Homework 0

[ 100 points - due by 11:59 pm, August?, 2017 ]

Submit these files to the CS submission system at the usual place by 11:59. You may work on your own or with 1-2 partners on the programming portions of this assignment. (The reading/response is individual only.) Groups of more than 3 should split into smaller groups. Remember that partners need to work in the same physical location, share composition time equally (or each compose on their own machines) and be fully equal owners and producers of their work. Have fun experimenting! [cs35 homepage](https://www.cs.hmc.edu/~dodds/cs35/)

**Downloads**

This week has two zip archives to download -- grab them at the start of class & follow along:

**Setup**

Our primary toolsets will be Python3's *Anaconda* edition and libraries, a text editor (VS Code is a good free one, but you're welcome to use another, like Atom), and the ipython command-line (terminal/shell) for execution. If you were in cs5 in the fall of 2016, you have all of this! If you need any of these things, grab and install them from [this term's cs5 page](https://www.cs.hmc.edu/twiki/bin/view/CS5/OwnMachines).

**Submission**

Although there are several files in each assignment, for ease of handling them, we'll ask you to submit a single zipped archive named hw0.zip with all of those files (except the reading response). *Please do not include the large folders of data -- we have those!* Submit your hw0.zip at the [usual submission site](http://cicero.cs.hmc.edu/).

**Problem 0: Reading and response**  (**hw0pr0.txt** - 5 pts)

Similar in spirit to cs5's reading/response assignments, each week there will be a short article overlapping that week's topics (sometimes broadly, other times narrowly). [This week's article, from the *NYTimes*, elaborates on the idea of word choice as a signal of deeper mood.](https://www.nytimes.com/2016/11/27/science/language-negative-words.html?_r=1) With each article, problem 0 asks you to compose a one-paragraph (4-5 sentences) reflection. For this article, either (a) reflect on whether you believe computational analysis is appropriate (or worthwhile) for investigating these kinds of multiple-layers of meaning in text or (b) share an example of these multiple-layers from your experience. Your answer should incorporate ideas from the article, ideally along with your own take/experiences. Submit your reflection in a file named hw0pr0.txt. Since this is individual, you don't need to include it in the archive.

**Problem 1: Review, functions, and challenges + Scavenger Hunt** (**hw0pr1.py** - 20 pts - lab problem)

Problem 1 this week should be a file you create and experiment with in lab -- and there will be time dedicated to understanding and making progress on these challenges. For this week, start by [downloading this zip file](https://drive.google.com/open?id=0BwPWh-3AmiLxU1ZPSU5fNzRjeWM) and trying out hw0pr1.py within it. You should include a number of functions that will help with the next two problems (so it's well worth it!) If you come to lab and work on this, you'll receive full credit on this problem, even if you don't complete all of the challenges. (Some, however, you'll *want and need* to complete later on…). This week the functions include

* times42( s ): which should print the string s 42 times (on separate lines)
* alien( N ): should return the string "aliii...iiien" with exactly N "i"s
* count\_digits( s ): *returns the number of digits in the input string s*
* clean\_digits( s ): *returns only the digits in the input string s*
* clean\_word( s ): *returns an all-lowercase, all-letter version of the input string s*
* The [string library at this link](https://docs.python.org/3.1/library/string.html) will be a good reference for the last three…
* Also, please submit your **examples.py** file from the inclass folder.

*All of your functions - in this problem and others - should include docstrings that indicate, at least briefly, what they do.*

**Scavenger Hunt [Lab Problem Part 2]**

Follow this link to [skip this story](#etfxwkcfvf8q) and go right to the instructions.

It’s the first day of CS 35. Prof. Dodds has just emitted a remarkably peppy and engaging “That’s it for today! See you \_all\_ in lab,” and you begin collecting your belongings before heading for the door. However, there’s something weighing on your mind -- earlier today, you finally decided to ask out that classmate you’ve had a crush on since orientation week. It seems to have gone well -- they accepted, and asked what you were planning on doing. To be perfectly honest, you hadn’t expected them to actually say yes, so you hadn’t really spent any time thinking up ideas for a romantic first date. Under the tremendous pressure of the moment, you began sweating profusely -- possibly jeopardizing the date you had somehow just secured. But then, salvation came in the form of a memory of your mother: when you were a small child, she had told you that cooking together is always a good way to get to know somebody. In fact, she had proclaimed that’s what she had always done on first dates.   
  
Much relieved, you wrested yourself from the growing pool of sweat accumulating under your feet, and sputtered “My mother…”  
  
Your date frowned -- *Oh god*, you had thought, *they think I want them to meet my parents already. Oh my god what have I done. What do I do. What do I say. aaAAAAAAAAA!* However, one of what you consider to be your best qualities is playing off mistakes as being wholly intentional. You added (quite smoothly), “My mother has some really good homemade pie recipes. I’ve been really missing them recently, but haven’t gotten around to making them yet -- #worklifebalance, I guess. I was wondering if you’d like to try making a few -- they probably won’t be any good, but maybe it’d be fun.”   
  
Your date broke into a huge smile, and said, “Wait, that’s actually one of the sweetest things I’ve ever heard -- of course I would! Let me know what groceries I can pick up on my way over.” They then stole a look at their phone, gave an apologetic “Gotta run -- I’m late for class!” and headed for the door. You remained there for quite a few moments, entirely petrified -- you knew your mom had never baked a pie before in her life. Pies, she always claimed, dredged up unsavory memories of her high school geometry class. In fact, you had suggested it only because a particularly impassioned math professor had just given you a long and involved speech in your previous lecture about how pies were the perfect foods -- not because they sound similar in name to the famous mathematical constant, but because they can be either sweet or savory, like crepes. Supposedly, this was some sort of “helpful” analogy for understanding implicit differentiation, but to be honest, she had lost you immediately after discussing the syllabus.   
  
You mentally snap back to the present and realize you’re still in the CS 35 lecture hall. Somehow, you’re not the last one in the room -- an all-knowing, benevolent grutor (who decided to sit in on the class) has been patiently holding the door for you for the past five minutes, and will probably wait for an extra three, as that one kid who sat in the front of the room compulsively asking questions appears to have brought 15 lbs of Japanese stationary with him and is having a hard time collecting his belongings. You begin making your way up the stairs, but then being somewhat mentally preoccupied, trip impressively on the very last step. All your papers and notes go flying into the air, and you are almost entirely certain you’ve just broken your shin.   
  
The grutor springs into action and quickly begins gathering up your dropped belongings in a strikingly approachable manner. You look into the grutor’s eyes, which sparkle with a mischievous, knowing glint. You are convinced that the grutor is actually George Clooney. The grutor asks you (worriedly, but not oppressively so) if everything is alright. You are struck by an overwhelming urge to tell the grutor everything. You know they would understand.   
  
You give in and explain the whole situation. The grutor tells you not to worry, saying “Actually, it’s funny -- the first time *I* tried to ask somebody out, I did the \_exact\_ same thing. Don’t worry, actually lots of people make up a mother’s-pie-recipe story when they ask somebody out for the first time. In fact when I first took the course, I asked a grutor this same question.” You feel comforted -- this grutor *understands* you. The grutor tells you that if you remember to remind them during lab, they’ll give you their personal repository of pie recipes, helpfully stored as .txt files on their hard drive. You begin to cry, and thank the grutor for their time.   
  
Lab day comes. You arrive early, but the benevolent grutor hasn’t shown up -- instead, there’s a hastily-scrawled message on the blackboard: “NO TIME TO EXPLAIN: PIES HERE,” with an arrow pointing to what you take to be the grutor’s laptop. You look around the room -- nobody else has arrived yet. You wonder if it’s right for you to take the laptop -- it seems vaguely morally suspect to do so with the grutor absent. But who else would they have left the message for? Besides, the message sounds ominous… Maybe the grutor is in trouble. You look around the room; you’re sure nobody else is there. Finally, curiosity gets the better of you, and you retrieve the laptop.   
  
The grutor’s laptop appears to be horribly organized. There are multiple folders on the desktop labeled “desktop stuff,” “desktop 1,” and “desktop 2016,” in addition to some folders with names like “screenshots,” “more screenshots,” and “dsjzkfadsf.” By itself in the corner of the screen is a folder called “recipes.” You figure this must be the pie recipe folder. You open it, but the folders here are arranged even more strangely -- all subfolders are named (quite unhelpfully) with numbers, and there doesn’t appear to be any pattern to the way recipes are distributed -- some meat pie recipes are in the same folder as apple pie recipes, and some folders actually contain no recipes at all. You frown -- but being an enterprising student of *Computer Science for Insight*, you imagine you can come up with some quick scripts to analyze the recipes: separating them manually is surely unfeasible, as they appear to number in the thousands. Most importantly, you are \_pretty\_ sure your date is a vegetarian -- so you figure a good place to start parsing the files would be checking which ones don’t contain meats, and moving them to a separate folder. You crack your knuckles, and get to work…

So, a grutor with the youthful glint of George Clooney in their eyes left you their computer. It’s an absolute mess and they still haven’t shown up for lab. You have a vague sense that something is wrong… but it’s time to get to work. For this scavenger hunt, answer the following questions about their hard drive - but don’t worry if you don’t get all the way through - if you came to lab you’ll still get full credit for this problem.

* How many directories are there?
* How many levels down is the deepest directory?
* How many files are in Downloads?
* How many files are there in the whole tree?
* Which directory contains the most files?
* What types of files exist?
* How many html files are there?
* How many of each type of file are there? -> store in a dictionary
* How many filenames contain the string “dog”?
* How many filenames contain the string “recipe”?
* How many files contain the string “dog”?
* How many files have numbers in them?
* What TYPES of files contain the word dog?

We have provided the small folder [test](https://github.com/ScriptingBeyondCS/CS-35/tree/master/week_0/inclass) as a test case. You can run your functions on this test folder and check their accuracy by manually navigating the folder.

[**Lab problem**] Because this is a lab problem, you are welcome to come to the Tuesday or Wed. evening labs (same time and place as cs5 labs: 6-8pm in Big Beckman, the Hot Air Lab, and the Birch Lab: HMC's BkB126, 105, and 102). We will have students and instructors available. If you are able to come, sign in, and make progress on the lab for at least 90 min, you will receive full credit for this problem, even if you don't complete all of it. Alternatively, you can work on the problem on your own time (though without that same incentive). The lab-time incentive is important because it motivates both starting early and seeking out help!

**Problem 2: Phone-forensics (*Phorensics*?!)** (**hw0pr2.py** - 25 pts)

***10,000 files? No problem at all!*** For Problem 2 this week your challenge is to read and analyze a large phonebook of files (almost 10,000). The files are available in the phonebook directory of the grutor’s hard drive. Your task is to compose several functions -- perhaps using and extending the functions from problem 1 -- in order to answer several questions about the phone-numbers+names in this set of files.

First, look through the folder, the subfolders, and a few of the files inside them to get a sense of their format. Each file has a phone number (in a variety of formats) on its first line and a name (in a couple of formats) in its second line. Your task is, first, to write functions that find the answer these questions:

* How many .txt files are in the entire folder?
* What is the maximum depth of directories in the entire folder (in other words, how many times can you dig deeper into a new subdirectory)?
* What is the path to the deepest directory?
* Across all of the files, how many of the phone numbers contain exactly 10 digits?   
  Of these exactly-ten-digit phone numbers, how many are in the area code 909 (the area code will be the first three digits of a ten-digit number).
* How many people have the last name Davis?   
  *Hint: Use the comma to determine whether the last name comes first or second. Caution: If you use str.endswith(), consider that each line ends with an ‘Enter’ (‘\n’)*
* How many people have the first name Davis?
* Are there any phone numbers that have more than 10 digits? Interesting...

Be sure to include the answers to these questions in a triple-quoted string at the bottom of your hw0pr2.py file. Also, so that we have a standard method for running each team's code, be sure that the overall analysis is executed by a function named **main()**.

Next, create three *more* questions -- of your own design -- that you can ask about the data in these files. For full credit, your questions should change not only the data, but they should expand the *kinds* of questions that can be answered. (They shouldn't ask only about different data items from those above.) Here are a few examples in this go-beyond spirit -- feel free to use at most *one* of these, but the other two should be your own algorithmic creations:

* Area codes beginning with "2" are in the northeast; those beginning with "9" are in the southwest. Are there more NE or SW phone numbers across the whole dataset (meaning area codes beginning with "2" or "9" respectively)?
* How many *different* last names (or first names) are present across the entire dataset?
* How many of the phone numbers contain the substring "42" somewhere within them? Or, how many have digits *that add up to 42* ?

When you submit your file, be sure to include a triple-quoted string that (a) lists the three questions you wanted to investigate, (b) what the answers were to your questions, and (c) how to run your code to replicate those answers.

[**Extra credit**] Especially creative and unusual custom-questions (with answers) are welcome and can earn extra credit of up to +5 pts. A full extra-credit problem involving *writing-out* a file with all of the phone numbers and names is available in problem 5, below. (Note that these are *not* real names and numbers!)

**Problem 3: A recipe for disaster** (**hw0pr3.py** - 30 pts)

Problem 3 this week is another file-analysis problem, but this one focuses on organizing and comparing files in contrast to problem 2, which summarizes data across many files. Complete this problem by composing functions in a new file (hw0pr3.py). And, as always, you're welcome to copy functions you wrote elsewhere into this file!

The files to compare are a series of experimental pie recipes. They can be found on the Hard Drive in the recipes directory. First, your task is to write functions to organize the files into a directory structure that looks like this:



Your code should search through the recipe files for keywords that will help you sort the files into folders according to the structure above. Look at a few of the recipes to get a sense of their structure and which words will be useful. Hint: there are three types of meat: pork, chicken, and beef. You should also delete all extra directories when you are done.

You’re welcome to approach this problem any way you want! Here is one strategy:

* Write 3 functions: is\_savory, is\_sweet, and is\_vegetarian that take a recipe text file and determine if it satisfies the conditions for savory, sweet, or vegetarian
* Use those functions to write a function that gets a list of all savory, sweet, and vegetarian pies.
* Use the listing function to move all savory pie recipes into a directory called savory\_recipes, and all sweet pie recipes into a directory called sweet\_recipes etc.
* Write a function to remove all empty directories

Also answer:

* Across all recipes, which recipe calls for the most kilograms of one ingredient?
* What is that ingredient and how much does the recipe call for?

Hmm… an interesting recipe...

Be sure your answers are in a triple-quoted string in your file - and let us know what function we can run to find them (**main()** is never a bad choice).

As with problem 2, you should ask -- and answer -- three more questions that stretch the analysis beyond the prior questions. Further organization of files also counts as a question! You're welcome to choose other ingredients/amounts to analyze for one of the questions (like the most tablespoons etc) -- but at least two should do something a bit different, e.g. which recipes are the longest/shortest, which cook in the hottest oven, sort all the files that cook in under 40 minutes into their own directory, or something else entirely: serious or whimsical are both welcome. Especially creative questions/answers will certainly be called out for extra attention/points… !

When you submit your file, be sure to include a triple-quoted string that (a) lists the three questions you chose, (b) what the answers were (!) to your questions, and (c) how to run your code to replicate those answers!

**Problem 4: Creating a GitHub account and repository** (**hw0pr4.txt** - 20 pts)

One of the goals of cs35 is to create a portfolio that will show off your final project (and, if you like, some of the intermediate projects along the way to the final project). One of the most common ways to share software projects and portfolios uses two tools. First is *Git*, software that manages projects among teams and across sites. (This is "source-control.") Second is *GitHub*, an online interface for sharing and managing software through Git.

For this final problem this week, you'll create a GitHub account, work through the brief introductory tutorial (which includes a good amount of source-control and GitHub vocabulary), and then edit the front page (README) of your new GitHib repository:

* First, if you don't yet have a GitHub account, join at <https://github.com/join>
* Choose the free plan (with public repositories)
* Then, work through the tutorial: <https://guides.github.com/activities/hello-world/> which will introduce the basics of handling different versions of files, branching versions, editing and then creating a pull request, and then merging them together.
* Finally, make at least a few edits to your README.md file. This is in a language named *markdown* - look up how to include an image and create a list and include one of each, either on your master branch or another accessible branch. As a simple example,
  + Here's my [main branch](https://github.com/zdodds-cs35/hello-world/tree/master) for cs35 - feel free to keep it simple!
  + And [another branch](https://github.com/zdodds-cs35/hello-world/tree/readme-edits) with a couple of additional items

That's all that's needed for this week's GitHub set-up. So that we know where your repository is located, submit its URL in a text file named hw0pr4.txt inside your hw0.zip archive.

**Problem 5: Extra Credit: *Writing files, too!*** (**hw0pr5.zip** - up to + 10 pts)

Reading and analyzing large sets of files is only part of the fun! Often, a project requires a set of files to be *transformed* into another format (to match downstream processing or visualization). Here, you'll create a spreadsheet-compatible file (a csv, or *comma-separated-value*, file) that contains all of the phone numbers and names from problem 2's data set.

For this problem, look up how to use Python's print statement in order to write out to files. Experiment on small examples so that you feel confident you know how it works! Then, create a function (or functions) in a file named hw0pr5.py that crawls over all of the phone-number-and-name files, reads and handles the data appropriately, and writes out a single file named **numbers.csv** . This phone.csv file should include one line for each of the .txt files in the original data set in the following format:

**Lastname, Firstname, phonenumber (with digits only)**

For example, the file 00.txt inside the folder named 00 would produce this line:

**LEWIS, NANCY, 3754601**

When you've saved the file (again, use Google for this!) as numbers.csv, open it up in Excel to show that you've now created a spreadsheet with the relevant data from nearly 10,000 files… . Submit your hw0pr5.py file (include a triple-quoted string about how to run your program), and include the phone.csv file you created as well!

**Problem 6: Extra Credit: The benevolent grutor’s *super secret* research files…**

Fearing for the benevolent grutor’s life, you decide to go talk to their roommate, and see if they’d noticed the grutor acting funny over the past few weeks. You head on over to the grutor’s dorm, and raise your hand to knock on their door---apparently it was left ajar, and it swings open with a squeak after the first rap…

Inside, you see the grutor’s roommate, holding their E4 hammer high overhead. They appear to have been about to smash the benevolent grutor’s desktop computer! Now, you’re *really* wondering what’s going on here. They turn to look at the source of the noise---it’s you. The roommate stares at you. You stare at the roommate. The roommate wonders what you’re doing here. YOU wonder what you’re doing here. You clear your throat awkwardly, wondering why you had ever thought this would be a good idea. This continues for about six minutes. Neither of you move. You wonder whether the roommate hopes that you can’t see them if they don’t move. Then, you start to wonder if the roommate can’t see *you* if you don’t move. Maybe you should say something… Eventually, you work up the courage tomumble, “Uh… Is this a bad time? I can come back later… It’s about your roommate.” The roommate looks startled, and takes a second to regain composure. They sit down at their desk, lovingly placing the hammer back in its toolbox, and sigh “I wasn’t going to do it, of course.” You stammer, “uh, what? I mean...uh, wha---do what? What were you going to do?” The roommate stares mistily off into the distance, and continues, “my roommate has been sleeptalking for the past few weeks. Lots of...weird stuff. I don’t remember any of it though, because I was asleep when I heard it.” You nod compulsively, but then pause to think about what the grutor’s roommate just said, and become extremely confused. The roommate continues, “my roommate came running in the door yesterday, and said that if I valued my own safety I should smash the computer. I wasn’t going to do it selfishly, of course, but I imagined it would help buy more time, or something of the sort. I’m not a bad person. I have principles. I was just doing what I thought was best for them, you know. I would never do something just for personal gain.” You frown, wondering why the roommate feels the need to tell you all of this. All you wanted to know was where the benevolent grutor was. The roommate continues, “but sometimes, the line between selfishness and selflessness is blurred. Trying your best to be ‘good’ becomes an absurd quest to conquer an impalpable greyness, mired in an atmosphere of tepid skepticism, and rage against the dying of the light. But I carry on nonetheless. I refuse to submit to the Man. I’ll reject cheap consolations, and use my full, lucid consciousness to ceaselessly push the summit of my mountain closer to the liminal cusp twixt light and day. For the unexamined life is not worth living.”

Slowly, you begin backing towards the door. *Maybe if I can just leave quietly, they won’t notice*, you think*.* The roommate has begun a long rant about *Le Mythe de Sisyphe*, and you really don’t want to be there anymore. You’re almost to the door, when suddenly (perhaps noticing you’re about to leave), the roomate says, “But if you’re looking for my roommate, this computer would probably be your best bet---I’m very worried, of course, but I have too many psets due tomorrow to concern myself with any of this. You know how it is. Here, the password should be password1234. Godspeed.” Before you can say anything, the roommate dumps the computer in your hands, and shoves you out the door, closing it behind you. You’re somewhat in shock---you have so many questions! You try knocking on the door to ask, but there’s no answer. Sighing, you return to your room. Thankfully, you accidentally acquired a monitor for 5$ during dorm auction earlier in the year, so you set up the grutor’s desktop with fairly little effort. Sure enough, the password works, and you soon find yourself in a familiar mess of folders and old files… At first, everything looks identical. But there’s something strange about the recipe directory in this one---as you’re clicking through the directories, you notice you’ve eventually gotten back to where you started. It seems that this file structure isn’t a tree---strange… But you need to see if this directory can offer you any extra clues about the grutor’s whereabouts---their life might be in your hands.

**Good luck with homework #0, everyone!**