**Goal:** Make python script that takes as input a file with sample names (one per row), and multiple files from metaSNV analysis. The script must then parse the results and out a new count matrix with sample names making up each column and the metaSNV results in the rows.

**Test data:**

* metaSNV results:
  + tab separated file names
    - called\_SNPs.best\_split\_0
    - called\_SNPs.best\_split\_1
    - called\_SNPs.best\_split\_2
  + These files are created based on the number of threads that are used to run metaSNV. In this case, we used 3 threads to run the metaSNV command.
  + Columns
    - 1-4 have to do with the unique gene variant that was identified and columns 5 and 6 have the count results for how many time each variant was found in a sample.
    - Column 5 represents the count results and uses the “|” pipe to separate the results for each sample. The order that these results are in are based on the sample name file.
* Sample name file
  + all\_samples.txt
  + This contains a sample name on each row and corresponds to the order that the results are reported in the metaSNV files. More information below.
* Example count matrix
  + This is an example of what it would like for these test data.

**Script specifications:**

* Must be able to take arguments, can be positional or with flags. 3 arguments are needed:
  + Sample name file
  + metaSNV results file
  + output file name
* Input:
  + File with 1 sample name per row (all\_samples.txt)
  + MetaSNV result files. This has to be able to take multiple files that are specified with a wildcard. Alternatively, you can take advantage of the fact that metaSNV always names it’s results “called\_SNPs.best\_split” and then adds the underscore and numerical value for each thread that is used when running the program.
* Function:
  + Parse the sample name file and maintain the same order of sample names. Could be a dictionary that you keep adding results to and the sample names are the indices.
  + Then, parse each metaSNV results file in the following way:
    - Make a unique variant ID by combining the text from columns 1-4 (0-3 in python). You can combine it using “|”.
    - Take column 5 (4 in python) and store those results such that the order of counts is identical to the sample name file.
    - Do this for each file (which could range based on how many threads we used).
  + Output a count matrix (csv) with each unique variant on the rows (skipping the first row for column names), sample names on the columns (except for the first column which has the variant names) and the counts in their corresponding place. Check out the “example\_matrix.csv” for how to label the first column with “variant\_accesssion”.