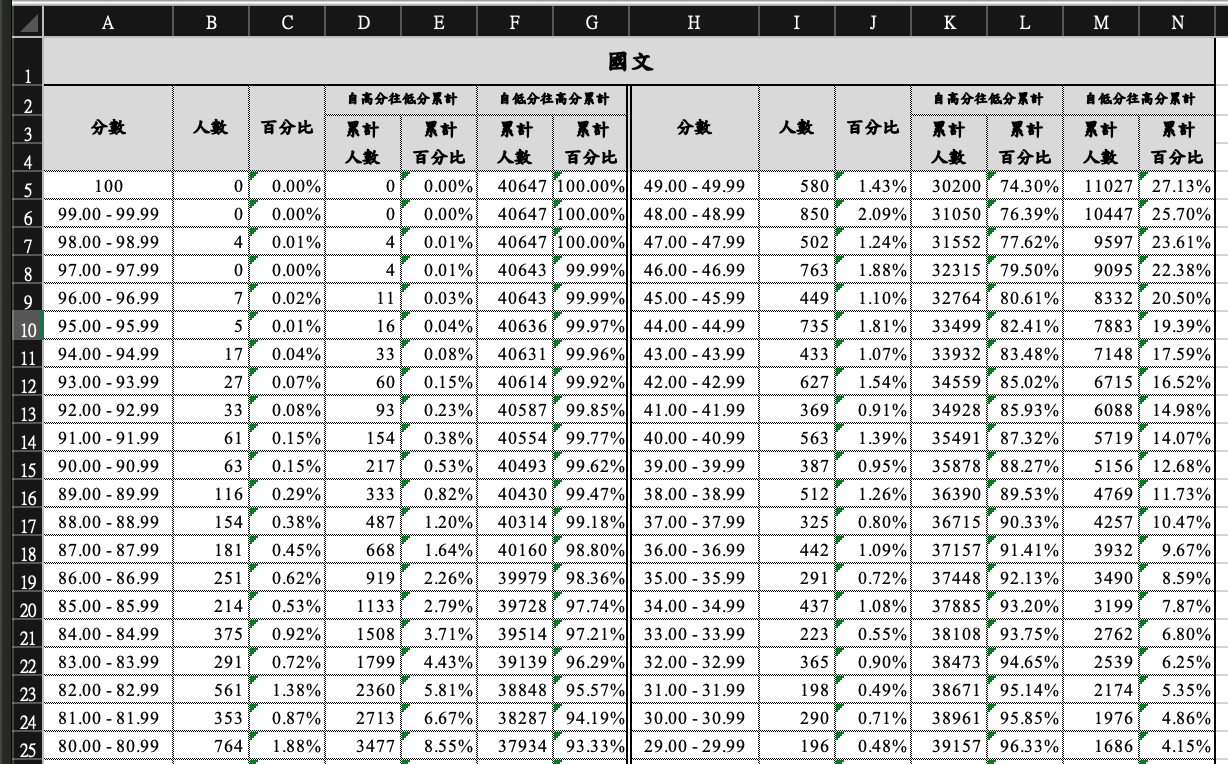
The Advanced Subject Test consists of ten subjects: Chinese, English, Math A, Math B, Chemistry, Physics, Biology, History, Geography, and Civics. Each student can choose the tests they take, and they can apply for the departments they want to enter by the scores. Note that each department requires scores of different subjects. In addition, the threshold total scores (weighted) of each department in previous years can be found online, see <https://www.com.tw/exam/group_list109.html>.

Every year, after students get their scores, they would like to know which department they are likely to be able enter. The most naïve way is to compare their scores with the previous thresholds. However, the difficulty of the test varies, and this way seems too rough. To get more precise estimations, one should “translate” their scores according to the difficulty of the previous tests, and then compare with the threshold.

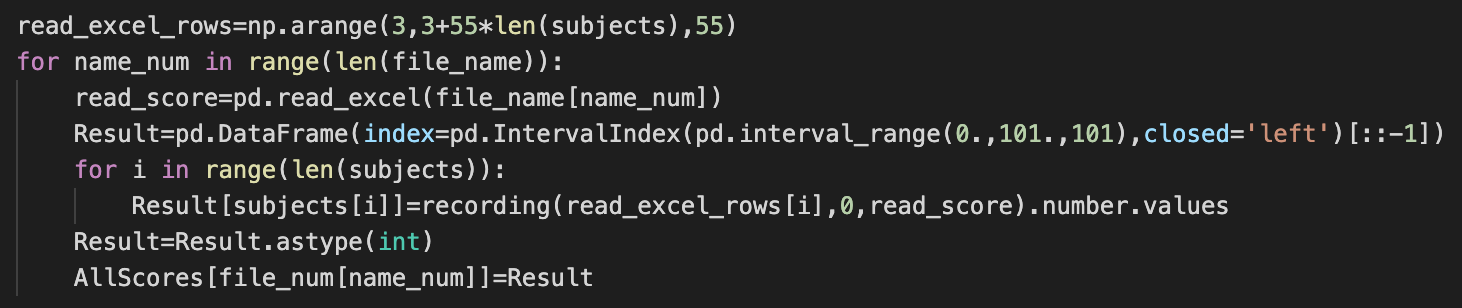
Sometimes, one gets only the quantiles of a test. For instance, in mock tests of the Advanced Subject Test, one gets only five quantiles. Or, one may happen to have the quantiles of the Advanced Subject Test this year (which is quite common since they are often shown on lecture notes or news). Given only fragmented quantiles, I want to estimate the percentage of students getting each score, from 100 to 0.

1. **Read files:**

One can download the following excel worksheets on the internet, see <https://www.ceec.edu.tw/xmdoc?xsmsid=0J018611000723433352> (The downloaded files are .xls files),



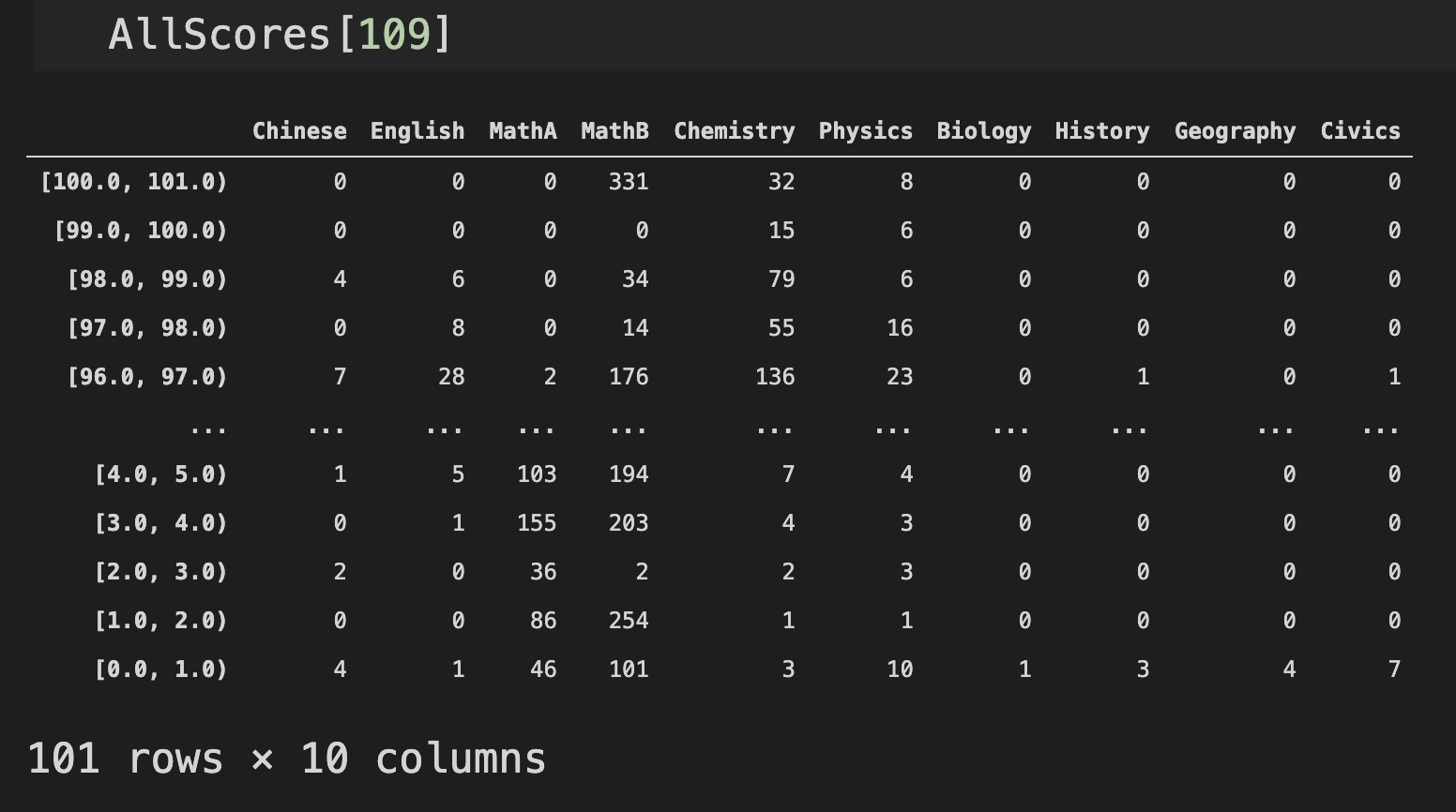
The worksheets contain the information of the number of people getting each score (the highlighted parts) in previous years. The python file starts by reading those worksheets. The technical part here is that we should carefully specify the elements we want the program to read, after all, we don’t need all information the tables serve, but the highlighted parts only. This can be done by



which carefully selects the rows and columns we need.

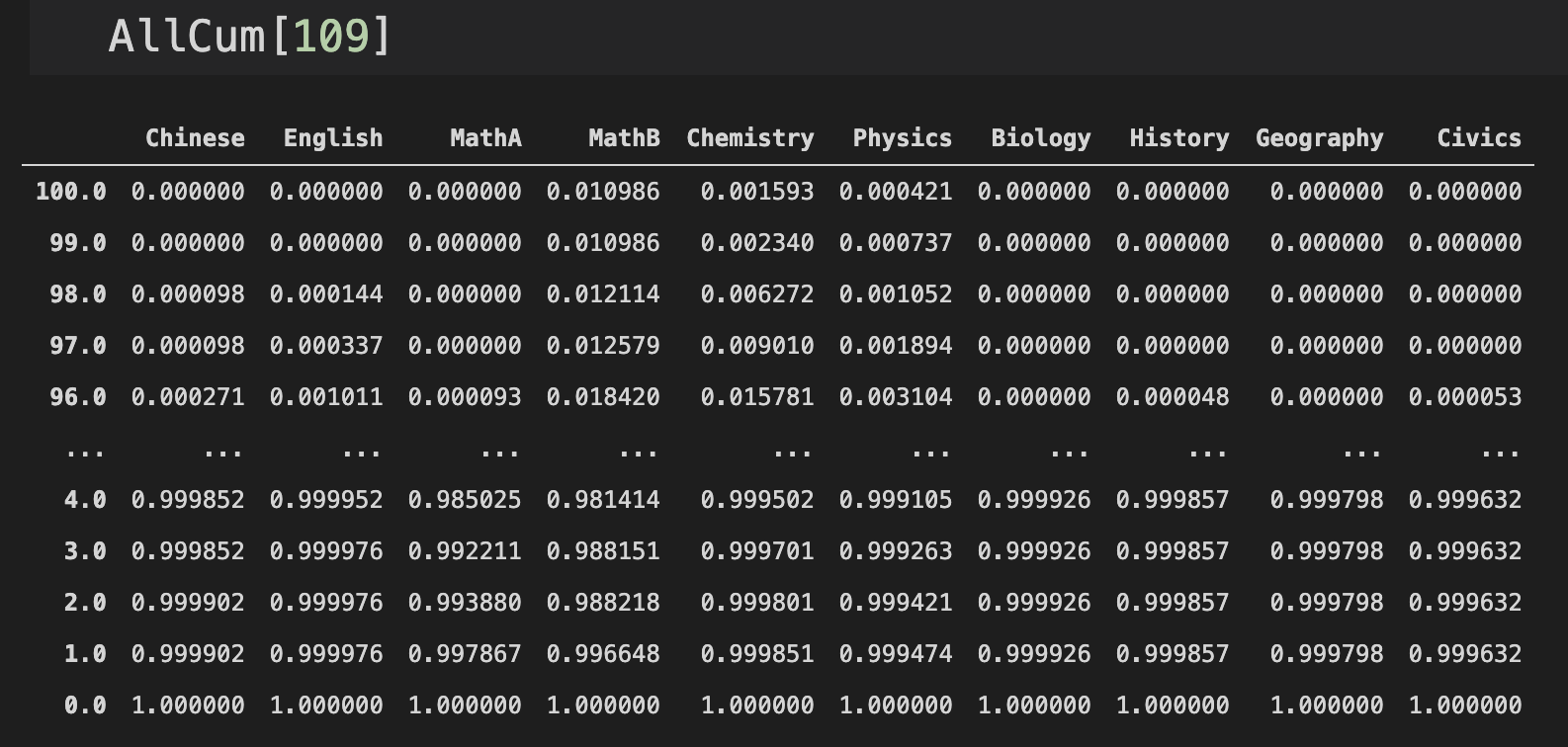
1. **Important DataFrames**

The highlighted parts are recorded in the dictionary AllScores, with keys the years in file\_num and values DataFrames in the following format

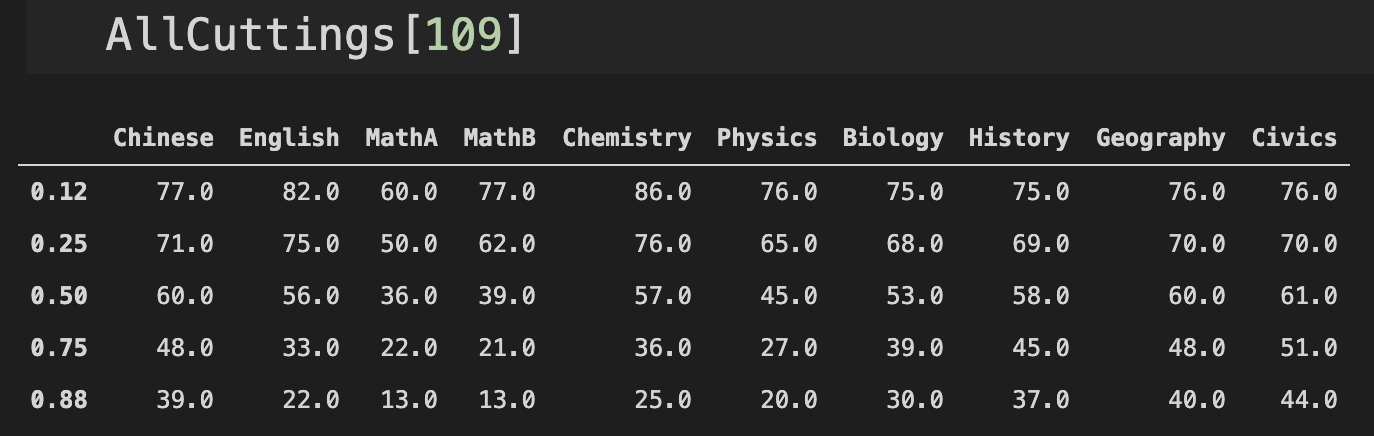


From AllScores, AllCum and AllCuttings can be deduced.

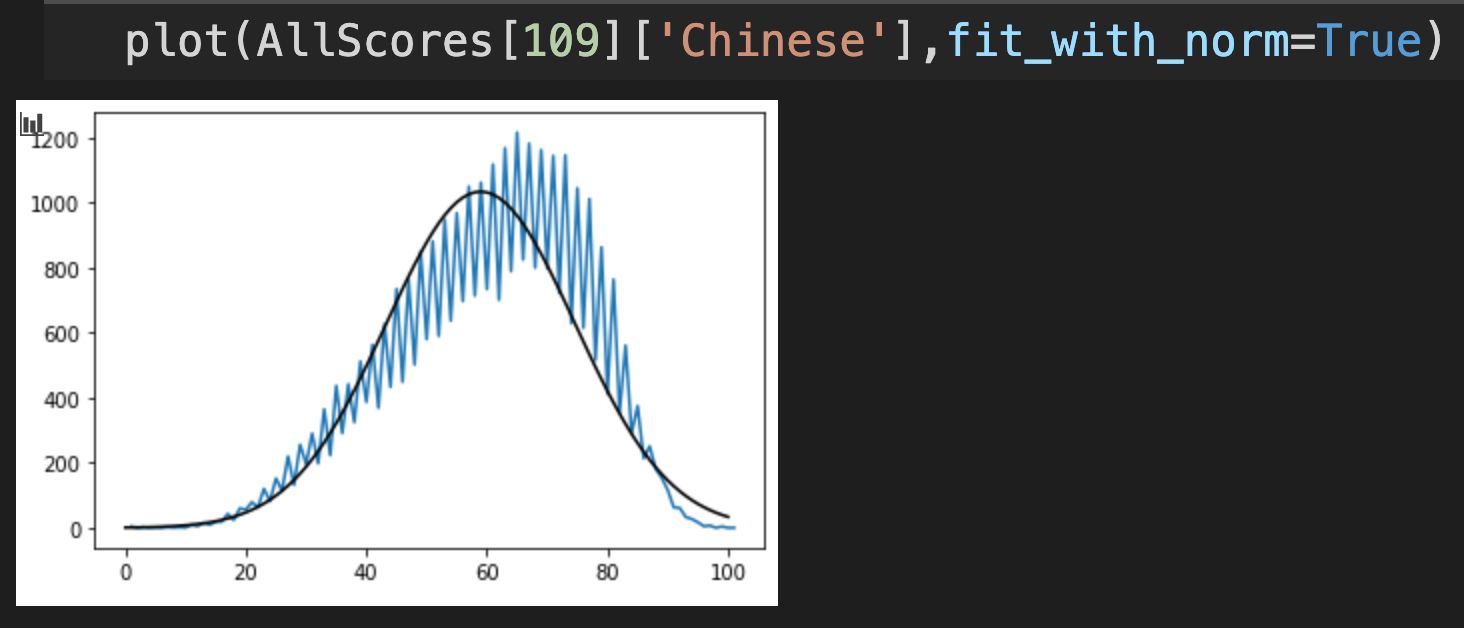
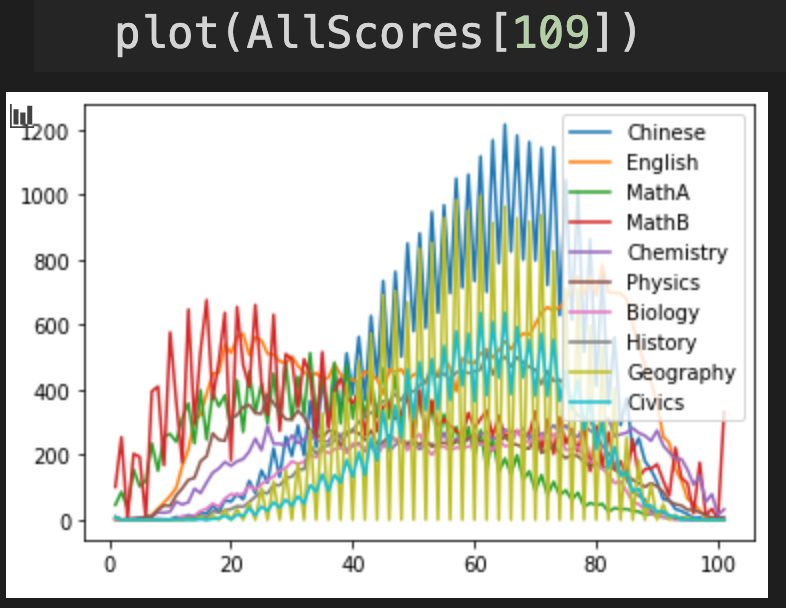
* AllCum: It is a dictionary, with keys being the years in file\_num. Each value corresponds to a DataFrame whose elements are the ratio of people scoring [the index of the element, 100).



* AllCuttings: It is also a dictionary with keys being the years in file\_num and values being DataFrames. The DataFrames have indexes [0.12,0.25,0.5,0.75,0.88], and the elements are the 7th octile, the 3rd quartile, the median, the 1st quartile, and the 1st octile, respectively.



One can use the plot function to plot the graphs of AllScores and AllCum

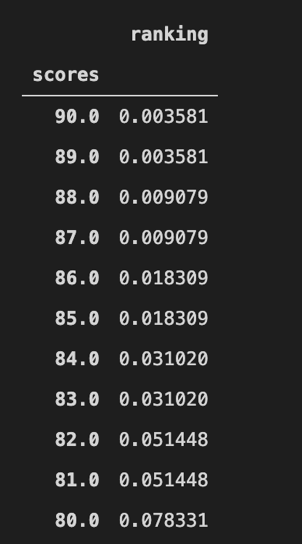
1. **Convert scores**

* The function ratio(a\_upper,a\_lower,a,b\_upper,b\_lower) is useful while doing interpolations, and it does the following
* Given the scores one gets this year (score\_now), and given the cumulation table this year (Scores\_cum\_now) (similar to the values in AllCum), we can translate the scores to the previous years by the function

ConvertCum(Scores\_cum\_now,score\_now,Scores\_cum\_past)

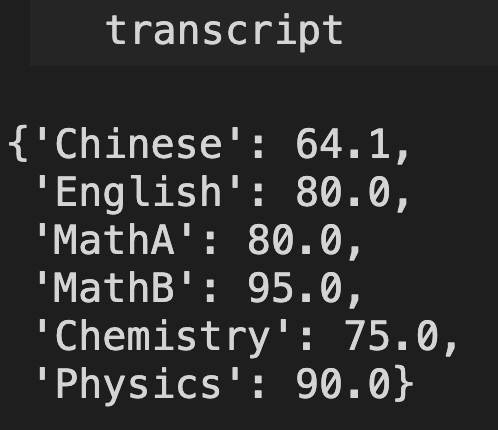
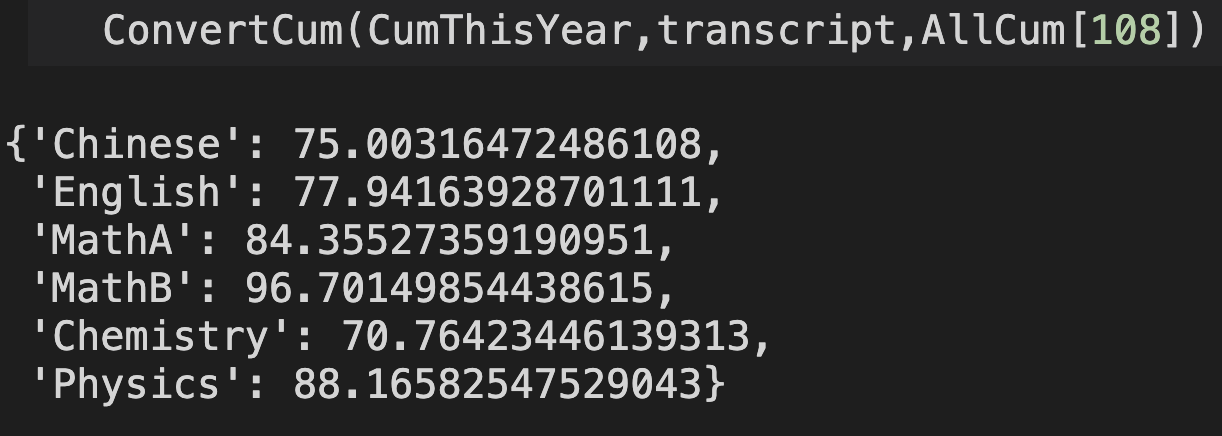
The output is a dictionary, which consists of the translated scores.

The way I translate the score starts from locating the ranking of the score\_now with respect to Scores\_cum\_now, then find the score that corresponds to the ranking in Scores\_cum\_past. Technical part: quite often, we can’t find exact score in Scores\_cum\_now, as a result, we need to interpolate the initial score into the cut that contains it, and use ratio to obtain the corresponding translated ranking. As an example, suppose the initial score is 86.4, and (part of) the cumulation table this year is



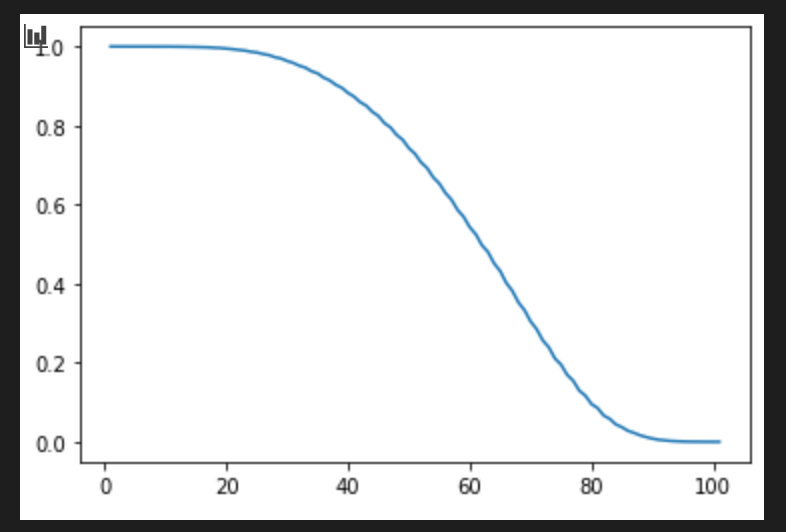
**86.4**

The translated ranking is ratio(87.0,86.0,86.4,0.009079,0.0018309). We then interpolate the ranking to Scores\_cum\_past to obtain the translated score.

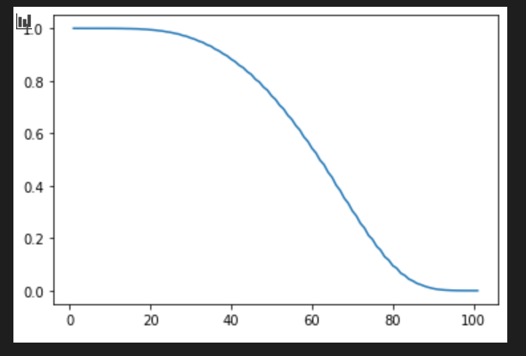
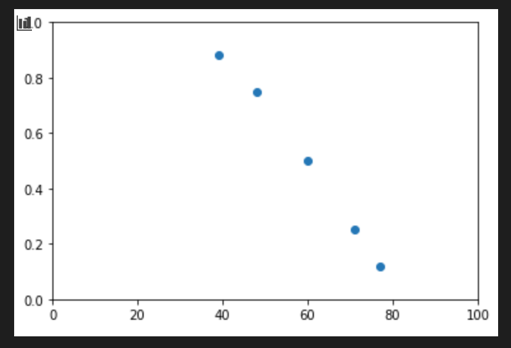
 

1. **Fit the cumulation curve**

If we plot the cumulation table, we get a graph like



However, one may not have access to the full cumulation table of the test he takes. This happens, for instance, when one takes the mock tests of the Advanced Subject Test. Rather, he gets the quantiles of the test. I want to generate the full cumulation curve given only the quantiles. Diagrammatically,



To do this, we have to have a reference cumulation table. Without it, we will have no idea how the curve might be. The reference cumulation table is given by AllCum, and there are two ways to do this.

1. ConvertCuttings\_toCum\_Interpol(Scores\_cut\_now,score\_now,Scores\_cum\_past)

Scores\_cut\_now: the quantiles one gets on his test

score\_now: the score one wants to translate

Scores\_cum\_past: the reference cumulation table

* However, one often gets the quartiles rather than the cumulation table (the official cumulation table each year is released after the application,) therefore, we want to construct the cumulation table given the quartiles. There are two ways to do this, and they both start with the matching:

untranslated score

translated score

previous score

Note that the indexes of the “untranslated score” and the “previous score” are both np.arange(0.,101.,1.), and the values of the two are cumulations.

1. Interpolate:

After matching the quantiles, the translated score’s index are no longer np.arange(0.,101.,1.), since it is “squeezed” or “stretched” while matching. However, the index must be somewhere between two consecutive integers, say “a” and “b”, and we can find those integers in previous score’s index, with corresponding values, say f(a) and f(b), which are previous score’s cumulations. As a result, we can obtain the cumulation of the translated score’s index by interpolation with respect to f(a) and f(b).

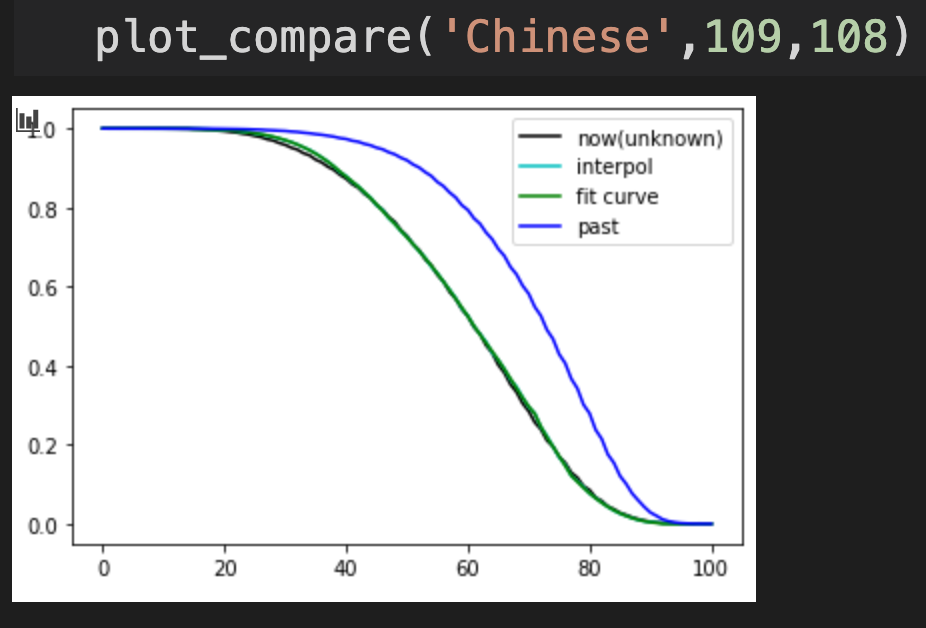
1. Fit curve:

We first find the best-fit curve to each interval of the previous score (intervals between the red dots and the endpoints). Then we calculate the relative positions of the translated score’s index with respect to the endpoints of each interval. Last, we apply the best-fit curve onto the relative positions to obtain the cumulation.

Once we get the translated cumulation, we can feed it to ConvertCum and get the translated scores. The two processes are combined in

ConvertCuttings(Scores\_cut\_now,score\_now,Scores\_cum\_past,method)

* To see how well the quartiles-to-cumulations algorithm does, we can use plot\_compare(sub,now,past). We apply the algorithm to AllCuttings[now], and fit it with respect to AllCum[past], and see how close it is to AllCum[now].



We can see that the black curve, the cyan curve, and the green curve nearly overlap with each other, indicating that our algorithm is quite accurate.

* plot\_cum(sub,Cut\_now,past) simply plots the cumulation we obtain using this algorithm, evaluated with respect to AllCum[past].

1. **Read scores**

This is the section where the scores of the users are read. The user should input their scores in transcript.xlsx in the following format

