js
1 import sqlite3
2 # page rank algorithm
3
4 conn = sqlite3.connect('spider.sqlite')
5 cur = conn.cursor()
6
7 # Find the ids that send out page rank - we only are interested
8 # in pages in the SCC that have in and out links
9 cur.execute(''':SELECT DISTINCT from_id FROM Links''')
10 from_ids = list()
11 for row in cur:
12     from_ids.append(row[0])
13
14 # Find the ids that receive page rank
15 to_ids = list()
16 links = list()
17 cur.execute(''':SELECT DISTINCT from_id, to_id FROM Links''')
18 for row in cur:
19     from_id = row[0]
20     to_id = row[1]
21     if from_id == to_id: continue
22     if from_id not in from_ids: continue
23     if to_id not in from_ids: continue
24     links.append(row)
25     if to_id not in to_ids: to_ids.append(to_id)
26
27 # Get latest page ranks for strongly connected component
28 prev_ranks = dict()
29 for node in from_ids:
30     cur.execute(''':SELECT new_rank FROM Pages WHERE id = ?''', (node, ))
31     row = cur.fetchone()
32     prev_ranks[node] = row[0]
33
34 sval = input('How many iterations:')
35 many = 1
36 if len(sval) > 0: many = int(sval)
37
38 # Sanity check
39 if len(prev_ranks) < 1:
40     print("Nothing to page rank. Check data.")
41     quit()
42
43 # Lets do Page Rank in memory so it is really fast
44 for i in range(many):
45     # print prev_ranks.items()[:5]
46     next_ranks = dict()
47     total = 0.0
48     for (node, old_rank) in list(prev_ranks.items()):
49         total = total + old_rank
50         next_ranks[node] = 0.0
51     # print total
52
53     # Find the number of outbound links and sent the page rank down each
54     for (node, old_rank) in list(prev_ranks.items()):
55         # print node, old_rank
56         give_ids = list()
57         for (from_id, to_id) in links:
58             if from_id != node: continue
59             # print ' ', from_id,to_id
60
61             if to_id not in to_ids: continue
62             give_ids.append(to_id)
63             if len(give_ids) < 1: continue
64             amount = old_rank / len(give_ids)
65             # print node, old_rank,amount, give_ids
66
67             for id in give_ids:
68                 next_ranks[id] = next_ranks[id] + amount
69
70     newtot = 0
71     for (node, next_rank) in list(next_ranks.items()):
72         newtot = newtot + next_rank
73     evap = (total - newtot) / len(next_ranks)
```

## Sprank.py (pt.2):

```
74
75     # print newtot, evap
76     for node in next_ranks:
77         next_ranks[node] = next_ranks[node] + evap
78
79     newtot = 0
80     for (node, next_rank) in list(next_ranks.items()):
81         newtot = newtot + next_rank
82
83     # Compute the per-page average change from old rank to new rank
84     # As indication of convergence of the algorithm
85     totdiff = 0
86     for (node, old_rank) in list(prev_ranks.items()):
87         new_rank = next_ranks[node]
88         diff = abs(old_rank-new_rank)
89         totdiff = totdiff + diff
90
91     avediff = totdiff / len(prev_ranks)
92     print(i+1, avediff)
93
94     # rotate
95     prev_ranks = next_ranks
96
97     # Put the final ranks back into the database
98     print(list(next_ranks.items())[5:])
99     cur.execute(''UPDATE Pages SET old_rank=new_rank''')
100    for (id, new_rank) in list(next_ranks.items()):
101        cur.execute(''UPDATE Pages SET new_rank=? WHERE id=?'', (new_rank, id))
102    conn.commit()
103    cur.close()
```

Results from running sprank.py (100 iterations to normalize the ranks as best as possible):

```
Run: sprank x
/Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/pagerank/sprank.py
How many iterations:100
1 0.4426507484446251
2 0.2592619011216506
3 0.15869898093894685
4 0.10951604950291145
5 0.07608047783909237
6 0.052714671803645974
7 0.04167050536391932
8 0.030501604299477708
9 0.02137668589156627
10 0.014286856560368844
11 0.009521795303278265
12 0.0066312847475213195
13 0.004627829426680675
14 0.003568134956289162
15 0.0026476388603147182
16 0.001852592337867123
17 0.0012193664985869254
18 0.0008821265283382987
19 0.0006666287245819747
20 0.00047813811116997684
21 0.00032091518125373095
22 0.00023713333704713689
23 0.00015519725585138528
24 0.00010883056210334659
25 8.13962367131095e-05
26 6.6379995323734e-05
27 5.065940166152224e-05
```

Spider.sqlite database, after running sprank.py, notice the rank values:

Table: Pages

New Record Delete Record

	<b>id</b>	<b>url</b>	<b>html</b>	<b>error</b>	<b>old_rank</b>	<b>new_rank</b>
1	1	python-data.dr...	NULL	-1	1.0	1.0
2	2	http://python-...	<html><head>	NULL	1.0	0.09965093527...
3	5	http://python-...	<html><head>	NULL	1.0	1.0
4	8	http://python-...	<html><head>	NULL	1.0	0.19930187054...
5	9	http://python-...	NULL	NULL	1.0	1.0
6	10	http://python-...	NULL	NULL	1.0	1.0
7	11	http://python-...	NULL	NULL	1.0	1.0
8	12	http://python-...	NULL	NULL	1.0	1.0
9	13	http://python-...	NULL	NULL	1.0	1.0
10	14	http://python-...	NULL	NULL	1.0	1.0
11	15	http://python-...	<html><head>	NULL	1.0	0.12456366909...
12	16	http://python-...	NULL	NULL	1.0	1.0
13	17	http://python-...	NULL	NULL	1.0	1.0
14	18	http://python-...	NULL	NULL	1.0	1.0
15	19	http://python-...	NULL	NULL	1.0	1.0
16	20	http://python-...	NULL	NULL	1.0	1.0
17	21	http://python-...	NULL	NULL	1.0	1.0
18	22	http://python-...	NULL	NULL	1.0	1.0
19	23	http://python-...	NULL	NULL	1.0	1.0

1 - 19 of 4809 Go to: 1

Spdump.py and output in terminal (Select statement w/JOIN to retrieve web page counts (frequencies), old rank, new rank, web page ID, URL of the web page, then prints it to the screen), screenshot needed for assignment:

The screenshot shows the PyCharm IDE interface. On the left, the project structure is visible with files like Course5w2\_Notes.py, spider.py, sprank.py, spdump.py (which is currently selected), and spjson.py. The right side shows the code editor for spdump.py and the run output.

```

import sqlite3
conn = sqlite3.connect('spider.sqlite')
cur = conn.cursor()
cur.execute(''':SELECT COUNT(from_id) AS inbound, old_rank, new_rank, id,
    FROM Pages JOIN Links ON Pages.id = Links.to_id
    WHERE html IS NOT NULL
    GROUP BY id ORDER BY inbound DESC''')
count = 0
for row in cur:
    if count < 50: print(row)
    count = count + 1
print(count, 'rows.')
cur.close()

```

The run output window shows the results of the SQL query. It lists 106 rows of data, each containing four fields: a frequency count (either 2 or 3), an old rank (float), a new rank (float), and a URL (string). The URLs are all from the domain "http://python-data.dr-chuck.net/known\_by\_".

Count	Old Rank	New Rank	URL
2	1.371268783651756	356	<a href="http://python-data.dr-chuck.net/known_by_Naeema.html">http://python-data.dr-chuck.net/known_by_Naeema.html</a>
2	2.6101611122066752	480	<a href="http://python-data.dr-chuck.net/known_by_Emilly.html">http://python-data.dr-chuck.net/known_by_Emilly.html</a>
2	3.5307430134223368	24	<a href="http://python-data.dr-chuck.net/known_by_Lex.html">http://python-data.dr-chuck.net/known_by_Lex.html</a>
2	1.712521548774718	61	<a href="http://python-data.dr-chuck.net/known_by_Fruin.html">http://python-data.dr-chuck.net/known_by_Fruin.html</a>
2	1.206076131491614	254	<a href="http://python-data.dr-chuck.net/known_by_Mahnoor.html">http://python-data.dr-chuck.net/known_by_Mahnoor.html</a>
2	2.0567224489938303	256	<a href="http://python-data.dr-chuck.net/known_by_Jaye.html">http://python-data.dr-chuck.net/known_by_Jaye.html</a>
2	1.4823523149501494	268	<a href="http://python-data.dr-chuck.net/known_by_Kaid.html">http://python-data.dr-chuck.net/known_by_Kaid.html</a>
2	1.0613576460786223	271	<a href="http://python-data.dr-chuck.net/known_by_Sharmaine.html">http://python-data.dr-chuck.net/known_by_Sharmaine.html</a>
2	0.49579562032785995	345	<a href="http://python-data.dr-chuck.net/known_by_Billy.html">http://python-data.dr-chuck.net/known_by_Billy.html</a>
2	2.3798270784088	627	<a href="http://python-data.dr-chuck.net/known_by_Anneroy.html">http://python-data.dr-chuck.net/known_by_Anneroy.html</a>
2	0.9826596087623303	931	<a href="http://python-data.dr-chuck.net/known_by_Maykayla.html">http://python-data.dr-chuck.net/known_by_Maykayla.html</a>
2	0.9587972383001718	1010	<a href="http://python-data.dr-chuck.net/known_by_Dyllan.html">http://python-data.dr-chuck.net/known_by_Dyllan.html</a>
2	1.0194918684486225	1871	<a href="http://python-data.dr-chuck.net/known_by_Ivan.html">http://python-data.dr-chuck.net/known_by_Ivan.html</a>
2	3.0320986272993253	2050	<a href="http://python-data.dr-chuck.net/known_by_Paris.html">http://python-data.dr-chuck.net/known_by_Paris.html</a>
2	1.0959432913766776	2079	<a href="http://python-data.dr-chuck.net/known_by_Lida.html">http://python-data.dr-chuck.net/known_by_Lida.html</a>
2	1.608356092008641	2783	<a href="http://python-data.dr-chuck.net/known_by_Alasdair.html">http://python-data.dr-chuck.net/known_by_Alasdair.html</a>
2	1.1603673826830831	3856	<a href="http://python-data.dr-chuck.net/known_by_Jasveer.html">http://python-data.dr-chuck.net/known_by_Jasveer.html</a>
3	0.7886053173815278	35	<a href="http://python-data.dr-chuck.net/known_by_Reynold.html">http://python-data.dr-chuck.net/known_by_Reynold.html</a>
3	0.8087863217830508	74	<a href="http://python-data.dr-chuck.net/known_by_Elyssa.html">http://python-data.dr-chuck.net/known_by_Elyssa.html</a>
3	2.8025739356634864	119	<a href="http://python-data.dr-chuck.net/known_by_Elizabeth.html">http://python-data.dr-chuck.net/known_by_Elizabeth.html</a>
3	0.956675929504834	148	<a href="http://python-data.dr-chuck.net/known_by_Stella.html">http://python-data.dr-chuck.net/known_by_Stella.html</a>
3	0.8421305433429495	157	<a href="http://python-data.dr-chuck.net/known_by_Kianna.html">http://python-data.dr-chuck.net/known_by_Kianna.html</a>
3	0.863268262586839	318	<a href="http://python-data.dr-chuck.net/known_by_Darach.html">http://python-data.dr-chuck.net/known_by_Darach.html</a>
3	1.4100393416327048	351	<a href="http://python-data.dr-chuck.net/known_by_Youssef.html">http://python-data.dr-chuck.net/known_by_Youssef.html</a>
3	1.20630318707544073	360	<a href="http://python-data.dr-chuck.net/known_by_Andra.html">http://python-data.dr-chuck.net/known_by_Andra.html</a>
3	0.6840207037589775	384	<a href="http://python-data.dr-chuck.net/known_by_Diaz.html">http://python-data.dr-chuck.net/known_by_Diaz.html</a>
3	1.775170697596545	412	<a href="http://python-data.dr-chuck.net/known_by_Cathal.html">http://python-data.dr-chuck.net/known_by_Cathal.html</a>
3	0.9492551969715776	579	<a href="http://python-data.dr-chuck.net/known_by_Loki.html">http://python-data.dr-chuck.net/known_by_Loki.html</a>
3	2.2640737267166986	714	<a href="http://python-data.dr-chuck.net/known_by_Mali.html">http://python-data.dr-chuck.net/known_by_Mali.html</a>
3	0.548691991734015	1057	<a href="http://python-data.dr-chuck.net/known_by_Madelyn.html">http://python-data.dr-chuck.net/known_by_Madelyn.html</a>
3	0.877947268635226	1172	<a href="http://python-data.dr-chuck.net/known_by_Mikee.html">http://python-data.dr-chuck.net/known_by_Mikee.html</a>
3	0.6053352253898773	1177	<a href="http://python-data.dr-chuck.net/known_by_Tyelor.html">http://python-data.dr-chuck.net/known_by_Tyelor.html</a>
3	0.9633477829000542	1310	<a href="http://python-data.dr-chuck.net/known_by_Vanni.html">http://python-data.dr-chuck.net/known_by_Vanni.html</a>
3	1.1420633924333952	1371	<a href="http://python-data.dr-chuck.net/known_by_Ekaterina.html">http://python-data.dr-chuck.net/known_by_Ekaterina.html</a>
3	2.480325079481248	1444	<a href="http://python-data.dr-chuck.net/known_by_Elias.html">http://python-data.dr-chuck.net/known_by_Elias.html</a>
3	1.1872379451253705	1457	<a href="http://python-data.dr-chuck.net/known_by_Lock.html">http://python-data.dr-chuck.net/known_by_Lock.html</a>
3	2.545131002793506	1534	<a href="http://python-data.dr-chuck.net/known_by_Sinem.html">http://python-data.dr-chuck.net/known_by_Sinem.html</a>
3	2.5031869359620815	1686	<a href="http://python-data.dr-chuck.net/known_by_Ralfs.html">http://python-data.dr-chuck.net/known_by_Ralfs.html</a>
3	1.114522275253917	2022	<a href="http://python-data.dr-chuck.net/known_by_Japieet.html">http://python-data.dr-chuck.net/known_by_Japieet.html</a>
3	0.47532505953306126	2094	<a href="http://python-data.dr-chuck.net/known_by_Karson.html">http://python-data.dr-chuck.net/known_by_Karson.html</a>
3	1.0743388039343025	2163	<a href="http://python-data.dr-chuck.net/known_by_Menna.html">http://python-data.dr-chuck.net/known_by_Menna.html</a>
3	2.926014988148159	2497	<a href="http://python-data.dr-chuck.net/known_by_Michee.html">http://python-data.dr-chuck.net/known_by_Michee.html</a>
3	1.4018901578237826	2650	<a href="http://python-data.dr-chuck.net/known_by_Name.html">http://python-data.dr-chuck.net/known_by_Name.html</a>
3	0.9019333438474566	3505	<a href="http://python-data.dr-chuck.net/known_by_Eilish.html">http://python-data.dr-chuck.net/known_by_Eilish.html</a>
3	1.5085180475098745	3661	<a href="http://python-data.dr-chuck.net/known_by_Chiamaka.html">http://python-data.dr-chuck.net/known_by_Chiamaka.html</a>
2	0.19973898085731162	59	<a href="http://python-data.dr-chuck.net/known_by_Anaia.html">http://python-data.dr-chuck.net/known_by_Anaia.html</a>
2	0.28323131977248245	65	<a href="http://python-data.dr-chuck.net/known_by_Aadam.html">http://python-data.dr-chuck.net/known_by_Aadam.html</a>
2	0.6274030182609494	107	<a href="http://python-data.dr-chuck.net/known_by_Finnlay.html">http://python-data.dr-chuck.net/known_by_Finnlay.html</a>
2	0.2903671501509767	122	<a href="http://python-data.dr-chuck.net/known_by_Aimie.html">http://python-data.dr-chuck.net/known_by_Aimie.html</a>
2	0.3738594890661475	161	<a href="http://python-data.dr-chuck.net/known_by_Aimee.html">http://python-data.dr-chuck.net/known_by_Aimee.html</a>

106 rows.

## Spjson.py, asking for 20 nodes:

The screenshot shows the PyCharm IDE interface with the following details:

- Project Structure:** The project tree on the left shows several files and folders, including "Reading Files - Employees", "PY4E ~/PycharmProjects/PY4E", "bs4", "pagerank", "Sample Code", and "venv".
- Code Editor:** The main editor window displays the content of `spjson.py`. The code is a Python script that connects to a SQLite database named `spider.sqlite`, executes SQL queries to get page rank data, and writes it to a JSON file named `spider.json`.
- Run Tab:** The bottom tab bar shows the current run configuration is `spjson`. The run history output shows:
  - The command: `/Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/pagerank/spjson.py`
  - The prompt: `Creating JSON output on spider.json...`
  - The user input: `How many nodes? 20`
  - The instruction: `Open force.html in a browser to view the visualization`
  - The status: `Process finished with exit code 0`

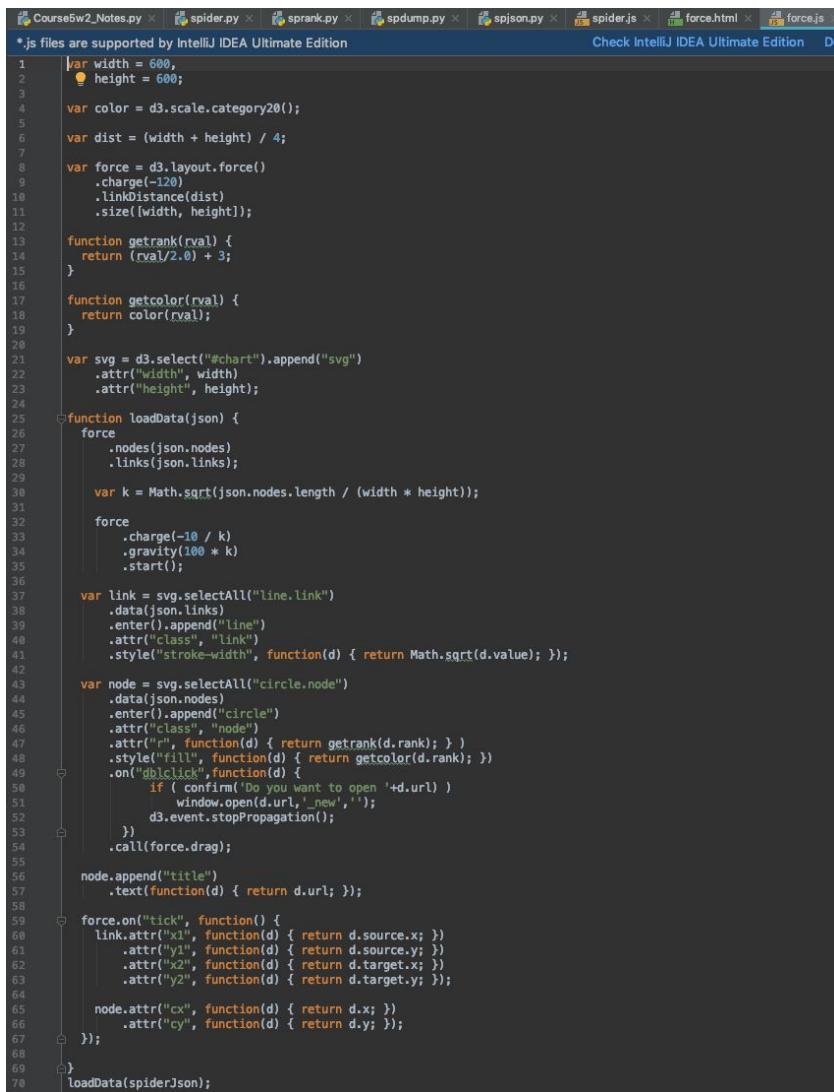
Spider.js, created from running spjson.py, information on the 20 nodes, feeds data to force.js and force.html (used screenshot from nfl data instead of Dr. Chucks website because I didn't want to run the whole over again on my network, they look very similar, just wanted to show you how the data looks)

\*.js files are supported by IntelliJ IDEA Ultimate Edition

Check IntelliJ IDEA Ultimate Edition   Do not suggest Ultimate Edition   Ignore extension

```
1 spiderJson = {"id":1,
2   "weight":131, "rank":13, "url":"https://www.nfl.com"}, 
3   {"weight":175, "rank":0, "id":2, "url":"https://www.nfl.com/news"}, 
4   {"weight":140, "rank":0, "id":3, "url":"https://www.nfl.com/scores"}, 
5   {"weight":132, "rank":19, "id":10, "url":"https://www.nfl.com/combine"}, 
6   {"weight":141, "rank":0, "id":12, "url":"https://www.nfl.com/stats/player"}, 
7   {"weight":131, "rank":13, "url":"https://www.nfl.com/films"}, 
8   {"weight":1, "rank":0, "id":17, "url":"https://www.nfl.com/photos?eventseason=1&event=combine"}, 
9   {"weight":7, "rank":0, "id":25, "url":"https://www.nfl.com/news/story/0ap300001021518/article/parris-campbell-among-prospects-trending-up-and-down"}, 
10  {"weight":4, "rank":0, "id":26, "url":"https://www.nfl.com/news/story/0ap300001019889/article/daniel-jeremiahs-nfl-scouting-combine-takeaways"}, 
11  {"weight":5, "rank":0, "id":30, "url":"https://www.nfl.com/news/story/0ap300001021069/article/2019-nfl-scouting-combine-team-top-offensive-line"}, 
12  {"weight":5, "rank":0, "id":31, "url":"https://www.nfl.com/news/story/0ap300001021076/article/2019-nfl-scouting-combine-team-top-defensive-line"}, 
13  {"weight":17, "rank":0, "id":35, "url":"https://www.nfl.com/videos/nfl-combine/0ap300001020923/Greedy-Williams-runs-an-official-4-37-40-yard-dash"}, 
14  {"weight":14, "rank":0, "id":37, "url":"https://www.nfl.com/news/story/0ap300001020899/article/sean-payton-likenks-trace-mcsorley-to-julian-edelman"}, 
15  {"weight":5, "rank":0, "id":40, "url":"https://www.nfl.com/news/story/0ap300001020821/article/2019-nfl-scouting-combine-winners-losers-montana"}, 
16  {"weight":5, "rank":0, "id":41, "url":"https://www.nfl.com/news/story/0ap300001020852/article/ole-miss-dk-metcalf-dominates-at-2019-nfl-scouting-combine"}, 
17  {"weight":5, "rank":0, "id":43, "url":"https://www.nfl.com/news/story/0ap300001020897/article/kendall-sheffield-suffered-partially-torn-pedicle"}, 
18  {"weight":17, "rank":0, "id":49, "url":"https://www.nfl.com/videos/nfl-combine/0ap300001020685/Rashan-Gary-runs-an-official-4-58-40-yard-dash"}, 
19  {"weight":17, "rank":0, "id":51, "url":"https://www.nfl.com/videos/nfl-combine/0ap300001020663/Silimcam-Quinnen-Williams-vs-Aaron-Donald"}, 
20  {"weight":21, "rank":0, "id":55, "url":"https://www.nfl.com/videos/pff/0ap300001018257/PFF-s-Top-10-highest-graded-defensive-players-at-combine"}, 
21  {"weight":17, "rank":0, "id":63, "url":"https://www.nfl.com/videos/nfl-combine/0ap300001020363/LeBron-reacts-to-Parris-Campbell-s-blazing-speed"}, 
22  {"weight":31, "rank":0, "id":79, "url":"https://www.nfl.com/cfb247"}], 
23  "links": [ 
24    {"source":0, "target":0, "value":3}, 
25    {"source":0, "target":1, "value":3}, 
26    {"source":0, "target":2, "value":3}, 
27    {"source":0, "target":3, "value":3}, 
28    {"source":0, "target":4, "value":3}, 
29    {"source":0, "target":5, "value":3}, 
30    {"source":3, "target":0, "value":3}, 
31    {"source":3, "target":1, "value":3}, 
32    {"source":3, "target":2, "value":3}, 
33    {"source":3, "target":3, "value":3}, 
34    {"source":3, "target":4, "value":3}, 
35    {"source":3, "target":5, "value":3}, 
36    {"source":3, "target":6, "value":3}, 
37    {"source":3, "target":7, "value":3}, 
38    {"source":3, "target":8, "value":3}, 
39    {"source":3, "target":9, "value":3}, 
40    {"source":3, "target":10, "value":3}, 
41    {"source":3, "target":11, "value":3}, 
42    {"source":3, "target":12, "value":3}, 
43    {"source":3, "target":13, "value":3}, 
44    {"source":3, "target":14, "value":3}, 
45    {"source":3, "target":15, "value":3}, 
46    {"source":3, "target":16, "value":3}, 
47    {"source":3, "target":17, "value":3}, 
48    {"source":3, "target":18, "value":3}, 
49    {"source":3, "target":19, "value":3}, 
50    {"source":16, "target":16, "value":3}, 
51    {"source":16, "target":0, "value":3}, 
52    {"source":16, "target":1, "value":3}, 
53    {"source":16, "target":2, "value":3}, 
54    {"source":16, "target":3, "value":3}, 
55    {"source":16, "target":4, "value":3}, 
56    {"source":16, "target":5, "value":3}, 
57    {"source":12, "target":0, "value":3}, 
58    {"source":12, "target":1, "value":3}, 
59    {"source":12, "target":2, "value":3}, 
60    {"source":12, "target":3, "value":3}, 
61    {"source":12, "target":4, "value":3}, 
62    {"source":12, "target":5, "value":3}, 
```

Force.js (d3 library javascript code, provided by instructor), contains design specifications for visualizations, feeds force.html:



```

1  var width = 600,
2      height = 600;
3
4  var color = d3.scale.category20();
5
6  var dist = (width + height) / 4;
7
8  var force = d3.layout.force()
9      .charge(-120)
10     .linkDistance(dist)
11     .size([width, height]);
12
13 function getrank(rval) {
14     return (rval/2.0) + 3;
15 }
16
17 function getcolor(rval) {
18     return color(rval);
19 }
20
21 var svg = d3.select("#chart").append("svg")
22     .attr("width", width)
23     .attr("height", height);
24
25 function loadData(json) {
26     force
27         .nodes(json.nodes)
28         .links(json.links);
29
30     var k = Math.sqrt(json.nodes.length / (width * height));
31
32     force
33         .charge(-10 / k)
34         .gravity(100 * k)
35         .start();
36
37     var link = svg.selectAll("line.link")
38         .data(json.links)
39         .enter().append("line")
40         .attr("class", "link")
41         .style("stroke-width", function(d) { return Math.sqrt(d.value); });
42
43     var node = svg.selectAll("circle.node")
44         .data(json.nodes)
45         .enter().append("circle")
46         .attr("class", "node")
47         .attr("r", function(d) { return getrank(d.rank); })
48         .style("fill", function(d) { return getcolor(d.rank); })
49         .on("dblclick", function(d) {
50             if (confirm('Do you want to open '+d.url+'))
51                 window.open(d.url,'_new','');
52             d3.event.stopPropagation();
53         })
54         .call(force.drag);
55
56     node.append("title")
57         .text(function(d) { return d.url; });
58
59     force.on("tick", function() {
60         link.attr("x1", function(d) { return d.source.x; })
61         .attr("y1", function(d) { return d.source.y; })
62         .attr("x2", function(d) { return d.target.x; })
63         .attr("y2", function(d) { return d.target.y; });
64
65         node.attr("cx", function(d) { return d.x; })
66         .attr("cy", function(d) { return d.y; });
67     });
68
69     loadData(spiderJson);
70 }

```

Force.html, notice the src's (sources), browser links in right corner:



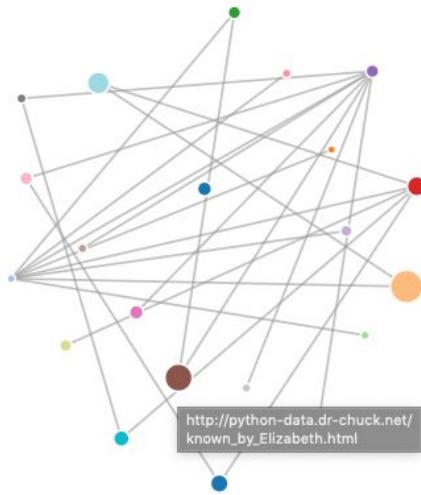
```

<!DOCTYPE html>
<html>
    <head>
        <title>Force-Directed Layout</title>
        <script type="text/javascript" src="d3.v2.js"></script>
        <script type="text/javascript" src="spider.js"></script>
        <link type="text/css" rel="stylesheet" href="force.css"/>
    </head>
    <body style="...>
        <script>
            document.write("<p>Starting url: "+spiderJson.nodes[0].url+"</p>");
        </script>
        <div id="chart" style="...></div>
        <script type="text/javascript" src="force.js"></script>
        <p>If you don't see a chart above, check the JavaScript console. You may need to use a different browser.</p>
    </body>
</html>

```

Visualization for Dr. Chuck's website, result from selecting the Chrome browser in force.html (screenshot need for assignment):

Starting url: [http://python-data.dr-chuck.net/comments\\_42.html](http://python-data.dr-chuck.net/comments_42.html)



If you don't see a chart above, check the JavaScript console. You may need to use a different browser.

Spreset.py, run to reset the “spidering” process (and spider.sqlitedatabase) so we can “spider” another webpage for the assignment:

```
1 import sqlite3
2
3 conn = sqlite3.connect('spider.sqlite')
4 cur = conn.cursor()
5
6 cur.execute(''':UPDATE Pages SET new_rank=1.0, old_rank=0.0'''')
7 conn.commit()
8
9 cur.close()
10
11 print("All pages set to a rank of 1.0")
12
```

Run: spreset x  
/Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/pag  
All pages set to a rank of 1.0  
Process finished with exit code 0

Results from running spider.py, “spidering” 100 pages from <https://www.nfl.com>:

```
Run: spider x
  ↑
  Enter web url or enter: https://www.nfl.com
  ['https://www.nfl.com']
  How many pages: 2
  1 https://www.nfl.com (134019) 18
  10 https://www.nfl.com/combine (199199) 226
  How many pages: 2
  49 https://www.nfl.com/videos/nfl-combine/0ap3000001020685/Rashan-Gary-runs-an-official-4-58-40-yard-dash-at-2019-NFL-combine (143529) 38
  37 https://www.nfl.com/news/story/0ap3000001020899/article/sean-payton-likenz-trace-mcsorley-to-julian-edelman (166518) 35
  22 https://www.nfl.com/photos?eventseason=1&event=combine (276765) 128
  43 https://www.nfl.com/news/story/0ap3000001020897/article/kendall-sheffield-suffered-partially-torn-pcl-at-nfl-combine (180457) 60
  93 https://www.nfl.com/photos/0ap300000473712 (154941) 51
  101 https://www.nfl.com/photos/0ap200000329158 (152847) 51
  115 https://www.nfl.com/photos/09000d5d81e740c9 (285098) 51
  86 https://www.nfl.com/teams/newenglandpatriots/profile?team=NE (168693) 102
  110 https://www.nfl.com/photos/09000d5d82741511 (158070) 51
  204 https://www.nfl.com/draft/2019/profiles/n'keal-harry?id=2562576 (181588) 21
  108 https://www.nfl.com/photos/0ap100000143242 (262132) 51
  157 https://www.nfl.com/photos/0ap3000001010166 (150336) 51
  35 https://www.nfl.com/videos/nfl-combine/0ap3000001020923/Greedy-Williams-runs-an-official-4-37-40-yard-dash-at-2019-combine (143428) 38
  124 https://www.nfl.com/photos/0ap300000771874 (157194) 51
  162 https://www.nfl.com/news/story/0ap300000662802/comments/16-for-16-countdowns-of-college-football-s-best (168629) 59
  114 https://www.nfl.com/photos/09000d5d8264309c (156081) 51
  79 https://www.nfl.com/cfb247 (143387) 28
  184 https://www.nfl.com/teams/newenglandpatriots/coaches?coaType=head&team=NE (127642) 40
  51 https://www.nfl.com/videos/nfl-combine/0ap3000001020663/Simulcam-Quinnen-Williams-vs-Aaron-Donald (143499) 38
  151 https://www.nfl.com/videos/nfl-game-highlights/0ap3000001042905/Bradley-Chubb-s-best-plays-Preseason-Week-2 (149789) 42
  230 https://www.nfl.com/player/dontrelleinman/2530700/profile (147077) 38
  99 https://www.nfl.com/photos/0ap100000140338 (157995) 51
  218 https://www.nfl.com/teams/profile?team=TEN (163874) 90
  178 https://www.nfl.com/rss/rsslanding?searchString=teamId=3200&name>New England Patriots&abbr=NE Ignore non text/html page
  250 https://www.nfl.com/teams/losangeleschargers/profile?team=LAC (164198) 91
  319 https://www.nfl.com/news/story/0ap300000791026/article/la-chargers-release-brandon-flowers-two-others (169088) 45
  137 https://www.nfl.com/photos/0ap300000718608 (156858) 51
  306 https://www.nfl.com/teams/losangeleschargers/statistics?team=LAC (244840) 112
  17 https://www.nfl.com/films (139940) 18
  219 https://www.nfl.com/gamecenter/2019081755/2019/PRE2/patriots@titans (894344) 18
  89 https://www.nfl.com/photos/0ap300000640051 (160976) 51
  63 https://www.nfl.com/videos/nfl-combine/0ap3000001020363/LeBron-reacts-to-Parris-Campbell-s-blazing-fast-40-yard-dash-on-Instagram (143946)
  366 https://www.nfl.com/players/justinjackson/profile?id=JAC283219 (148259) 44
  146 https://www.nfl.com/prospects/billy-price?id=32462018-0002-5599-11cc-88054ff74390 (93226) 19
  118 https://www.nfl.com/photos/09000d5d8173d33 (162315) 51
  190 https://www.nfl.com/news/story/0ap3000001039879/article/tom-brady-set-to-be-fa-in-2020-despite-new-extension (170396) 44
  40 https://www.nfl.com/news/story/0ap3000001020821/article/2019-nfl-scouting-combine-winnerslosers-montez-sweat-soars (191856) 65
  406 https://www.nfl.com/players/justinjones/profile?id=JON518335 (147121) 39
  379 https://www.nfl.com/players/jordansmallwood/profile?id=SMAT756207 (143599) 38
  442 https://www.nfl.com/news/story/0ap3000001042693/article/preseason-grades-week-2-tony-pollard-rolls-kyler-murray-fails (212552) 132
  112 https://www.nfl.com/photos/09000d5d8271e16a (272749) 51
  41 https://www.nfl.com/news/story/0ap3000001020521/article/ole-miss-dk-metcalf-dominates-at-2019-nfl-scouting-combine (186382) 59
  286 https://www.nfl.com/gamecenter/2019080861/2019/PRE1/titans@eagles (917110) 18
  477 https://www.nfl.com/teams/detroitlions/profile?team=DET (137041) 95
  358 https://www.nfl.com/player/t.j.lang/89746/profile (141929) 36
  217 https://www.nfl.com/gamecenter/2019080856/2019/PRE1/patriots@lions (878348) 18
  396 https://www.nfl.com/players/elijahzeise/profile?id=ZEI437210 (143342) 37
  145 https://www.nfl.com/news/story/0ap300000866868/article/super-sleeper-is-dallas-goedert-nfls-next-matchup-nightmare (208853) 70
  299 https://www.nfl.com/teams/houstontexans/profile?team=HOU (166872) 92
  608 https://www.nfl.com/teams/profile?team=DAL (167819) 96
  216 https://www.nfl.com/teams/profile?team=DET (163943) 95
  661 https://www.nfl.com/player/devinsmith/2553434/profile (147745) 40
  255 https://www.nfl.com/tickets/tennessee-titans?campaign=Teams\_Tel\_Tickets Unable to retrieve or parse page
  337 https://www.nfl.com/teams/profile?team=SEA (167287) 103
  241 https://www.nfl.com/standings (152018) 19
  689 https://www.nfl.com/news/story/0ap3000001041163/article/seahawks-oc-chris-carson-should-get-50-targets (171296) 48
  279 https://www.nfl.com/draft/2019/profiles/a.l.brown?id=2562238 (177498) 21
  545 https://www.nfl.com/teams/schedule?team=DET (130554) 58
  212 https://www.nfl.com/draft/2019/profiles/jake-bailey?id=2562226 (173270) 21
  125 https://www.nfl.com/photos/0ap300000768046 (130123) 51
  748 https://www.nfl.com/players/davidtales/profile?id=FAL381917 (154121) 49
  249 https://www.nfl.com/teams/chicagobears/profile?team=CHI (163190) 87
  504 https://www.nfl.com/teams/greenbaypackers/profile?team=GB (164305) 95
  848 https://www.nfl.com/player/roberttonyan/2559270/profile (146893) 39
  767 https://www.nfl.com/fantasy/story/0ap3000001035854/article/fantasy-football-roster-resets-nfc (188115) 310
  482 https://www.nfl.com/player/irvsmith/2562738/profile (144108) 41
  389 https://www.nfl.com/players/kemonhall/profile?id=HAL409771 (116284) 37
  483 https://www.nfl.com/player/adamthielan/2541785/profile (152315) 48
  527 https://www.nfl.com/news/story/0ap3000001037458/article/lions-release-rb-theo-ridick-after-six-seasons (167541) 45
  25 https://www.nfl.com/news/story/0ap3000001021518/article/parris-campbell-among-prospects-trending-up-after-combine (195538) 74
  793 https://www.nfl.com/draft/2019/profiles/duke-shelley?id=2562957 (162620) 21
  921 https://www.nfl.com/player/elimanning/2505996/profile (166950) 58
```

Results from running sprank.py for nfl.com (100 iterations to normalize the ranks as best as possible):

```
Run: sprank x
/Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/pagerank/sprank.py
How many iterations: 100
1 1.4379398048018128
2 0.5272215279284813
3 0.2521660801487082
4 0.08638291191365961
5 0.03052908232347833
6 0.012645639451475937
7 0.0051597819703775635
8 0.001881971929332455
9 0.000669961107589387
10 0.0002911954782331515
11 0.00011453373084136624
12 4.4220702166356384e-05
13 1.6205066148997997e-05
14 6.693869895913812e-06
15 2.5148725966821486e-06
16 1.0539318984254368e-06
17 4.1816125139001724e-07
18 1.6873695189881728e-07
19 6.299730099967458e-08
20 2.5773235183187456e-08
21 1.05084882021142e-08
22 4.291553262414804e-09
23 1.5305856839477786e-09
24 6.283600903232782e-10
25 2.5612976763820226e-10
26 1.0785000004855018e-10
```

Spider.sqlite database, after running sprank.py for nfl.com, notice the rank values, nfl has more traffic than Dr. Chuck's website:

Table: Pages

	<input type="button" value="New Record"/>	<input type="button" value="Delete Record"/>				
	<input type="button" value="Filter"/>	<input type="button" value="Filter"/>	<input type="button" value="Filter"/>	<input type="button" value="Filter"/>	<input type="button" value="Filter"/>	
1	1	<a href="https://www.nfl.com">https://www.nfl.com</a>	<!doctype html...>	NULL	1.0	18.253579856
2	2	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>		NULL	1.0	1.0
3	3	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>	<!doctype html...>	NULL	1.0	1.0
4	4	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>	NULL	NULL	1.0	1.0
5	5	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>	NULL	NULL	1.0	1.0
6	6	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>	NULL	NULL	1.0	1.0
7	7	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>	NULL	NULL	1.0	1.0
8	8	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>	NULL	NULL	1.0	1.0
9	9	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>	NULL	NULL	1.0	1.0
10	10	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>		NULL	1.0	25.669096673
11	11	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>	NULL	NULL	1.0	1.0
12	12	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>		NULL	1.0	1.0
13	13	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>	NULL	NULL	1.0	1.0
14	14	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>	NULL	NULL	1.0	1.0
15	15	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>	NULL	NULL	1.0	1.0
16	16	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>	NULL	NULL	1.0	1.0
17	17	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>	<!doctype html...>	NULL	1.0	18.253579856
18	18	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>	NULL	NULL	1.0	1.0
19	19	<a href="https://www.nfl.com/">https://www.nfl.com/...</a>	NULL	NULL	1.0	1.0

1 - 19 of 2297 Go to: 1

Spdump.py and output in terminal (Select statement w/JOIN to retrieve web page counts (frequencies), old rank, new rank, web page ID, URL of the web page, then prints it to the screen), screenshot needed for assignment (nfl.com):

```

Run: spdump x
↑ /Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/pagerank/spdump.py
(175, 1, 0, 1, 0, 2, 'https://www.nfl.com/news')
(141, 1, 0, 1, 0, 12, 'https://www.nfl.com/stats/player')
(140, 1, 0, 1, 0, 3, 'https://www.nfl.com/scores')
(132, 1, 0, 25.669096673005036, 10, 'https://www.nfl.com/combine')
(131, 1, 0, 18.253579856359135, 1, 'https://www.nfl.com')
(131, 1, 0, 18.253579856359135, 17, 'https://www.nfl.com/films')
(97, 1, 0, 0.492916311831964, 477, 'https://www.nfl.com/games/detroitlions/profile?team=DET')
(85, 1, 0, 1, 0, 156, 'https://www.nfl.com/photos/0ap30000001011682')
(82, 1, 0, 1.7217353333672507, 157, 'https://www.nfl.com/photos/0ap300000010101666')
(47, 1, 0, 0.4710553304343639, 504, 'https://www.nfl.com/teams/greenbaypackers/profile?team=GB')
(35, 1, 0, 0.3775252459389664, 250, 'https://www.nfl.com/teams/losangeleschargers/profile?team=LAC')
(31, 1, 0, 2.699413353645514, 79, 'https://www.nfl.com/cfb242')
(30, 1, 0, 0.6777888371161555, 86, 'https://www.nfl.com/teams/newenglandpatriots/profile?team=NE')
(29, 1, 0, 1.369414339863397311, 151, 'https://www.nfl.com/videos/nfl-game-highlights/0ap30000001042905/Bradley-Chubb-s-best-plays-Preseason-Week-2')
(21, 1, 0, 1, 0, 55, 'https://www.nfl.com/videos/nfl/0ap30000001018257/PFF-s-Top-10-highest-graded-defensive-players-at-combine')
(20, 1, 0, 2.2159727259296408, 216, 'https://www.nfl.com/teams/profile?team=DET')
(18, 1, 0, 0.18403603273620614, 249, 'https://www.nfl.com/teams/chicagobears/profile?team=CHI')
(17, 1, 0, 1.712731115336688, 35, 'https://www.nfl.com/videos/nfl-combine/0ap3000001020923/Williams-runs-an-official-4-37-40-yard-dash-at-2019-combine')
(17, 1, 0, 1.712731115336688, 49, 'https://www.nfl.com/videos/nfl-combine/0ap3000001020685/Rashan-Gary-runs-an-official-4-58-40-yard-dash-at-2019-NFL-combine')
(17, 1, 0, 1.712731115336688, 51, 'https://www.nfl.com/videos/nfl-combine/0ap3000001020663/Smulcan-Quinn-Williams-vs-Aaron-Donald')
(17, 1, 0, 1.712731115336688, 63, 'https://www.nfl.com/videos/nfl-combine/0ap3000001020363/LeBron-reacts-to-Parris-Campbell-s-blazing-fast-40-yard-dash-on-Instagram')
(17, 1, 0, 1.737837702626266, 162, 'https://www.nfl.com/news/story/0ap3000000662802/comments/16-for-16-countdowns-of-college-football-s-best')
(13, 1, 0, 0.3618140649522587, 299, 'https://www.nfl.com/teams/houstontexans/profile?team=HOU')
(13, 1, 0, 0.06700250462819582, 608, 'https://www.nfl.com/teams/profile?team=DAL')
(11, 1, 0, 0.04853247770554054, 337, 'https://www.nfl.com/teams/profile?team=SEA')
(11, 1, 0, 0.222798916084543, 442, 'https://www.nfl.com/news/story/0ap3000001042693/article/preseason-grades-week-2-tony-pollard-rolls-kyler-murray-fails')
(10, 1, 0, 0.079486690718503858, 218, 'https://www.nfl.com/players/0ap300000102786/PFF-s-Ten')
(10, 1, 0, 0.024330795908762786, 241, 'https://www.nfl.com/standings')
(10, 1, 0, 0.05870578098534443, 336, 'https://www.nfl.com/gamcenter/0ap9081851/2019/PRE2/saints@chargers')
(10, 1, 0, 0.01340052092563902, 656, 'https://www.nfl.com/teams/profile?team=TB')
(9, 1, 0, 1, 0, 556, 'https://www.nfl.com/teams/coaches?coaType=head&team=DET')
(7, 1, 0, 1.712731115336688, 25, 'https://www.nfl.com/news/story/0ap3000001021518/article/parris-campbell-among-prospects-trending-up-after-combine')
(7, 1, 0, 1, 0, 489, 'https://www.nfl.com/player/milessanders/2562722/profile')
(7, 1, 0, 0.07088898037761534, 545, 'https://www.nfl.com/teams/schedule?team=DET')
(6, 1, 0, 0.056482403093012806, 184, 'https://www.nfl.com/teams/newenglandpatriots/coaches?coaType=head&team=NE')
(6, 1, 0, 0.012380064545094266, 482, 'https://www.nfl.com/player/irvsmith/2562738/profile')
(6, 1, 0, 0.0027117027488321984, 739, 'https://www.nfl.com/player/alvinamarai/2558019/profile')
(6, 1, 0, 0.00269618403378855, 767, 'https://www.nfl.com/rantans/story/0ap3000001035854/article/fantasy-football-roster-resets-nfc')
(6, 1, 0, 1, 0, 1091, 'https://www.nfl.com/team/profile?team=NY')
(5, 1, 0, 1.712731115336688, 30, 'https://www.nfl.com/news/story/0ap3000001021009/article/2019-nfl-scouting-combine-team-top-offensive-prospects')
(5, 1, 0, 1.712731115336688, 31, 'https://www.nfl.com/news/story/0ap3000001021076/article/2019-nfl-scouting-combine-team-top-defensive-prospects')
(5, 1, 0, 1.712731115336688, 40, 'https://www.nfl.com/news/story/0ap3000001020821/article/2019-nfl-scouting-combine-winnerslosers-monbez-sweat-soars')
(5, 1, 0, 1.712731115336688, 41, 'https://www.nfl.com/news/story/0ap3000001020521/article/ole-miss-dk-metcalf-dominates-at-2019-nfl-scouting-combine')
(5, 1, 0, 1.712731115336688, 43, 'https://www.nfl.com/news/story/0ap3000001020897/article/kendall-sheffield-suffered-partially-torn-pec-at-nfl-combine')
(5, 1, 0, 0.404050595778727, 145, 'https://www.nfl.com/news/story/0ap300000066868/article/super-sleeper-is-dallas-goedert-nfls-next-matchup-nightmare')
(5, 1, 0, 0.135374061381, 217, 'https://www.nfl.com/gamcenter/0ap9081856/2019/PRE1/patriots@lions')
(5, 1, 0, 0.011937919382769507, 484, 'https://www.nfl.com/player/stefondiggs/2552608/profile')
(5, 1, 0, 0.02300450409205622, 806, 'https://www.nfl.com/teams/coaches?coaType=assist&team=CHI')
(5, 1, 0, 0.263272786389795, 1201, 'https://www.nfl.com/news/story/0ap3000001029901/article/2019-nfl-all-rookie-team-kyler-murray-devin-bush-to-show-out')
5, None, 1, 0, 2157, 'https://www.nfl.com/news/story/0ap200000051331/article/jeff-demps-picks-new-england-patriots-from-nfl-suitors')
131 rows.

```

Spjson.py asking for 20 nodes from nfl.com:

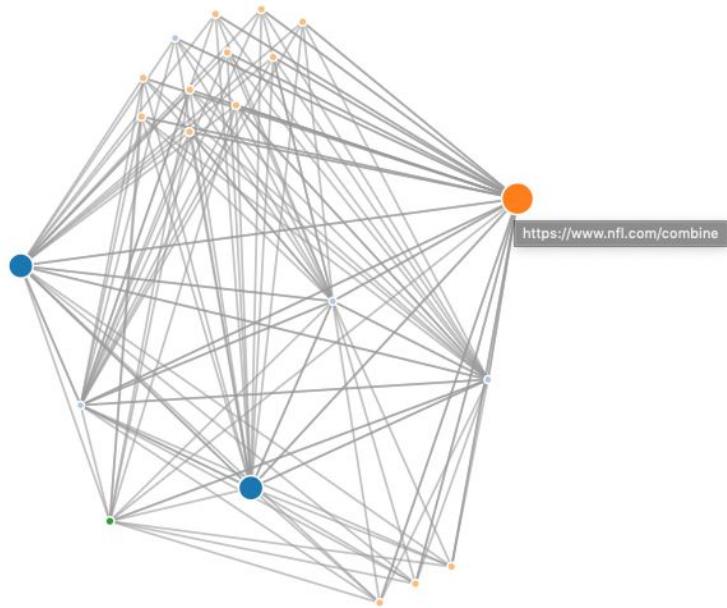
```

Run: spjson x
↑ /Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/pagerank/spjson.py
Creating JSON output on spider.js...
How many nodes? 20
Open force.html in a browser to view the visualization
Process finished with exit code 0
| 

```

Visualization for nfl.com, result from selecting the Chrome browser in force.html  
(screenshot need for assignment):

Starting url: <https://www.nfl.com>



If you don't see a chart above, check the JavaScript console. You may need to use a different browser.

## F.2: Week 4 - Loading and Modeling Mail Data Assignment (Gmane/Mail pt.1):

### Prompt for the assignment:

Welcome Alex Huerta from Capstone: Retrieving, Processing, and Visualizing Data with Python

#### Mailing List Data - Part I

In this assignment you will download some of the mailing list data from <http://mbox.dr-chuck.net/> and run the data cleaning / modeling process and take some screen shots..

Don't take off points for little mistakes. If they seem to have done the assignment give them full credit. Feel free to make suggestions if there are small mistakes. Please keep your comments positive and useful. The course staff will assign the last 30% of the grade and it will take a few days to get graded.

#### Please Upload Your Submission:

A screen shot of your SQLiteBrowser showing messages downloaded from mbox.dr-chuck.net into the content.sqlite database

No file chosen

(Please use PNG or JPG files)

A screen shot of you running the gmodel.py application to produce the index.sqlite database.

No file chosen

(Please use PNG or JPG files)

A screen shot of your SQLiteBrowser showing messages in the index.sqlite database after the gmodel.py has executed.

No file chosen

(Please use PNG or JPG files)

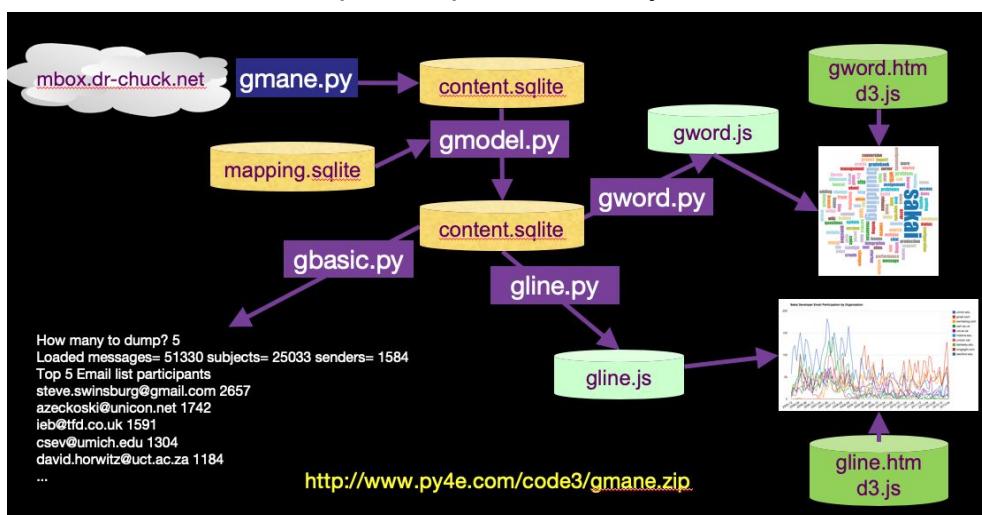
A screen shot of you running the gbasic.py program to compute basic histogram data on the messages you have retrieved.

No file chosen

(Please use PNG or JPG files)

Enter optional comments below

Diagram from the Ch.16 lecture slides, shows how each of the pages of code provided relate to each other, helps to explain what they do:



## Gmane.py (pt.1):

```
Course5w4_Notes.py x README.txt x gmane.py x gmodel.py x Course5w2_Notes.py x spic
1 import sqlite3
2 import time
3 import ssl
4 import urllib.request, urllib.parse, urllib.error
5 from urllib.parse import urljoin
6 from urllib.parse import urlparse
7 import re
8 from datetime import datetime, timedelta
9
10 # Not all systems have this so conditionally define parser
11 try:
12     import dateutil.parser as parser
13 except:
14     pass
15
16 def parsemaildate(md):
17     # See if we have dateutil
18     try:
19         pdate = parser.parse(tdate)
20         test_at = pdate.isoformat()
21         return test_at
22     except:
23         pass
24
25     # Non-dateutil version - we try our best
26
27     pieces = md.split()
28     notz = ".join(pieces[:4]).strip()"
29
30     # Try a bunch of format variations - strftime() is *lame*
31     dnotz = None
32     for form in [ "%d %b %Y %H:%M:%S", "%d %b %Y %H:%M:%S",
33                  "%d %b %Y %H:%M", "%d %b %Y %H:%M:%S",
34                  "%d %b %Y %H:%M:S", "%d %b %Y %H:%M", "%d %b %Y %H:%M" ]:
35         try:
36             dnotz = datetime.strptime(notz, form)
37             break
38         except:
39             continue
40
41     if dnotz is None:
42         # print 'Bad Date:',md
43         return None
44
45     iso = dnotz.isoformat()
46
47     tz = "+0000"
48     try:
49         tz = pieces[4]
50         jval = int(tz) # Only want numeric timezone values
51         if tz == '-0000': tz = '+0000'
52         tzh = tz[:3]
53         tzm = tz[3:]
54         tz = tzh+":"+tzm
55     except:
56         pass
57
58     return iso+tz
59 # End of dr chuck's date|time parser code
60
61 # Ignore SSL certificate errors
62 ctx = ssl.create_default_context()
63 ctx.check_hostname = False
64 ctx.verify_mode = ssl.CERT_NONE
65
66 conn = sqlite3.connect('content.sqlite')
67 cur = conn.cursor()
68
69 baseurl = "http://mbox.dr-chuck.net/sakai.devel/"
70
71 cur.execute('''CREATE TABLE IF NOT EXISTS Messages
72     (id INTEGER UNIQUE, email TEXT, sent_at TEXT,
73      subject TEXT, headers TEXT, body TEXT)'''')
74
75 # Pick up where we left off
76 start = None
77 cur.execute('SELECT max(id) FROM Messages')
78 +try:
```

Gmane.py (pt.2):

```
gmane.py x gmodel.py x
77     cur.execute('SELECT id FROM messages')
78     try:
79         row = cur.fetchone()
80         if row is None:
81             start = 0
82         else:
83             start = row[0]
84     except:
85         start = 0
86
87     if start is None: start = 0
88
89     many = 0
90     count = 0
91     fail = 0
92     while True:
93         if (many < 1):
94             conn.commit()
95             sval = input('How many messages: ')
96             if (len(sval) < 1): break
97             many = int(sval)
98
99         start = start + 1
100        cur.execute('SELECT id FROM Messages WHERE id=?', (start,))
101        try:
102            row = cur.fetchone()
103            if row is not None: continue
104        except:
105            row = None
106
107        many = many - 1
108        url = baseurl + str(start) + '/' + str(start + 1)
109
110        text = "None"
111    try:
```

### Gmane.py (pt.3):

```
Course5w4_Notes.py x README.txt x gmane.py x gmodel.py x Course5w2_Notes.py x spider.py x
111     try:
112         # Open with a timeout of 30 seconds
113         document = urllib.request.urlopen(url, None, 30, context=ctx)
114         text = document.read().decode()
115         if document.getcode() != 200:
116             print("Error code=%d, document.getcode(), url")
117             break
118     except KeyboardInterrupt:
119         print('')
120         print('Program interrupted by user...')
121         break
122     except Exception as e:
123         print("Unable to retrieve or parse page", url)
124         print("Error", e)
125         fail = fail + 1
126         if fail > 5: break
127         continue
128
129     print(url, len(text))
130     count = count + 1
131
132     if not text.startswith("From "):
133         print(text)
134         print("Did not find From ")
135         fail = fail + 1
136         if fail > 5: break
137         continue
138
139     pos = text.find("\n\n")
140     if pos > 0:
141         hdr = text[:pos]
142         body = text[pos+2:]
143     else:
144         print(text)
145         print("Could not find break between headers and body")
146         fail = fail + 1
147         if fail > 5: break
148         continue
149
150     email = None
151     x = re.findall('\nFrom: .*<(\S+@\S+)>\n', hdr)
152     if len(x) == 1:
153         email = x[0]
154         email = email.strip().lower()
155         email = email.replace("<", "")
156     else:
157         x = re.findall('\nFrom: (\S+@\S+)\n', hdr)
158         if len(x) == 1:
159             email = x[0]
160             email = email.strip().lower()
161             email = email.replace("<", "")
162
163     date = None
164     y = re.findall('\nDate: .*, (.*)\n', hdr)
165     if len(y) == 1:
166         tdate = y[0]
167         tdate = tdate[:26]
168         try:
169             sent_at = parsemaildate(tdate)
170         except:
171             print(text)
172             print("Parse fail", tdate)
173             fail = fail + 1
174             if fail > 5: break
175             continue
176
177     subject = None
178     z = re.findall('\nSubject: (.*)\n', hdr)
179     if len(z) == 1: subject = z[0].strip().lower();
180
181     # Reset the fail counter
182     fail = 0
183     print("email", email, sent_at, subject)
184     cur.execute("""INSERT OR IGNORE INTO Messages (id, email, sent_at, subject, headers, body)
185                 VALUES ( ?, ?, ?, ?, ?, ? )""", ( _start, email, sent_at, subject, hdr, body))
186     if count % 50 == 0: conn.commit()
187     if count % 100 == 0: time.sleep(1)
188
189     conn.commit()
190     cur.close()
```

Results from running gmane.py, spidering email messages and creating content.sqlite:

```
Run: gmane
/usr/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/gmane/gmane.py -l 10
How many messages: 10
http://mbox.dr-chuck.net/60421/60422 375
This server contains an email list.

This server expects a URL of the form:
http://mbox.dr-chuck.net/sakai.devel/4/5

And returns messages in an mbox format. No more than 10
messages can be requested at one time.

There is a copy of this hosted at:
http://mbox.dr-chuck.net/

That is served through CloudFlare and so it should provide
fast access anywhere in the world.

Did not find From
http://mbox.dr-chuck.net/60422/60423 375
This server contains an email list.

This server expects a URL of the form:
http://mbox.dr-chuck.net/sakai.devel/4/5
```

Content.sqlite database (raw data), created by running gmane.py (“spidering” 100 messages), screenshot needed for assignment:

The screenshot shows the DB Browser for SQLite interface. The main window displays the 'Messages' table with 59823 rows. The table has columns: id, email, sent\_at, subject, headers, and body. A specific row is selected, showing an email from ggolden@umich.edu. The right panel shows the SQL log with the following queries:

```

1 PRAGMA foreign_keys = '1';
2 PRAGMA database_list;
3 SELECT type, name, sql, tbl_name FROM "main".sqlite_master;
4 PRAGMA encoding;
5 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "main"."DNSMapping");
6 SELECT "_rowid_","* FROM "main"."DNSMapping" ORDER BY "_rowid_";
7 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "main"."Mapping");
8 SELECT "_rowid_","* FROM "main"."Mapping" ORDER BY "_rowid_";
9 SELECT COUNT(*) FROM (SELECT "_rowid_","* FROM "main"."Message");
10 SELECT "_rowid_","* FROM "main".Messages" ORDER BY "_rowid_";
11

```

## Gmodel.py (pt.1):

The screenshot shows a code editor window with several tabs at the top, including "Course5w4\_Notes.py", "README.txt", "gname.py", "gmodel.py", "gbasic.py", "Course5w2\_Notes.py", and "sprank.py". The main area displays the content of the "gmodel.py" file.

```
1 import sqlite3
2 import time
3 import re
4 import zlib
5 from datetime import datetime, timedelta
6
7 # Not all systems have this
8 try:
9     import dateutil.parser as parser
10 except:
11     pass
12
13 dnsmapping = dict()
14 mapping = dict()
15
16 def fixsender(sender, allsenders=None):
17     global dnsmapping
18     global mapping
19     if sender is None:
20         return None
21     sender = sender.strip().lower()
22     sender = sender.replace('<') + '>'.replace('>', '<')
23
24     # Check if we have a hacked gname.org from address
25     if allsenders is not None and sender.endswith('gname.org'):
26         pieces = sender.split('-')
27         realsender = None
28         for s in allsenders:
29             if s.startswith(pieces[0]):
30                 realsender = sender
31                 sender = s
32                 # print(realsender, sender)
33                 break
34         if realsender is None:
35             for s in mapping:
36                 if s.startswith(pieces[0]):
37                     realsender = sender
38                     sender = mapping[s]
39                     # print(realsender, sender)
40                     break
41         if realsender is None:
42             sender = pieces[0]
43
44         mpieces = sender.split("@")
45         if len(mpieces) != 2:
46             return sender
47         dns = mpieces[1]
48         x = dns
49         pieces = dns.split(".")
50         if dns.endswith(".edu") or dns.endswith(".com") or dns.endswith(".org") or dns.endswith(".net"):
51             dns = ".".join(pieces[-2:])
52         else:
53             dns = ".".join(pieces[-3:])
54         # if dns != x: print(x,dns)
55         # if dns != dnsmapping.get(dns,dns): print(dns,dnsmapping.get(dns,dns))
56         dns = dnsmapping.get(dns, dns)
57     return mpieces[0] + '@' + dns
58
59 def parsemaildate(md):
60     # See if we have dateutil
61     try:
62         pdate = parser.parse(md)
63         test_at = pdate.isoformat()
64     except:
65         pass
66
67     # Non-dateutil version - we try our best
68
69     pieces = md.split()
70     notz = ".join(pieces[:4]).strip()
71
72     # Try a bunch of format variations - strftime() is lame
73     dnotz = None
74     for form in ['%d %b %Y %H:%M:%S', '%d %b %Y %H:%M:%S',
75                 '%d %b %Y %H:%M', '%d %b %Y %H:%M:%S',
76                 '%d %b %Y %H:%M', '%d %b %Y %H:%M']:
77         try:
78             dnotz = datetime.strptime(notz, form)
79             break
80         except:
81             continue
```

## Gmodel.py (pt.2):

```
Course5w4_Notes.py x README.txt x gmane.py x gmodel.py x gbasic.py x Course5w2_Notes.py x
81     if dnotz is None:
82         # print('Bad Date:',md)
83         return None
84
85     iso = dnotz.isoformat()
86
87     tz = "+0000"
88     try:
89         tz = pieces[4]
90         jval = int(tz).# Only want numeric timezone values
91         if tz == '-0000': tz = '+0000'
92         tzh = tz[:3]
93         tzm = tz[3:]
94         tz = tzh+":"+tzm
95     except:
96         pass
97
98     return iso+tz
99
100    # Parse out the info...
101    def parseheader(hdr, allsenders=None):
102        if hdr is None or len(hdr) < 1: return None
103        sender = None
104        x = re.findall('\nFrom:.*<(\S+@\S+)>\n', hdr)
105        if len(x) >= 1:
106            sender = x[0]
107        else:
108            x = re.findall('\nFrom: (\S+@\S+)\n', hdr)
109            if len(x) >= 1:
110                sender = x[0]
111
112        # normalize the domain name of Email addresses
113        sender = fixsender(sender, allsenders)
114
115        date = None
116        y = re.findall('\nDate: .*, (.*)\n', hdr)
117        sent_at = None
118        if len(y) >= 1:
119            tdate = y[0]
120            tdate = tdate[:26]
121            try:
122                sent_at = parsemaildate(tdate)
123            except Exception as e:
124                # print('Date ignored ',tdate, e)
125                return None
126
127        subject = None
128        z = re.findall('\nSubject: (.*)\n', hdr)
129        if len(z) >= 1: subject = z[0].strip().lower()
130
131        guid = None
132        z = re.findall('\nMessage-ID: (.*)\n', hdr)
133        if len(z) >= 1: guid = z[0].strip().lower()
134
135        if sender is None or sent_at is None or subject is None or guid is None:
136            return None
137        return (guid, sender, subject, sent_at)
138
139    conn = sqlite3.connect('index.sqlite')
140    cur = conn.cursor()
141
142    cur.execute('''DROP TABLE IF EXISTS Messages ''')
143    cur.execute('''DROP TABLE IF EXISTS Senders ''')
144    cur.execute('''DROP TABLE IF EXISTS Subjects ''')
145    cur.execute('''DROP TABLE IF EXISTS Replies ''')
146
147    cur.execute('''CREATE TABLE IF NOT EXISTS Messages
148        (id INTEGER PRIMARY KEY, guid TEXT UNIQUE, sent_at INTEGER,
149        sender_id INTEGER, subject_id INTEGER,
150        headers BLOB, body BLOB)''')
151    cur.execute('''CREATE TABLE IF NOT EXISTS Senders
152        (id INTEGER PRIMARY KEY, sender TEXT UNIQUE)''')
153    cur.execute('''CREATE TABLE IF NOT EXISTS Subjects
154        (id INTEGER PRIMARY KEY, subject TEXT UNIQUE)''')
155    cur.execute('''CREATE TABLE IF NOT EXISTS Replies
156        (from_id INTEGER, to_id INTEGER)''')
157
158    conn_1 = sqlite3.connect('mapping.sqlite')
159    cur_1 = conn_1.cursor()
```

### Gmodel.py (pt.3):

```
157 conn_1 = sqlite3.connect('mapping.sqlite')
158 cur_1 = conn_1.cursor()
159
160 cur_1.execute('''SELECT old,new FROM DNSMapping''')
161 for message_row in cur_1:
162     dnsmapping[message_row[0].strip().lower()] = message_row[1].strip().lower()
163
164 mapping = dict()
165 cur_1.execute('''SELECT old,new FROM Mapping''')
166 for message_row in cur_1:
167     old = fixsender(message_row[0])
168     new = fixsender(message_row[1])
169     mapping[old] = new
170
171 # Done with mapping.sqlite
172 conn_1.close()
173
174 # Open the main content (Read only)
175 conn_1 = sqlite3.connect('file:content.sqlite?mode=ro', uri=True)
176 cur_1 = conn_1.cursor()
```

### Gmodel.py (pt.4):

```
177 cur_1 = conn_1.cursor()
178
179 allsenders = list()
180 cur_1.execute('''SELECT email FROM Messages''')
181 for message_row in cur_1:
182     sender = fixsender(message_row[0])
183     if sender is None: continue
184     if 'gmane.org' in sender: continue
185     if sender in allsenders: continue
186     allsenders.append(sender)
187
188 print("Loaded allsenders", len(allsenders), "and mapping", len(mapping), "dns mapping", len(dnsmapping))
189
190 cur_1.execute('''SELECT headers, body, sent_at
191                 FROM Messages ORDER BY sent_at''')
192
193 senders = dict()
194 subjects = dict()
195 guids = dict()
196
197 count = 0
198
199 for message_row in cur_1:
200     hdr = message_row[0]
201     parsed = parseheader(hdr, allsenders)
202     if parsed is None: continue
203     (guid, sender, subject, sent_at) = parsed
204
205     # Apply the sender mapping
206     sender = mapping.get(sender, sender)
207
208     count = count + 1
209     if count % 250 == 1: print(count, sent_at, sender)
210     # print(guid, sender, subject, sent_at)
211
212     if 'gmane.org' in sender:
213         print("Error in sender ==>", sender)
214
215     sender_id = senders.get(sender, None)
216     subject_id = subjects.get(subject, None)
217     guid_id = guids.get(guid, None)
218
219     if sender_id is None:
220         cur.execute('INSERT OR IGNORE INTO Senders (sender) VALUES ( ? )', (sender,))
221         conn.commit()
222         cur.execute('SELECT id FROM Senders WHERE sender=? LIMIT 1', (sender,))
223         try:
224             row = cur.fetchone()
225             sender_id = row[0]
226             senders[sender] = sender_id
227         except:
228             print('Could not retrieve sender id', sender)
229             break
230
231     if subject_id is None:
232         cur.execute('INSERT OR IGNORE INTO Subjects (subject) VALUES ( ? )', (subject,))
233         conn.commit()
234         cur.execute('SELECT id FROM Subjects WHERE subject=? LIMIT 1', (subject,))
235         try:
236             row = cur.fetchone()
237             subject_id = row[0]
238             subjects[subject] = subject_id
239         except:
240             print('Could not retrieve subject id', subject)
241
242     # print(sender_id, subject_id)
243     cur.execute('INSERT OR IGNORE INTO Messages (guid,sender_id,subject_id,sent_at,headers,body) VALUES ( ?, ?, ?, ?, ?, ? )',
244                (guid, sender_id, subject_id, sent_at,
245                 zlib.compress(message_row[0].encode()), zlib.compress(message_row[1].encode())))
246     conn.commit()
247
248     cur.execute('SELECT id FROM Messages WHERE guid=? LIMIT 1', (guid,))
249     try:
250         row = cur.fetchone()
251         message_id = row[0]
252         guids[guid] = message_id
253     except:
254         print('Could not retrieve guid id', guid)
255         break
256
257 cur.close()
258 cur_1.close()
```

Results from running gmodel.py, creates index.sqlite, prints data (guid, date/time message sent, and the sender's email) that's being inserted into database (screenshot needed for assignment):

```

Run: gmodel x
/Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/gmane/gmodel.py
Loaded allsenders 1803 and mapping 29 dns mapping 1
1 2005-12-08T23:34:30-06:00 ggolden22@mac.com
251 2005-12-22T10:03:20-08:00 tpamsler@ucdavis.edu
501 2006-01-12T11:17:34-05:00 lance@indiana.edu
751 2006-01-24T11:13:28-08:00 vrajgopalan@ucmerced.edu
1001 2006-02-02T08:27:30-07:00 john.ellis@rsmart.com
1251 2006-02-15T19:13:17-05:00 ggolden22@mac.com
1501 2006-02-23T16:49:00-05:00 csev@umich.edu
1751 2006-03-13T13:48:36-05:00 csev@umich.edu
2001 2006-03-27T13:34:32-06:00 swgithen@tu.edu
2251 2006-04-05T12:38:43-04:00 ccount@mit.edu
2501 2006-04-17T13:28:54-07:00 jholzman@berkeley.edu
2751 2006-04-25T14:26:03-07:00 vrajgopalan@ucmerced.edu
3001 2006-05-03T13:49:11-04:00 cheryl.wogahn@yale.edu
3251 2006-05-10T17:14:01+00:00 nuno@ufp.pt
3501 2006-05-17T10:37:49-04:00 ys2n@virginia.edu
3751 2006-05-23T14:52:13-04:00 azekoski@unicon.net
4001 2006-06-06T12:51:32-04:00 ajpoland@indiana.edu
4251 2006-06-14T08:28:01-04:00 lance@indiana.edu
4501 2006-06-21T14:31:06-05:00 mustansar@rice.edu
4751 2006-06-29T15:52:15+01:00 antranig@cam.ac.uk
5001 2006-07-12T11:50:55-04:00 markjnorton@earthlink.net
5251 2006-07-20T17:24:14-07:00 sinou@etudes.org
5501 2006-07-31T18:45:12-04:00 ggolden22@mac.com
5751 2006-08-08T07:44:09-04:00 dave.ross@gmail.com
6001 2006-08-14T15:32:02-04:00 daisy.flemming@gmail.com
6251 2006-08-23T02:02:38-12:00 woleraymond@genownow.com
6501 2006-08-31T13:53:24-04:00 ryanaby@gmail.com
6751 2006-09-12T17:52:18-05:00 zach@aeroplanesoftware.com
7001 2006-09-19T08:34:39+01:00 b.steele@blueyonder.co.uk
7251 2006-09-27T10:56:27+02:00 wilfrid.tabeta@gmail.com
7501 2006-10-06T10:03:19+02:00 b.toeter@uva.nl
7751 2006-10-17T10:09:17-07:00 sfischbein@ucdavis.edu
8001 2006-10-27T12:40:07+01:00 nuno@ufp.pt
8251 2006-11-08T10:16:19+00:00 harriet@cam.ac.uk
8501 2006-11-19T17:08:08+01:00 jcreyesbe@ya.com
8751 2006-11-30T16:03:54-05:00 markjnorton@earthlink.net
9001 2006-12-13T15:35:15-05:00 jmpease@syr.edu
9251 2006-12-27T17:27:22+00:00 nuno@ufp.pt
9501 2007-01-12T22:27:47+00:00 matthew.bucket@ox.ac.uk
9751 2007-01-24T22:26:10-05:00 ys2n@virginia.edu
10001 2007-02-03T10:14:31-05:00 jlmeng@umich.edu
10251 2007-02-14T10:04:45+00:00 liam@k-int.com
10501 2007-02-23T10:19:58-05:00 ajpoland@indiana.edu

```

Index.sqlite (cleaned-up data)m created by gmodel.py (screenshot needed for assignment):

	id	guid	sent_at	sender_id	subject_id	headers	body
1	12861	<pine.osx.4.64...	2007-05-25 19...	8	6333	BLOB	BLOB
2	23024	<4810a18f.703...	2008-04-24 15...	541	10897	BLOB	BLOB
3	29350	<bay139-dav...	2009-03-06 17...	928	13788	BLOB	BLOB
4	25335	<488e361f.6d3...	2008-07-28 21...	28	11885	BLOB	BLOB
5	23796	<be5dfc6d080...	2008-05-17 11...	23	11156	BLOB	BLOB
6	21890	<47e15af3.304...	2008-03-19 18...	21	10382	BLOB	BLOB
7	17540	<pine.osx.4.64...	2007-10-12 18...	8	8424	BLOB	BLOB
8	17194	<be5dfc6d071...	2007-10-04 21...	23	8300	BLOB	BLOB
9	28708	<2bd34e75f6ca...	2009-01-27 21...	79	13469	BLOB	BLOB
10	23347	<0741497-a4...	2008-05-06 22...	600	11017	BLOB	BLOB
11	19586	<4761b484.10...	2007-12-13 22...	114	9320	BLOB	BLOB
12	25443	<4283be85-88...	2008-07-30 22...	672	11936	BLOB	BLOB
13	17986	<ffba1060-674...	2007-11-01 00...	600	8252	BLOB	BLOB
14	24756	<35c6b060080...	2008-07-02 12...	742	11609	BLOB	BLOB
15	25959	<35c6b060080...	2008-08-14 20...	742	12157	BLOB	BLOB
16	23563	<1c617430-c0f...	2008-05-12 14...	291	11004	BLOB	BLOB
17	20911	<47a8a0a7.7fb...	2008-02-05 17...	28	9917	BLOB	BLOB
18	23683	<482c5366.20...	2008-05-15 15...	513	11125	BLOB	BLOB
19	18375	<20071117225...	2007-11-17 22...	47	8791	BLOB	REDACTED

Type of data currently in cell: Binary  
127 byte(s)

Show SQL submitted by Application

```

23 SELECT COUNT(*) FROM (SELECT "_rowid_"," FROM "main"."Senders" ORDER BY
24 _rowid_") FROM "main"."Senders" ORDER BY _rowid_ ASC LIMIT 0, 4!
25 SELECT COUNT(*) FROM (SELECT "_rowid_"," FROM "main"."Subjects" ORDER BY
26 _rowid_") FROM "main"."Subjects" ORDER BY _rowid_ ASC LIMIT 0, 4
27 SELECT COUNT(*) FROM (SELECT "_rowid_"," FROM "main"."Senders" ORDER BY
28 _rowid_") FROM "main"."Senders" ORDER BY _rowid_ ASC LIMIT 0, 4!
29 SELECT COUNT(*) FROM (SELECT "_rowid_"," FROM "main"."Replies" ORDER BY
30 _rowid_") FROM "main"."Replies" ORDER BY _rowid_ ASC LIMIT 0, 49
31 SELECT COUNT(*) FROM (SELECT "_rowid_"," FROM "main"."Messages" ORDER I
32 SELECT "_rowid_"," FROM "main"."Messages" ORDER BY "body" ASC LIMIT 0, 49
33

```

## F.3: Week 6 - Visualizing Email Data Assignment (Gmane/Mail pt.2):

### Prompt for the assignment:

Welcome Alex Huerta from Capstone: Retrieving, Processing, and Visualizing Data with Python

#### Mailing List Data - Part II

In this assignment you will visualize the mailing list data you have downloaded from <http://mbox.dr-chuck.net/> and take some screen shots. Important: You do not have to download all of the data - 3 months is 1700 messages, 6 mths 3200 and 1 yr 9000. Do not use low counts like 1, 20, 100, 300. You can run gmane multiple times to download more messages. It is completely acceptable to visualize a small subset of the data in the gbasic screenshot.

Don't take off points for little mistakes. If they seem to have done the assignment give them full credit. Feel free to make suggestions if there are small mistakes. Please keep your comments positive and useful. The course staff will assign the last 30% of the grade and it may take a few days to get graded. - For students in mainland China that can't access the Google API for the timeline, take a screenshot of gline.js open in your editor showing the numbers, dates of messages at the top. Please add a note to your assignment.

#### Please Upload Your Submission:

A screen shot of you running the gbasic.py program to compute basic histogram data on the messages you have retrieved.

No file chosen

(Please use PNG or JPG files)

A screen shot of word cloud visualization for the messages you have retrieved.

No file chosen

(Please use PNG or JPG files)

A screen shot of time line visualization for the messages you have retrieved.

No file chosen

(Please use PNG or JPG files)

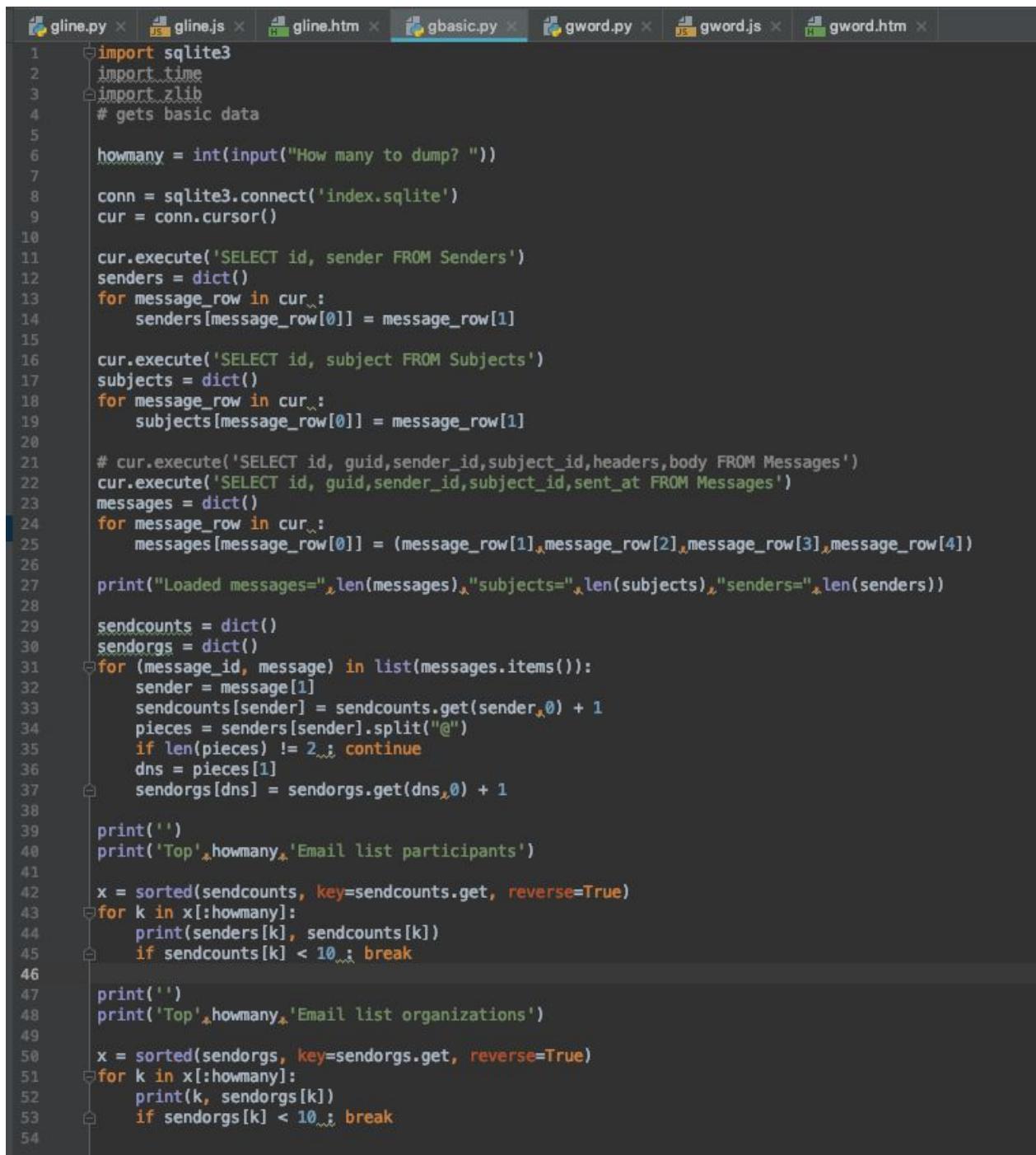
Optional Challenge: Change the gline.py program to show the message count by year instead of by month and take a screen shot of the by year visualization. You can switch from a by-month to a by-year visualization by changing only a few lines in gline.py. The puzzle is to figure out the smallest change to accomplish the change. If you do not want to do this optional challenge - just upload the above image a second time.

No file chosen

(Please use PNG or JPG files)

Enter optional comments below

## Gbasic.py:



The screenshot shows a code editor with multiple tabs at the top, including gline.py, gline.js, gline.htm, gbasic.py (which is the active tab), gword.py, gword.js, and gword.htm. The gbasic.py tab contains the following Python script:

```
1 import sqlite3
2 import time
3 import zlib
4 # gets basic data
5
6 howmany = int(input("How many to dump? "))
7
8 conn = sqlite3.connect('index.sqlite')
9 cur = conn.cursor()
10
11 cur.execute('SELECT id, sender FROM Senders')
12 senders = dict()
13 for message_row in cur:
14     senders[message_row[0]] = message_row[1]
15
16 cur.execute('SELECT id, subject FROM Subjects')
17 subjects = dict()
18 for message_row in cur:
19     subjects[message_row[0]] = message_row[1]
20
21 # cur.execute('SELECT id, guid, sender_id, subject_id, headers, body FROM Messages')
22 cur.execute('SELECT id, guid, sender_id, subject_id, sent_at FROM Messages')
23 messages = dict()
24 for message_row in cur:
25     messages[message_row[0]] = (message_row[1], message_row[2], message_row[3], message_row[4])
26
27 print("Loaded messages=%d, subjects=%d, senders=%d" % (len(messages), len(subjects), len(senders)))
28
29 sendcounts = dict()
30 sendorgs = dict()
31 for (message_id, message) in list(messages.items()):
32     sender = message[1]
33     sendcounts[sender] = sendcounts.get(sender, 0) + 1
34     pieces = senders[sender].split("@")
35     if len(pieces) != 2: continue
36     dns = pieces[1]
37     sendorgs[dns] = sendorgs.get(dns, 0) + 1
38
39 print('')
40 print('Top %d Email list participants' % howmany)
41
42 x = sorted(sendcounts, key=sendcounts.get, reverse=True)
43 for k in x[:howmany]:
44     print(senders[k], sendcounts[k])
45     if sendcounts[k] < 10: break
46
47 print('')
48 print('Top %d Email list organizations' % howmany)
49
50 x = sorted(sendorgs, key=sendorgs.get, reverse=True)
51 for k in x[:howmany]:
52     print(k, sendorgs[k])
53     if sendorgs[k] < 10: break
54
```

Results from running gbasic.py, top 20 email list participants & organizations  
(screenshot need for week 4 assignment and this assignment):

```
Run: gbasic x
▶ Top 20 Email list participants
steve.swinsburg@swinsborg.com 3337
azeckoski@unicon.net 1907
ian@cam.ac.uk 1591
csev@umich.edu 1490
david.horwitz@uct.ac.za 1221
matthew@longsight.com 1209
neal.caidin@apereo.org 1035
stephen.marquard@uct.ac.za 931
adam.marshall@ox.ac.uk 929
arwhyte@umich.edu 868
adrian.r.fish@gmail.com 764
slt@columbia.edu 732
jimeng@umich.edu 709
hedrick@rutgers.edu 672
clay.fenlason@gatech.edu 670
markjnorton@earthlink.net 669
ottenhoff@longsight.com 653
chmaurer@indiana.edu 619
bkirschn@umich.edu 599
swgithen@mtu.edu 585

Top 20 Email list organizations
umich.edu 6805
gmail.com 5683
swinsborg.com 3337
cam.ac.uk 2626
uct.ac.za 2582
indiana.edu 2333
longsight.com 2315
unicon.net 2310
ox.ac.uk 1544
berkeley.edu 1388
stanford.edu 1355
ucdavis.edu 1184
rsmart.com 1109
apereo.org 1038
rutgers.edu 908
etudes.org 895
gatech.edu 865
virginia.edu 765
columbia.edu 746
earthlink.net 670

Process finished with exit code 0
```

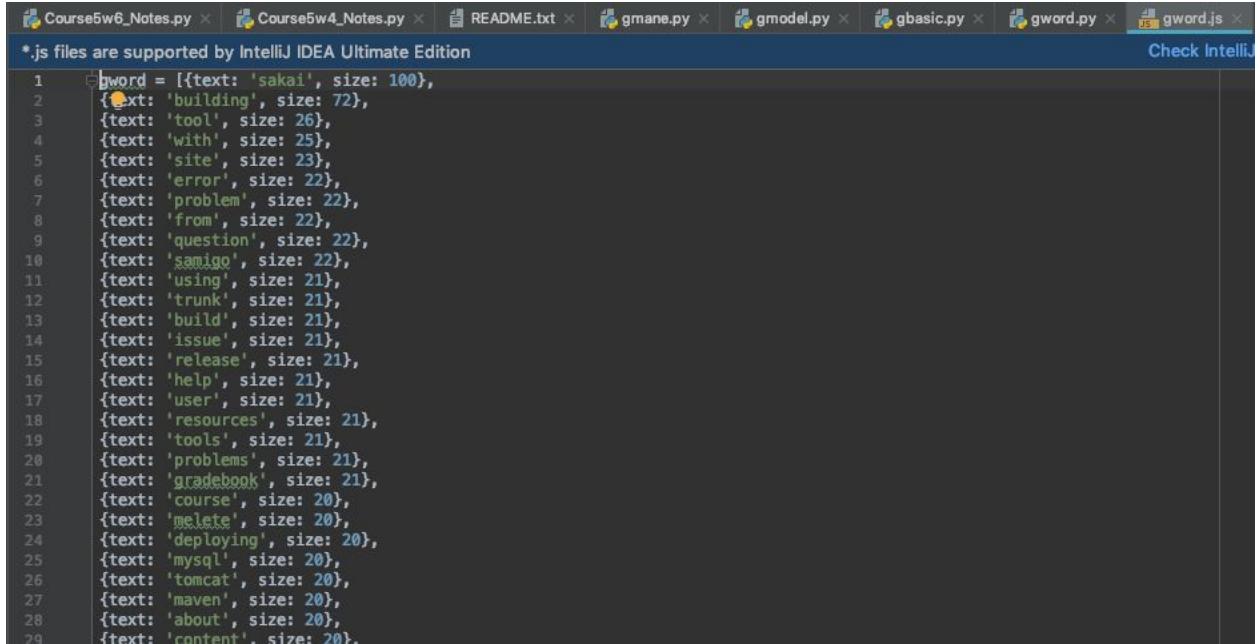
Gword.py, creates gword.js:

The screenshot shows the PyCharm IDE interface. The top navigation bar has tabs for Course5w6\_Notes.py, Course5w4\_Notes.py, README.txt, gmane.py, gmodel.py, gbasic.py, and gword.py. The main code editor window displays Gword.py, which reads from index.sqlite and gmane.sqlite to generate a word frequency distribution and write it to gword.js. The code uses SQLite3, string manipulation, and file I/O. The bottom run tab shows the command run, the output of the script, and the exit status.

```
1 import sqlite3
2 import time
3 import zlib
4 import string
5
6 conn = sqlite3.connect('index.sqlite')
7 cur = conn.cursor()
8
9 cur.execute('SELECT id, subject FROM Subjects')
10 subjects = dict()
11 for message_row in cur:
12     subjects[message_row[0]] = message_row[1]
13
14 # cur.execute('SELECT id, guid, sender_id, subject_id, headers, body FROM Messages')
15 cur.execute('SELECT subject_id FROM Messages')
16 counts = dict()
17 for message_row in cur:
18     text = subjects[message_row[0]]
19     text = text.translate(str.maketrans(string.punctuation))
20     text = text.translate(str.maketrans(string.punctuation))
21     text = text.strip()
22     text = text.lower()
23     words = text.split()
24     for word in words:
25         if len(word) < 4: continue
26         counts[word] = counts.get(word, 0) + 1
27
28 x = sorted(counts, key=counts.get, reverse=True)
29 highest = None
30 lowest = None
31 for k in x[:100]:
32     if highest is None or highest < counts[k]:
33         highest = counts[k]
34     if lowest is None or lowest > counts[k]:
35         lowest = counts[k]
36 print('Range of counts:', highest, lowest)
37
38 # Spread the font sizes across 20-100 based on the count
39 bigsize = 80
40 smallsize = 20
41
42 fhand = open('gword.js', 'w')
43 fhand.write("gword = [")
44 first = True
45 for k in x[:100]:
46     if not first: fhand.write(",\n")
47     first = False
48     size = counts[k]
49     size = (size - lowest) / float(highest - lowest)
50     size = int((size * bigsize) + smallsize)
51     fhand.write("{text: '"+k+"', size: "+str(size)+"}")
52 fhand.write("\n];\n")
53 fhand.close()
54
55 print("Output written to gword.js")
56 print("Open gword.htm in a browser to see the visualization")
57
```

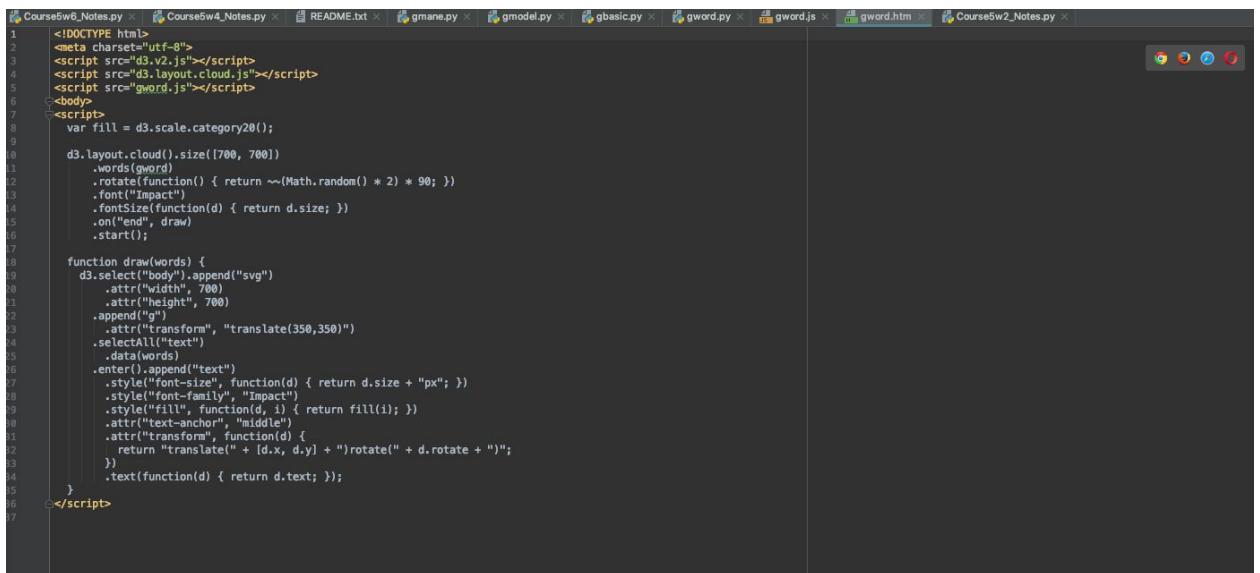
Run: gword x  
/Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/gmane/gword.py  
Range of counts: 43506 330  
Output written to gword.js  
Open gword.htm in a browser to see the visualization  
Process finished with exit code 0

## Gword.js (dimension and design info for d3 visualization):



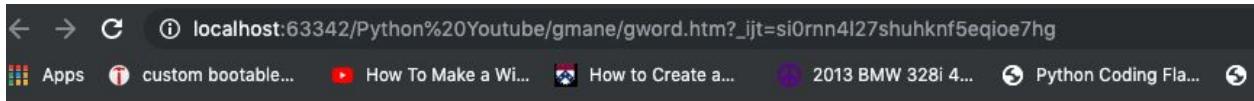
```
1  bword = [{text: 'sakai', size: 100},
2  {text: 'building', size: 72},
3  {text: 'tool', size: 26},
4  {text: 'with', size: 25},
5  {text: 'site', size: 23},
6  {text: 'error', size: 22},
7  {text: 'problem', size: 22},
8  {text: 'from', size: 22},
9  {text: 'question', size: 22},
10 {text: 'amigo', size: 22},
11 {text: 'using', size: 21},
12 {text: 'trunk', size: 21},
13 {text: 'build', size: 21},
14 {text: 'issue', size: 21},
15 {text: 'release', size: 21},
16 {text: 'help', size: 21},
17 {text: 'user', size: 21},
18 {text: 'resources', size: 21},
19 {text: 'tools', size: 21},
20 {text: 'problems', size: 21},
21 {text: 'gradebook', size: 21},
22 {text: 'course', size: 20},
23 {text: 'delete', size: 20},
24 {text: 'deploying', size: 20},
25 {text: 'mysql', size: 20},
26 {text: 'tomcat', size: 20},
27 {text: 'maven', size: 20},
28 {text: 'about', size: 20},
29 {text: 'content', size: 20},
```

## Gword.htm (link to Word Cloud Visualization):



```
1  <!DOCTYPE html>
2  <meta charset="utf-8">
3  <script src="d3.v2.js"></script>
4  <script src="d3.layout.cloud.js"></script>
5  <script src="gword.js"></script>
6  <body>
7  <script>
8    var fill = d3.scale.category20();
9
10   d3.layout.cloud().size([1700, 700])
11     .words(gword)
12     .rotate(function() { return ~ (Math.random() * 2) * 90; })
13     .font("Impact")
14     .fontSize(function(d) { return d.size; })
15     .on("end", draw)
16     .start();
17
18   function draw(words) {
19     d3.select("body").append("svg")
20       .attr("width", 1700)
21       .attr("height", 700)
22       .append("g")
23       .attr("transform", "translate(350,350)");
24     d3.selectAll("text")
25       .data(words)
26       .enter().append("text")
27       .style("font-size", function(d) { return d.size + "px"; })
28       .style("font-family", "Impact")
29       .style("fill", function(d, i) { return fill(i); })
30       .attr("text-anchor", "middle")
31       .attr("transform", function(d) {
32         return "translate(" + [d.x, d.y] + ")rotate(" + d.rotate + ")";
33       })
34       .text(function(d) { return d.text; });
35   }
36 </script>
```

Word Cloud Visualization (screenshot needed for assignment):



## Gline.py:

```
1 import sqlite3
2 import time
3 import zlib
4
5 conn = sqlite3.connect('index.sqlite')
6 cur = conn.cursor()
7
8 cur.execute('SELECT id, sender FROM Senders')
9 senders = dict()
10 for message_row in cur:
11     senders[message_row[0]] = message_row[1]
12
13 cur.execute('SELECT id, guid, sender_id, subject_id, sent_at FROM Messages')
14 messages = dict()
15 for message_row in cur:
16     messages[message_row[0]] = (message_row[1], message_row[2], message_row[3], message_row[4])
17
18 print("Loaded messages=%d, senders=%d" % (len(messages), len(senders)))
19
20 sendorgs = dict()
21 for (message_id, message) in list(messages.items()):
22     sender = message[1]
23     pieces = senders[sender].split("@")
24     if len(pieces) != 2: continue
25     dns = pieces[1]
26     sendorgs[dns] = sendorgs.get(dns, 0) + 1
27
28 # pick the top schools
29 orgs = sorted(sendorgs, key=sendorgs.get, reverse=True)
30 orgs = orgs[:10]
31 print("Top 10 Organizations")
32 print(orgs)
33
34 counts = dict()
35 months = list()
36 # cur.execute('SELECT id, guid, sender_id, subject_id, sent_at FROM Messages')
37 for (message_id, message) in list(messages.items()):
38     sender = message[1]
39     pieces = senders[sender].split("@")
40     if len(pieces) != 2: continue
41     dns = pieces[1]
42     if dns not in orgs: continue
43     month = message[3][:7]
44     if month not in months: months.append(month)
45     key = (month, dns)
46     counts[key] = counts.get(key, 0) + 1
47
48 months.sort()
49 # print counts
50 # print months
51
52 fhand = open('gline.js', 'w')
53 fhand.write("gline = [ ['Month']")
54 for org in orgs:
55     fhand.write(", '" + org + "'")
56 fhand.write("]")
57
58 for month in months:
59     fhand.write(",\n[ '" + month + "'")
60     for org in orgs:
61         key = (month, org)
62         val = counts.get(key, 0)
63         fhand.write(", '" + str(val) + "'")
64     fhand.write("]")
65
66 fhand.write("\n];\n")
67 fhand.close()
68
69 print("Output written to gline.js")
70 print("Open gline.htm to visualize the data")
```

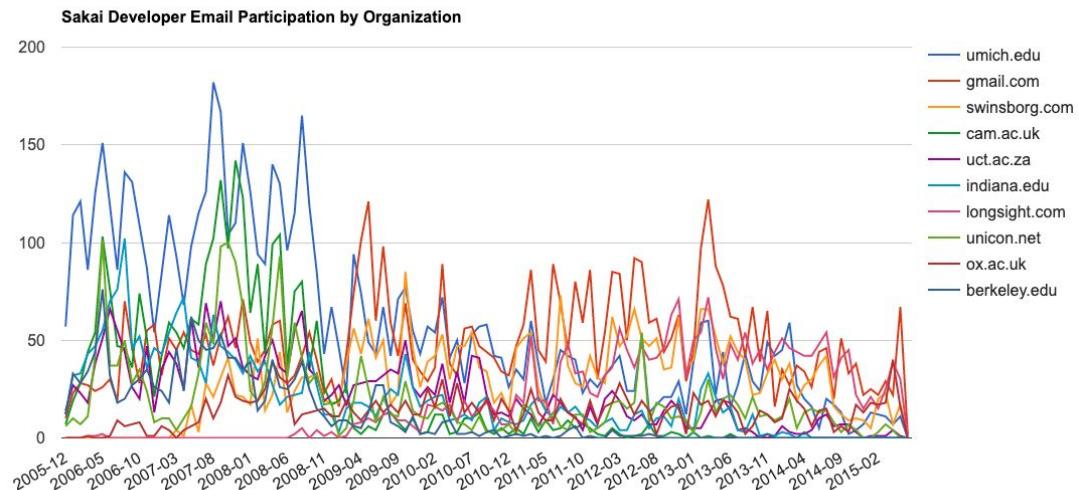
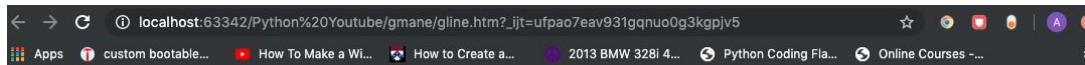
Results from running gline.py (top 10 organizations):

```
Run: gline x
    /Users/alex/PycharmProjects/PY4E/venv/bin/python /Users/alex/PycharmProjects/PY4E/gmane/gline.py
    Loaded messages= 59720 senders= 1800
    Top 10 Organizations
    ['umich.edu', 'gmail.com', 'swinsborg.com', 'cam.ac.uk', 'uct.ac.za', 'indiana.edu', 'longsight.com', 'unicon.net', 'ox.ac.uk', 'berkeley.edu']
    Output written to gline.js
    Open gline.htm to visualize the data
    Process finished with exit code 0
```

Gline.js (dimension and design info for d3 visualization):

```
*.js files are supported by IntelliJ IDEA Ultimate Edition
Check IntelliJ IDEA Ultimate Edition  Do not suggest Ultimate Edition  Ignore extension
1  gline = [ !'Month', 'umich.edu', 'gmail.com', 'swinsborg.com', 'cam.ac.uk', 'uct.ac.za', 'indiana.edu', 'longsight.com', 'unicon.net', 'ox.ac.uk', 'berkeley.edu'],
2  [ '2005-12', 57, 10, 7, 14, 12, 0, 6, 0, 12],
3  [ '2006-01', 114, 23, 0, 19, 27, 32, 0, 10, 0, 33],
4  [ '2006-02', 121, 28, 0, 28, 23, 33, 0, 7, 0, 28],
5  [ '2006-03', 86, 27, 0, 44, 18, 43, 1, 11, 1, 34],
6  [ '2006-04', 125, 24, 0, 54, 38, 47, 1, 42, 0, 44],
7  [ '2006-05', 151, 26, 0, 103, 51, 55, 2, 101, 0, 76],
8  [ '2006-06', 119, 30, 0, 76, 66, 66, 70, 0, 37, 1, 32],
9  [ '2006-07', 86, 19, 0, 47, 55, 76, 0, 37, 9, 18],
10 [ '2006-08', 136, 70, 0, 46, 44, 102, 0, 50, 6, 20],
11 [ '2006-09', 131, 46, 0, 36, 26, 46, 0, 28, 7, 27],
12 [ '2006-10', 109, 28, 0, 74, 20, 52, 0, 35, 8, 30],
13 [ '2006-11', 87, 55, 0, 51, 47, 36, 0, 24, 1, 35],
14 [ '2006-12', 54, 58, 0, 21, 13, 46, 0, 8, 1, 26],
15 [ '2007-01', 84, 32, 0, 42, 35, 43, 0, 10, 6, 24],
16 [ '2007-02', 114, 51, 0, 59, 44, 54, 0, 10, 4, 18],
17 [ '2007-03', 93, 45, 4, 54, 38, 64, 0, 4, 0, 46],
18 [ '2007-04', 68, 54, 1, 46, 25, 72, 0, 10, 4, 24],
19 [ '2007-05', 98, 45, 17, 61, 61, 41, 0, 16, 6, 62],
20 [ '2007-06', 115, 43, 3, 58, 36, 39, 0, 33, 8, 50],
21 [ '2007-07', 126, 53, 28, 89, 69, 28, 0, 59, 20, 45],
22 [ '2007-08', 182, 37, 21, 102, 58, 63, 0, 48, 10, 46],
23 [ '2007-09', 167, 52, 31, 132, 70, 47, 0, 98, 18, 53],
24 [ '2007-10', 104, 62, 41, 97, 47, 44, 0, 100, 32, 41],
25 [ '2007-11', 118, 46, 22, 142, 51, 48, 0, 98, 21, 41],
26 [ '2007-12', 151, 71, 21, 123, 36, 33, 0, 67, 19, 35],
27 [ '2008-01', 126, 49, 17, 64, 32, 42, 0, 24, 18, 39],
28 [ '2008-02', 94, 39, 51, 89, 38, 34, 0, 19, 19, 14],
29 [ '2008-03', 89, 45, 14, 43, 42, 39, 0, 29, 25, 19],
30 [ '2008-04', 140, 58, 22, 99, 50, 27, 0, 57, 39, 40],
31 [ '2008-05', 130, 60, 44, 104, 36, 17, 0, 95, 31, 26],
32 [ '2008-06', 96, 28, 13, 36, 33, 21, 0, 31, 28, 25],
33 [ '2008-07', 115, 32, 24, 75, 55, 22, 2, 59, 7, 30],
34 [ '2008-08', 165, 42, 31, 80, 65, 23, 5, 41, 12, 39],
35 [ '2008-09', 119, 54, 31, 35, 35, 44, 0, 28, 13, 25],
36 [ '2008-10', 85, 40, 33, 66, 31, 27, 4, 33, 14, 15],
```

Gline.htm (link to Time-Line Visualization), screenshot needed for assignment:



## G. Course Certificates

### G.1: Course 1



### G.2: Course 2



### G.3: Course 3



### G.4: Course 4



## G.5: Course 5

