Description
A tool to compare variants contained within multiple vcf format files, to determine common and unique variants to each file. To write these variants to a new file also in vcf format.

Idea:

Take two or more vcf files, compare the snvs and then write out a venn like diagram. Also list which snvs were found in which files.

Possibility:
A mathematical algorithm to determine file comparison to allow the program to be scalable to multiple file comparisons. python module

```
>>> import itertools
>>> itertools.combinations('abcd',2)
<itertools.combinations object at 0x01348F30>
>>> list(itertools.combinations('abcd',2))
[('a', 'b'), ('a', 'c'), ('a', 'd'), ('b', 'c'), ('b', 'd'), ('c', 'd')]
>>> [''.join(x) for x in itertools.combinations('abcd',2)]
['ab', 'ac', 'ad', 'bc', 'bd', 'cd']
```

So we need a loop from n-1 files to 2 with itertools.combinations([filelist], n-1..2)
Add all of these combinations to a list of output generators..

Comparisons and snv lists will be produced for the following for a 3 file case:
file 1 uniq
file 2 uniq
file 3 uniq
file 1, 2 common not 3
file 1, 3 common not 2
file 2, 3 common not 1
common to all

Dictionary - key; value
Key - chromosome;postion;altAllele (separator “:”)
value - array: [0]: a list of files which this snv occurs
                        [1]: the vcf line in file if the same, no meta-data if different (cut down line
            instead)

Output filenames to be generated by the progam using a prefix specified by the user followed by the comparison performed. Outputed into a directory specified by the user (default venn_results/).
 e.g. prefix.File1VFile2.vcf etc; prefix.1UNIQ.vcf etc; prefix.ALL.vcf
NB - use math function to write file names.

Requirements

As where are multiple versions of the human genome, comparison should allow for both formats of chromosome naming, (e.g. chr1 or 1).

SIMON'S BIT:

Getting files and combo's:

```
def make_file_dictionary(file_list):
        '''
        subroutine to create and return a dictionary of the supplied file_list
        using an integer from 1 to len(file_list) as the keys

        >>>make_file_dictionary(['file_1','file2','file_3'])
        {1: 'file_1', 2: 'file_2', 3: 'file_3'}

        '''

def make_file_combinations(num_files):
        '''
        subroutine to make a list of file combinations (as lists) for the number of files
        supplied as the parameter. Returns a list of lists

        >>>make_file_combinations(3)
        [[1], [2], [3], [1, 2], [1, 3], [2, 3], [1, 2, 3]]
        '''
```