from modeller import \*
from modeller.automodel import \*    # Load the AutoModel class
log.verbose()
# Override the 'special\_restraints' and 'user\_after\_single\_model' methods:
class MyModel(AutoModel):
    def special\_restraints(self, aln):
        # Constrain the A and B chains to be identical (but only restrain
        # the C-alpha atoms, to reduce the number of interatomic distances
        # that need to be calculated):
        s1 = Selection(self.chains['F']).only\_atom\_types('CA')
        s2 = Selection(self.chains['A']).only\_atom\_types('CA')
        self.restraints.symmetry.append(Symmetry(s1, s2, 1.0))
    def user\_after\_single\_model(self):
        # Report on symmetry violations greater than 1A after building
        # each model:
        self.restraints.symmetry.report(1.0)
env = Environ()
# directories for input atom files
env.io.atom\_files\_directory = ['.', '../atom\_files']

# Be sure to use 'MyModel' rather than 'AutoModel' here!
a = MyModel(env,
            alnfile  = 'two\_chain\_5ht7r\_Gs.ali' ,     # alignment filename
            knowns   = '5ht7r\_Gs\_protein',              # codes of the templates
            sequence = '5ht7r\_Gs')  # code of the target

a.starting\_model = 1       # index of the first model
a.ending\_model   = 2     # index of the last model (determines how many models to calculate)
a.make()                           # do comparative modeling