From: **Jonathan Soma** <[js4571@columbia.edu](mailto:js4571@columbia.edu" \t "_blank)>  
Date: 26 May 2017 at 15:30  
Subject: Foundations Homework 3 (next Wednesday)  
To: Jonathan Soma <[js4571@columbia.edu](mailto:js4571@columbia.edu" \t "_blank)>  
Cc: Jager William Hartman <[jwh2163@columbia.edu](mailto:jwh2163@columbia.edu" \t "_blank)>, Georgia Kromrei <[kromreig@gmail.com](mailto:kromreig@gmail.com" \t "_blank)>

Here's another homework for you to refine your skills with! I don't want you to forget everything we've done before next Wednesday. And don't worry if you find it to be magic or difficult, trying to get lists and dictionaries to do what you want is practically 60% of programming, so we're staying on this for a while.

It's the same topics - lists and dictionaries - but instead of boring fake tasks, these are more focused on common techniques and patterns you'll see in the real world.

The first two parts are definitely easier than the Spotify part of Homework #2. The third part might be easier, too. As always, just finish what you can.

**HOMEWORK 3**

Please create a new file named homework-3-lastname.py

The first line should be a comment with your full name

The second line should be the date

The third line should be a comment "Homework 3"

Submit via email before class on Wednesday

**PART ONE: Lists**

There are a lot of common ways to deal with lists of things, whether you're counting, totaling, or finding maxes or mins. Sometimes when you do those things you're only interested in certain subcategories - you want to total up the value of only red cars, or finding the fastest car built before 1965. The problems below will help you get there.

I have part of a writeup at <http://jonathansoma.com/lede/foundations-2017/classes/data%20structures/looping-patterns/>, but I don't want to make these too easy so I only have one example there. It should be helpful, though!

Most of the problems below have easy answers using Python functions like len(), max(), sum(), etc. The fact that I'm making you do them with for loops, then, seems completely absurd.... which it is! Once you have lists of dictionaries, though, everything shifts and the patterns below become very very very useful. It's nice to start with simpler problems sometimes!

**numbers = [4, 5, 1, 10, 200, 34, 22, 19, 43, 56, 32, 11, 40, 82, 23, 43, 12, 65, 10]**

1) Count how many numbers are in the list. Use a for loop, do NOT use len.

2) Add another number to the list. You can pick the number!

3) Count how many even numbers are in the list. Use a for loop.

4) Count how many values are above the mean and how many are below the mean. Use a for loop.

5) Total up the numbers. Use a for loop, do NOT use sum().

6) Total up the numbers that are above the mean and the numbers below the mean. Use a for loop, do not use sum().

7) Find the largest number. Use a for loop.

8) You have a list of dogs, shown below. BUT YOU GOT ANOTHER DOG!!! His name is Maxwell, please add him to the list (and no, you don't just add him to the end of that line). Use a for loop.

**dogs = ["Sparky", "Jane", "Matilda", "Blartsburg"]**

9) Make a list of all dogs that have names of 5 characters or less. Use a for loop.

10) I'm on a web page with some links about Zurich, and the URL looks like this: [**http://important-swiss-things.ch/docs/download/ZH**](http://important-swiss-things.ch/docs/download/ZH)

I want to get this link for every canton in Switzerland, not just Zurich, but I don't want to type the links manually. If I give you a list of the abbreviations, can you write out all of the URLs?

**cantons = [ "ZH", "BE", "LU", "UR", "SZ", "OW", "NW", "GL", "ZG", "FR", "SO", "BS", "BL", "AR", "AI", "SG", "GR", "AG", "TG", "TI", "VD", "VS", "NE", "GE", "JU"]**

11) I'm trying to get some top-secret documents from [top-secret-secrets.com](http://top-secret-secrets.com), and while I know the URL pattern I don't want to type them all out individually!

Each secret document has a document ID and is made up of 12 pages, pages "001.pdf" through "012.pdf". Each page is available at a different URL. For example, for the document ID of QQ7LTHM, the pages are available at

[www.top-secret-pdfs.com/content/secrets/QQ7LTHM/page/001.pdf](http://www.top-secret-pdfs.com/content/secrets/QQ7LTHM/page/001.pdf)

[www.top-secret-pdfs.com/content/secrets/QQ7LTHM/page/002.pdf](http://www.top-secret-pdfs.com/content/secrets/QQ7LTHM/page/002.pdf)

[www.top-secret-pdfs.com/content/secrets/QQ7LTHM/page/003.pdf](http://www.top-secret-pdfs.com/content/secrets/QQ7LTHM/page/003.pdf" \t "_blank)

...

[www.top-secret-pdfs.com/content/secrets/QQ7LTHM/page/012.pdf](http://www.top-secret-pdfs.com/content/secrets/QQ7LTHM/page/012.pdf" \t "_blank)

I have the following document IDs:

**qq7lthm**

**jemsqhg**

**O6itcke**

**cp4Iua0**

**bkijcmo**

**ctoyjrm**

**z8wc6xy**

**zk4Bmm0**

Can you print out all of the URLs? Note that the document IDs need to be capitalized in the URL!

**TIPS**

Every looping question is built up of THREE parts

1. The Initial Condition

2. The Condition

3. The Update

What does that mean?! Learn more details one the site at <http://jonathansoma.com/lede/foundations-2017/classes/data%20structures/looping-patterns/>

Instead of always adding one to a number, maybe you should think about adding to a running total, or putting another element into a list.

When getting the mean and median, DON'T use numpy, use the statistics package.

If you'd like to get fancy for the last one, you can look up formatting with "leading zeroes"

Those document IDs are lowercase, but you need them to be capitalized!

**PART TWO: Dictionaries**

Dictionaries are super useful and you use them all of the time, but I wouldn't say there are very many 'tricks' like there are with lists. Most of the time you just want to put things in and take things out using keys.

1) Let's say we are terrible doctors and we have a patient. Define a dictionary called patient that works with the following code.

**print("Doctor, it looks like the patient's is complaining about", patient['complaint'])**

2) We aren't really listening to their complaint, but as more test results come in, we'll add these things to the patient's record. Add the following to the patient dictionary:

Heart rate: 70

Temperature: 98.6

Infection: No

3) Now let's be doctors! First, if they have a heart rate about 100, we should tell them to relax. Store your diagnosis in a key called 'diagnosis'.

\* If their temperature is above 102 but they do not have an infection, they have heat stroke.

\* If they have a high temperature and they have an infection, they have the flu.

\* If they have an infection but no high temperature, it's probably a cold.

\* If none of the above, tell them to take an aspirin and call you in the morning.

When you are finished, print "Your diagnosis is \_\_\_\_\_\_\_."

4) Make a list of 3 different patients, each with different complaint, heart rate, temperature, and whether they have an infection or not. Use a for loop to diagnose each of them.

5) Because you're such a bad doctor, they've put you in the back. You don't get to talk to patients any more, you just get to diagnose them based on their temperatures. And these ones aren't even sick! They just might have heat stroke.

Using the following list and a for loop, create some new patient records (a list of dictionaries). Each dictionary should include a tempearture and whether the patient has heat stroke or not. When completed, my "for patient in patients..." code should be able to run.

**temperatures = [ 99, 105, 98, 99, 100, 104, 105, 100 ]**

**# YOUR CODE GOES HERE**

**for record in records:**

**if record['diagnosis'] == 'heat stroke':**

**print("This patient has heat stroke!")**

**else:**

**print("This patient does not have heat stroke")**

**PART THREE: Reading CSV files**

If we didn't know a better method, we might use csv.DictReader to analyze CSV files in Python (a csv is like a less fancy Excel spreadsheet). In a few weeks we'll learn the "better method," but for now we'll use it to practice with lists and dictionaries.

**import csv**

**csvfile = open('countries.csv', 'r')**

**reader = csv.DictReader(csvfile)**

**data = list(reader)**

**csvfile.close()**

When you run this code, it will open up a file and converting it into a list of dictionaries. Each row becomes a dictionary, and each column becomes a key. Not sure what I mean? Print it out to get a good look!

***Note:*** *With Python 3, reading in CSV files is friendlier than it used to be, and uses something called an "ordered dictionary." It means the keys have a specific order to them, but most importantly they don't have { } around them!*

Answer the following questions based on my favorite boring dataset, **countries.csv** (attached).

1) What are the rows in our dataset?

2) How many countries do we have in our dataset?

3) How many countries in Asia? How about North America?

4) What is the total population of the world?

5) Which has a larger population, Africa or South America?

6) Calculate the total GDP of each country and print it out (right now it's per capita).

7) What is the median life expectancy of the world?

8) What is the median life expectancy of Europe?

9) Print out each country that has a below-average life expectancy.

10) Print out each country that has a below-average GDP but an above-average life expectancy.

11) Calculate the 75th percentile of GDP.

12) What percent of the world population has a life expectancy of below 50 years? Above 80 years?

**TIPS**

A quick way to find out the rows is to select the first dictionary and look at its keys.

You don't have to use a for loop for number two!

For most of these, you'll use the same approaches as you did with Part One and the lists, you just need to figure out which patterns are the right ones.

You know how to get a median if given a list of numbers. For some of these it might be helpful to use your for loop to create a new list of numbers, then calculate the median from it.

You know how to calculate the 50th percentile - it's the median of ALL of the values. The 75th percentile should be the 50th percentile of only the top 50% of the values.

--

Jonathan Soma

Director, The Lede Program

Columbia University Graduate School of Journalism

2950 Broadway, New York, NY, 10027

[212-851-1816](tel:%28212%29%20851-1816) | [js4571@columbia.edu](mailto:js4571@columbia.edu)