Start the python interpreter and load the module

```
import pynmrstar
```

Load an entry directly from the BMRB database

```
>>> ent15000 = pynmrstar.Entry.from_database(15000)
```

View the heirarchy of saveframes and loops as a tree

```
>>> ent15000.print_tree()
<pynmrstar.Entry '15000' from_database(15000)>
    [0] <pynmrstar.Saveframe 'entry_information'>
        [0] <pynmrstar.Loop '_Entry_author'>
        [1] <pynmrstar.Loop '_SG_project'>
        [2] <pynmrstar.Loop '_Struct_keywords'>
        [3] <pynmrstar.Loop '_Data_set'>
        [4] <pynmrstar.Loop '_Datum'>
        [5] <pynmrstar.Loop '_Release'>
        [6] <pynmrstar.Loop '_Related_entries'>
    [1] <pynmrstar.Saveframe 'citation_1'>
        [0] <pynmrstar.Loop '_Citation_author'>
    [2] <pynmrstar.Saveframe 'assembly'>
        [0] <pynmrstar.Loop '_Entity_assembly'>
    [3] <pynmrstar.Saveframe 'F5-Phe-cVHP'>
```

```
        [0] <pynmrstar.Loop '_Entity_db_link'>
        [1] <pynmrstar.Loop '_Entity_comp_index'>
        [2] <pynmrstar.Loop '_Entity_poly_seq'>
    [4] <pynmrstar.Saveframe 'natural_source'>
        [0] <pynmrstar.Loop '_Entity_natural_src'>
    [5] <pynmrstar.Saveframe 'experimental_source'>
        [0] <pynmrstar.Loop '_Entity_experimental_src'>
    [6] <pynmrstar.Saveframe 'chem_comp_PHF'>
        [0] <pynmrstar.Loop '_Chem_comp_descriptor'>
        [1] <pynmrstar.Loop '_Chem_comp_atom'>
        [2] <pynmrstar.Loop '_Chem_comp_bond'>
    [7] <pynmrstar.Saveframe 'unlabeled_sample'>
        [0] <pynmrstar.Loop '_Sample_component'>
    [8] <pynmrstar.Saveframe 'selectively_labeled_sample'>
        [0] <pynmrstar.Loop '_Sample_component'>
    [9] <pynmrstar.Saveframe 'sample_conditions'>
        [0] <pynmrstar.Loop '_Sample_condition_variable'>
    [10] <pynmrstar.Saveframe 'NMRPipe'>
        [0] <pynmrstar.Loop '_Vendor'>
        [1] <pynmrstar.Loop '_Task'>
    [11] <pynmrstar.Saveframe 'PIPP'>
        [0] <pynmrstar.Loop '_Vendor'>
        [1] <pynmrstar.Loop '_Task'>
    [12] <pynmrstar.Saveframe 'SPARKY'>
        [0] <pynmrstar.Loop '_Vendor'>
        [1] <pynmrstar.Loop '_Task'>
    [13] <pynmrstar.Saveframe 'CYANA'>
        [0] <pynmrstar.Loop '_Vendor'>
        [1] <pynmrstar.Loop '_Task'>
    [14] <pynmrstar.Saveframe 'X-PLOR_NIH'>
        [0] <pynmrstar.Loop '_Vendor'>
        [1] <pynmrstar.Loop '_Task'>
    [15] <pynmrstar.Saveframe 'spectrometer_1'>
    [16] <pynmrstar.Saveframe 'spectrometer_2'>
    [17] <pynmrstar.Saveframe 'spectrometer_3'>
    [18] <pynmrstar.Saveframe 'spectrometer_4'>
    [19] <pynmrstar.Saveframe 'spectrometer_5'>
    [20] <pynmrstar.Saveframe 'spectrometer_6'>
    [21] <pynmrstar.Saveframe 'NMR_spectrometer_list'>
        [0] <pynmrstar.Loop '_NMR_spectrometer_view'>
    [22] <pynmrstar.Saveframe 'experiment_list'>
        [0] <pynmrstar.Loop '_Experiment'>
    [23] <pynmrstar.Saveframe 'chemical_shift_reference_1'>
        [0] <pynmrstar.Loop '_Chem_shift_ref'>
    [24] <pynmrstar.Saveframe 'assigned_chem_shift_list_1'>
        [0] <pynmrstar.Loop '_Chem_shift_experiment'>
        [1] <pynmrstar.Loop '_Atom_chem_shift'>
```

There is a shorthand way to access saveframes by name (look at the tree for the saveframe names):

```
>>> ent15000['entry_information']
<pynmrstar.Saveframe 'entry_information'>
```

And a shorthand way to access loops by category (only one loop of a given category can exist in a saveframe):

```
>>> ent15000['entry_information']['_Entry_author']
<pynmrstar.Loop '_Entry_author'>
```

Next, we'll load the same entry again from the database, remove a saveframe, and compare it to the original.

```
>>> ent15000_2 = pynmrstar.Entry.from_database(15000)
>>> del ent15000_2['entry_information']
>>> pynmrstar.diff(ent15000, ent15000_2)
The number of saveframes in the entries are not equal: '25' v
s '24'.
No saveframe with name 'entry_information' in other entry.
```

Let's look at a loop's tags and its data:

```
>>> ent15000['entry_information']['_Entry_author'].get_tag_na
mes()
[u'_Entry_author.Ordinal',
 u'_Entry_author.Given_name',
 u'_Entry_author.Family_name',
 u'_Entry_author.First_initial',
 u'_Entry_author.Middle_initials',
 u'_Entry_author.Family_title',
 u'_Entry_author.Entry_ID']
```

Get the first names of the authors using direct saveframe name and loop reference:

```
>>> ent15000['entry_information']['_Entry_author'].get_data_b
y_tag('Given_name')
[[u'Claudia', u'Gabriel', u'Erik', u'Samuel', u'John']]
```

Get the first and last names of the authors by providing multiple tags to `get_data_by_tag`.

```
>>> ent15000['entry_information']['_Entry_author'].get_data_b
y_tag(['Given_name', 'Family_name'])
[[u'Claudia', u'Cornilescu'],
 [u'Gabriel', u'Cornilescu'],
 [u'Erik', u'Hadley'],
 [u'Samuel', u'Gellman'],
 [u'John', u'Markley']]
```

Write the modified entry to disk in NMR-STAR format:

```
>>> ent15000_2.write_to_file("example_entry.str")
```

Get a list of validation errors. (The line numbers are only available if an entry is loaded from a file. When an entry is loaded from the API the line numbers are not preserved.)

```
>>> ent15000.validate()
["Value cannot be NULL but is: '_Chem_comp.Provenance':'.' on
 line 'None'."]
```

Here is how to create a loop from scratch

```
>>> new_loop = pynmrstar.Loop.from_scratch()
>>> new_loop.add_tag("_Example.age")
>>> new_loop.add_tag("name")
>>> new_loop.add_tag("description")
>>> new_loop.add_tag("notes")
```

Alternatively, you could replace above with:

```
>>> new_loop.add_tag(["_Example.age","name","description","no
tes"])
```

Now let’s add data to the loop

```
>>> new_loop.add_data([29,"Jon","A BRMB employee", None])
```

Notice that data is automatically encapsulated as necessary to meet the NMR-STAR format (quotes around the data containing a space). You never have to worry about encapsulating data you insert to make it syntactically valid STAR. Notice also that python None types are automatically converted to the NMR-STAR “null” value, “.”.

```
>>> print new_loop
   loop_
      _Example.age
      _Example.name
      _Example.description
      _Example.notes


     29   Jon   'A BRMB employee'   .


   stop_
```

Add the loop to the entry_information saveframe

```
>>> ent15000['entry_information'].add_loop(new_loop)
```

See that the loop has been added to the saveframe

```
>>> ent15000['entry_information'].print_tree()
<pynmrstar.Saveframe 'entry_information'>
[0] <pynmrstar.Loop '_Entry_author'>
[1] <pynmrstar.Loop '_SG_project'>
[2] <pynmrstar.Loop '_Struct_keywords'>
[3] <pynmrstar.Loop '_Data_set'>
[4] <pynmrstar.Loop '_Datum'>
[5] <pynmrstar.Loop '_Release'>
[6] <pynmrstar.Loop '_Related_entries'>
```

```
[7] <pynmrstar.Loop '_Example'>
```

View the value of the tag `ID` in the assembly saveframe

```
>>> ent15000['assembly'].get_tag('ID')
1
```

To get the NMR-STAR representation of any object, just request its string representation:

```
>>> print ent15000['assembly']
#############################################
#  Molecular system (assembly) description  #
#############################################

save_assembly
   _Assembly.Sf_category                     assembly
   _Assembly.Sf_framecode                    assembly
   _Assembly.Entry_ID                        15000
   _Assembly.ID                              1
   _Assembly.Name                            F5-Phe-cVHP
   _Assembly.BMRB_code                       .
   _Assembly.Number_of_components            1
   _Assembly.Organic_ligands                 .
   _Assembly.Metal_ions                      .
   _Assembly.Non_standard_bonds              .
   _Assembly.Ambiguous_conformational_states .
```

```
    _Assembly.Ambiguous_chem_comp_sites         .
    _Assembly.Molecules_in_chemical_exchange    .
    _Assembly.Paramagnetic                      no
    _Assembly.Thiol_state                       'all free'
    _Assembly.Molecular_mass                    .
    _Assembly.Enzyme_commission_number          .
    _Assembly.Details                           .
    _Assembly.DB_query_date                     .
    _Assembly.DB_query_revised_last_date        .


    loop_
       _Entity_assembly.ID
       _Entity_assembly.Entity_assembly_name
       _Entity_assembly.Entity_ID
       _Entity_assembly.Entity_label
       _Entity_assembly.Asym_ID
       _Entity_assembly.PDB_chain_ID
       _Entity_assembly.Experimental_data_reported
       _Entity_assembly.Physical_state
       _Entity_assembly.Conformational_isomer
       _Entity_assembly.Chemical_exchange_state
       _Entity_assembly.Magnetic_equivalence_group_code
       _Entity_assembly.Role
       _Entity_assembly.Details
       _Entity_assembly.Entry_ID
       _Entity_assembly.Assembly_ID


      1   F5-Phe-cVHP   1   $F5-Phe-cVHP   K   .   yes   nativ
```

```
e   no   no   .   .   .   15000   1

   stop_

save_

```

Reading Spectral Peaks from a NMR-STAR file:

First load the file and get a list of the peak list saveframes

```
>>> ent6577 = pynmrstar.Entry.from_database(6577)
>>> spectral_peaks = ent6577.get_saveframes_by_category('spectral_peak_list')
```

Lets look at how many spectral peak list saveframes we have

```
>>> spectral_peaks
[<pynmrstar.Saveframe 'peak_list_1'>,
 <pynmrstar.Saveframe 'peak_list_2'>,
 <pynmrstar.Saveframe 'peak_list_3'>]
```

For this demo we’ll just look at one individual peak list

```
>>> peak1 = spectral_peaks[0]
```

We can see what loops this peak list saveframe contains

```
>>> peak1.print_tree()
<pynmrstar.Saveframe 'peak_list_1'>
    [0] <pynmrstar.Loop '_Spectral_peak_software'>
    [1] <pynmrstar.Loop '_Peak_general_char'>
    [2] <pynmrstar.Loop '_Peak_char'>
    [3] <pynmrstar.Loop '_Assigned_peak_chem_shift'>
```

Let’s see what the `_Peak_char` loop looks like in NMR-STAR format

```
>>> print peak1['_Peak_char']
   loop_
      _Peak_char.Peak_ID
      _Peak_char.Spectral_dim_ID
      _Peak_char.Chem_shift_val
      _Peak_char.Chem_shift_val_err
      _Peak_char.Line_width_val
      _Peak_char.Line_width_val_err
      _Peak_char.Phase_val
      _Peak_char.Phase_val_err
      _Peak_char.Decay_rate_val
      _Peak_char.Decay_rate_val_err
      _Peak_char.Coupling_pattern
      _Peak_char.Bounding_box_upper_val
      _Peak_char.Bounding_box_lower_val
      _Peak_char.Bounding_box_range_val
      _Peak_char.Details
      _Peak_char.Derivation_method_ID
      _Peak_char.Entry_ID
      _Peak_char.Spectral_peak_list_ID


     1      1   9.857   .   .   .   .   .   .   .   .   .   .
   .   .   .   6577   1
     1      2   4.922   .   .   .   .   .   .   .   .   .   .
   .   .   .   6577   1
     2      1   9.857   .   .   .   .   .   .   .   .   .   .
   .   .   .   6577   1
     2      2   2.167   .   .   .   .   .   .   .   .   .   .
   .   .   .   6577   1
     ...etc...
```

That is more information than we want right now. Lets get just the columns we need (we'll get a list of lists, each inner list corresponds to a row:

```
>>> our_data = peak1['_Peak_char'].get_data_by_tag(['Peak_ID'
,'Chem_shift_val'])
>>> print our_data
[[u'1', u'9.857'],
 [u'1', u'4.922'],
 [u'2', u'9.857'],
 [u'2', u'2.167'],
 [u'3', u'9.855'],
 [u'3', u'1.994'],
 ...]
```

Excellent! Now we can iterate through each spectral peak and corresponding shift easily. The data is stored as a python list of lists (2 dimensional array) and we can modify or access it any of the normal ways python allows.

```
>>> for x in our_data:
>>>    print "Sprectral chemical shift value is: " + str(x[1]
)
Sprectral chemical shift value is: 9.857
Sprectral chemical shift value is: 4.922
Sprectral chemical shift value is: 9.857
...
```

It is also easy to dump the table in a loop as a CSV

```
>>> print peak1['_Peak_char'].get_data_as_csv()
_Peak_char.Peak_ID,_Peak_char.Spectral_dim_ID,_Peak_char.Chem
_shift_val,_Peak_char.Chem_shift_val_err,_Peak_char.Line_widt
h_val,_Peak_char.Line_width_val_err,_Peak_char.Phase_val,_Pea
k_char.Phase_val_err,_Peak_char.Decay_rate_val,_Peak_char.Dec
ay_rate_val_err,_Peak_char.Coupling_pattern,_Peak_char.Boundi
ng_box_upper_val,_Peak_char.Bounding_box_lower_val,_Peak_char
.Bounding_box_range_val,_Peak_char.Details,_Peak_char.Derivat
ion_method_ID,_Peak_char.Entry_ID,_Peak_char.Spectral_peak_li
st_ID
```

```
1,1,9.857,.,.,.,.,.,.,.,.,.,.,.,.,.,6577,1
1,2,4.922,.,.,.,.,.,.,.,.,.,.,.,.,.,6577,1
2,1,9.857,.,.,.,.,.,.,.,.,.,.,.,.,.,6577,1
2,2,2.167,.,.,.,.,.,.,.,.,.,.,.,.,.,6577,1
3,1,9.855,.,.,.,.,.,.,.,.,.,.,.,.,.,6577,1
3,2,1.994,.,.,.,.,.,.,.,.,.,.,.,.,.,6577,1
...
```