# A simple working example of the BestMSM package

The following provides a minimal example to how the BestMSM package works. We construct an MSM using data from simulations of the ala5 peptide, which is already discretized (!). We start by importing the package, which should be installed.

In [3]:
```
import bestmsm.trajectory as traj
```

The only initial input is the name of the file with strings corresponding to time-stamps and state names (or indexes or whatever you want; they'll be handled as strings). In the current example states look like the following:

In [2]:
```
1 11111
2 11111
3 11111
4 11111
5 11111
6 11111
7 11111
8 11111
9 11111
...
```

```
  File "<ipython-input-2-dd2f9c752768>", line 1
    1 11111
          ^
SyntaxError: invalid syntax
```

The strings of ones "11111" are just helix-coil states for the ala5 pentapeptide (all we see here are fully helical states). Then we generate an instance of the TimeSeries class.

In [4]:
```
traj_ala5 = traj.TimeSeries("ala5_32states_timeseries.dat")
```

The TimeSeries class has a number of atributes, like the time stamps ('time'), the corresponding time series for the states ('states'), the names of the states ('keys'), the lag between snapshots ('dt') and the filename ('filename').

In [4]:
```
traj_ala5.dt
```

Out[4]:
```
1.0
```

Having read one or multiple trajectories we now invoke the MSM class.

In [6]:
```
import bestmsm.msm as msm
msm_ala5 = msm.MSM([traj_ala5])
```

The only thing we have to start off is the set of states from the combined trajectories.

In [7]:
```
print msm_ala5.keys
```

```
['11111', '11110', '01111', '01110', '01101', '01100', '01000', '01001', '
11000', '00000', '11001', '01011', '01010', '00010', '00110', '10110', '10
111', '10100', '10101', '10001', '10000', '10010', '00111', '11101', '0010
1', '00100', '00011', '10011', '11011', '11100', '00001', '11010']
```

Then we start computing interesting things like the count matrix.

In [9]:
```
nji = msm_ala5.calc_count(lagt=10)
```

The array that we see is just the count matrix for a lag time of 10 (time units). Having the count matrix you can now calculate the transition matrix. Or we can directly calculate the transition matrix.

In []:
```
trans = msm_ala5.calc_trans(lagt=10)
```

In the absence of a count matrix the library will just use the information of the trajectoy and the lag time.