There are two json files that contains the required courses for each major, but they are in different format.

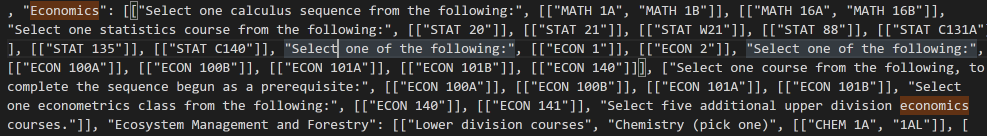
1. The original one is named as “data.json.” Basically, it contains a very long dictionary whose keys are the names of majors, such as “African American Studies” or “Computer Science,” and whose values are lists of contents of each table. The idea of table can be understood better if you could look into the major website. The following is a screenshot of Economics Major Website:



Each table is started with its heading and then followed by a list of courses or other information. In the case shown, there are two tables in all.

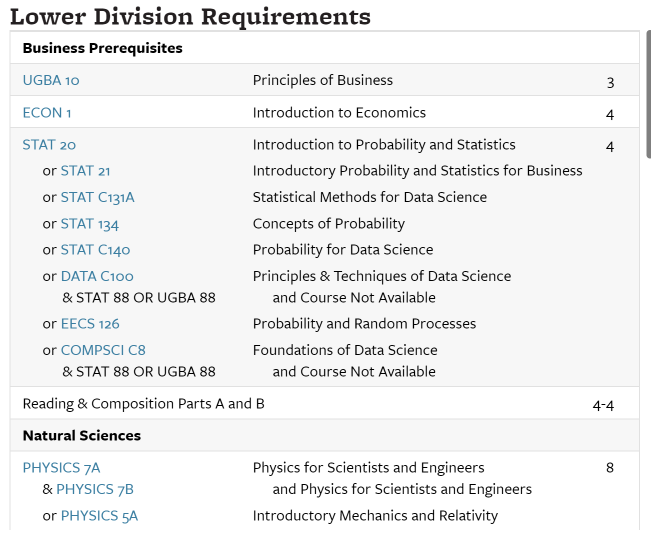
* 1. Tables

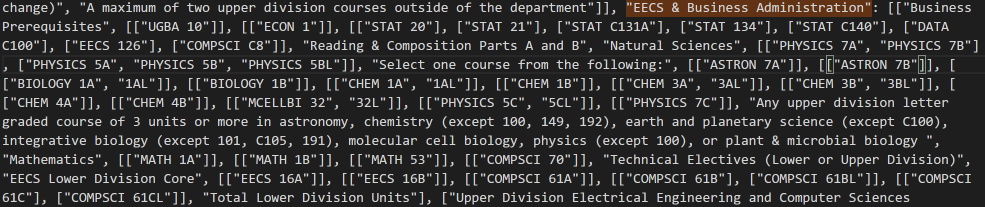
Back to values for the dictionary. Let’s take the value for “Economics” as an example.



The value is clearly nested list. Since there are two tables as discussed before, the length of the outermost list should be. In other words, the key to value format in this situation could be interpreted as “Major Name”: [Table1, Table2].

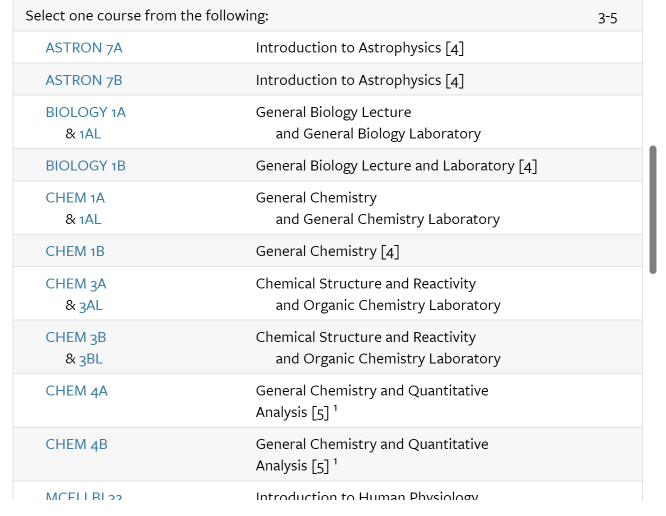
* 1. Rows

After understanding the concept of table, now we look into the contents of each table within it. Let’s take the rows of the first table for the EECS and BA major as an example. Both the website itself and the data collected are shown as below:



In each table, we define the row as the region of one color separated by another color. To be specific, Business Prerequisite is the first row, UGBA is the second one, and the third one is very complicated going from STAT 20 way to &STAT 88 OR UGBA 88. If the first thing of each row is a course type (formatted in the type of Major+CourseNumber), my algorithm appends each row as a nested list into the table list. If it is not, (for example, the first row is Business Prerequisites), we call it course comment and append it as a string into the table list. Then the simple question left is why I would do each row for courses as a nested list. The reason is that I want to save the possibilities for permutation for each row. Let’s take the fourth row starting from STAT 20 as an example. We could append many things into the table as long as it both satisfies the “or” and “and” conditions. I could append “STAT 20,” “STAT 21,” “DATA C100 & STAT88,” “DATA C100 & UGBA 88.” (Please note the blue color in the web page indicates the existence of such class, in contrast, and the black color indicates it no longer exists so I just omit it)

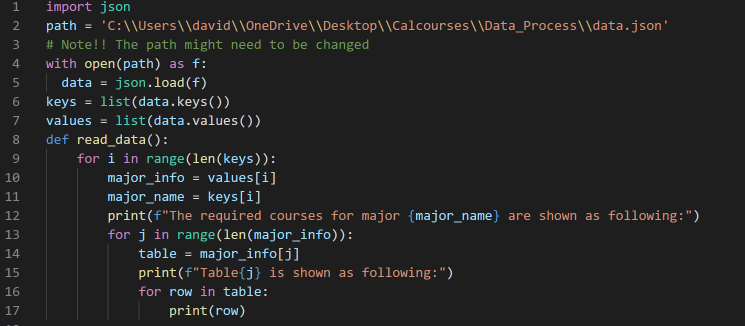
Within each table, it is formatted as table = [row1, row2, row3…]Within each row, it is formatted in the following way: [alter1, alter2, alter3…]. Each alter is a list of course that might be connected by the “and” logic.



For example, in the webpage shown ahead, alter1 = [[ATRON 7A]], alter2 = [[ASTRON 7B]], alter 3 = [[BIOLOGY 1A, BIOLOGY 1AL]].

Overall, this table can be read as [[[ASTRON 7A]], [[ASTRON 7B]], [[BIOLOGY 1A, BIOLOGY 1AL]] …]

* 1. Iterate through the data.json

If you want to iterate through this, you just need to go into the row level. It is shown as follows:

The code in Python is provided in the method called read\_data in the main.py. You can also access to this method in the branch called “David.” There is one file called “Data\_process,” where you can find all the json files and codes, I extract the data and process them, of course including the main.py you need.

1. Another one is called “table.json.” The format is used for uploading it into the database. I suggest not to use it because it’s unclean and contains many useless empty tables. If you really want to use it, you can ask me privately.
2. My WeChat ID is yinyida0609. If you have any question about data, please feel free to reach me.