Multi-state design is a Rosetta protocol that allows us to design a protein using multiple structures as inputs. This is a very powerful concept, because it increases the structural constraints that we can include to define protein function.

The python script GenerateSMSDFiles.py will generate all of the files needed to run multi-state design:
1) an entity file - this declares the number of residues that will be designed by
2) a correspondence file - this maps a position in the entity to a position in a pdb file
 - In our case, all of the correspondence files are redundant, so we use just one.
3) a secondary resfile - this tells Rosetta how to
 - The default is that everything is fixed in its native rotamer, but anything within 10 Å of a design residue (a residue in mapped to a position in the entity) is allowed to be sample rotamers of its native amino acid.
4) a fitness file - this file states which pdb files are connected to which correspondence files and secondary resfiles

The GenerateMSDFiles.py script takes in several parameters. Some of these have been optimized for you, and some of these will be your "experimental knobs".

1) the list of residues in the patches
 - These have been selected for you.
2) the starting energies used to determine the offset in the fitness function
 - These values have been pre-optimized based on the default patches.
3) the weights (k_values) for each state
 - These are the values we want you to experiment with.
4) the steepness values for each state

The first set of inputs is the name of a patch and a set of residues that corresponds to that patch. These will be the design positions in your entity file. For the purpose of this class, a patch should be around 6-8 residues and no more than 10 residues. To go beyond this would require more time than we have for this class.

NOTE: The default settings already design residues on the interface for the relevant complex. If you want to change these values, talk to Samuel, because the starting energies (described below) will need to be re-optimized.

The second set of inputs are the paths to the input structures. For this class, structures that have been pre-optimized for multi-state design have been provided for you. Use the paths to the structures that are already in GenerateMSDFiles.py and not your own structures. Otherwise, the results of the simulation will be wildly inaccurate.

The third set of inputs is a start energy for each patch within each input pdb. These values have been optimized for you. This was done by running multi-state design, but allowing only one input structure. The best score for that run represents the energy at optimal fitness, at least according to what Rosetta can design. If you want to make your own patches, you will have to repeat this process for each of your patches.

—KEY EXPERIMENTAL PARAMETER—
The fourth set of inputs are the weights for each state. This determines how the fitness of each state contributes to the overall fitness function. We have provided data for the simple case when all weights are set to 1.0, and for each of the cases when only one input structure is included. You should alter the weights in a way that makes sense for your perturbation an see how that affects the tolerated sequence space.

The final set of parameters are the steepness values. You should not need to alter these, but values between 0.4 and 2.5 are typical. The use of these values is based on empirical observation of the outcome rather than any interpretable physical meaning.

Once you have set all of these parameters correctly, run the script. First, ssh into an interactive node so that python has access to BioPython, then change into the directory where the GenerateMSDFiles.py script is, and run it.

```
#log into the cluster
$
#log into an interactive not
$ssh iqint
$cd [path to your MSD folder]
$python GenerateMSDFiles.py
```

Once the GenerateMSDFiles.py script has finished running, it should give you the command lines to run.

Example command line: `$qsub MSD_MPI_patch1.sh`

You will notice that the fitness scores in the log file are negative. This is because the multi-state design protocol was developed before the fuzzy fitness and expects to minimize the fitness function (lower energy = lower fitness).

For more information on operating the MPI_MSD protcol, look at the Rosetta documentation: https://www.rosettacommons.org/manuals/archive/rosetta3.4_user_guide/db/d10/multistate_design.html

For more information on the multi-state design and fuzzy logic fitness functions, look at the following references:

A generic algorithm for multistate protein design. Leaver-Fay, R. Jacak, B. Stranges, and B. Kuhlman B. PLoS-ONE, RosettaCon2010 special collection. 2010

A 'fuzzy'-logic language for encoding multiple physical traits in biomolecules. Warszawski S, Netzer R, Tawfik DS, Fleishman SJ. J Mol Biol. 2014