Overview of NMR-STAR Objects

Entry

Data:

- BMRB ID : 15000
- Saveframes: [<bmrb.saveframe 'entry_information'>, <bmrb.saveframe 'citation_1'>, ...]

Methods:

- Construct:
  - From scratch, from a file, from a string, from the public database of entries
- Add and remove saveframes
- Search for saveframes by name or by category
- Generate a string of the entry in NMR-STAR format
- Validate and compare to other entries
- Print structure of entry as tree

Saveframe

Data:

- Name : 'entry_information'
- Tag prefix: '_Entry'
- Tags: {'Sf_category':'entry_information', 'Sf_framecode':'entry_information', ...}
- Loops: [<bmrb.loop '_Entry_author'>, <bmrb.loop '_SG_project'>, ...]

Methods:

- Construct:
  - From scratch, from a file, from a string
- Add and remove tags
- Add and remove loops
- Search for loops by category
- Generate a string of the saveframe in NMR-STAR format
- Generate a CSV containing the data in the tags
- Validate and compare to other saveframes
- Print tree structure of saveframe

Loop

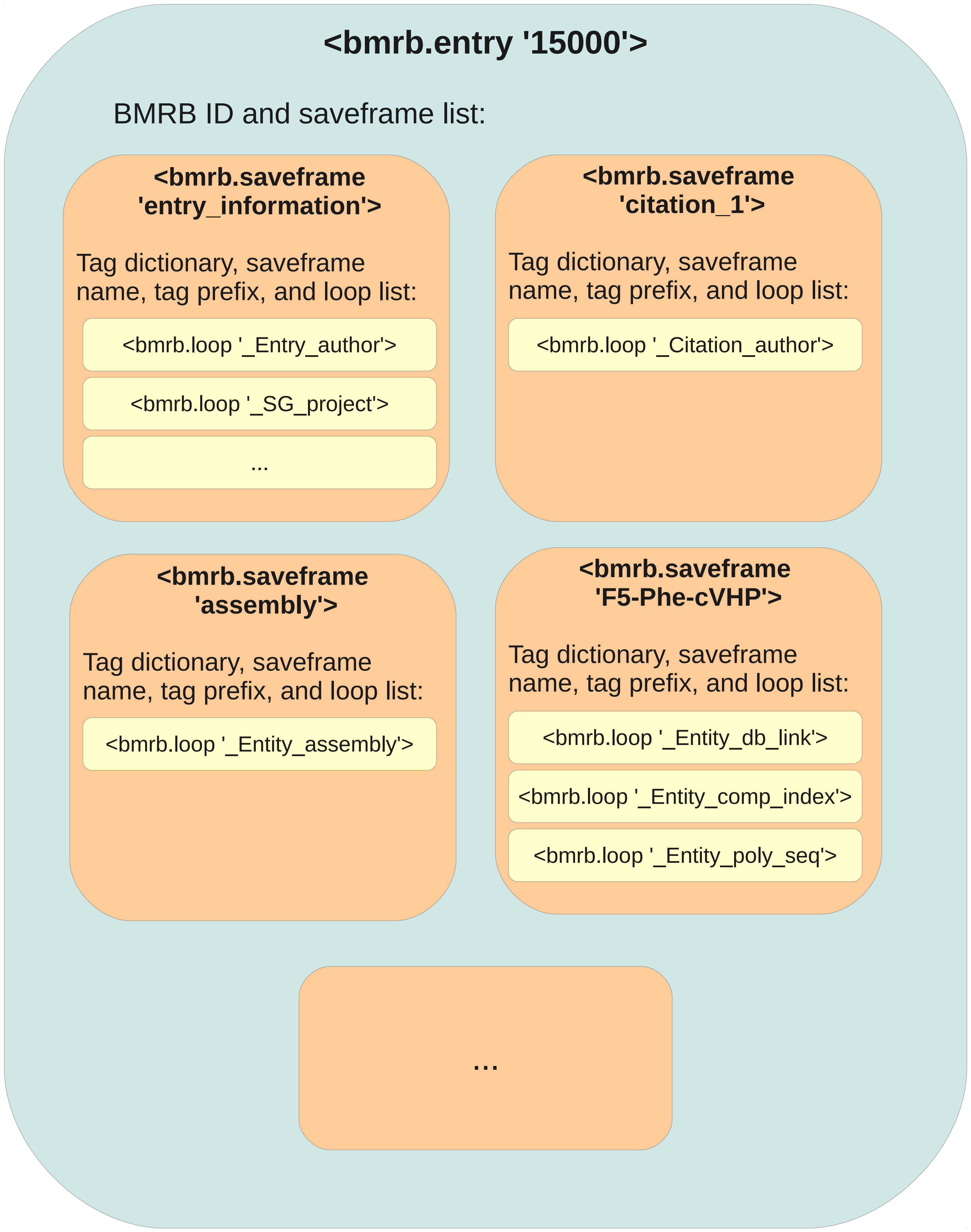
Data:

- Category : '_Entry_author'
- Columns:
  - ['Ordinal', 'Given_name', 'Family_name', 'First_initial', 'Middle_initials', 'Entry_ID1']
- Data:
  - [['1', 'Claudia', 'Cornilescu', '.', 'C.', '15000'], ['2', 'Gabriel', 'Cornilescu', '.', '.', '15000'],...]

Methods:

- Construct:
  - From scratch, from a file, from a string
- Add columns
- Add, remove, and renumber data
- Fetch data by column and/or fetch row matching certain data
- Generate a string of the loop in NMR-STAR format
- Generate a CSV containing the data
- Validate and compare to other loops

An example entry



<bmrb.entry '15000'>
BMRB ID and saveframe list:
<bmrb.saveframe 'entry_information'>
Tag dictionary, saveframe name, tag prefix, and loop list:
<bmrb.loop '_Entry_author'>
<bmrb.loop '_SG_project'>
...
<bmrb.saveframe 'citation_1'>
Tag dictionary, saveframe name, tag prefix, and loop list:
<bmrb.loop '_Citation_author'>
<bmrb.saveframe 'assembly'>
Tag dictionary, saveframe name, tag prefix, and loop list:
<bmrb.loop '_Entity_assembly'>
<bmrb.saveframe 'F5-Phe-cVHP'>
Tag dictionary, saveframe name, tag prefix, and loop list:
<bmrb.loop '_Entity_db_link'>
<bmrb.loop '_Entity_comp_index'>
<bmrb.loop '_Entity_poly_seq'>
...

More detailed example of a saveframe:

<bmrb.saveframe 'assigned_chem_shift_list_1'>

Tag_prefix: '_Assigned_chem_shift_list'
Tags: {'Sf_category':'assigned_chemical_shifts', 'Sf_framecode':'assigned_chem_shift_list_1','Entry_ID':'15000','ID': '1'), ...}
Loops: [<bmrb.loop '_Chem_shift_experiment'>, <bmrb.loop '_Atom_chem_shift'>]

<bmrb.loop '_Chem_shift_experiment'>

Category: '_Chem_shift'
Columns: ['Experiment_ID','Experiment_name',...]
Data: 2x2 array of data

In NMR-STAR format:

```
loop_
      _Chem_shift_experiment.Experiment_ID
      _Chem_shift_experiment.Experiment_name
      _Chem_shift_experiment.Sample_ID
      _Chem_shift_experiment.Sample_label
      _Chem_shift_experiment.Sample_state
      _Chem_shift_experiment.Entry_ID
      _Chem_shift_experiment.Assigned_chem_shift_list_ID

     .   '2D 1H-15N HSQC' 1   $unlabeled_sample             isotropic   15000   1
     .   '2D 1H-13C HSQC' 1   $unlabeled_sample             isotropic   15000   1
     .   '2D COSY'        1   $unlabeled_sample             isotropic   15000   1
     .   '2D TOCSY'       1   $unlabeled_sample             isotropic   15000   1
   stop_
```

<bmrb.loop '_Atom_chem_shift'>

Some sample code:

```
# Start the python interpreter and load the module
$: python
>>> import bmrb
>>> ent15000 = bmrb.entry.fromDatabase(15000)
>>> ent15000.printTree()
<bmrb.entry '15000'>
     [0] <bmrb.saveframe 'citation_1'>
          [0] <bmrb.loop '_Citation_author'>
     [1] <bmrb.saveframe 'assembly'>
          [0] <bmrb.loop '_Entity_assembly'>
     [2] <bmrb.saveframe 'F5-Phe-cVHP'>
          [0] <bmrb.loop '_Entity_db_link'>
          [1] <bmrb.loop '_Entity_comp_index'>
          [2] <bmrb.loop '_Entity_poly_seq'>
     [3] <bmrb.saveframe 'natural_source'>
          [0] <bmrb.loop '_Entity_natural_src'>
     [4] <bmrb.saveframe 'experimental_source'>
          [0] <bmrb.loop '_Entity_experimental_src'>
     [5] ....

# There is a shorthand way to access saveframes by name and loops by
category.
>>> ent15000['entry_information']
<bmrb.saveframe 'entry_information'>
>>> ent15000['entry_information']['_Entry_author']
<bmrb.loop '_Entry_author'>

>>> ent15000_2 = bmrb.entry.fromDatabase(15000)
>>> del ent15000_2['entry_information']
>>> bmrb.diff(ent15000,ent15000_2)
The number of saveframes in the entries are not equal: 25 vs 24
No saveframe with name 'entry_information' in other entry.

# Let's look at a loop's column headers and its data
>>> ent15000['entry_information']['_Entry_author'].columns
['Ordinal', 'Given_name', 'Family_name', 'First_initial',
'Middle_initials', 'Family_title', 'Entry_ID']
>>> ent15000['entry_information']
['_Entry_author'].getDataByTag('Given_name')
[['Claudia'], ['Gabriel'], ['Erik'], ['Samuel'], ['John']]

# Write the entry to disk in NMR-STAR format
>>> with open('/tmp/entry','w') as tmp_file:
...     tmp_file.write(str(ent15000))

# Get a list of validation errors
>>> ent15000.validate()
["Tag '_Entry.Type' not found in schema. Line 4.", "Value cannot be NULL
but is: '_Chem_comp.Provenance':'.' on line 4.", ...]
```

```
# Here is how to create a loop from scratch
>>> new_loop = bmrb.loop.fromScratch()
>>> new_loop.addColumn("_Example.day")
>>> new_loop.addColumn("month")
>>> new_loop.addColumn("year")
>>> new_loop.addColumn("event")
# You could replace above with:
>>new_loop.addColumn(["_Example.day","month",”year”,”event”])
# Add data to the loop
>>> new_loop.addData([25,11,13,"Nothing exciting happened."])

# Notice that data is automatically encapsulated as necessary to meet STAR
format (quotes around the data containing a space). You never have to worry
about encapsulating data you insert.
>>> print new_loop

   loop_
      _Example.day
      _Example.month
      _Example.year
      _Example.event

     25   11   13   'Nothing exciting happened.'
   stop_

# Add the loop to the entry_information saveframe
>>> ent15000['entry_information'].addLoop(new_loop)
>>> ent15000['entry_information'].printTree()
<bmrb.saveframe 'entry_information'>
     [0] <bmrb.loop '_Entry_author'>
     [1] <bmrb.loop '_SG_project'>
     [2] <bmrb.loop '_Struct_keywords'>
     [3] <bmrb.loop '_Data_set'>
     [4] <bmrb.loop '_Datum'>
     [5] <bmrb.loop '_Release'>
     [6] <bmrb.loop '_Related_entries'>
     [7] <bmrb.loop '_Example'>

>>> print ent15000['assembly'].tags
OrderedDict([('Sf_category', 'assembly'), ('Sf_framecode', 'assembly'),
('Entry_ID', '15000'), ('ID', '1'), ('Name', 'F5-Phe-cVHP'), ('BMRB_code',
'.'), ('Number_of_components', '1'), ...])
>>> ent15000['assembly'].getTag('ID')
1
>>> ent15000['assembly'].printTree()
<bmrb.saveframe 'assembly'>
     [0] <bmrb.loop '_Entity_assembly'>
```

# Get a saveframe in NMR-STAR format

```
>>> print ent15000['assembly']
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   native   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 = bmrb.entry.fromDatabase(6577)
>>> spectral_peaks = ent6577.getSaveframesByCategory('spectral_peak_list')

# Lets look at how many spectral peak list saveframes we have
>>> spectral_peaks
[<bmrb.saveframe 'peak_list_1'>, <bmrb.saveframe 'peak_list_2'>,
<bmrb.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.printTree()
<bmrb.saveframe 'peak_list_1'>
     [0] <bmrb.loop '_Spectral_peak_software'>
     [1] <bmrb.loop '_Peak_general_char'>
     [2] <bmrb.loop '_Peak_char'>
     [3] <bmrb.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.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
     ...
stop_
```

Reading Spectral Peaks from a NMR-STAR file (cont.):

```
# 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'].getDataByTag(
                                     ['Peak_ID','Chem_shift_val'])

>>> print our_data
[['1', '9.857'],
 ['1', '4.922'],
 ['2', '9.857'],
 ['2', '2.167'],
 ['3', '9.855'],
 ...]

# 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'].getDataAsCSV()
Peak_ID,Spectral_dim_ID,Chem_shift_val,Chem_shift_val_err,Line_width_
val,Line_width_val_err,Phase_val,Phase_val_err,Decay_rate_val,Decay_r
ate_val_err,Coupling_pattern,Bounding_box_upper_val,Bounding_box_lowe
r_val,Bounding_box_range_val,Details,Derivation_method_ID,Entry_ID,Sp
ectral_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
3,1,9.855,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,6577,1
3,2,1.994,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,6577,1
4,1,9.854,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,6577,1
```