

---

# **dpdata Documentation**

*Release 0.0.0-rc*

**Han Wang**

**May 08, 2022**



## CONTENTS:

<b>1</b>	<b>Supported Formats</b>	<b>1</b>
<b>2</b>	<b>API documentation</b>	<b>3</b>
2.1	dpdata package . . . . .	3
<b>3</b>	<b>Installation</b>	<b>89</b>
<b>4</b>	<b>Quick start</b>	<b>91</b>
4.1	Load data . . . . .	91
4.2	Access data . . . . .	93
4.3	Dump data . . . . .	94
4.4	replicate . . . . .	94
4.5	perturb . . . . .	94
4.6	replace . . . . .	95
<b>5</b>	<b>BondOrderSystem</b>	<b>97</b>
5.1	Bond Order Assignment . . . . .	97
5.2	Formal Charge Assignment . . . . .	98
<b>6</b>	<b>Plugins</b>	<b>99</b>
<b>7</b>	<b>Indices and tables</b>	<b>101</b>
	<b>Python Module Index</b>	<b>103</b>
	<b>Index</b>	<b>105</b>



## SUPPORTED FORMATS

dpdata supports the following formats:

Table 1: Supported

Class	Alias	Supported Function
<i>ListFormat</i>	list	<i>LabeledSystem.t</i>
<i>DeepMDRawFormat</i>	deepmd/raw deepmd	<i>MultiSystems.lo</i>
<i>DeepMDCompFormat</i>	deepmd/comp deepmd/npz	<i>MultiSystems.lo</i>
<i>DeepMDHDF5Format</i>	deepmd/hdf5	<i>MultiSystems.lo</i>
<i>GaussianLogFormat</i>	gaussian/log	<i>LabeledSystem()</i>
<i>GaussianMDFormat</i>	gaussian/md	<i>LabeledSystem()</i>
<i>GromacsGroFormat</i>	gromacs/gro gro	<i>LabeledSystem.t</i>
<i>VASPPoscarFormat</i>	vasp/poscar vasp/poscar contcar poscar	<i>LabeledSystem.t</i>
<i>VASPStringFormat</i>	vasp/string	<i>LabeledSystem.t</i>
<i>VASPOutcarFormat</i>	vasp/outcar outcar	<i>LabeledSystem()</i>
<i>VASPXMLFormat</i>	vasp/xml xml	<i>LabeledSystem()</i>
<i>CP2KAIMDOutputFormat</i>	cp2k/aimd_output	<i>LabeledSystem()</i>
<i>CP2KOutputFormat</i>	cp2k/output	<i>LabeledSystem()</i>
<i>XYZFormat</i>	xyz	<i>LabeledSystem.t</i>
<i>QuipGapXYZFormat</i>	quip/gap/xyz_file quip/gap/xyz	<i>MultiSystems.lo</i>
<i>AbacusSCFFormat</i>	abacus/lcao/scf abacus/pw/scf abacus/scf	<i>LabeledSystem()</i>
<i>AbacusMDFormat</i>	abacus/lcao/md abacus/pw/md abacus/md	<i>LabeledSystem()</i>
<i>SiestaOutputFormat</i>	siesta/output	<i>LabeledSystem()</i>
<i>SiestaAIMDOutputFormat</i>	siesta/aimd_output	<i>LabeledSystem()</i>
<i>MolFormat</i>	mol_file mol	<i>BondOrderSystem</i>
<i>SdfFormat</i>	sdf_file sdf	<i>BondOrderSystem</i>
<i>LAMMPSLmpFormat</i>	lammps/lmp lmp	<i>LabeledSystem.t</i>
<i>LAMMPSDumpFormat</i>	lammps/dump dump	<i>System()</i>
<i>AmberMDFormat</i>	amber/md	<i>LabeledSystem()</i>
<i>SQMOutFormat</i>	sqm/out	<i>LabeledSystem()</i>
<i>SQMInFormat</i>	sqm/in	<i>LabeledSystem.t</i>
<i>PyMatgenStructureFormat</i>	pymatgen/structure	<i>LabeledSystem.t</i>
<i>PyMatgenMoleculeFormat</i>	pymatgen/molecule	<i>LabeledSystem.t</i>
<i>PyMatgenCSEFormat</i>	pymatgen/computedstructureentry	<i>LabeledSystem.t</i>
<i>PwmatOutputFormat</i>	pwmat/output pwmat/mlmd pwmat/movement mlmd movement	<i>LabeledSystem()</i>
<i>PwmatAtomconfigFormat</i>	pwmat/final.config pwmat/atom.config final.config atom.config	<i>LabeledSystem.t</i>
<i>QECPTrajFormat</i>	qe/cp/traj	<i>LabeledSystem()</i>
<i>QECPPWSCFFormat</i>	qe/pw/scf	<i>LabeledSystem()</i>
<i>FhiMDFormat</i>	fhi_aims/output fhi_aims/md	<i>LabeledSystem()</i>

Table 1 – continued from

Class	Alias	Supported Functions
<i>FhiSCFFormat</i>	fhi_aims/scf	<i>LabeledSystem()</i>
<i>ASEStructureFormat</i>	ase/structure	<i>MultiSystems.Lo</i>

## 2.1 dpdata package

### 2.1.1 Subpackages

#### dpdata.amber package

##### Submodules

##### dpdata.amber.mask module

Amber mask

`dpdata.amber.mask.load_param_file(param_file)`

`dpdata.amber.mask.pick_by_amber_mask(param, maskstr, coords=None)`

Pick atoms by amber masks

##### Parameters

**param:** **str** or **parmed.Structure** filename of Amber param file or parmed.Structure

**maskstr:** **str** Amber masks

**coords:** **np.ndarray (optional)** frame coordinates, shape: N\*3

##### dpdata.amber.md module

`dpdata.amber.md.read_amber_traj(parm7_file, nc_file, mdfrc_file=None, mden_file=None, mdout_file=None, use_element_symbols=None, labeled=True)`

The amber trajectory includes: \* nc, NetCDF format, stores coordinates \* mdfrc, NetCDF format, stores forces \* mden (optional), text format, stores energies \* mdout (optional), text format, may store energies if there is no mden\_file \* parm7, text format, stores types

##### Parameters

**parm7\_file, nc\_file, mdfrc\_file, mden\_file, mdout\_file:** filenames

**use\_element\_symbols:** **None** or **list** or **str** If use\_element\_symbols is a list of atom indexes, these atoms will use element symbols instead of amber types. For example, a ligand will use C, H, O, N, and so on instead of h1, hc, o, os, and so on. IF use\_element\_symbols is str, it will be considered as Amber mask.

## dpdata.amber.sqm module

`dpdata.amber.sqm.make_sqm_in(data, fname=None, frame_idx=0, **kwargs)`

`dpdata.amber.sqm.parse_sqm_out(fname)`

Read atom symbols, charges and coordinates from ambertools sqm.out file

## dpdata.cp2k package

### Submodules

#### dpdata.cp2k.cell module

`dpdata.cp2k.cell.cell_to_low_triangle(A, B, C, alpha, beta, gamma)`

Convert cell to low triangle matrix.

#### Parameters

**A** [float] cell length A

**B** [float] cell length B

**C** [float] cell length C

**alpha** [float] radian. The angle between vector B and vector C.

**beta** [float] radian. The angle between vector A and vector C.

**gamma** [float] radian. The angle between vector B and vector C.

#### Returns

**cell** [list] The cell matrix used by dpdata in low triangle form.

#### dpdata.cp2k.output module

`class dpdata.cp2k.output.Cp2kSystems(log_file_name, xyz_file_name, restart=False)`

Bases: `object`

deal with cp2k outputfile

#### Methods

<code>get_log_block_generator</code>	
<code>get_xyz_block_generator</code>	
<code>handle_single_log_frame</code>	
<code>handle_single_xyz_frame</code>	

`get_log_block_generator()`

`get_xyz_block_generator()`

`handle_single_log_frame(lines)`



`handle_single_xyz_frame`(*lines*)

`dpdata.cp2k.output.get_frames`(*fname*)

## dpdata.deepmd package

### Submodules

#### dpdata.deepmd.comp module

`dpdata.deepmd.comp.dump`(*folder*, *data*, *set\_size=5000*, *comp\_prec=<class 'numpy.float32'>*, *remove\_sets=True*)

`dpdata.deepmd.comp.to_system_data`(*folder*, *type\_map=None*, *labels=True*)

#### dpdata.deepmd.hdf5 module

Utils for deepmd/hdf5 format.

`dpdata.deepmd.hdf5.dump`(*f: h5py.\_hl.files.File*, *folder: str*, *data: dict*, *set\_size=5000*, *comp\_prec=<class 'numpy.float32'>*) → `None`

Dump data to a HDF5 file.

##### Parameters

- f** [h5py.File] HDF5 file object
- folder** [str] path in the HDF5 file
- data** [dict] System or LabeledSystem data
- set\_size** [int, default: 5000] size of a set
- comp\_prec** [np.dtype, default: np.float32] precision of data

`dpdata.deepmd.hdf5.to_system_data`(*f: h5py.\_hl.files.File*, *folder: str*, *type\_map: Optional[list] = None*, *labels: bool = True*)

Load a HDF5 file.

##### Parameters

- f** [h5py.File] HDF5 file object
- folder** [str] path in the HDF5 file
- type\_map** [list] type map
- labels** [bool] labels

## dpdata.deepmd.raw module

dpdata.deepmd.raw.**dump**(*folder, data*)

dpdata.deepmd.raw.**load\_type**(*folder, type\_map=None*)

dpdata.deepmd.raw.**to\_system\_data**(*folder, type\_map=None, labels=True*)

## dpdata.fhi\_aims package

### Submodules

#### dpdata.fhi\_aims.output module

dpdata.fhi\_aims.output.**analyze\_block**(*lines, first\_blk=False, md=True*)

dpdata.fhi\_aims.output.**get\_fhi\_aims\_block**(*fp*)

dpdata.fhi\_aims.output.**get\_frames**(*fname, md=True, begin=0, step=1*)

dpdata.fhi\_aims.output.**get\_info**(*lines, type\_idx\_zero=False*)

## dpdata.gaussian package

### Submodules

#### dpdata.gaussian.log module

dpdata.gaussian.log.**to\_system\_data**(*file\_name, md=False*)

## dpdata.gromacs package

### Submodules

#### dpdata.gromacs.gro module

dpdata.gromacs.gro.**file\_to\_system\_data**(*fname, format\_atom\_name=True, \*\*kwargs*)

dpdata.gromacs.gro.**from\_system\_data**(*system, f\_idx=0, \*\*kwargs*)

## dpdata.lammps package

### Submodules

#### dpdata.lammps.dump module

dpdata.lammps.dump.**box2dumpbox**(*orig, box*)

---

```
dpdata.lammps.dump.dumpbox2box(bounds, tilt)
dpdata.lammps.dump.get_atype(lines, type_idx_zero=False)
dpdata.lammps.dump.get_coordtype_and_scalefactor(keys)
dpdata.lammps.dump.get_dumpbox(lines)
dpdata.lammps.dump.get_natoms(lines)
dpdata.lammps.dump.get_natoms_vec(lines)
dpdata.lammps.dump.get_natomtypes(lines)
dpdata.lammps.dump.load_file(fname, begin=0, step=1)
dpdata.lammps.dump.safe_get_posi(lines, cell, orig=array([0.0, 0.0, 0.0]))
dpdata.lammps.dump.split_traj(dump_lines)
dpdata.lammps.dump.system_data(lines, type_map=None, type_idx_zero=True)
```

### dpdata.lammps.lmp module

```
dpdata.lammps.lmp.box2lmpbox(orig, box)
dpdata.lammps.lmp.from_system_data(system, f_idx=0)
dpdata.lammps.lmp.get_atoms(lines)
dpdata.lammps.lmp.get_atype(lines, type_idx_zero=False)
dpdata.lammps.lmp.get_lmpbox(lines)
dpdata.lammps.lmp.get_natoms(lines)
dpdata.lammps.lmp.get_natoms_vec(lines)
dpdata.lammps.lmp.get_natomtypes(lines)
dpdata.lammps.lmp.get_posi(lines)
dpdata.lammps.lmp.lmpbox2box(lohi, tilt)
dpdata.lammps.lmp.system_data(lines, type_map=None, type_idx_zero=True)
dpdata.lammps.lmp.to_system_data(lines, type_map=None, type_idx_zero=True)
```

### dpdata.plugins package

#### Submodules

#### dpdata.plugins.abacus module

```
class dpdata.plugins.abacus.AbacusMDFormat
```

```
    Bases: dpdata.format.Format
```

## Methods

<code>MultiModes()</code>	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<code><i>from_labeled_system</i>(file_name, **kwargs)</code>	
<code>from_multi_systems(directory, **kwargs)</code>	MultiSystems.from
<code>from_system(file_name, **kwargs)</code>	System.from
<code>register(key)</code>	Register a virtual subclass of an ABC.
<code>to_system(data, *args, **kwargs)</code>	System.to

<b>from_bond_order_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

`from_labeled_system(file_name, **kwargs)`

**class** `dpdata.plugins.abacus.AbacusSCFFormat`

Bases: `dpdata.format.Format`

## Methods

<code>MultiModes()</code>	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<code><i>from_labeled_system</i>(file_name, **kwargs)</code>	
<code>from_multi_systems(directory, **kwargs)</code>	MultiSystems.from
<code>from_system(file_name, **kwargs)</code>	System.from
<code>register(key)</code>	Register a virtual subclass of an ABC.
<code>to_system(data, *args, **kwargs)</code>	System.to

<b>from_bond_order_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_labeled\_system**(*file\_name*, **\*\*kwargs**)

## dpdata.plugins.amber module

**class** dpdata.plugins.amber.**AmberMDFormat**

Bases: *dpdata.format.Format*

### Methods

<b>MultiModes</b> ()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<b>from_multi_systems</b> ( <i>directory</i> , <b>**kwargs</b> )	MultiSystems.from
<b>from_system</b> ([ <i>file_name</i> , <i>parm7_file</i> , ...])	System.from
<b>register</b> ( <i>key</i> )	Register a virtual subclass of an ABC.
<b>to_system</b> ( <i>data</i> , <i>*args</i> , <b>**kwargs</b> )	System.to

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_labeled\_system**(*file\_name*=None, *parm7\_file*=None, *nc\_file*=None, *mdfrc\_file*=None, *mden\_file*=None, *mdout\_file*=None, *use\_element\_symbols*=None, **\*\*kwargs**)

**from\_system**(*file\_name*=None, *parm7\_file*=None, *nc\_file*=None, *use\_element\_symbols*=None, **\*\*kwargs**)  
System.from

#### Parameters

**file\_name**: str file name

### Returns

**data:** dict system data

**class** dpdata.plugins.amber.SQMINFormat

Bases: *dpdata.format.Format*

### Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data[, fname, frame_idx])	Generate input files for semi-empirical calculation in sqm software

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**to\_system**(data, fname=None, frame\_idx=0, \*\*kwargs)

Generate input files for semi-empirical calculation in sqm software

**class** dpdata.plugins.amber.SQMOutFormat

Bases: *dpdata.format.Format*

### Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<i>from_labeled_system</i> (fname, **kwargs)	Read from ambertools sqm.out
from_multi_systems(directory, **kwargs)	MultiSystems.from
<i>from_system</i> (fname, **kwargs)	Read from ambertools sqm.out
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

<b>from_bond_order_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_labeled\_system**(*fname*, **\*\*kwargs**)

Read from ambertools sqm.out

**from\_system**(*fname*, **\*\*kwargs**)

Read from ambertools sqm.out

## dpdata.plugins.ase module

**class** dpdata.plugins.ase.ASEStructureFormat

Bases: *dpdata.format.Format*

Format for the Atomic Simulation Environment (ase).

ASE supports parsing a few dozen of data formats. As described in [the documentation](#), many of these formats can be determined automatically. Use the *ase\_fmt* keyword argument to supply the format if automatic detection fails.

## Methods

<b>MultiModes</b> ()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<i>from_multi_systems</i> ( <i>file_name</i> [, <i>begin</i> , <i>end</i> , ...])	MultiSystems.from
<b>from_system</b> ( <i>file_name</i> , <b>**kwargs</b> )	System.from
<b>register</b> ( <i>key</i> )	Register a virtual subclass of an ABC.
<i>to_labeled_system</i> ( <i>data</i> , <i>*args</i> , <b>**kwargs</b> )	Convert System to ASE Atoms object.
<i>to_system</i> ( <i>data</i> , <b>**kwargs</b> )	convert System to ASE Atom obj

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_multi_systems</b>	

`from_labeled_system(data, **kwargs)`

`from_multi_systems(file_name, begin=None, end=None, step=None, ase_fmt=None, **kwargs)`

MultiSystems.from

#### Parameters

**directory:** `str` directory of system

#### Returns

**filenames:** `list[str]` list of filenames

`to_labeled_system(data, *args, **kwargs)`

Convert System to ASE Atoms object.

`to_system(data, **kwargs)`

convert System to ASE Atom obj

### dpdata.plugins.cp2k module

`class dpdata.plugins.cp2k.CP2KAIMDOutputFormat`

Bases: `dpdata.format.Format`

#### Methods

<code>MultiModes()</code>	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<code>from_multi_systems(directory, **kwargs)</code>	MultiSystems.from
<code>from_system(file_name, **kwargs)</code>	System.from
<code>register(key)</code>	Register a virtual subclass of an ABC.
<code>to_system(data, *args, **kwargs)</code>	System.to

<code>from_bond_order_system</code>	
<code>from_labeled_system</code>	
<code>get_formats</code>	
<code>get_from_methods</code>	
<code>get_to_methods</code>	
<code>post</code>	
<code>register_from</code>	
<code>register_to</code>	
<code>to_bond_order_system</code>	
<code>to_labeled_system</code>	
<code>to_multi_systems</code>	

`from_labeled_system(file_name, restart=False, **kwargs)`

`class dpdata.plugins.cp2k.CP2KOutputFormat`

Bases: `dpdata.format.Format`



## Methods

<code>MultiModes()</code>	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<code>from_multi_systems(directory, **kwargs)</code>	MultiSystems.from
<code>from_system(file_name, **kwargs)</code>	System.from
<code>register(key)</code>	Register a virtual subclass of an ABC.
<code>to_system(data, *args, **kwargs)</code>	System.to

<code>from_bond_order_system</code>	
<code>from_labeled_system</code>	
<code>get_formats</code>	
<code>get_from_methods</code>	
<code>get_to_methods</code>	
<code>post</code>	
<code>register_from</code>	
<code>register_to</code>	
<code>to_bond_order_system</code>	
<code>to_labeled_system</code>	
<code>to_multi_systems</code>	

`from_labeled_system(file_name, restart=False, **kwargs)`

## dpdata.plugins.deepmd module

**class** `dpdata.plugins.deepmd.DeePMDCompFormat`

Bases: `dpdata.format.Format`

## Methods

<code>MultiModes()</code>	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<code>from_multi_systems(directory, **kwargs)</code>	MultiSystems.from
<code>from_system(file_name[, type_map])</code>	System.from
<code>register(key)</code>	Register a virtual subclass of an ABC.
<code>to_system(data, file_name[, set_size, prec])</code>	Dump the system in deepmd compressed format (numpy binary) to <i>folder</i> .

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**MultiMode = 1**

**from\_labeled\_system**(*file\_name*, *type\_map=None*, *\*\*kwargs*)

**from\_system**(*file\_name*, *type\_map=None*, *\*\*kwargs*)

System.from

#### Parameters

**file\_name:** str file name

#### Returns

**data:** dict system data

**to\_system**(*data*, *file\_name*, *set\_size=5000*, *prec=<class 'numpy.float64'>*, *\*\*kwargs*)

Dump the system in deepmd compressed format (numpy binary) to *folder*.

The frames are firstly split to sets, then dumped to separated subfolders named as *folder/set.000*, *folder/set.001*, ...

Each set contains *set\_size* frames. The last set may have less frames than *set\_size*.

#### Parameters

**data:** dict System data

**file\_name** [str] The output folder

**set\_size** [int] The size of each set.

**prec:** {**numpy.float32**, **numpy.float64**} The floating point precision of the compressed data

**class** dpdata.plugins.deepmd.DeepMDHDF5Format

Bases: *dpdata.format.Format*

HDF5 format for DeePMD-kit.

## Examples

Dump a MultiSystems to a HDF5 file: `>>> import dpdata >>> dpdata.MultiSystems().from_deepmd_npy("data").to_deepmd_hdf5("data.hdf5")`

## Methods

<code>MultiModes()</code>	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<code>from_multi_systems(directory, **kwargs)</code>	MultiSystems.from
<code>from_system(file_name[, type_map])</code>	System.from
<code>register(key)</code>	Register a virtual subclass of an ABC.
<code>to_system(data, file_name[, set_size, comp_prec])</code>	System.to

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

`from_labeled_system(file_name, type_map=None, **kwargs)`

`from_multi_systems(directory, **kwargs)`

MultiSystems.from

### Parameters

**directory:** str directory of system

### Returns

**filenames:** list[str] list of filenames

`from_system(file_name, type_map=None, **kwargs)`

System.from

### Parameters

**file\_name:** str file name

### Returns

**data:** dict system data

`to_multi_systems(formulas, directory, **kwargs)`

`to_system(data: dict, file_name: str, set_size: int = 5000, comp_prec: numpy.dtype = <class 'numpy.float64'>, **kwargs)`

System.to

**Parameters**

**data: dict** system data

**class** dpdata.plugins.deepmd.DeePMDRawFormat

Bases: *dpdata.format.Format*

**Methods**

<code>MultiModes()</code>	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<code>from_multi_systems(directory, **kwargs)</code>	MultiSystems.from
<code>from_system(file_name[, type_map])</code>	System.from
<code>register(key)</code>	Register a virtual subclass of an ABC.
<code>to_system(data, file_name, **kwargs)</code>	Dump the system in deepmd raw format to directory <i>file_name</i>

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**MultiMode = 1**

`from_labeled_system(file_name, type_map=None, **kwargs)`

`from_system(file_name, type_map=None, **kwargs)`

System.from

**Parameters**

**file\_name: str** file name

**Returns**

**data: dict** system data

`to_system(data, file_name, **kwargs)`

Dump the system in deepmd raw format to directory *file\_name*

**dpdata.plugins.fhi\_aims module****class** dpdata.plugins.fhi\_aims.FhiMDFormatBases: *dpdata.format.Format***Methods**

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_labeled\_system**(*file\_name*, *md=True*, *begin=0*, *step=1*, \*\**kwargs*)**class** dpdata.plugins.fhi\_aims.FhiSCFFormatBases: *dpdata.format.Format***Methods**

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

`from_labeled_system(file_name, **kwargs)`

## dpdata.plugins.gaussian module

**class** dpdata.plugins.gaussian.GaussianLogFormat

Bases: *dpdata.format.Format*

### Methods

<code>MultiModes()</code>	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<code>from_multi_systems(directory, **kwargs)</code>	MultiSystems.from
<code>from_system(file_name, **kwargs)</code>	System.from
<code>register(key)</code>	Register a virtual subclass of an ABC.
<code>to_system(data, *args, **kwargs)</code>	System.to

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

`from_labeled_system(file_name, md=False, **kwargs)`

**class** dpdata.plugins.gaussian.GaussianMDFormat

Bases: *dpdata.format.Format*

## Methods

<code>MultiModes()</code>	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<code>from_multi_systems(directory, **kwargs)</code>	MultiSystems.from
<code>from_system(file_name, **kwargs)</code>	System.from
<code>register(key)</code>	Register a virtual subclass of an ABC.
<code>to_system(data, *args, **kwargs)</code>	System.to

<code>from_bond_order_system</code>	
<code>from_labeled_system</code>	
<code>get_formats</code>	
<code>get_from_methods</code>	
<code>get_to_methods</code>	
<code>post</code>	
<code>register_from</code>	
<code>register_to</code>	
<code>to_bond_order_system</code>	
<code>to_labeled_system</code>	
<code>to_multi_systems</code>	

`from_labeled_system(file_name, **kwargs)`

## dpdata.plugins.gromacs module

`class dpdata.plugins.gromacs.GromacsGroFormat`

Bases: `dpdata.format.Format`

## Methods

<code>MultiModes()</code>	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<code>from_multi_systems(directory, **kwargs)</code>	MultiSystems.from
<code>from_system(file_name[, format_atom_name])</code>	Load gromacs .gro file
<code>register(key)</code>	Register a virtual subclass of an ABC.
<code>to_system(data[, file_name, frame_idx])</code>	Dump the system in gromacs .gro format

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_system**(*file\_name*, *format\_atom\_name=True*, *\*\*kwargs*)

Load gromacs .gro file

**Parameters**

**file\_name** [str] The input file name

**to\_system**(*data*, *file\_name=None*, *frame\_idx=- 1*, *\*\*kwargs*)

Dump the system in gromacs .gro format

**Parameters**

**file\_name** [str or None] The output file name. If None, return the file content as a string

**frame\_idx** [int] The index of the frame to dump

**dpdata.plugins.lammps module**

**class** dpdata.plugins.lammps.LAMPSDumpFormat

Bases: *dpdata.format.Format*

**Methods**

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, <i>**kwargs</i> )	MultiSystems.from
<i>from_system</i> ( <i>file_name</i> [, <i>type_map</i> , <i>begin</i> , <i>step</i> ])	System.from
register( <i>key</i> )	Register a virtual subclass of an ABC.
to_system( <i>data</i> , <i>*args</i> , <i>**kwargs</i> )	System.to



<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_system**(*file\_name*, *type\_map*=None, *begin*=0, *step*=1, *\*\*kwargs*)

System.from

#### Parameters

**file\_name**: str file name

#### Returns

**data**: dict system data

**class** dpdata.plugins.lammps.LAMMPSLmpFormat

Bases: *dpdata.format.Format*

#### Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, <i>**kwargs</i> )	MultiSystems.from
<i>from_system</i> ( <i>file_name</i> [, <i>type_map</i> ])	System.from
register(key)	Register a virtual subclass of an ABC.
<i>to_system</i> (data, <i>file_name</i> [, <i>frame_idx</i> ])	Dump the system in lammps data format

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_system**(*file\_name*, *type\_map*=None, *\*\*kwargs*)

System.from

#### Parameters

**file\_name:** str file name

**Returns**

**data:** dict system data

**to\_system**(data, file\_name, frame\_idx=0, \*\*kwargs)

Dump the system in lammps data format

**Parameters**

**data:** dict System data

**file\_name** [str] The output file name

**frame\_idx** [int] The index of the frame to dump

**dpdata.plugins.list module**

**class** dpdata.plugins.list.**ListFormat**

Bases: *dpdata.format.Format*

**Methods**

<b>MultiModes</b> ()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<b>from_multi_systems</b> (directory, **kwargs)	MultiSystems.from
<b>from_system</b> (file_name, **kwargs)	System.from
<b>register</b> (key)	Register a virtual subclass of an ABC.
<b>to_system</b> (data, **kwargs)	convert system to list, usefull for data collection

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**to\_system**(data, \*\*kwargs)

convert system to list, usefull for data collection

**dpdata.plugins.pwmat module**

**class** dpdata.plugins.pwmat.PwmatAtomconfigFormat

Bases: *dpdata.format.Format*

**Methods**

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, file_name[, frame_idx])	Dump the system in pwmat atom.config format

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_system**(*file\_name*, \*\**kwargs*)

System.from

**Parameters**

**file\_name:** str file name

**Returns**

**data:** dict system data

**to\_system**(*data*, *file\_name*, *frame\_idx=0*, \**args*, \*\**kwargs*)

Dump the system in pwmat atom.config format

**Parameters**

**file\_name** [str] The output file name

**frame\_idx** [int] The index of the frame to dump

**class** dpdata.plugins.pwmat.PwmatOutputFormat

Bases: *dpdata.format.Format*

## Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

`from_labeled_system(file_name, begin=0, step=1, **kwargs)`

## dpdata.plugins.pymatgen module

**class** dpdata.plugins.pymatgen.PyMatgenCSEFormat

Bases: *dpdata.format.Format*

## Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
<i>to_labeled_system</i> (data, *args, **kwargs)	convert System to Pymagen ComputedStructureEntry obj
to_system(data, *args, **kwargs)	System.to

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_multi_systems</b>	

**to\_labeled\_system**(*data*, \**args*, \*\**kwargs*)

convert System to Pymagen ComputedStructureEntry obj

**class** dpdata.plugins.pymatgen.PyMatgenMoleculeFormat

Bases: *dpdata.format.Format*

## Methods

<b>MultiModes</b> ()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<b>from_multi_systems</b> ( <i>directory</i> , ** <i>kwargs</i> )	MultiSystems.from
<i>from_system</i> ( <i>file_name</i> , ** <i>kwargs</i> )	System.from
<b>register</b> ( <i>key</i> )	Register a virtual subclass of an ABC.
<i>to_system</i> ( <i>data</i> , ** <i>kwargs</i> )	convert System to Pymatgen Molecule obj

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_system**(*file\_name*, \*\**kwargs*)

System.from

### Parameters

**file\_name:** str file name

### Returns

**data:** dict system data

**to\_system**(data, \*\*kwargs)  
 convert System to Pymatgen Molecule obj

**class** dpdata.plugins.pymatgen.PyMatgenStructureFormat

Bases: *dpdata.format.Format*

### Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, **kwargs)	convert System to Pymatgen Structure obj

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**to\_system**(data, \*\*kwargs)  
 convert System to Pymatgen Structure obj

### dpdata.plugins.qe module

**class** dpdata.plugins.qe.QECPWSCFFormat

Bases: *dpdata.format.Format*

### Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_labeled\_system**(*file\_name*, *\*\*kwargs*)

**class** dpdata.plugins.qe.QECPTrajFormat

Bases: *dpdata.format.Format*

### Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<b>from_multi_systems</b> ( <i>directory</i> , <i>**kwargs</i> )	MultiSystems.from
<i>from_system</i> ( <i>file_name</i> [, <i>begin</i> , <i>step</i> ])	System.from
<b>register</b> ( <i>key</i> )	Register a virtual subclass of an ABC.
<b>to_system</b> ( <i>data</i> , <i>*args</i> , <i>**kwargs</i> )	System.to

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_labeled\_system**(*file\_name*, *begin=0*, *step=1*, *\*\*kwargs*)

**from\_system**(*file\_name*, *begin=0*, *step=1*, *\*\*kwargs*)

System.from

#### Parameters

**file\_name**: str file name

#### Returns

**data**: dict system data

## dpdata.plugins.rdkit module

class dpdata.plugins.rdkit.MolFormat

Bases: *dpdata.format.Format*

## Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_bond\_order\_system**(*file\_name*, \*\*kwargs)

**to\_bond\_order\_system**(*data*, *mol*, *file\_name*, *frame\_idx=0*, \*\*kwargs)

class dpdata.plugins.rdkit.SdfFormat

Bases: *dpdata.format.Format*

## Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<i>from_bond_order_system</i> ( <i>file_name</i> , **kwargs)	Note that it requires all molecules in .sdf file must be of the same topology
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to



<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_bond\_order\_system**(*file\_name*, **\*\*kwargs**)

Note that it requires all molecules in .sdf file must be of the same topology

**to\_bond\_order\_system**(*data*, *mol*, *file\_name*, *frame\_idx=-1*, **\*\*kwargs**)

## dpdata.plugins.siesta module

**class** dpdata.plugins.siesta.SiestaAIMDOutputFormat

Bases: *dpdata.format.Format*

### Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<b>from_multi_systems</b> ( <i>directory</i> , <b>**kwargs</b> )	MultiSystems.from
<i>from_system</i> ( <i>file_name</i> , <b>**kwargs</b> )	System.from
<b>register</b> ( <i>key</i> )	Register a virtual subclass of an ABC.
<b>to_system</b> ( <i>data</i> , <i>*args</i> , <b>**kwargs</b> )	System.to

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_labeled\_system**(*file\_name*, **\*\*kwargs**)

**from\_system**(*file\_name*, **\*\*kwargs**)

System.from

**Parameters**

**file\_name: str** file name

**Returns**

**data: dict** system data

**class** dpdata.plugins.siesta.SiestaOutputFormat

Bases: *dpdata.format.Format*

**Methods**

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
<i>from_system</i> (file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_labeled\_system**(*file\_name*, \*\**kwargs*)

**from\_system**(*file\_name*, \*\**kwargs*)

System.from

**Parameters**

**file\_name: str** file name

**Returns**

**data: dict** system data

**dpdata.plugins.vasp module****class** dpdata.plugins.vasp.VASPOutcarFormatBases: *dpdata.format.Format***Methods**

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_labeled\_system**(*file\_name*, *begin=0*, *step=1*, \*\**kwargs*)**class** dpdata.plugins.vasp.VASPPoscarFormatBases: *dpdata.format.Format***Methods**

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
<i>from_system</i> (file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
<i>to_system</i> (data, file_name[, frame_idx])	Dump the system in vasp POSCAR format

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_system**(*file\_name*, **\*\*kwargs**)

System.from

**Parameters**

**file\_name**: str file name

**Returns**

**data**: dict system data

**to\_system**(*data*, *file\_name*, *frame\_idx=0*, **\*\*kwargs**)

Dump the system in vasp POSCAR format

**Parameters**

**file\_name** [str] The output file name

**frame\_idx** [int] The index of the frame to dump

**class** dpdata.plugins.vasp.VASPStringFormat

Bases: *dpdata.format.Format*

**Methods**

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, <b>**kwargs</b> )	MultiSystems.from
from_system(file_name, <b>**kwargs</b> )	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data[, frame_idx])	Dump the system in vasp POSCAR format string

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**to\_system**(*data*, *frame\_idx=0*, *\*\*kwargs*)

Dump the system in vasp POSCAR format string

#### Parameters

**frame\_idx** [int] The index of the frame to dump

**class** dpdata.plugins.vasp.VASPXMLFormat

Bases: *dpdata.format.Format*

#### Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, <i>**kwargs</i> )	MultiSystems.from
from_system(file_name, <i>**kwargs</i> )	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, <i>*args</i> , <i>**kwargs</i> )	System.to

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_labeled\_system**(*file\_name*, *begin=0*, *step=1*, *\*\*kwargs*)

## dpdata.plugins.xyz module

**class** dpdata.plugins.xyz.QuipGapXYZFormat

Bases: *dpdata.format.Format*

### Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<i>from_multi_systems</i> (file_name, **kwargs)	MultiSystems.from
<i>from_system</i> (file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**from\_labeled\_system**(data, \*\*kwargs)

**from\_multi\_systems**(file\_name, \*\*kwargs)

MultiSystems.from

#### Parameters

**directory:** str directory of system

#### Returns

**filenames:** list[str] list of filenames

**class** dpdata.plugins.xyz.XYZFormat

Bases: *dpdata.format.Format*

XYZ format.

## Examples

```
>>> s.to("xyz", "a.xyz")
```

## Methods

<code>MultiModes()</code>	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<code>from_multi_systems(directory, **kwargs)</code>	MultiSystems.from
<code>from_system(file_name, **kwargs)</code>	System.from
<code>register(key)</code>	Register a virtual subclass of an ABC.
<code>to_system(data, file_name, **kwargs)</code>	System.to

<code>from_bond_order_system</code>	
<code>from_labeled_system</code>	
<code>get_formats</code>	
<code>get_from_methods</code>	
<code>get_to_methods</code>	
<code>post</code>	
<code>register_from</code>	
<code>register_to</code>	
<code>to_bond_order_system</code>	
<code>to_labeled_system</code>	
<code>to_multi_systems</code>	

`to_system(data, file_name, **kwargs)`

System.to

### Parameters

**data: dict** system data

## dpdata.pwmat package

### Submodules

#### dpdata.pwmat.atomconfig module

`dpdata.pwmat.atomconfig.from_system_data(system, f_idx=0, skip_zeros=True)`

`dpdata.pwmat.atomconfig.to_system_data(lines)`

## dpdata.pwmat.movement module

dpdata.pwmat.movement.**analyze\_block**(*lines*, *ntot*, *nelm*)

dpdata.pwmat.movement.**get\_frames**(*fname*, *begin=0*, *step=1*)

dpdata.pwmat.movement.**get\_movement\_block**(*fp*)

dpdata.pwmat.movement.**system\_info**(*lines*, *type\_idx\_zero=False*)

## dpdata.pymatgen package

### Submodules

#### dpdata.pymatgen.molecule module

dpdata.pymatgen.molecule.**to\_system\_data**(*file\_name*, *protect\_layer=9*)

## dpdata.qe package

### Submodules

#### dpdata.qe.scf module

dpdata.qe.scf.**get\_block**(*lines*, *keyword*, *skip=0*)

dpdata.qe.scf.**get\_cell**(*lines*)

dpdata.qe.scf.**get\_coords**(*lines*, *cell*)

dpdata.qe.scf.**get\_energy**(*lines*)

dpdata.qe.scf.**get\_force**(*lines*)

dpdata.qe.scf.**get\_frame**(*fname*)

dpdata.qe.scf.**get\_stress**(*lines*)

#### dpdata.qe.traj module

dpdata.qe.traj.**convert\_celldm**(*ibrav*, *celldm*)

dpdata.qe.traj.**load\_atom\_names**(*lines*, *ntypes*)

dpdata.qe.traj.**load\_atom\_types**(*lines*, *natoms*, *atom\_names*)

dpdata.qe.traj.**load\_block**(*lines*, *key*, *nlines*)

dpdata.qe.traj.**load\_cell\_parameters**(*lines*)

dpdata.qe.traj.**load\_celldm**(*lines*)



`dpdata.qe.traj.load_data(fname, natoms, begin=0, step=1, convert=1.0)`

`dpdata.qe.traj.load_energy(fname, begin=0, step=1)`

`dpdata.qe.traj.load_key(lines, key)`

`dpdata.qe.traj.load_param_file(fname)`

`dpdata.qe.traj.to_system_data(input_name, prefix, begin=0, step=1)`

`dpdata.qe.traj.to_system_label(input_name, prefix, begin=0, step=1)`

## dpdata.rdkit package

### Submodules

#### dpdata.rdkit.sanitize module

#### dpdata.rdkit.utils module

`dpdata.rdkit.utils.check_molecule_list(mols)`

`dpdata.rdkit.utils.check_same_atom(atom_1, atom_2)`

`dpdata.rdkit.utils.check_same_molecule(mol_1, mol_2)`

`dpdata.rdkit.utils.combine_molecules(mols)`

`dpdata.rdkit.utils.mol_to_system_data(mol)`

`dpdata.rdkit.utils.system_data_to_mol(data)`

## dpdata.siesta package

### Submodules

#### dpdata.siesta.aiMD\_output module

`dpdata.siesta.aiMD_output.covert_dimension(arr, num)`

`dpdata.siesta.aiMD_output.extract_keyword(fout, keyword, down_line_num, begin_column,  
read_column_num, is_repeated_read, column_num)`

`dpdata.siesta.aiMD_output.get_aiMD_frame(fname)`

`dpdata.siesta.aiMD_output.get_atom_name(fout)`

`dpdata.siesta.aiMD_output.get_atom_numbs(atomtypes)`

`dpdata.siesta.aiMD_output.get_atom_types(fout, atomnums)`

`dpdata.siesta.aiMD_output.get_single_line_tail(fin, keyword, num=1)`

`dpdata.siesta.aiMD_output.get_virial(fout, cell)`

`dpdata.siesta.aiMD_output.obtain_nframe(fname)`

### dpdata.siesta.output module

dpdata.siesta.output.**extract\_keyword**(*fout*, *keyword*, *down\_line\_num*, *begin\_column*, *column\_num*)

dpdata.siesta.output.**get\_atom\_name**(*fout*)

dpdata.siesta.output.**get\_atom\_numbs**(*atomtypes*)

dpdata.siesta.output.**get\_atom\_types**(*fout*, *atomnums*)

dpdata.siesta.output.**get\_single\_line\_tail**(*fin*, *keyword*, *num=1*)

dpdata.siesta.output.**get\_virial**(*fout*, *cells*)

dpdata.siesta.output.**obtain\_frame**(*fname*)

### dpdata.vasp package

#### Submodules

#### dpdata.vasp.outcar module

dpdata.vasp.outcar.**analyze\_block**(*lines*, *ntot*, *nelm*)

dpdata.vasp.outcar.**get\_frames**(*fname*, *begin=0*, *step=1*)

dpdata.vasp.outcar.**get\_outcar\_block**(*fp*)

dpdata.vasp.outcar.**system\_info**(*lines*, *type\_idx\_zero=False*)

#### dpdata.vasp.poscar module

dpdata.vasp.poscar.**from\_system\_data**(*system*, *f\_idx=0*, *skip\_zeros=True*)

dpdata.vasp.poscar.**to\_system\_data**(*lines*)

#### dpdata.vasp.xml module

dpdata.vasp.xml.**analyze**(*fname*, *type\_idx\_zero=False*, *begin=0*, *step=1*)

can deal with broken xml file

dpdata.vasp.xml.**analyze\_atominfo**(*atominfo\_xml*)

dpdata.vasp.xml.**analyze\_calculation**(*cc*)

dpdata.vasp.xml.**check\_name**(*item*, *name*)

dpdata.vasp.xml.**formulate\_config**(*eles*, *types*, *posi*, *cell*, *ener*, *forc*, *strs\_*)

dpdata.vasp.xml.**get\_varray**(*varray*)

## 2.1.2 Submodules

### 2.1.3 dpdata.bond\_order\_system module

### 2.1.4 dpdata.cli module

Command line interface for dpdata.

```
dpdata.cli.convert(*, from_file: str, from_format: str = 'auto', to_file: Optional[str] = None, to_format:
    Optional[str] = None, no_labeled: bool = False, multi: bool = False, type_map:
    Optional[list] = None, **kwargs)
```

Convert files from one format to another one.

#### Parameters

**from\_file** [str] read data from a file  
**from\_format** [str] the format of from\_file  
**to\_file** [str] dump data to a file  
**to\_format** [str] the format of to\_file  
**no\_labeled** [bool] labels aren't provided  
**multi** [bool] the system contains multiple directories  
**type\_map** [list] type map

```
dpdata.cli.dpdata_cli()
dpdata cli.
```

#### Examples

```
$ dpdata -iposcar POSCAR -odeepmd/npz -O data -n
```

### 2.1.5 dpdata.format module

Implement the format plugin system.

```
class dpdata.format.Format
    Bases: abc.ABC
```

#### Methods

<i>MultiModes()</i>	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
<i>from_multi_systems(directory, **kwargs)</i>	MultiSystems.from
<i>from_system(file_name, **kwargs)</i>	System.from
<i>register(key)</i>	Register a virtual subclass of an ABC.
<i>to_system(data, *args, **kwargs)</i>	System.to

<b>from_bond_order_system</b>	
<b>from_labeled_system</b>	
<b>get_formats</b>	
<b>get_from_methods</b>	
<b>get_to_methods</b>	
<b>post</b>	
<b>register_from</b>	
<b>register_to</b>	
<b>to_bond_order_system</b>	
<b>to_labeled_system</b>	
<b>to_multi_systems</b>	

**MultiMode = 0**

**class MultiModes**

Bases: `object`

File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system

**Directory = 1**

**NotImplemented = 0**

**from\_bond\_order\_system**(*file\_name*, *\*\*kwargs*)

**from\_labeled\_system**(*file\_name*, *\*\*kwargs*)

**from\_multi\_systems**(*directory*, *\*\*kwargs*)

MultiSystems.from

**Parameters**

**directory: str** directory of system

**Returns**

**filenames: list[str]** list of filenames

**from\_system**(*file\_name*, *\*\*kwargs*)

System.from

**Parameters**

**file\_name: str** file name

**Returns**

**data: dict** system data

**static get\_formats**()

**static get\_from\_methods**()

**static get\_to\_methods**()

**static post**(*func\_name*)

**static register**(*key*)  
 Register a virtual subclass of an ABC.  
 Returns the subclass, to allow usage as a class decorator.

**static register\_from**(*key*)

**static register\_to**(*key*)

**to\_bond\_order\_system**(*data*, *rdkit\_mol*, *\*args*, *\*\*kwargs*)

**to\_labeled\_system**(*data*, *\*args*, *\*\*kwargs*)

**to\_multi\_systems**(*formulas*, *directory*, *\*\*kwargs*)

**to\_system**(*data*, *\*args*, *\*\*kwargs*)  
 System.to

**Parameters**

**data: dict** system data

## 2.1.6 dpdata.periodic\_table module

**class** dpdata.periodic\_table.**Element**(*symbol: str*)

Bases: `object`

### Attributes

**X**

**Z**

**calculated\_radius**

**mass**

**name**

**radius**

### Methods

from_Z	
--------	--

**property X**

**property Z**

**property calculated\_radius**

**classmethod from\_Z(Z)**

**property mass**

**property name**

**property radius**

## 2.1.7 dpdata.plugin module

Base of plugin systems.

**class** dpdata.plugin.Plugin

Bases: `object`

A class to register plugins.

### Examples

```
>>> Plugin = Register()
>>> @Plugin.register("xx")
    def xxx():
        pass
>>> print(Plugin.plugins['xx'])
```

### Methods

---

<code>register(key)</code>	Register a plugin.
----------------------------	--------------------

---

<code>get_plugin</code>	
-------------------------	--

`get_plugin(key)`

`register(key)`

Register a plugin.

## 2.1.8 dpdata.system module

**class** dpdata.system.LabeledSystem(*file\_name=None, fmt='auto', type\_map=None, begin=0, step=1, data=None, \*\*kwargs*)

Bases: `dpdata.system.System`

The labeled data System

**For example, a labeled water system named `d_example` has two molecules (6 atoms) and `nframes` frames. The labels can be**

- `d_example['energies']`: a numpy array of size `nframes`
- `d_example['forces']`: a numpy array of size `nframes x 6 x 3`
- `d_example['virials']`: optional, a numpy array of size `nframes x 3 x 3`

#### It is noted that

- The order of frames stored in `'energies'`, `'forces'` and `'virials'` should be consistent with `'atom_types'`, `'cells'` and `'coords'`.
- The order of atoms in **every** frame of `'forces'` should be consistent with `'coords'` and `'atom_types'`.

#### Attributes

**formula** Return the formula of this system, like C3H5O2

**nopbc**

**uniq\_formula** Return the uniq\_formula of this system.

## Methods

<code>add_atom_names(atom_names)</code>	Add atom_names that do not exist.
<code>append(system)</code>	Append a system to this system
<code>apply_pbc()</code>	Append periodic boundary condition
<code>as_dict()</code>	Returns data dict of System instance
<code>check_type_map(type_map)</code>	Assign atom_names to type_map if type_map is given and different from atom_names.
<code>copy()</code>	Returns a copy of the system.
<code>correction(hl_sys)</code>	Get energy and force correction between self and a high-level LabeledSystem.
<code>dump(filename[, indent])</code>	dump .json or .yaml file
<code>extend(systems)</code>	Extend a system list to this system
<code>from_abacus_lcao_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>from_abacus_lcao_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>from_abacus_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>from_abacus_pw_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>from_abacus_pw_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>from_abacus_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>from_amber_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.amber.AmberMDFormat</code> format.
<code>from_ase_structure(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.ase.ASEStructureFormat</code> format.
<code>from_atomconfig(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>from_contcar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>from_cp2k_aimd_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.cp2k.CP2KAIMDOutputFormat</code> format.
<code>from_cp2k_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.cp2k.CP2KOutputFormat</code> format.
<code>from_deepmd(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDRawFormat</code> format.
<code>from_deepmd_comp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.
<code>from_deepmd_hdf5(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDHDF5Format</code> format.
<code>from_deepmd_npy(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.

continues on next page

Table 1 – continued from previous page

<code>from_deepmd_raw(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDRawFormat</code> format.
<code>from_dict(d)</code>	<b>param d</b> Dict representation.
<code>from_dump(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>from_fhi_aims_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>from_fhi_aims_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>from_fhi_aims_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiSCFFormat</code> format.
<code>from_finalconfig(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>from_gaussian_log(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianLogFormat</code> format.
<code>from_gaussian_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianMDFormat</code> format.
<code>from_gro(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>from_gromacs_gro(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>from_lammps_dump(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>from_lammps_lmp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>from_list(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.list.ListFormat</code> format.
<code>from_lmp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>from_mlmd(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>from_mol(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>from_mol_file(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>from_movement(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>from_outcar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>from_poscar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>from_pwmat_atomconfig(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>from_pwmat_finalconfig(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>from_pwmat_mlmd(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>from_pwmat_movement(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.

continues on next page



Table 1 – continued from previous page

<code>from_pwmat_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>from_pymatgen_computedstructureentry(...)</code>	Read data from <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>from_pymatgen_molecule(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pymatgen.PyMatgenMoleculeFormat</code> format.
<code>from_pymatgen_structure(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pymatgen.PyMatgenStructureFormat</code> format.
<code>from_qe_cp_traj(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.qe.QECPTrajFormat</code> format.
<code>from_qe_pw_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.qe.QECPWSCFFFormat</code> format.
<code>from_quip_gap_xyz(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>from_quip_gap_xyz_file(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>from_sdf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>from_sdf_file(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>from_siesta_aiMD_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.siesta.SiestaAIMDOutputFormat</code> format.
<code>from_siesta_aimd_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.siesta.SiestaAIMDOutputFormat</code> format.
<code>from_siesta_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.siesta.SiestaOutputFormat</code> format.
<code>from_sqm_in(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.amber.SQMInFormat</code> format.
<code>from_sqm_out(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.amber.SQMOutFormat</code> format.
<code>from_vasp_contcar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>from_vasp_outcar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>from_vasp_poscar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>from_vasp_string(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPStringFormat</code> format.
<code>from_vasp_xml(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>from_xml(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>from_xyz(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.xyz.XYZFormat</code> format.
<code>get_atom_names()</code>	Returns name of atoms
<code>get_atom_numbs()</code>	Returns number of atoms
<code>get_atom_types()</code>	Returns type of atoms
<code>get_natoms()</code>	Returns total number of atoms in the system
<code>get_nframes()</code>	Returns number of frames in the system
<code>load(filename)</code>	rebuild System obj.

continues on next page

Table 1 – continued from previous page

<code>map_atom_types([type_map])</code>	Map the atom types of the system Parameters -- ----- <code>type_map : dict : {"H":0,"O":1}</code> or list ["H","C","O","N"] The map between elements and index if no <code>map_dict</code> is given, index will be set ac- cording to atomic number
<code>perturb(pert_num, cell_pert_fraction, ..., ...)</code>	Perturb each frame in the system randomly.
<code>pick_atom_idx(idx[, nopbc])</code>	Pick atom index
<code>pick_by_amber_mask(param, maskstr[, ...])</code>	Pick atoms by amber mask
<code>predict(dp)</code>	Predict energies and forces by deepmd-kit.
<code>remove_atom_names(atom_names)</code>	Remove atom names and all such atoms.
<code>remove_pbc([protect_layer])</code>	This method does NOT delete the definition of the cells, it (1) revises the cell to a cubic cell and ensures that the cell boundary to any atom in the system is no less than <i>protect_layer</i> (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.
<code>replicate(ncopy)</code>	Replicate the each frame in the system in 3 dimen- sions.
<code>shuffle()</code>	Also shuffle labeled data e.g.
<code>sort_atom_names([type_map])</code>	Sort <code>atom_names</code> of the system and reorder <code>atom_numbs</code> and <code>atom_types</code> according to <code>atom_names</code> .
<code>sub_system(f_idx)</code>	Construct a subsystem from the system
<code>to(fmt, *args, **kwargs)</code>	Dump systems to the specific format.
<code>to_abacus_lcao_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus. AbacusMDFormat</code> format.
<code>to_abacus_lcao_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus. AbacusSCFFormat</code> format.
<code>to_abacus_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus. AbacusMDFormat</code> format.
<code>to_abacus_pw_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus. AbacusMDFormat</code> format.
<code>to_abacus_pw_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus. AbacusSCFFormat</code> format.
<code>to_abacus_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus. AbacusSCFFormat</code> format.
<code>to_amber_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber. AmberMDFormat</code> format.
<code>to_ase_structure(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.ase. ASEStructureFormat</code> format.
<code>to_atomconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat. PwmatAtomconfigFormat</code> format.
<code>to_contcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp. VASPPoscarFormat</code> format.
<code>to_cp2k_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k. CP2KAIMDOutputFormat</code> format.
<code>to_cp2k_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k. CP2KOutputFormat</code> format.
<code>to_deepmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd. DeePMDRawFormat</code> format.

continues on next page

Table 1 – continued from previous page

<code>to_deepmd_comp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.
<code>to_deepmd_hdf5(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDHDF5Format</code> format.
<code>to_deepmd_npy(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.
<code>to_deepmd_raw(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDRawFormat</code> format.
<code>to_dump(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>to_fhi_aims_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>to_fhi_aims_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>to_fhi_aims_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiSCFFormat</code> format.
<code>to_finalconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_gaussian_log(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianLogFormat</code> format.
<code>to_gaussian_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianMDFormat</code> format.
<code>to_gro(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>to_gromacs_gro(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>to_json()</code>	Returns a json string representation of the MSONable object.
<code>to_lammps_dump(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>to_lammps_lmp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>to_list(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.list.ListFormat</code> format.
<code>to_lmp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>to_mlmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_mol(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>to_mol_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>to_movement(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_outcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>to_poscar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_pwmat_atomconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.

continues on next page

Table 1 – continued from previous page

<code>to_pwmat_finalconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_pwmat_mlmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_movement(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pymatgen_ComputedStructureEntry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>to_pymatgen_computedstructureentry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>to_pymatgen_molecule(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenMoleculeFormat</code> format.
<code>to_pymatgen_structure(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenStructureFormat</code> format.
<code>to_qe_cp_traj(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPTrajFormat</code> format.
<code>to_qe_pw_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPWSCFFormat</code> format.
<code>to_quip_gap_xyz(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_quip_gap_xyz_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_sdf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_sdf_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_siesta_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaAIMDOutputFormat</code> format.
<code>to_siesta_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaOutputFormat</code> format.
<code>to_sqm_in(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMInFormat</code> format.
<code>to_sqm_out(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMOutFormat</code> format.
<code>to_vasp_contcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_outcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>to_vasp_poscar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_string(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPStringFormat</code> format.
<code>to_vasp_xml(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>to_xml(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>to_xyz(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.XYZFormat</code> format.
<code>unsafe_hash()</code>	Returns an hash of the current object.

continues on next page

Table 1 – continued from previous page

validate_monty(v)	pydantic Validator for MSONable pattern
-------------------	---

<b>affine_map</b>	
<b>affine_map_fv</b>	
<b>apply_type_map</b>	
<b>from_fmt</b>	
<b>from_fmt_obj</b>	
<b>has_virial</b>	
<b>replace</b>	
<b>rot_frame_lower_triangular</b>	
<b>rot_lower_triangular</b>	
<b>sort_atom_types</b>	
<b>to_fmt_obj</b>	

**affine\_map\_fv**(*trans*, *f\_idx*)

**append**(*system*)

Append a system to this system

**Parameters**

**system** [System] The system to append

**correction**(*hl\_sys*)

Get energy and force correction between self and a high-level LabeledSystem. The self's coordinates will be kept, but energy and forces will be replaced by the correction between these two systems.

Note: The function will not check whether coordinates and elements of two systems are the same. The user should make sure by itself.

**Parameters**

**hl\_sys: LabeledSystem** high-level LabeledSystem

**Returns**

\_\_\_\_\_

**corrected\_sys: LabeledSystem** Corrected LabeledSystem

**from\_abacus\_lcao\_md**(*file\_name*, **\*\*kwargs**)

Read data from `dpdata.plugins.abacus.AbacusMDFormat` format.

**from\_abacus\_lcao\_scf**(*file\_name*, **\*\*kwargs**)

Read data from `dpdata.plugins.abacus.AbacusSCFFormat` format.

**from\_abacus\_md**(*file\_name*, **\*\*kwargs**)

Read data from `dpdata.plugins.abacus.AbacusMDFormat` format.

**from\_abacus\_pw\_md**(*file\_name*, **\*\*kwargs**)

Read data from `dpdata.plugins.abacus.AbacusMDFormat` format.

**from\_abacus\_pw\_scf**(*file\_name*, **\*\*kwargs**)

Read data from `dpdata.plugins.abacus.AbacusSCFFormat` format.

**from\_abacus\_scf**(*file\_name*, **\*\*kwargs**)

Read data from `dpdata.plugins.abacus.AbacusSCFFormat` format.

**from\_amber\_md**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.amber.AmberMDFormat` format.

**from\_ase\_structure**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.ase.ASEStructureFormat` format.

**from\_atomconfig**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_contcar**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_cp2k\_aimd\_output**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.cp2k.CP2KAIMDOutputFormat` format.

**from\_cp2k\_output**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.cp2k.CP2KOutputFormat` format.

**from\_deepmd**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**from\_deepmd\_comp**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**from\_deepmd\_hdf5**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.deepmd.DeepMDHDF5Format` format.

**from\_deepmd\_npy**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**from\_deepmd\_raw**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**from\_dump**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**from\_fhi\_aims\_md**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_output**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_scf**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**from\_finalconfig**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_fmt\_obj**(*fmtobj*, *file\_name*, **\*\*kwargs**)

**from\_gaussian\_log**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.gaussian.GaussianLogFormat` format.

**from\_gaussian\_md**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.gaussian.GaussianMDFormat` format.

**from\_gro**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.gromacs.GromacsGroFormat` format.

**from\_gromacs\_gro**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.gromacs.GromacsGroFormat` format.

**from\_lammps\_dump**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**from\_lammps\_lmp**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**from\_list**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.list.ListFormat` format.

**from\_lmp**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**from\_mlmd**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_mol**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_mol\_file**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_movement**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_outcar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPOutcarFormat` format.

**from\_poscar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_pwmat\_atomconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_finalconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_mlmd**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_movement**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pymatgen\_computedstructureentry**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**from\_pymatgen\_molecule**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**from\_pymatgen\_structure**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**from\_qe\_cp\_traj**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.qe.QECPTrajFormat` format.

**from\_qe\_pw\_scf**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.qe.QECPWSCFFormat` format.

**from\_quip\_gap\_xyz**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_quip\_gap\_xyz\_file**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_sdf**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_sdf\_file**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_siesta\_aiMD\_output**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_aimd\_output**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_output**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.siesta.SiestaOutputFormat` format.

**from\_sqm\_in**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.amber.SQMInFormat` format.

**from\_sqm\_out**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.amber.SQMOutFormat` format.

**from\_vasp\_contcar**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_vasp\_outcar**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPOutcarFormat` format.

**from\_vasp\_poscar**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_vasp\_string**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPStringFormat` format.

**from\_vasp\_xml**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPXMLFormat` format.

**from\_xml**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPXMLFormat` format.

**from\_xyz**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.xyz.XYZFormat` format.

**has\_virial**()



**pick\_atom\_idx**(*idx*, *nopbc=None*)

Pick atom index

#### Parameters

**idx**: int or list or slice atom index

**nopbc**: Boolean (default: None) If nopbc is True or False, set nopbc

#### Returns

**new\_sys**: LabeledSystem new system

**post\_funcs** = <dpdata.plugin.Plugin object>

**rot\_frame\_lower\_triangular**(*f\_idx=0*)

**rot\_lower\_triangular**()

**shuffle**()

Also shuffle labeled data e.g. energies and forces.

**sort\_atom\_types**()

**sub\_system**(*f\_idx*)

Construct a subsystem from the system

#### Parameters

**f\_idx** [int or index] Which frame to use in the subsystem

#### Returns

**sub\_system** [LabeledSystem] The subsystem

**to\_abacus\_lcao\_md**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.abacus.AbacusMDFormat` format.

**to\_abacus\_lcao\_scf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.abacus.AbacusSCFFormat` format.

**to\_abacus\_md**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.abacus.AbacusMDFormat` format.

**to\_abacus\_pw\_md**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.abacus.AbacusMDFormat` format.

**to\_abacus\_pw\_scf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.abacus.AbacusSCFFormat` format.

**to\_abacus\_scf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.abacus.AbacusSCFFormat` format.

**to\_amber\_md**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.amber.AmberMDFormat` format.

**to\_ase\_structure**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.ase.ASEStructureFormat` format.

**to\_atomconfig**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_contcar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_cp2k\_aimd\_output**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.cp2k.CP2KAIMDOutputFormat` format.

**to\_cp2k\_output**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.cp2k.CP2KOutputFormat` format.

**to\_deepmd**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**to\_deepmd\_comp**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**to\_deepmd\_hdf5**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.deepmd.DeepMDHDF5Format` format.

**to\_deepmd\_npy**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**to\_deepmd\_raw**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**to\_dump**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**to\_fhi\_aims\_md**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_output**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_scf**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**to\_finalconfig**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_fmt\_obj**(fmtobj, \*args, \*\*kwargs)

**to\_gaussian\_log**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.gaussian.GaussianLogFormat` format.

**to\_gaussian\_md**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.gaussian.GaussianMDFormat` format.

**to\_gro**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.gromacs.GromacsGroFormat` format.

**to\_gromacs\_gro**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.gromacs.GromacsGroFormat` format.

**to\_lammps\_dump**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**to\_lammps\_lmp**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**to\_list**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.list.ListFormat` format.

**to\_lmp**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**to\_mlmd**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_mol**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.rdkit.MolFormat` format.

**to\_mol\_file**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.rdkit.MolFormat` format.

**to\_movement**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_outcar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPOutcarFormat` format.

**to\_poscar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_pwmat\_atomconfig**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_pwmat\_finalconfig**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_pwmat\_mlmd**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_movement**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_output**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pymatgen\_ComputedStructureEntry**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_computedstructureentry**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_molecule**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**to\_pymatgen\_structure**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**to\_qe\_cp\_traj**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.qe.QECPTrajFormat` format.

**to\_qe\_pw\_scf**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.qe.QEPPWSCFFormat` format.

**to\_quip\_gap\_xyz**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**to\_quip\_gap\_xyz\_file**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**to\_sdf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.rdkit.SdfFormat` format.

**to\_sdf\_file**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.rdkit.SdfFormat` format.

**to\_siesta\_aimd\_output**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**to\_siesta\_output**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.siesta.SiestaOutputFormat` format.

**to\_sqm\_in**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.amber.SQMInFormat` format.

**to\_sqm\_out**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.amber.SQMOutFormat` format.

**to\_vasp\_contcar**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_vasp\_outcar**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.vasp.VASPOutcarFormat` format.

**to\_vasp\_poscar**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_vasp\_string**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.vasp.VASPStringFormat` format.

**to\_vasp\_xml**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.vasp.VASPXMLFormat` format.

**to\_xml**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.vasp.VASPXMLFormat` format.

**to\_xyz**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.xyz.XYZFormat` format.

**class** `dpdata.system.MultiSystems`(\*systems, type\_map=None)

Bases: `object`

A set containing several systems.

## Methods

<i>append</i> (*systems)	Append systems or MultiSystems to systems
<i>check_atom_names</i> (system)	Make atom_names in all systems equal, prevent inconsistent atom_types.
<i>from_abacus_lcao_md</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.
<i>from_abacus_lcao_scf</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusSCFFormat</i> format.
<i>from_abacus_md</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.
<i>from_abacus_pw_md</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.
<i>from_abacus_pw_scf</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusSCFFormat</i> format.
<i>from_abacus_scf</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusSCFFormat</i> format.
<i>from_amber_md</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.amber.AmberMDFormat</i> format.
<i>from_ase_structure</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.ase.ASEStructureFormat</i> format.
<i>from_atomconfig</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatAtomconfigFormat</i> format.
<i>from_contcar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPPoscarFormat</i> format.
<i>from_cp2k_aimd_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.cp2k.CP2KAIMDOutputFormat</i> format.
<i>from_cp2k_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.cp2k.CP2KOutputFormat</i> format.
<i>from_deepmd</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.deepmd.DeePMDRawFormat</i> format.
<i>from_deepmd_comp</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.deepmd.DeePMDCompFormat</i> format.
<i>from_deepmd_hdf5</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.deepmd.DeePMDHDF5Format</i> format.
<i>from_deepmd_npy</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.deepmd.DeePMDCompFormat</i> format.
<i>from_deepmd_raw</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.deepmd.DeePMDRawFormat</i> format.
<i>from_dump</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.lammps.LAMMPSDumpFormat</i> format.
<i>from_fhi_aims_md</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.fhi_aims.FhiMDFormat</i> format.
<i>from_fhi_aims_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.fhi_aims.FhiMDFormat</i> format.
<i>from_fhi_aims_scf</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.fhi_aims.FhiSCFFormat</i> format.
<i>from_finalconfig</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatAtomconfigFormat</i> format.
<i>from_gaussian_log</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.gaussian.GaussianLogFormat</i> format.

continues on next page

Table 2 – continued from previous page

<i>from_gaussian_md</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.gaussian.GaussianMDFormat</i> format.
<i>from_gro</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.gromacs.GromacsGroFormat</i> format.
<i>from_gromacs_gro</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.gromacs.GromacsGroFormat</i> format.
<i>from_lammps_dump</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.lammps.LAMMPSDumpFormat</i> format.
<i>from_lammps_lmp</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.lammps.LAMMPSLmpFormat</i> format.
<i>from_list</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.list.ListFormat</i> format.
<i>from_lmp</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.lammps.LAMMPSLmpFormat</i> format.
<i>from_mlmd</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatOutputFormat</i> format.
<i>from_mol</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.rdkit.MolFormat</i> format.
<i>from_mol_file</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.rdkit.MolFormat</i> format.
<i>from_movement</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatOutputFormat</i> format.
<i>from_outcar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPOutcarFormat</i> format.
<i>from_poscar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPPoscarFormat</i> format.
<i>from_pwmat_atomconfig</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatAtomconfigFormat</i> format.
<i>from_pwmat_finalconfig</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatAtomconfigFormat</i> format.
<i>from_pwmat_mlmd</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatOutputFormat</i> format.
<i>from_pwmat_movement</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatOutputFormat</i> format.
<i>from_pwmat_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatOutputFormat</i> format.
<i>from_pymatgen_computedstructureentry</i> (...)	Read data from <i>dpdata.plugins.pymatgen.PyMatgenCSEFormat</i> format.
<i>from_pymatgen_molecule</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pymatgen.PyMatgenMoleculeFormat</i> format.
<i>from_pymatgen_structure</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pymatgen.PyMatgenStructureFormat</i> format.
<i>from_qe_cp_traj</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.qe.QECPTrajFormat</i> format.
<i>from_qe_pw_scf</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.qe.QECPWSCFFFormat</i> format.
<i>from_quip_gap_xyz</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.xyz.QuipGapXYZFormat</i> format.
<i>from_quip_gap_xyz_file</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.xyz.QuipGapXYZFormat</i> format.

continues on next page

Table 2 – continued from previous page

<i>from_sdf</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.rdkit.SdfFormat</i> format.
<i>from_sdf_file</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.rdkit.SdfFormat</i> format.
<i>from_siesta_aiMD_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.siesta.SiestaAIMDOutputFormat</i> format.
<i>from_siesta_aimd_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.siesta.SiestaAIMDOutputFormat</i> format.
<i>from_siesta_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.siesta.SiestaOutputFormat</i> format.
<i>from_sqm_in</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.amber.SQMInFormat</i> format.
<i>from_sqm_out</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.amber.SQMOutFormat</i> format.
<i>from_vasp_contcar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPPoscarFormat</i> format.
<i>from_vasp_outcar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPOutcarFormat</i> format.
<i>from_vasp_poscar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPPoscarFormat</i> format.
<i>from_vasp_string</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPStringFormat</i> format.
<i>from_vasp_xml</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPXMLFormat</i> format.
<i>from_xml</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPXMLFormat</i> format.
<i>from_xyz</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.xyz.XYZFormat</i> format.
<i>get_nframes</i> ()	Returns number of frames in all systems
<i>pick_atom_idx</i> (idx[, nopbc])	Pick atom index
<i>to</i> (fmt, *args, **kwargs)	Dump systems to the specific format.
<i>to_abacus_lcao_md</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.
<i>to_abacus_lcao_scf</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusSCFFormat</i> format.
<i>to_abacus_md</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.
<i>to_abacus_pw_md</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.
<i>to_abacus_pw_scf</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusSCFFormat</i> format.
<i>to_abacus_scf</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusSCFFormat</i> format.
<i>to_amber_md</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.amber.AmberMDFormat</i> format.
<i>to_ase_structure</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.ase.ASEStructureFormat</i> format.
<i>to_atomconfig</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.pwmat.PwmatAtomconfigFormat</i> format.
<i>to_contcar</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.vasp.VASPPoscarFormat</i> format.

continues on next page

Table 2 – continued from previous page

<code>to_cp2k_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k.CP2KAIMDOutputFormat</code> format.
<code>to_cp2k_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k.CP2KOutputFormat</code> format.
<code>to_deepmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDRawFormat</code> format.
<code>to_deepmd_comp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDCompFormat</code> format.
<code>to_deepmd_hdf5(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDHDF5Format</code> format.
<code>to_deepmd_npy(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDCompFormat</code> format.
<code>to_deepmd_raw(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDRawFormat</code> format.
<code>to_dump(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>to_fhi_aims_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>to_fhi_aims_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>to_fhi_aims_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiSCFFormat</code> format.
<code>to_finalconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_gaussian_log(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianLogFormat</code> format.
<code>to_gaussian_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianMDFormat</code> format.
<code>to_gro(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>to_gromacs_gro(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>to_lammps_dump(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>to_lammps_lmp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>to_list(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.list.ListFormat</code> format.
<code>to_lmp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>to_mlmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_mol(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>to_mol_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>to_movement(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_outcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.

continues on next page



Table 2 – continued from previous page

<code>to_poscar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_pwmat_atomconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_pwmat_finalconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_pwmat_mlmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_movement(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pymatgen_ComputedStructureEntry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>to_pymatgen_computedstructureentry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>to_pymatgen_molecule(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenMoleculeFormat</code> format.
<code>to_pymatgen_structure(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenStructureFormat</code> format.
<code>to_qe_cp_traj(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPTrajFormat</code> format.
<code>to_qe_pw_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPWSCFFormat</code> format.
<code>to_quip_gap_xyz(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_quip_gap_xyz_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_sdf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_sdf_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_siesta_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaAIMDOutputFormat</code> format.
<code>to_siesta_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaOutputFormat</code> format.
<code>to_sqm_in(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMInFormat</code> format.
<code>to_sqm_out(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMOutFormat</code> format.
<code>to_vasp_contcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_outcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>to_vasp_poscar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_string(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPStringFormat</code> format.
<code>to_vasp_xml(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.

continues on next page

Table 2 – continued from previous page

<code>to_xml(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>to_xyz(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.XYZFormat</code> format.

<code>from_dir</code>	
<code>from_file</code>	
<code>from_fmt_obj</code>	
<code>load_systems_from_file</code>	
<code>predict</code>	
<code>to_fmt_obj</code>	

`append(*systems)`

Append systems or MultiSystems to systems

**Parameters**

`system` [System] The system to append

`check_atom_names(system)`

Make atom\_names in all systems equal, prevent inconsistent atom\_types.

`from_abacus_lcao_md(file_name, **kwargs)`

Read data from `dpdata.plugins.abacus.AbacusMDFormat` format.

`from_abacus_lcao_scf(file_name, **kwargs)`

Read data from `dpdata.plugins.abacus.AbacusSCFFormat` format.

`from_abacus_md(file_name, **kwargs)`

Read data from `dpdata.plugins.abacus.AbacusMDFormat` format.

`from_abacus_pw_md(file_name, **kwargs)`

Read data from `dpdata.plugins.abacus.AbacusMDFormat` format.

`from_abacus_pw_scf(file_name, **kwargs)`

Read data from `dpdata.plugins.abacus.AbacusSCFFormat` format.

`from_abacus_scf(file_name, **kwargs)`

Read data from `dpdata.plugins.abacus.AbacusSCFFormat` format.

`from_amber_md(file_name, **kwargs)`

Read data from `dpdata.plugins.amber.AmberMDFormat` format.

`from_ase_structure(file_name, **kwargs)`

Read data from `dpdata.plugins.ase.ASEStructureFormat` format.

`from_atomconfig(file_name, **kwargs)`

Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

`from_contcar(file_name, **kwargs)`

Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

`from_cp2k_aimd_output(file_name, **kwargs)`

Read data from `dpdata.plugins.cp2k.CP2KAIMDOutputFormat` format.

**from\_cp2k\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.cp2k.CP2KOutputFormat` format.

**from\_deepmd**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**from\_deepmd\_comp**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**from\_deepmd\_hdf5**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDHDF5Format` format.

**from\_deepmd\_npy**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**from\_deepmd\_raw**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**classmethod from\_dir**(*dir\_name*, *file\_name*, *fmt*='auto', *type\_map*=None)

**from\_dump**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**from\_fhi\_aims\_md**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_scf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**classmethod from\_file**(*file\_name*, *fmt*, **\*\*kwargs**)

**from\_finalconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_fmt\_obj**(*fmtobj*, *directory*, *labeled*=True, **\*\*kwargs**)

**from\_gaussian\_log**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.gaussian.GaussianLogFormat` format.

**from\_gaussian\_md**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.gaussian.GaussianMDFormat` format.

**from\_gro**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.gromacs.GromacsGroFormat` format.

**from\_gromacs\_gro**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.gromacs.GromacsGroFormat` format.

**from\_lammps\_dump**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**from\_lammps\_lmp**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.lammps.LAMPSLmpFormat` format.

**from\_list**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.list.ListFormat` format.

**from\_lmp**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**from\_mlmd**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_mol**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_mol\_file**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_movement**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_outcar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPOutcarFormat` format.

**from\_poscar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_pwmat\_atomconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_finalconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_mlmd**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_movement**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pymatgen\_computedstructureentry**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**from\_pymatgen\_molecule**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**from\_pymatgen\_structure**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**from\_qe\_cp\_traj**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.qe.QECPTrajFormat` format.

**from\_qe\_pw\_scf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.qe.QECPPWSCFFormat` format.

**from\_quip\_gap\_xyz**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_quip\_gap\_xyz\_file**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_sdf**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_sdf\_file**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_siesta\_aiMD\_output**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_aimd\_output**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_output**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.siesta.SiestaOutputFormat` format.

**from\_sqm\_in**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.amber.SQMInFormat` format.

**from\_sqm\_out**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.amber.SQMOutFormat` format.

**from\_vasp\_contcar**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_vasp\_outcar**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPOutcarFormat` format.

**from\_vasp\_poscar**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_vasp\_string**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPStringFormat` format.

**from\_vasp\_xml**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPXMLFormat` format.

**from\_xml**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPXMLFormat` format.

**from\_xyz**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.xyz.XYZFormat` format.

**get\_nframes**()  
 Returns number of frames in all systems

**load\_systems\_from\_file**(*file\_name=None*, *fmt=None*, **\*\*kwargs**)

**pick\_atom\_idx**(*idx*, *nopbc=None*)  
 Pick atom index

**Parameters**

- idx**: **int or list or slice** atom index
- nopbc**: **Boolean (default: None)** If nopbc is True or False, set nopbc

**Returns**

- new\_sys**: **MultiSystems** new system

**predict**(*dp*)

**to**(*fmt: str, \*args, \*\*kwargs*) → *dpdata.system.MultiSystems*

Dump systems to the specific format.

**Parameters**

**fmt** [str] format

**Returns**

**MultiSystems** self

**to\_abacus\_lcao\_md**(\*args, \*\*kwargs)

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

**to\_abacus\_lcao\_scf**(\*args, \*\*kwargs)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

**to\_abacus\_md**(\*args, \*\*kwargs)

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

**to\_abacus\_pw\_md**(\*args, \*\*kwargs)

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

**to\_abacus\_pw\_scf**(\*args, \*\*kwargs)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

**to\_abacus\_scf**(\*args, \*\*kwargs)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

**to\_amber\_md**(\*args, \*\*kwargs)

Dump data to *dpdata.plugins.amber.AmberMDFormat* format.

**to\_ase\_structure**(\*args, \*\*kwargs)

Dump data to *dpdata.plugins.ase.ASEStructureFormat* format.

**to\_atomconfig**(\*args, \*\*kwargs)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

**to\_contcar**(\*args, \*\*kwargs)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

**to\_cp2k\_aimd\_output**(\*args, \*\*kwargs)

Dump data to *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.

**to\_cp2k\_output**(\*args, \*\*kwargs)

Dump data to *dpdata.plugins.cp2k.CP2KOutputFormat* format.

**to\_deepmd**(\*args, \*\*kwargs)

Dump data to *dpdata.plugins.deepmd.DeepMDRawFormat* format.

**to\_deepmd\_comp**(\*args, \*\*kwargs)

Dump data to *dpdata.plugins.deepmd.DeepMDCompFormat* format.

**to\_deepmd\_hdf5**(\*args, \*\*kwargs)

Dump data to *dpdata.plugins.deepmd.DeepMDHDF5Format* format.

**to\_deepmd\_npy**(\*args, \*\*kwargs)

Dump data to *dpdata.plugins.deepmd.DeepMDCompFormat* format.

**to\_deepmd\_raw**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.deepmd.DeePMDRawFormat` format.

**to\_dump**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**to\_fhi\_aims\_md**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_output**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_scf**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**to\_finalconfig**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_fmt\_obj**(fmtobj, directory, \*args, \*\*kwargs)

**to\_gaussian\_log**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.gaussian.GaussianLogFormat` format.

**to\_gaussian\_md**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.gaussian.GaussianMDFormat` format.

**to\_gro**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.gromacs.GromacsGroFormat` format.

**to\_gromacs\_gro**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.gromacs.GromacsGroFormat` format.

**to\_lammps\_dump**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**to\_lammps\_lmp**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**to\_list**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.list.ListFormat` format.

**to\_lmp**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**to\_mlmd**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_mol**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.rdkit.MolFormat` format.

**to\_mol\_file**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.rdkit.MolFormat` format.

**to\_movement**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_outcar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPOutcarFormat` format.

**to\_poscar**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_pwmat\_atomconfig**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_pwmat\_finalconfig**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_pwmat\_mlmd**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_movement**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_output**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pymatgen\_ComputedStructureEntry**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_computedstructureentry**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_molecule**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**to\_pymatgen\_structure**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**to\_qe\_cp\_traj**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.qe.QECPTrajFormat` format.

**to\_qe\_pw\_scf**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.qe.QEPPWSCFFormat` format.

**to\_quip\_gap\_xyz**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**to\_quip\_gap\_xyz\_file**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**to\_sdf**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.rdkit.SdfFormat` format.

**to\_sdf\_file**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.rdkit.SdfFormat` format.

**to\_siesta\_aimd\_output**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**to\_siesta\_output**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.siesta.SiestaOutputFormat` format.

**to\_sqm\_in**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.amber.SQMInFormat` format.



`to_sqm_out(*args, **kwargs)`

Dump data to `dpdata.plugins.amber.SQMOutFormat` format.

`to_vasp_contcar(*args, **kwargs)`

Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

`to_vasp_outcar(*args, **kwargs)`

Dump data to `dpdata.plugins.vasp.VASPOutcarFormat` format.

`to_vasp_poscar(*args, **kwargs)`

Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

`to_vasp_string(*args, **kwargs)`

Dump data to `dpdata.plugins.vasp.VASPStringFormat` format.

`to_vasp_xml(*args, **kwargs)`

Dump data to `dpdata.plugins.vasp.VASPXMLFormat` format.

`to_xml(*args, **kwargs)`

Dump data to `dpdata.plugins.vasp.VASPXMLFormat` format.

`to_xyz(*args, **kwargs)`

Dump data to `dpdata.plugins.xyz.XYZFormat` format.

`class dpdata.system.System(file_name=None, fmt='auto', type_map=None, begin=0, step=1, data=None, **kwargs)`

Bases: `monty.json.MSONable`

The data System

A data System (a concept used by `deepmd-kit`) contains frames (e.g. produced by an MD simulation) that has the same number of atoms of the same type. The order of the atoms should be consistent among the frames in one System.

**For example, a water system named `d_example` has two molecules. The properties can be accessed by**

- `d_example['atom_nums']`: [2, 4]
- `d_example['atom_names']`: ['O', 'H']
- `d_example['atom_types']`: [0, 1, 1, 0, 1, 1]
- `d_example['orig']`: [0, 0, 0]
- `d_example['cells']`: a numpy array of size nframes x 3 x 3
- `d_example['coords']`: a numpy array of size nframes x natoms x 3

**It is noted that**

- The order of frames stored in `'atom_types'`, `'cells'` and `'coords'` should be consistent.
- The order of atoms in **all** frames of `'atom_types'` and `'coords'` should be consistent.

**Restrictions:**

- `d_example['orig']` is always [0, 0, 0]
- `d_example['cells'][ii]` is always lower triangular (lammmps cell tensor convention)

**Attributes**

**`formula`** Return the formula of this system, like C3H5O2

**nopbc**

*uniq\_formula* Return the `uniq_formula` of this system.

**Methods**

<i>add_atom_names</i> (atom_names)	Add atom_names that do not exist.
<i>append</i> (system)	Append a system to this system
<i>apply_pbc</i> ()	Append periodic boundary condition
<i>as_dict</i> ()	Returns data dict of System instance
<i>check_type_map</i> (type_map)	Assign atom_names to type_map if type_map is given and different from atom_names.
<i>copy</i> ()	Returns a copy of the system.
<i>dump</i> (filename[, indent])	dump .json or .yaml file
<i>extend</i> (systems)	Extend a system list to this system
<i>from_abacus_lcao_md</i> (file_name, **kwargs)	Read data from <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<i>from_abacus_lcao_scf</i> (file_name, **kwargs)	Read data from <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<i>from_abacus_md</i> (file_name, **kwargs)	Read data from <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<i>from_abacus_pw_md</i> (file_name, **kwargs)	Read data from <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<i>from_abacus_pw_scf</i> (file_name, **kwargs)	Read data from <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<i>from_abacus_scf</i> (file_name, **kwargs)	Read data from <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<i>from_amber_md</i> (file_name, **kwargs)	Read data from <code>dpdata.plugins.amber.AmberMDFormat</code> format.
<i>from_ase_structure</i> (file_name, **kwargs)	Read data from <code>dpdata.plugins.ase.ASEStructureFormat</code> format.
<i>from_atomconfig</i> (file_name, **kwargs)	Read data from <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<i>from_contcar</i> (file_name, **kwargs)	Read data from <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<i>from_cp2k_aimd_output</i> (file_name, **kwargs)	Read data from <code>dpdata.plugins.cp2k.CP2KAIMDOutputFormat</code> format.
<i>from_cp2k_output</i> (file_name, **kwargs)	Read data from <code>dpdata.plugins.cp2k.CP2KOutputFormat</code> format.
<i>from_deepmd</i> (file_name, **kwargs)	Read data from <code>dpdata.plugins.deepmd.DeepMDRawFormat</code> format.
<i>from_deepmd_comp</i> (file_name, **kwargs)	Read data from <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.
<i>from_deepmd_hdf5</i> (file_name, **kwargs)	Read data from <code>dpdata.plugins.deepmd.DeepMDHDF5Format</code> format.
<i>from_deepmd_npy</i> (file_name, **kwargs)	Read data from <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.
<i>from_deepmd_raw</i> (file_name, **kwargs)	Read data from <code>dpdata.plugins.deepmd.DeepMDRawFormat</code> format.

continues on next page

Table 3 – continued from previous page

<code>from_dict(d)</code>	<b>param d</b> Dict representation.
<code>from_dump(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>from_fhi_aims_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>from_fhi_aims_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>from_fhi_aims_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiSCFFormat</code> format.
<code>from_finalconfig(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>from_gaussian_log(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianLogFormat</code> format.
<code>from_gaussian_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianMDFormat</code> format.
<code>from_gro(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>from_gromacs_gro(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>from_lammps_dump(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>from_lammps_lmp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>from_list(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.list.ListFormat</code> format.
<code>from_lmp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>from_mlmd(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>from_mol(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>from_mol_file(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>from_movement(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>from_outcar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>from_poscar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>from_pwmat_atomconfig(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>from_pwmat_finalconfig(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>from_pwmat_mlmd(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>from_pwmat_movement(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>from_pwmat_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.

continues on next page

Table 3 – continued from previous page

<code>from_pymatgen_computedstructureentry(...)</code>	Read data from <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>from_pymatgen_molecule(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pymatgen.PyMatgenMoleculeFormat</code> format.
<code>from_pymatgen_structure(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pymatgen.PyMatgenStructureFormat</code> format.
<code>from_qe_cp_traj(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.qe.QECPTrajFormat</code> format.
<code>from_qe_pw_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.qe.QECPWSCFFFormat</code> format.
<code>from_quip_gap_xyz(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>from_quip_gap_xyz_file(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>from_sdf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>from_sdf_file(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>from_siesta_aiMD_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.siesta.SiestaAIMDOutputFormat</code> format.
<code>from_siesta_aimd_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.siesta.SiestaAIMDOutputFormat</code> format.
<code>from_siesta_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.siesta.SiestaOutputFormat</code> format.
<code>from_sqm_in(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.amber.SQMInFormat</code> format.
<code>from_sqm_out(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.amber.SQMOutFormat</code> format.
<code>from_vasp_contcar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>from_vasp_outcar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>from_vasp_poscar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>from_vasp_string(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPStringFormat</code> format.
<code>from_vasp_xml(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>from_xml(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>from_xyz(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.xyz.XYZFormat</code> format.
<code>get_atom_names()</code>	Returns name of atoms
<code>get_atom_numbs()</code>	Returns number of atoms
<code>get_atom_types()</code>	Returns type of atoms
<code>get_natoms()</code>	Returns total number of atoms in the system
<code>get_nframes()</code>	Returns number of frames in the system
<code>load(filename)</code>	rebuild System obj.

continues on next page

Table 3 – continued from previous page

<code>map_atom_types([type_map])</code>	Map the atom types of the system Parameters -- ----- type_map : dict : {"H":0,"O":1} or list ["H","C","O","N"] The map between elements and index if no map_dict is given, index will be set ac- cording to atomic number
<code>perturb(pert_num, cell_pert_fraction, ..., ...)</code>	Perturb each frame in the system randomly.
<code>pick_atom_idx(idx[, nopbc])</code>	Pick atom index
<code>pick_by_amber_mask(param, maskstr[, ...])</code>	Pick atoms by amber mask
<code>predict(dp)</code>	Predict energies and forces by deepmd-kit.
<code>remove_atom_names(atom_names)</code>	Remove atom names and all such atoms.
<code>remove_pbc([protect_layer])</code>	This method does NOT delete the definition of the cells, it (1) revises the cell to a cubic cell and ensures that the cell boundary to any atom in the system is no less than <i>protect_layer</i> (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.
<code>replicate(ncopy)</code>	Replicate the each frame in the system in 3 dimen- sions.
<code>shuffle()</code>	Shuffle frames randomly.
<code>sort_atom_names([type_map])</code>	Sort atom_names of the system and reorder atom_numbs and atom_types according to atom_names.
<code>sub_system(f_idx)</code>	Construct a subsystem from the system
<code>to(fmt, *args, **kwargs)</code>	Dump systems to the specific format.
<code>to_abacus_lcao_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus. AbacusMDFormat</code> format.
<code>to_abacus_lcao_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus. AbacusSCFFormat</code> format.
<code>to_abacus_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus. AbacusMDFormat</code> format.
<code>to_abacus_pw_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus. AbacusMDFormat</code> format.
<code>to_abacus_pw_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus. AbacusSCFFormat</code> format.
<code>to_abacus_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus. AbacusSCFFormat</code> format.
<code>to_amber_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber. AmberMDFormat</code> format.
<code>to_ase_structure(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.ase. ASEStructureFormat</code> format.
<code>to_atomconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat. PwmatAtomconfigFormat</code> format.
<code>to_contcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp. VASPPoscarFormat</code> format.
<code>to_cp2k_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k. CP2KAIMDOutputFormat</code> format.
<code>to_cp2k_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k. CP2KOutputFormat</code> format.
<code>to_deepmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd. DeePMDRawFormat</code> format.

continues on next page

Table 3 – continued from previous page

<code>to_deepmd_comp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.
<code>to_deepmd_hdf5(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDHDF5Format</code> format.
<code>to_deepmd_npy(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.
<code>to_deepmd_raw(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDRawFormat</code> format.
<code>to_dump(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMPSDumpFormat</code> format.
<code>to_fhi_aims_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>to_fhi_aims_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>to_fhi_aims_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiSCFFormat</code> format.
<code>to_finalconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_gaussian_log(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianLogFormat</code> format.
<code>to_gaussian_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianMDFormat</code> format.
<code>to_gro(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>to_gromacs_gro(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>to_json()</code>	Returns a json string representation of the MSONable object.
<code>to_lammps_dump(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMPSDumpFormat</code> format.
<code>to_lammps_lmp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMPSLmpFormat</code> format.
<code>to_list(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.list.ListFormat</code> format.
<code>to_lmp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMPSLmpFormat</code> format.
<code>to_mlmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_mol(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>to_mol_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>to_movement(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_outcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>to_poscar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_pwmat_atomconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.

continues on next page

Table 3 – continued from previous page

<code>to_pwmat_finalconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_pwmat_mlmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_movement(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pymatgen_ComputedStructureEntry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>to_pymatgen_computedstructureentry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>to_pymatgen_molecule(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenMoleculeFormat</code> format.
<code>to_pymatgen_structure(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenStructureFormat</code> format.
<code>to_qe_cp_traj(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPTrajFormat</code> format.
<code>to_qe_pw_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPWSCFFormat</code> format.
<code>to_quip_gap_xyz(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_quip_gap_xyz_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_sdf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_sdf_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_siesta_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaAIMDOutputFormat</code> format.
<code>to_siesta_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaOutputFormat</code> format.
<code>to_sqm_in(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMInFormat</code> format.
<code>to_sqm_out(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMOutFormat</code> format.
<code>to_vasp_contcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_outcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>to_vasp_poscar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_string(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPStringFormat</code> format.
<code>to_vasp_xml(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>to_xml(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>to_xyz(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.XYZFormat</code> format.
<code>unsafe_hash()</code>	Returns an hash of the current object.

continues on next page

Table 3 – continued from previous page

validate_monty(v)	pydantic Validator for MSONable pattern
-------------------	---

<b>affine_map</b>	
<b>apply_type_map</b>	
<b>from_fmt</b>	
<b>from_fmt_obj</b>	
<b>replace</b>	
<b>rot_frame_lower_triangular</b>	
<b>rot_lower_triangular</b>	
<b>sort_atom_types</b>	
<b>to_fmt_obj</b>	

**add\_atom\_names**(*atom\_names*)

Add atom\_names that do not exist.

**affine\_map**(*trans*, *f\_idx=0*)

**append**(*system*)

Append a system to this system

**Parameters**

**system** [System] The system to append

**apply\_pbc**()

Append periodic boundary condition

**apply\_type\_map**(*type\_map*)

**as\_dict**()

Returns data dict of System instance

**check\_type\_map**(*type\_map*)

Assign atom\_names to type\_map if type\_map is given and different from atom\_names.

**Parameters**

**type\_map** [list] type\_map

**copy**()

Returns a copy of the system.

**dump**(*filename*, *indent=4*)

dump .json or .yaml file

**extend**(*systems*)

Extend a system list to this system

**Parameters**

**systems** [[System1, System2, System3 ]] The list to extend

**property formula**

Return the formula of this system, like C3H5O2

**from\_abacus\_lcao\_md**(*file\_name*, *\*\*kwargs*)

Read data from `dpdata.plugins.abacus.AbacusMDFormat` format.



**from\_abacus\_lcao\_scf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.abacus.AbacusSCFFormat` format.

**from\_abacus\_md**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.abacus.AbacusMDFormat` format.

**from\_abacus\_pw\_md**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.abacus.AbacusMDFormat` format.

**from\_abacus\_pw\_scf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.abacus.AbacusSCFFormat` format.

**from\_abacus\_scf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.abacus.AbacusSCFFormat` format.

**from\_amber\_md**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.amber.AmberMDFormat` format.

**from\_ase\_structure**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.ase.ASEStructureFormat` format.

**from\_atomconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_contcar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_cp2k\_aimd\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.cp2k.CP2KAIMDOutputFormat` format.

**from\_cp2k\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.cp2k.CP2KOutputFormat` format.

**from\_deepmd**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**from\_deepmd\_comp**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**from\_deepmd\_hdf5**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDHDF5Format` format.

**from\_deepmd\_npy**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**from\_deepmd\_raw**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**from\_dump**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**from\_fhi\_aims\_md**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_scf**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**from\_finalconfig**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_fmt**(*file\_name*, *fmt*='auto', **\*\*kwargs**)

**from\_fmt\_obj**(*fmtobj*, *file\_name*, **\*\*kwargs**)

**from\_gaussian\_log**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.gaussian.GaussianLogFormat` format.

**from\_gaussian\_md**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.gaussian.GaussianMDFormat` format.

**from\_gro**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.gromacs.GromacsGroFormat` format.

**from\_gromacs\_gro**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.gromacs.GromacsGroFormat` format.

**from\_lammps\_dump**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**from\_lammps\_lmp**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**from\_list**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.list.ListFormat` format.

**from\_lmp**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**from\_mlmd**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_mol**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_mol\_file**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_movement**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_outcar**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPOutcarFormat` format.

**from\_poscar**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_pwmat\_atomconfig**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_finalconfig**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_mlm**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_movement**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pymatgen\_computedstructureentry**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**from\_pymatgen\_molecule**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**from\_pymatgen\_structure**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**from\_qe\_cp\_traj**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.qe.QECPTrajFormat` format.

**from\_qe\_pw\_scf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.qe.QECPWSCFFormat` format.

**from\_quip\_gap\_xyz**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_quip\_gap\_xyz\_file**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_sdf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_sdf\_file**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_siesta\_aiMD\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_aimd\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.siesta.SiestaOutputFormat` format.

**from\_sqm\_in**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.amber.SQMInFormat` format.

**from\_sqm\_out**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.amber.SQMOutFormat` format.

**from\_vasp\_contcar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_vasp\_outcar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPOutcarFormat` format.

**from\_vasp\_poscar**(*file\_name*, **\*\*kwargs**)

Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_vasp\_string**(*file\_name*, **\*\*kwargs**)

Read data from `dpdata.plugins.vasp.VASPStringFormat` format.

**from\_vasp\_xml**(*file\_name*, **\*\*kwargs**)

Read data from `dpdata.plugins.vasp.VASPXMLFormat` format.

**from\_xml**(*file\_name*, **\*\*kwargs**)

Read data from `dpdata.plugins.vasp.VASPXMLFormat` format.

**from\_xyz**(*file\_name*, **\*\*kwargs**)

Read data from `dpdata.plugins.xyz.XYZFormat` format.

**get\_atom\_names**()

Returns name of atoms

**get\_atom\_numbs**()

Returns number of atoms

**get\_atom\_types**()

Returns type of atoms

**get\_natoms**()

Returns total number of atoms in the system

**get\_nframes**()

Returns number of frames in the system

**static load**(*filename*)

rebuild System obj. from .json or .yaml file

**map\_atom\_types**(*type\_map=None*)

Map the atom types of the system Parameters ——— type\_map :

dict : {“H”:0,“O”:1} or list [“H”,“C”,“O”,“N”] The map between elements and index if no map\_dict is given, index will be set according to atomic number

### Returns

**new\_atom\_types** [list] The mapped atom types

**property nopbc**

**perturb**(*pert\_num*, *cell\_pert\_fraction*, *atom\_pert\_distance*, *atom\_pert\_style='normal'*)

Perturb each frame in the system randomly. The cell will be deformed randomly, and atoms will be displaced by a random distance in random direction.

### Parameters

**pert\_num** [int] Each frame in the system will make *pert\_num* copies, and all the copies will be perturbed. That means the system to be returned will contain *pert\_num* \* frame\_num of the input system.

**cell\_pert\_fraction** [float] A fraction determines how much (relatively) will cell deform. The cell of each frame is deformed by a symmetric matrix perturbed from identity. The perturbation to the diagonal part is subject to a uniform distribution in [-cell\_pert\_fraction, cell\_pert\_fraction), and the perturbation to the off-diagonal part is subject to a uniform distribution in [-0.5\*cell\_pert\_fraction, 0.5\*cell\_pert\_fraction).

**atom\_pert\_distance: float** unit: Angstrom. A distance determines how far atoms will move. Atoms will move about *atom\_pert\_distance* in random direction. The distribution of the distance atoms move is determined by *atom\_pert\_style*

**atom\_pert\_style** [str] Determines the distribution of the distance atoms move is subject to. Available options are

- **‘normal’**: the distance will be object to *chi-square distribution with 3 degrees of freedom after normalization*. The mean value of the distance is *atom\_pert\_fraction\*side\_length*
- **‘uniform’**: will generate uniformly random points in a 3D-balls with radius as *atom\_pert\_distance*. These points are treated as vector used by atoms to move. Obviously, the max length of the distance atoms move is *atom\_pert\_distance*.
- **‘const’**: The distance atoms move will be a constant *atom\_pert\_distance*.

#### Returns

**perturbed\_system** [System] The perturbed\_system. It contains *pert\_num* \* *frame\_num* of the input system frames.

**pick\_atom\_idx**(*idx*, *nopbc=None*)

Pick atom index

#### Parameters

**idx**: int or list or slice atom index

**nopbc**: Boolean (default: None) If *nopbc* is True or False, set *nopbc*

#### Returns

**new\_sys**: System new system

**pick\_by\_amber\_mask**(*param*, *maskstr*, *pass\_coords=False*, *nopbc=None*)

Pick atoms by amber mask

#### Parameters

**param**: str or parmed.Structure filename of Amber param file or parmed.Structure

**maskstr**: str Amber masks

**pass\_coords**: Boolean (default: False) If *pass\_coords* is true, the function will pass coordinates and return a MultiSystem. Otherwise, the result is coordinate-independent, and the function will return System or LabeledSystem.

**nopbc**: Boolean (default: None) If *nopbc* is True or False, set *nopbc*

**post\_funcs** = <dpdata.plugin.Plugin object>

**predict**(*dp*)

Predict energies and forces by deepmd-kit.

#### Parameters

**dp** [deepmd.DeepPot or str] The deepmd-kit potential class or the filename of the model.

#### Returns

**labeled\_sys** [LabeledSystem] The labeled system.

**remove\_atom\_names**(*atom\_names*)

Remove atom names and all such atoms. For example, you may not remove EP atoms in TIP4P/Ew water, which is not a real atom.

**remove\_pbc**(*protect\_layer=9*)

This method does NOT delete the definition of the cells, it (1) revises the cell to a cubic cell and ensures that the cell boundary to any atom in the system is no less than *protect\_layer* (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.

**Parameters**

**protect\_layer** [the protect layer between the atoms and the cell] boundary

**replace**(*initial\_atom\_type, end\_atom\_type, replace\_num*)

**replicate**(*ncopy*)

Replicate the each frame in the system in 3 dimensions. Each frame in the system will become a supercell.

**Parameters**

**ncopy** list: [4,2,3] or tuple: (4,2,3,) make *ncopy[0]* copys in x dimensions, make *ncopy[1]* copys in y dimensions, make *ncopy[2]* copys in z dimensions.

**Returns**

**tmp** [System] The system after replication.

**rot\_frame\_lower\_triangular**(*f\_idx=0*)

**rot\_lower\_triangular**()

**shuffle**()

Shuffle frames randomly.

**sort\_atom\_names**(*type\_map=None*)

Sort atom\_names of the system and reorder atom\_numbs and atom\_types according to atom\_names. If type\_map is not given, atom\_names will be sorted by alphabetical order. If type\_map is given, atom\_names will be type\_map.

**Parameters**

**type\_map** [list] type\_map

**sort\_atom\_types**()

**sub\_system**(*f\_idx*)

Construct a subsystem from the system

**Parameters**

**f\_idx** [int or index] Which frame to use in the subsystem

**Returns**

**sub\_system** [System] The subsystem

**to**(*fmt: str, \*args, \*\*kwargs*) → *dpdata.system.System*

Dump systems to the specific format.

**Parameters**

**fmt** [str] format

**Returns**

**System** self

**to\_abacus\_lcao\_md**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.abacus.AbacusMDFormat` format.

**to\_abacus\_lcao\_scf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.abacus.AbacusSCFFormat` format.

**to\_abacus\_md**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.abacus.AbacusMDFormat` format.

**to\_abacus\_pw\_md**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.abacus.AbacusMDFormat` format.

**to\_abacus\_pw\_scf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.abacus.AbacusSCFFormat` format.

**to\_abacus\_scf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.abacus.AbacusSCFFormat` format.

**to\_amber\_md**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.amber.AmberMDFormat` format.

**to\_ase\_structure**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.ase.ASEStructureFormat` format.

**to\_atomconfig**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_contcar**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_cp2k\_aimd\_output**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.cp2k.CP2KAIMDOutputFormat` format.

**to\_cp2k\_output**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.cp2k.CP2KOutputFormat` format.

**to\_deepmd**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**to\_deepmd\_comp**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**to\_deepmd\_hdf5**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDHDF5Format` format.

**to\_deepmd\_npy**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**to\_deepmd\_raw**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**to\_dump**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**to\_fhi\_aims\_md**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_output**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_scf**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**to\_finalconfig**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_fmt\_obj**(fmtobj, \*args, \*\*kwargs)

**to\_gaussian\_log**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.gaussian.GaussianLogFormat` format.

**to\_gaussian\_md**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.gaussian.GaussianMDFormat` format.

**to\_gro**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.gromacs.GromacsGroFormat` format.

**to\_gromacs\_gro**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.gromacs.GromacsGroFormat` format.

**to\_lammps\_dump**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**to\_lammps\_lmp**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**to\_list**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.list.ListFormat` format.

**to\_lmp**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**to\_mlmd**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_mol**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.rdkit.MolFormat` format.

**to\_mol\_file**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.rdkit.MolFormat` format.

**to\_movement**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_outcar**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.vasp.VASPOutcarFormat` format.

**to\_poscar**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_pwmat\_atomconfig**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_pwmat\_finalconfig**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.



**to\_pwmat\_mlmd**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_movement**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_output**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pymatgen\_ComputedStructureEntry**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_computedstructureentry**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_molecule**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**to\_pymatgen\_structure**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**to\_qe\_cp\_traj**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.qe.QECPTrajFormat` format.

**to\_qe\_pw\_scf**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.qe.QECPWSCFFFormat` format.

**to\_quip\_gap\_xyz**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**to\_quip\_gap\_xyz\_file**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**to\_sdf**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.rdkit.SdfFormat` format.

**to\_sdf\_file**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.rdkit.SdfFormat` format.

**to\_siesta\_aimd\_output**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**to\_siesta\_output**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.siesta.SiestaOutputFormat` format.

**to\_sqm\_in**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.amber.SQMInFormat` format.

**to\_sqm\_out**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.amber.SQMOutFormat` format.

**to\_vasp\_contcar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_vasp\_outcar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPOutcarFormat` format.

**to\_vasp\_poscar**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_vasp\_string**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.vasp.VASPStringFormat` format.

**to\_vasp\_xml**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.vasp.VASPXMLFormat` format.

**to\_xml**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.vasp.VASPXMLFormat` format.

**to\_xyz**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.xyz.XYZFormat` format.

**property uniq\_formula**

Return the `uniq_formula` of this system. The `uniq_formula` sort the elements in formula by names. Systems with the same `uniq_formula` can be append together.

`dpdata.system.add_format_methods()`

Add format methods to System, LabeledSystem, and MultiSystems.

## Notes

Ensure all plugins have been loaded before executing this function!

`dpdata.system.check_LabeledSystem(data)`

`dpdata.system.check_System(data)`

`dpdata.system.get_atom_perturb_vector(atom_pert_distance, atom_pert_style='normal')`

`dpdata.system.get_cell_perturb_matrix(cell_pert_fraction)`

`dpdata.system.get_cls_name(cls: object) → str`

Returns the fully qualified name of a class, such as `np.ndarray`.

### Parameters

**cls** [object] the class

### Returns

**str** the fully qualified name of a class

`dpdata.system.load_format(fmt)`

## 2.1.9 dpdata.unit module

`class dpdata.unit.Conversion(unitA, unitB, check=True)`

Bases: `abc.ABC`

**Methods**

<b>set_value</b>	
<b>value</b>	

**set\_value**(*value*)**value**()**class** dpdata.unit.**EnergyConversion**(*unitA*, *unitB*)Bases: *dpdata.unit.Conversion***Methods**

<b>set_value</b>	
<b>value</b>	

**class** dpdata.unit.**ForceConversion**(*unitA*, *unitB*)Bases: *dpdata.unit.Conversion***Methods**

<b>set_value</b>	
<b>value</b>	

**class** dpdata.unit.**LengthConversion**(*unitA*, *unitB*)Bases: *dpdata.unit.Conversion***Methods**

<b>set_value</b>	
<b>value</b>	

**class** dpdata.unit.**PressureConversion**(*unitA*, *unitB*)Bases: *dpdata.unit.Conversion***Methods**

<b>set_value</b>	
<b>value</b>	

dpdata.unit.**check\_unit**(*unit*)

### 2.1.10 dpdata.utils module

`dpdata.utils.add_atom_names(data, atom_names)`

Add atom\_names that do not exist.

`dpdata.utils.elements_index_map(elements, standard=False, inverse=False)`

`dpdata.utils.remove_pbc(system, protect_layer=9)`

`dpdata.utils.sort_atom_names(data, type_map=None)`

Sort atom\_names of the system and reorder atom\_numbs and atom\_types according to atom\_names. If type\_map is not given, atom\_names will be sorted by alphabetical order. If type\_map is given, atom\_names will be type\_map.

#### Parameters

**type\_map** [list] type\_map

`dpdata.utils.uniq_atom_names(data)`

Make the atom names uniq. For example ['O', 'H', 'O', 'H', 'O'] -> ['O', 'H']

#### Parameters

**data** [dict] data dict of *System*, *LabeledSystem*

**dpdata** is a python package for manipulating data formats of software in computational science, including DeePMD-kit, VASP, LAMMPS, GROMACS, Gaussian. dpdata only works with python 3.x.

## INSTALLATION

One can download the source code of dpdata by

```
git clone https://github.com/deepmodeling/dpdata.git dpdata
```

then use `setup.py` to install the module

```
cd dpdata  
python setup.py install
```

dpdata can also be installed via pip

```
pip3 install dpdata
```



## QUICK START

This section gives some examples on how `dpdata` works. Firstly one needs to import the module in a python 3.x compatible code.

```
import dpdata
```

The typical workflow of `dpdata` is

1. Load data from vasp or lammmps or deepmd-kit data files.
2. Manipulate data
3. Dump data to in a desired format

### 4.1 Load data

```
d_poscar = dpdata.System('POSCAR', fmt = 'vasp/poscar')
```

or let `dpdata` infer the format (`vasp/poscar`) of the file from the file name extension

```
d_poscar = dpdata.System('my.POSCAR')
```

The number of atoms, atom types, coordinates are loaded from the POSCAR and stored to a data System called `d_poscar`. A data System (a concept used by `deepmd-kit`) contains frames that has the same number of atoms of the same type. The order of the atoms should be consistent among the frames in one System. It is noted that POSCAR only contains one frame. If the multiple frames stored in, for example, a OUTCAR is wanted,

```
d_outcar = dpdata.LabeledSystem('OUTCAR')
```

The labels provided in the OUTCAR, i.e. energies, forces and virials (if any), are loaded by `LabeledSystem`. It is noted that the forces of atoms are always assumed to exist. `LabeledSystem` is a derived class of `System`.

The `System` or `LabeledSystem` can be constructed from the following file formats with the `format` key in the table passed to argument `fmt`:

Software	format	multi frames	labeled	class	format key
vasp	poscar	False	False	System	'vasp/poscar'
vasp	outcar	True	True	LabeledSystem	'vasp/outcar'
vasp	xml	True	True	LabeledSystem	'vasp/xml'
lammmps	lmp	False	False	System	'lammmps/lmp'
lammmps	dump	True	False	System	'lammmps/dump'

continues on next page

Table 1 – continued from previous page

Software	format	multi frames	labeled	class	format key
deepmd	raw	True	False	System	'deepmd/raw'
deepmd	npz	True	False	System	'deepmd/npz'
deepmd	raw	True	True	LabeledSystem	'deepmd/raw'
deepmd	npz	True	True	LabeledSystem	'deepmd/npz'
gaussian	log	False	True	LabeledSystem	'gaussian/log'
gaussian	log	True	True	LabeledSystem	'gaussian/md'
siesta	output	False	True	LabeledSystem	'siesta/output'
siesta	aimd_output	True	True	LabeledSystem	'siesta/aimd_output'
cp2k	output	False	True	LabeledSystem	'cp2k/output'
cp2k	aimd_output	True	True	LabeledSystem	'cp2k/aimd_output'
QE	log	False	True	LabeledSystem	'qe/pw/scf'
QE	log	True	False	System	'qe/cp/traj'
QE	log	True	True	LabeledSystem	'qe/cp/traj'
Fhi-aims	output	True	True	LabeledSystem	'fhi_aims/md'
Fhi-aims	output	False	True	LabeledSystem	'fhi_aims/scf'
quip/gap	xyz	True	True	MultiSystems	'quip/gap/xyz'
PWmat	atom.config	False	False	System	'pwm/atom.config'
PWmat	movement	True	True	LabeledSystem	'pwm/movement'
PWmat	OUT.MLMD	True	True	LabeledSystem	'pwm/out.mlmd'
Amber	multi	True	True	LabeledSystem	'amber/md'
Amber/sqm	sqm.out	False	False	System	'sqm/out'
Gromacs	gro	True	False	System	'gromacs/gro'
ABACUS	STRU	False	True	LabeledSystem	'abacus/scf'
ABACUS	cif	True	True	LabeledSystem	'abacus/md'
ase	structure	True	True	MultiSystems	'ase/structure'

The Class `dpdata.MultiSystems` can read data from a dir which may contains many files of different systems, or from single xyz file which contains different systems.

Use `dpdata.MultiSystems.from_dir` to read from a directory, `dpdata.MultiSystems` will walk in the directory Recursively and find all file with specific file\_name. Supports all the file formats that `dpdata.LabeledSystem` supports.

Use `dpdata.MultiSystems.from_file` to read from single file. Single-file support is available for the `quip/gap/xyz` and `ase/structure` formats.

For example, for `quip/gap xyz` files, single `.xyz` file may contain many different configurations with different atom numbers and atom type.

The following commands relating to Class `dpdata.MultiSystems` may be useful.

```
# load data

xyz_multi_systems = dpdata.MultiSystems.from_file(file_name='tests/xyz/xyz_unittest.xyz',
↳ fmt='quip/gap/xyz')
vasp_multi_systems = dpdata.MultiSystems.from_dir(dir_name='./mgal_outcar', file_name=
↳ 'OUTCAR', fmt='vasp/outcar')

# use wildcard
vasp_multi_systems = dpdata.MultiSystems.from_dir(dir_name='./mgal_outcar', file_name=
↳ '*OUTCAR', fmt='vasp/outcar')
```

(continues on next page)



(continued from previous page)

```

# print the multi_system infomation
print(xyz_multi_systems)
print(xyz_multi_systems.systems) # return a dictionaries

# print the system infomation
print(xyz_multi_systems.systems['B1C9'].data)

# dump a system's data to ./my_work_dir/B1C9_raw folder
xyz_multi_systems.systems['B1C9'].to_deepmd_raw('./my_work_dir/B1C9_raw')

# dump all systems
xyz_multi_systems.to_deepmd_raw('./my_deepmd_data/')

```

You may also use the following code to parse muti-system:

```

from dpdata import LabeledSystem, MultiSystems
from glob import glob
"""
process multi systems
"""
fs=glob('./*/OUTCAR') # remeber to change here !!!
ms=MultiSystems()
for f in fs:
    try:
        ls=LabeledSystem(f)
    except:
        print(f)
    if len(ls)>0:
        ms.append(ls)

ms.to_deepmd_raw('deepmd')
ms.to_deepmd_npy('deepmd')

```

## 4.2 Access data

These properties stored in System and LabeledSystem can be accessed by operator [] with the key of the property supplied, for example

```
coords = d_outcar['coords']
```

Available properties are (nframe: number of frames in the system, natoms: total number of atoms in the system)

key	type	dimension	are labels	description
'atom_names'	list of str	ntypes	False	The name of each atom type
'atom_numbs'	list of int	ntypes	False	The number of atoms of each atom type
'atom_types'	np.ndarray	natoms	False	Array assigning type to each atom
'cells'	np.ndarray	nframes x 3 x 3	False	The cell tensor of each frame
'coords'	np.ndarray	nframes x natoms x 3	False	The atom coordinates
'energies'	np.ndarray	nframes	True	The frame energies
'forces'	np.ndarray	nframes x natoms x 3	True	The atom forces
'virials'	np.ndarray	nframes x 3 x 3	True	The virial tensor of each frame

## 4.3 Dump data

The data stored in `System` or `LabeledSystem` can be dumped in ‘lammmps/lmp’ or ‘vasp/poscar’ format, for example:

```
d_outcar.to('lammmps/lmp', 'conf.lmp', frame_idx=0)
```

The first frames of `d_outcar` will be dumped to ‘conf.lmp’

```
d_outcar.to('vasp/poscar', 'POSCAR', frame_idx=-1)
```

The last frames of `d_outcar` will be dumped to ‘POSCAR’.

The data stored in `LabeledSystem` can be dumped to deepmd-kit raw format, for example

```
d_outcar.to('deepmd/raw', 'dpmd_raw')
```

Or a simpler command:

```
dpdata.LabeledSystem('OUTCAR').to('deepmd/raw', 'dpmd_raw')
```

Frame selection can be implemented by

```
dpdata.LabeledSystem('OUTCAR').sub_system([0,-1]).to('deepmd/raw', 'dpmd_raw')
```

by which only the first and last frames are dumped to `dpmd_raw`.

## 4.4 replicate

`dpdata` will create a super cell of the current atom configuration.

```
dpdata.System('./POSCAR').replicate((1,2,3,))
```

tuple(1,2,3) means don’t copy atom configuration in x direction, make 2 copys in y direction, make 3 copys in z direction.

## 4.5 perturb

By the following example, each frame of the original system (`dpdata.System('./POSCAR')`) is perturbed to generate three new frames. For each frame, the cell is perturbed by 5% and the atom positions are perturbed by 0.6 Angstrom. `atom_pert_style` indicates that the perturbation to the atom positions is subject to normal distribution. Other available options to `atom_pert_style` are `uniform` (uniform in a ball), and `const` (uniform on a sphere).

```
perturbed_system = dpdata.System('./POSCAR').perturb(pert_num=3,
    cell_pert_fraction=0.05,
    atom_pert_distance=0.6,
    atom_pert_style='normal')
print(perturbed_system.data)
```

## 4.6 replace

By the following example, Random 8 Hf atoms in the system will be replaced by Zr atoms with the atom position unchanged.

```
s=dpdata.System('tests/poscars/POSCAR.P42nmc',fmt='vasp/poscar')
s.replace('Hf', 'Zr', 8)
s.to_vasp_poscar('POSCAR.P42nmc.replace')
```



## BONDORDERSYSTEM

A new class `BondOrderSystem` which inherits from class `System` is introduced in `dpdata`. This new class contains information of chemical bonds and formal charges (stored in `BondOrderSystem.data['bonds']`, `BondOrderSystem.data['formal_charges']`). Now `BondOrderSystem` can only read from `.mol/.sdf` formats, because of its dependency on `rdkit` (which means `rdkit` must be installed if you want to use this function). Other formats, such as `pdb`, must be converted to `.mol/.sdf` format (maybe with software like `open babel`).

```
import dpdata
system_1 = dpdata.BondOrderSystem("tests/bond_order/CH3OH.mol", fmt="mol") # read from .
↳mol file
system_2 = dpdata.BondOrderSystem("tests/bond_order/methane.sdf", fmt="sdf") # read from .
↳.sdf file
```

In `sdf` file, all molecules must be of the same topology (i.e. conformers of the same molecular configuration). `BondOrderSystem` also supports initialize from a `rdkit.Chem.rdchem.Mol` object directly.

```
from rdkit import Chem
from rdkit.Chem import AllChem
import dpdata

mol = Chem.MolFromSmiles("CC")
mol = Chem.AddHs(mol)
AllChem.EmbedMultipleConfs(mol, 10)
system = dpdata.BondOrderSystem(rdkit_mol=mol)
```

## 5.1 Bond Order Assignment

The `BondOrderSystem` implements a more robust sanitize procedure for `rdkit Mol`, as defined in `dpdata.rdkit.sanitize.Sanitizer`. This class defines 3 level of sanitization process by: low, medium and high. (default is medium).

- low: use `rdkit.Chem.SanitizeMol()` function to sanitize molecule.
- medium: before using `rdkit`, the program will first assign formal charge of each atom to avoid inappropriate valence exceptions. However, this mode requires the rightness of the bond order information in the given molecule.
- high: the program will try to fix inappropriate bond orders in aromatic hetrocycles, phosphate, sulfate, carboxyl, nitro, nitrine, guanidine groups. If this procedure fails to sanitize the given molecule, the program will then try to call `obabel` to pre-process the `mol` and repeat the sanitization procedure. **That is to say, if you want to use this level of sanitization, please ensure ``obabel`` is installed in the environment.** According to our test, our sanitization procedure can successfully read 4852 small molecules in the `PDBBind-refined-set`. It is necessary

to point out that the in the molecule file (mol/sdf), the number of explicit hydrogens has to be correct. Thus, we recommend to use `obabel xxx -O xxx -h` to pre-process the file. The reason why we do not implement this hydrogen-adding procedure in dpdata is that we can not ensure its correctness.

```
import dpdata

for sdf_file in glob.glob("bond_order/refined-set-ligands/obabel/*sdf"):
    syst = dpdata.BondOrderSystem(sdf_file, sanitize_level='high', verbose=False)
```

## 5.2 Formal Charge Assignment

BondOrderSystem implement a method to assign formal charge for each atom based on the 8-electron rule (see below). Note that it only supports common elements in bio-system: B,C,N,O,P,S,As

```
import dpdata

syst = dpdata.BondOrderSystem("tests/bond_order/CH3NH3+.mol", fmt='mol')
print(syst.get_formal_charges()) # return the formal charge on each atom
print(syst.get_charge()) # return the total charge of the system
```

If a valence of 3 is detected on carbon, the formal charge will be assigned to -1. Because for most cases (in alkynyl anion, isonitrile, cyclopentadienyl anion), the formal charge on 3-valence carbon is -1, and this is also consistent with the 8-electron rule.

**PLUGINS**

One can follow a [simple example](#) to add their own format by creating and installing plugins. It's critical to add the `Format` class to `entry_points['dpdata.plugins']` in `setup.py`:

```
entry_points={
    'dpdata.plugins': [
        'random=dpdata_random:RandomFormat'
    ]
},
```





## INDICES AND TABLES

- genindex
- modindex
- search



## PYTHON MODULE INDEX

### d

- dpdata, 3
- dpdata.amber, 3
- dpdata.amber.mask, 3
- dpdata.amber.md, 3
- dpdata.amber.sqm, 4
- dpdata.cli, 39
- dpdata.cp2k, 4
- dpdata.cp2k.cell, 4
- dpdata.cp2k.output, 4
- dpdata.deepmd, 5
- dpdata.deepmd.comp, 5
- dpdata.deepmd.hdf5, 5
- dpdata.deepmd.raw, 6
- dpdata.fhi\_aims, 6
- dpdata.fhi\_aims.output, 6
- dpdata.format, 39
- dpdata.gaussian, 6
- dpdata.gaussian.log, 6
- dpdata.gromacs, 6
- dpdata.gromacs.gro, 6
- dpdata.lammps, 6
- dpdata.lammps.dump, 6
- dpdata.lammps.lmp, 7
- dpdata.periodic\_table, 41
- dpdata.plugin, 42
- dpdata.plugins, 7
- dpdata.plugins.abacus, 7
- dpdata.plugins.amber, 9
- dpdata.plugins.ase, 11
- dpdata.plugins.cp2k, 12
- dpdata.plugins.deepmd, 13
- dpdata.plugins.fhi\_aims, 17
- dpdata.plugins.gaussian, 18
- dpdata.plugins.gromacs, 19
- dpdata.plugins.lammps, 20
- dpdata.plugins.list, 22
- dpdata.plugins.pwmat, 23
- dpdata.plugins.pymatgen, 24
- dpdata.plugins.qe, 26
- dpdata.plugins.rdkit, 28
- dpdata.plugins.siesta, 29
- dpdata.plugins.vasp, 31
- dpdata.plugins.xyz, 34
- dpdata.pwmat, 35
- dpdata.pwmat.atomconfig, 35
- dpdata.pwmat.movement, 36
- dpdata.pymatgen, 36
- dpdata.pymatgen.molecule, 36
- dpdata.qe, 36
- dpdata.qe.scf, 36
- dpdata.qe.traj, 36
- dpdata.rdkit, 37
- dpdata.rdkit.utils, 37
- dpdata.siesta, 37
- dpdata.siesta.aiMD\_output, 37
- dpdata.siesta.output, 38
- dpdata.system, 42
- dpdata.unit, 86
- dpdata.utils, 88
- dpdata.vasp, 38
- dpdata.vasp.outcar, 38
- dpdata.vasp.poscar, 38
- dpdata.vasp.xml, 38



## A

AbacusMDFormat (class in *dpdata.plugins.abacus*), 7  
 AbacusSCFFormat (class in *dpdata.plugins.abacus*), 8  
 add\_atom\_names() (*dpdata.system.System* method), 76  
 add\_atom\_names() (in module *dpdata.utils*), 88  
 add\_format\_methods() (in module *dpdata.system*), 86  
 affine\_map() (*dpdata.system.System* method), 76  
 affine\_map\_fv() (*dpdata.system.LabeledSystem* method), 49  
 AmberMDFormat (class in *dpdata.plugins.amber*), 9  
 analyze() (in module *dpdata.vasp.xml*), 38  
 analyze\_atominfo() (in module *dpdata.vasp.xml*), 38  
 analyze\_block() (in module *dpdata.fhi\_aims.output*), 6  
 analyze\_block() (in module *dpdata.pwmat.movement*), 36  
 analyze\_block() (in module *dpdata.vasp.outcar*), 38  
 analyze\_calculation() (in module *dpdata.vasp.xml*), 38  
 append() (*dpdata.system.LabeledSystem* method), 49  
 append() (*dpdata.system.MultiSystems* method), 62  
 append() (*dpdata.system.System* method), 76  
 apply\_pbc() (*dpdata.system.System* method), 76  
 apply\_type\_map() (*dpdata.system.System* method), 76  
 as\_dict() (*dpdata.system.System* method), 76  
 ASEStructureFormat (class in *dpdata.plugins.ase*), 11

## B

box2dumpbox() (in module *dpdata.lammps.dump*), 6  
 box2lmpbox() (in module *dpdata.lammps.lmp*), 7

## C

calculated\_radius (*dpdata.periodic\_table.Element* property), 41  
 cell\_to\_low\_triangle() (in module *dpdata.cp2k.cell*), 4  
 check\_atom\_names() (*dpdata.system.MultiSystems* method), 62  
 check\_LabeledSystem() (in module *dpdata.system*), 86  
 check\_molecule\_list() (in module *dpdata.rdkit.utils*), 37

check\_name() (in module *dpdata.vasp.xml*), 38  
 check\_same\_atom() (in module *dpdata.rdkit.utils*), 37  
 check\_same\_molecule() (in module *dpdata.rdkit.utils*), 37  
 check\_System() (in module *dpdata.system*), 86  
 check\_type\_map() (*dpdata.system.System* method), 76  
 check\_unit() (in module *dpdata.unit*), 87  
 combine\_molecules() (in module *dpdata.rdkit.utils*), 37  
 Conversion (class in *dpdata.unit*), 86  
 convert() (in module *dpdata.cli*), 39  
 convert\_celldm() (in module *dpdata.qe.traj*), 36  
 copy() (*dpdata.system.System* method), 76  
 correction() (*dpdata.system.LabeledSystem* method), 49  
 covert\_dimension() (in module *dpdata.siesta.aiMD\_output*), 37  
 CP2KAIMDOutputFormat (class in *dpdata.plugins.cp2k*), 12  
 CP2KOutputFormat (class in *dpdata.plugins.cp2k*), 12  
 Cp2kSystems (class in *dpdata.cp2k.output*), 4

## D

DeePMDCompFormat (class in *dpdata.plugins.deepmd*), 13  
 DeePMDHDF5Format (class in *dpdata.plugins.deepmd*), 14  
 DeePMDRawFormat (class in *dpdata.plugins.deepmd*), 16  
 Directory (*dpdata.format.Format.MultiModes* attribute), 40  
 dpdata  
   module, 3  
 dpdata.amber  
   module, 3  
 dpdata.amber.mask  
   module, 3  
 dpdata.amber.md  
   module, 3  
 dpdata.amber.sqm  
   module, 4  
 dpdata.cli  
   module, 39

dpdata.cp2k  
  module, 4

dpdata.cp2k.cell  
  module, 4

dpdata.cp2k.output  
  module, 4

dpdata.deepmd  
  module, 5

dpdata.deepmd.comp  
  module, 5

dpdata.deepmd.hdf5  
  module, 5

dpdata.deepmd.raw  
  module, 6

dpdata.fhi\_aims  
  module, 6

dpdata.fhi\_aims.output  
  module, 6

dpdata.format  
  module, 39

dpdata.gaussian  
  module, 6

dpdata.gaussian.log  
  module, 6

dpdata.gromacs  
  module, 6

dpdata.gromacs.gro  
  module, 6

dpdata.lammps  
  module, 6

dpdata.lammps.dump  
  module, 6

dpdata.lammps.lmp  
  module, 7

dpdata.periodic\_table  
  module, 41

dpdata.plugin  
  module, 42

dpdata.plugins  
  module, 7

dpdata.plugins.abacus  
  module, 7

dpdata.plugins.amber  
  module, 9

dpdata.plugins.ase  
  module, 11

dpdata.plugins.cp2k  
  module, 12

dpdata.plugins.deepmd  
  module, 13

dpdata.plugins.fhi\_aims  
  module, 17

dpdata.plugins.gaussian  
  module, 18

dpdata.plugins.gromacs  
  module, 19

dpdata.plugins.lammps  
  module, 20

dpdata.plugins.list  
  module, 22

dpdata.plugins.pwmat  
  module, 23

dpdata.plugins.pymatgen  
  module, 24

dpdata.plugins.qe  
  module, 26

dpdata.plugins.rdkit  
  module, 28

dpdata.plugins.siesta  
  module, 29

dpdata.plugins.vasp  
  module, 31

dpdata.plugins.xyz  
  module, 34

dpdata.pwmat  
  module, 35

dpdata.pwmat.atomconfig  
  module, 35

dpdata.pwmat.movement  
  module, 36

dpdata.pymatgen  
  module, 36

dpdata.pymatgen.molecule  
  module, 36

dpdata.qe  
  module, 36

dpdata.qe.scf  
  module, 36

dpdata.qe.traj  
  module, 36

dpdata.rdkit  
  module, 37

dpdata.rdkit.utils  
  module, 37

dpdata.siesta  
  module, 37

dpdata.siesta.aiMD\_output  
  module, 37

dpdata.siesta.output  
  module, 38

dpdata.system  
  module, 42

dpdata.unit  
  module, 86

dpdata.utils  
  module, 88

dpdata.vasp  
  module, 38

dpdata.vasp.outcar  
 module, 38  
 dpdata.vasp.poscar  
 module, 38  
 dpdata.vasp.xml  
 module, 38  
 dpdata\_cli() (in module dpdata.cli), 39  
 dump() (dpdata.system.System method), 76  
 dump() (in module dpdata.deepmd.comp), 5  
 dump() (in module dpdata.deepmd.hdf5), 5  
 dump() (in module dpdata.deepmd.raw), 6  
 dumpbox2box() (in module dpdata.lammps.dump), 6

## E

Element (class in dpdata.periodic\_table), 41  
 elements\_index\_map() (in module dpdata.utils), 88  
 EnergyConversion (class in dpdata.unit), 87  
 extend() (dpdata.system.System method), 76  
 extract\_keyword() (in module dp-  
 data.siesta.aiMD\_output), 37  
 extract\_keyword() (in module dpdata.siesta.output),  
 38

## F

FhiMDFormat (class in dpdata.plugins.fhi\_aims), 17  
 FhiSCFFormat (class in dpdata.plugins.fhi\_aims), 17  
 file\_to\_system\_data() (in module dp-  
 data.gromacs.gro), 6  
 ForceConversion (class in dpdata.unit), 87  
 Format (class in dpdata.format), 39  
 Format.MultiModes (class in dpdata.format), 40  
 formula (dpdata.system.System property), 76  
 formulate\_config() (in module dpdata.vasp.xml), 38  
 from\_abacus\_lcao\_md() (dp-  
 data.system.LabeledSystem method), 49  
 from\_abacus\_lcao\_md() (dpdata.system.MultiSystems  
 method), 62  
 from\_abacus\_lcao\_md() (dpdata.system.System  
 method), 76  
 from\_abacus\_lcao\_scf() (dp-  
 data.system.LabeledSystem method), 49  
 from\_abacus\_lcao\_scf() (dp-  
 data.system.MultiSystems method), 62  
 from\_abacus\_lcao\_scf() (dpdata.system.System  
 method), 76  
 from\_abacus\_md() (dpdata.system.LabeledSystem  
 method), 49  
 from\_abacus\_md() (dpdata.system.MultiSystems  
 method), 62  
 from\_abacus\_md() (dpdata.system.System method), 77  
 from\_abacus\_pw\_md() (dpdata.system.LabeledSystem  
 method), 49  
 from\_abacus\_pw\_md() (dpdata.system.MultiSystems  
 method), 62

from\_abacus\_pw\_md() (dpdata.system.System method),  
 77  
 from\_abacus\_pw\_scf() (dpdata.system.LabeledSystem  
 method), 49  
 from\_abacus\_pw\_scf() (dpdata.system.MultiSystems  
 method), 62  
 from\_abacus\_pw\_scf() (dpdata.system.System  
 method), 77  
 from\_abacus\_scf() (dpdata.system.LabeledSystem  
 method), 49  
 from\_abacus\_scf() (dpdata.system.MultiSystems  
 method), 62  
 from\_abacus\_scf() (dpdata.system.System method), 77  
 from\_amber\_md() (dpdata.system.LabeledSystem  
 method), 49  
 from\_amber\_md() (dpdata.system.MultiSystems  
 method), 62  
 from\_amber\_md() (dpdata.system.System method), 77  
 from\_ase\_structure() (dpdata.system.LabeledSystem  
 method), 50  
 from\_ase\_structure() (dpdata.system.MultiSystems  
 method), 62  
 from\_ase\_structure() (dpdata.system.System  
 method), 77  
 from\_atomconfig() (dpdata.system.LabeledSystem  
 method), 50  
 from\_atomconfig() (dpdata.system.MultiSystems  
 method), 62  
 from\_atomconfig() (dpdata.system.System method), 77  
 from\_bond\_order\_system() (dpdata.format.Format  
 method), 40  
 from\_bond\_order\_system() (dp-  
 data.plugins.rdkit.MolFormat method), 28  
 from\_bond\_order\_system() (dp-  
 data.plugins.rdkit.SdfFormat method), 29  
 from\_contcar() (dpdata.system.LabeledSystem  
 method), 50  
 from\_contcar() (dpdata.system.MultiSystems method),  
 62  
 from\_contcar() (dpdata.system.System method), 77  
 from\_cp2k\_aimd\_output() (dp-  
 data.system.LabeledSystem method), 50  
 from\_cp2k\_aimd\_output() (dp-  
 data.system.MultiSystems method), 62  
 from\_cp2k\_aimd\_output() (dpdata.system.System  
 method), 77  
 from\_cp2k\_output() (dpdata.system.LabeledSystem  
 method), 50  
 from\_cp2k\_output() (dpdata.system.MultiSystems  
 method), 62  
 from\_cp2k\_output() (dpdata.system.System method),  
 77  
 from\_deepmd() (dpdata.system.LabeledSystem method),  
 50

from\_deepmd() (*dpdata.system.MultiSystems* method), 63  
 from\_deepmd() (*dpdata.system.System* method), 77  
 from\_deepmd\_comp() (*dpdata.system.LabeledSystem* method), 50  
 from\_deepmd\_comp() (*dpdata.system.MultiSystems* method), 63  
 from\_deepmd\_comp() (*dpdata.system.System* method), 77  
 from\_deepmd\_hdf5() (*dpdata.system.LabeledSystem* method), 50  
 from\_deepmd\_hdf5() (*dpdata.system.MultiSystems* method), 63  
 from\_deepmd\_hdf5() (*dpdata.system.System* method), 77  
 from\_deepmd\_npy() (*dpdata.system.LabeledSystem* method), 50  
 from\_deepmd\_npy() (*dpdata.system.MultiSystems* method), 63  
 from\_deepmd\_npy() (*dpdata.system.System* method), 77  
 from\_deepmd\_raw() (*dpdata.system.LabeledSystem* method), 50  
 from\_deepmd\_raw() (*dpdata.system.MultiSystems* method), 63  
 from\_deepmd\_raw() (*dpdata.system.System* method), 77  
 from\_dir() (*dpdata.system.MultiSystems* class method), 63  
 from\_dump() (*dpdata.system.LabeledSystem* method), 50  
 from\_dump() (*dpdata.system.MultiSystems* method), 63  
 from\_dump() (*dpdata.system.System* method), 77  
 from\_fhi\_aims\_md() (*dpdata.system.LabeledSystem* method), 50  
 from\_fhi\_aims\_md() (*dpdata.system.MultiSystems* method), 63  
 from\_fhi\_aims\_md() (*dpdata.system.System* method), 77  
 from\_fhi\_aims\_output() (*dpdata.system.LabeledSystem* method), 50  
 from\_fhi\_aims\_output() (*dpdata.system.MultiSystems* method), 63  
 from\_fhi\_aims\_output() (*dpdata.system.System* method), 77  
 from\_fhi\_aims\_scf() (*dpdata.system.LabeledSystem* method), 50  
 from\_fhi\_aims\_scf() (*dpdata.system.MultiSystems* method), 63  
 from\_fhi\_aims\_scf() (*dpdata.system.System* method), 77  
 from\_file() (*dpdata.system.MultiSystems* class method), 63  
 from\_finalconfig() (*dpdata.system.LabeledSystem* method), 50  
 from\_finalconfig() (*dpdata.system.MultiSystems* method), 63  
 from\_finalconfig() (*dpdata.system.System* method), 78  
 from\_fmt() (*dpdata.system.System* method), 78  
 from\_fmt\_obj() (*dpdata.system.LabeledSystem* method), 50  
 from\_fmt\_obj() (*dpdata.system.MultiSystems* method), 63  
 from\_fmt\_obj() (*dpdata.system.System* method), 78  
 from\_gaussian\_log() (*dpdata.system.LabeledSystem* method), 50  
 from\_gaussian\_log() (*dpdata.system.MultiSystems* method), 63  
 from\_gaussian\_log() (*dpdata.system.System* method), 78  
 from\_gaussian\_md() (*dpdata.system.LabeledSystem* method), 50  
 from\_gaussian\_md() (*dpdata.system.MultiSystems* method), 63  
 from\_gaussian\_md() (*dpdata.system.System* method), 78  
 from\_gro() (*dpdata.system.LabeledSystem* method), 50  
 from\_gro() (*dpdata.system.MultiSystems* method), 63  
 from\_gro() (*dpdata.system.System* method), 78  
 from\_gromacs\_gro() (*dpdata.system.LabeledSystem* method), 50  
 from\_gromacs\_gro() (*dpdata.system.MultiSystems* method), 63  
 from\_gromacs\_gro() (*dpdata.system.System* method), 78  
 from\_labeled\_system() (*dpdata.format.Format* method), 40  
 from\_labeled\_system() (*dpdata.plugins.abacus.AbacusMDFormat* method), 8  
 from\_labeled\_system() (*dpdata.plugins.abacus.AbacusSCFFormat* method), 9  
 from\_labeled\_system() (*dpdata.plugins.amber.AmberMDFormat* method), 9  
 from\_labeled\_system() (*dpdata.plugins.amber.SQMOutFormat* method), 11  
 from\_labeled\_system() (*dpdata.plugins.ase.ASEStructureFormat* method), 11  
 from\_labeled\_system() (*dpdata.plugins.cp2k.CP2KAIMDOutputFormat* method), 12  
 from\_labeled\_system() (*dpdata.plugins.cp2k.CP2KOutputFormat* method), 13  
 from\_labeled\_system() (*dp-*



*data.plugins.deepmd.DeepMDCompFormat method*), 14

`from_labeled_system()` (*dpdata.plugins.deepmd.DeepMDHDF5Format method*), 15

`from_labeled_system()` (*dpdata.plugins.deepmd.DeepMDRawFormat method*), 16

`from_labeled_system()` (*dpdata.plugins.fhi\_aims.FhiMDFormat method*), 17

`from_labeled_system()` (*dpdata.plugins.fhi\_aims.FhiSCFFormat method*), 18

`from_labeled_system()` (*dpdata.plugins.gaussian.GaussianLogFormat method*), 18

`from_labeled_system()` (*dpdata.plugins.gaussian.GaussianMDFormat method*), 19

`from_labeled_system()` (*dpdata.plugins.pwmat.PwmatOutputFormat method*), 24

`from_labeled_system()` (*dpdata.plugins.qe.QECPWSCFFormat method*), 27

`from_labeled_system()` (*dpdata.plugins.qe.QECPTrajFormat method*), 27

`from_labeled_system()` (*dpdata.plugins.siesta.SiestaAIMDOutputFormat method*), 29

`from_labeled_system()` (*dpdata.plugins.siesta.SiestaOutputFormat method*), 30

`from_labeled_system()` (*dpdata.plugins.vasp.VASPOutcarFormat method*), 31

`from_labeled_system()` (*dpdata.plugins.vasp.VASPXMLFormat method*), 33

`from_labeled_system()` (*dpdata.plugins.xyz.QuipGapXYZFormat method*), 34

`from_lammps_dump()` (*dpdata.system.LabeledSystem method*), 51

`from_lammps_dump()` (*dpdata.system.MultiSystems method*), 63

`from_lammps_dump()` (*dpdata.system.System method*), 78

`from_lammps_lmp()` (*dpdata.system.LabeledSystem method*), 51

`from_lammps_lmp()` (*dpdata.system.MultiSystems method*), 63

`from_lammps_lmp()` (*dpdata.system.System method*), 78

`from_list()` (*dpdata.system.LabeledSystem method*), 51

`from_list()` (*dpdata.system.MultiSystems method*), 63

`from_list()` (*dpdata.system.System method*), 78

`from_lmp()` (*dpdata.system.LabeledSystem method*), 51

`from_lmp()` (*dpdata.system.MultiSystems method*), 63

`from_lmp()` (*dpdata.system.System method*), 78

`from_mlmd()` (*dpdata.system.LabeledSystem method*), 51

`from_mlmd()` (*dpdata.system.MultiSystems method*), 64

`from_mlmd()` (*dpdata.system.System method*), 78

`from_mol()` (*dpdata.system.LabeledSystem method*), 51

`from_mol()` (*dpdata.system.MultiSystems method*), 64

`from_mol()` (*dpdata.system.System method*), 78

`from_mol_file()` (*dpdata.system.LabeledSystem method*), 51

`from_mol_file()` (*dpdata.system.MultiSystems method*), 64

`from_mol_file()` (*dpdata.system.System method*), 78

`from_movement()` (*dpdata.system.LabeledSystem method*), 51

`from_movement()` (*dpdata.system.MultiSystems method*), 64

`from_movement()` (*dpdata.system.System method*), 78

`from_multi_systems()` (*dpdata.format.Format method*), 40

`from_multi_systems()` (*dpdata.plugins.ase.ASEStructureFormat method*), 12

`from_multi_systems()` (*dpdata.plugins.deepmd.DeepMDHDF5Format method*), 15

`from_multi_systems()` (*dpdata.plugins.xyz.QuipGapXYZFormat method*), 34

`from_outcar()` (*dpdata.system.LabeledSystem method*), 51

`from_outcar()` (*dpdata.system.MultiSystems method*), 64

`from_outcar()` (*dpdata.system.System method*), 78

`from_poscar()` (*dpdata.system.LabeledSystem method*), 51

`from_poscar()` (*dpdata.system.MultiSystems method*), 64

`from_poscar()` (*dpdata.system.System method*), 78

`from_pwmat_atomconfig()` (*dpdata.system.LabeledSystem method*), 51

`from_pwmat_atomconfig()` (*dpdata.system.MultiSystems method*), 64

`from_pwmat_atomconfig()` (*dpdata.system.System method*), 78

`from_pwmat_finalconfig()` (*dpdata.system.LabeledSystem method*), 51

from\_pwmat\_finalconfig() (dpdata.system.MultiSystems method), 64  
 from\_pwmat\_finalconfig() (dpdata.system.System method), 78  
 from\_pwmat\_m1md() (dpdata.system.LabeledSystem method), 51  
 from\_pwmat\_m1md() (dpdata.system.MultiSystems method), 64  
 from\_pwmat\_m1md() (dpdata.system.System method), 78  
 from\_pwmat\_movement() (dpdata.system.LabeledSystem method), 51  
 from\_pwmat\_movement() (dpdata.system.MultiSystems method), 64  
 from\_pwmat\_movement() (dpdata.system.System method), 79  
 from\_pwmat\_output() (dpdata.system.LabeledSystem method), 51  
 from\_pwmat\_output() (dpdata.system.MultiSystems method), 64  
 from\_pwmat\_output() (dpdata.system.System method), 79  
 from\_pymatgen\_computedstructureentry() (dpdata.system.LabeledSystem method), 51  
 from\_pymatgen\_computedstructureentry() (dpdata.system.MultiSystems method), 64  
 from\_pymatgen\_computedstructureentry() (dpdata.system.System method), 79  
 from\_pymatgen\_molecule() (dpdata.system.LabeledSystem method), 51  
 from\_pymatgen\_molecule() (dpdata.system.MultiSystems method), 64  
 from\_pymatgen\_molecule() (dpdata.system.System method), 79  
 from\_pymatgen\_structure() (dpdata.system.LabeledSystem method), 51  
 from\_pymatgen\_structure() (dpdata.system.MultiSystems method), 64  
 from\_pymatgen\_structure() (dpdata.system.System method), 79  
 from\_qe\_cp\_traj() (dpdata.system.LabeledSystem method), 51  
 from\_qe\_cp\_traj() (dpdata.system.MultiSystems method), 64  
 from\_qe\_cp\_traj() (dpdata.system.System method), 79  
 from\_qe\_pw\_scf() (dpdata.system.LabeledSystem method), 52  
 from\_qe\_pw\_scf() (dpdata.system.MultiSystems method), 64  
 from\_qe\_pw\_scf() (dpdata.system.System method), 79  
 from\_quip\_gap\_xyz() (dpdata.system.LabeledSystem method), 52  
 from\_quip\_gap\_xyz() (dpdata.system.MultiSystems method), 64  
 from\_quip\_gap\_xyz() (dpdata.system.System method), 79  
 from\_quip\_gap\_xyz\_file() (dpdata.system.LabeledSystem method), 52  
 from\_quip\_gap\_xyz\_file() (dpdata.system.MultiSystems method), 64  
 from\_quip\_gap\_xyz\_file() (dpdata.system.System method), 79  
 from\_sdf() (dpdata.system.LabeledSystem method), 52  
 from\_sdf() (dpdata.system.MultiSystems method), 64  
 from\_sdf() (dpdata.system.System method), 79  
 from\_sdf\_file() (dpdata.system.LabeledSystem method), 52  
 from\_sdf\_file() (dpdata.system.MultiSystems method), 65  
 from\_sdf\_file() (dpdata.system.System method), 79  
 from\_siesta\_aimd\_output() (dpdata.system.LabeledSystem method), 52  
 from\_siesta\_aimd\_output() (dpdata.system.LabeledSystem method), 52  
 from\_siesta\_aimd\_output() (dpdata.system.MultiSystems method), 65  
 from\_siesta\_aimd\_output() (dpdata.system.MultiSystems method), 65  
 from\_siesta\_aimd\_output() (dpdata.system.System method), 79  
 from\_siesta\_aimd\_output() (dpdata.system.System method), 79  
 from\_siesta\_output() (dpdata.system.LabeledSystem method), 52  
 from\_siesta\_output() (dpdata.system.MultiSystems method), 65  
 from\_siesta\_output() (dpdata.system.System method), 79  
 from\_sqm\_in() (dpdata.system.LabeledSystem method), 52  
 from\_sqm\_in() (dpdata.system.MultiSystems method), 65  
 from\_sqm\_in() (dpdata.system.System method), 79  
 from\_sqm\_out() (dpdata.system.LabeledSystem method), 52  
 from\_sqm\_out() (dpdata.system.MultiSystems method), 65  
 from\_sqm\_out() (dpdata.system.System method), 79  
 from\_system() (dpdata.format.Format method), 40  
 from\_system() (dpdata.plugins.amber.AmberMDFormat method), 9  
 from\_system() (dpdata.plugins.amber.SQMOutFormat method), 11  
 from\_system() (dpdata.plugins.deepmd.DeePMDCompFormat method), 14  
 from\_system() (dpdata.plugins.deepmd.DeePMDHDF5Format method), 15  
 from\_system() (dpdata.plugins.deepmd.DeePMDRawFormat method), 16

- from\_system() (*dpdata.plugins.gromacs.GromacsGroFormat* method), 20  
 from\_system() (*dpdata.plugins.lammps.LAMMPSDumpFormat* method), 21  
 from\_system() (*dpdata.plugins.lammps.LAMMPSLmpFormat* method), 21  
 from\_system() (*dpdata.plugins.pwmat.PwmatAtomconfigFormat* method), 23  
 from\_system() (*dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* method), 25  
 from\_system() (*dpdata.plugins.qe.QECPTrajFormat* method), 27  
 from\_system() (*dpdata.plugins.siesta.SiestaAIMDOutputFormat* method), 29  
 from\_system() (*dpdata.plugins.siesta.SiestaOutputFormat* method), 30  
 from\_system() (*dpdata.plugins.vasp.VASPPoscarFormat* method), 32  
 from\_system\_data() (in module *dpdata.gromacs.gro*), 6  
 from\_system\_data() (in module *dpdata.lammps.lmp*), 7  
 from\_system\_data() (in module *dpdata.pwmat.atomconfig*), 35  
 from\_system\_data() (in module *dpdata.vasp.poscar*), 38  
 from\_vasp\_contcar() (*dpdata.system.LabeledSystem* method), 52  
 from\_vasp\_contcar() (*dpdata.system.MultiSystems* method), 65  
 from\_vasp\_contcar() (*dpdata.system.System* method), 79  
 from\_vasp\_outcar() (*dpdata.system.LabeledSystem* method), 52  
 from\_vasp\_outcar() (*dpdata.system.MultiSystems* method), 65  
 from\_vasp\_outcar() (*dpdata.system.System* method), 79  
 from\_vasp\_poscar() (*dpdata.system.LabeledSystem* method), 52  
 from\_vasp\_poscar() (*dpdata.system.MultiSystems* method), 65  
 from\_vasp\_poscar() (*dpdata.system.System* method), 79  
 from\_vasp\_string() (*dpdata.system.LabeledSystem* method), 52  
 from\_vasp\_string() (*dpdata.system.MultiSystems* method), 65  
 from\_vasp\_string() (*dpdata.system.System* method), 80  
 from\_vasp\_xml() (*dpdata.system.LabeledSystem* method), 52  
 from\_vasp\_xml() (*dpdata.system.MultiSystems* method), 65  
 from\_vasp\_xml() (*dpdata.system.System* method), 80  
 from\_xml() (*dpdata.system.LabeledSystem* method), 52  
 from\_xml() (*dpdata.system.MultiSystems* method), 65  
 from\_xml() (*dpdata.system.System* method), 80  
 from\_xyz() (*dpdata.system.LabeledSystem* method), 52  
 from\_xyz() (*dpdata.system.MultiSystems* method), 65  
 from\_xyz() (*dpdata.system.System* method), 80  
 from\_Z() (*dpdata.periodic\_table.Element* class method), 41
- ## G
- GaussianLogFormat (class in *dpdata.plugins.gaussian*), 18  
 GaussianMDFormat (class in *dpdata.plugins.gaussian*), 18  
 get\_aiMD\_frame() (in module *dpdata.siesta.aiMD\_output*), 37  
 get\_atom\_name() (in module *dpdata.siesta.aiMD\_output*), 37  
 get\_atom\_name() (in module *dpdata.siesta.output*), 38  
 get\_atom\_names() (*dpdata.system.System* method), 80  
 get\_atom\_nums() (*dpdata.system.System* method), 80  
 get\_atom\_nums() (in module *dpdata.siesta.aiMD\_output*), 37  
 get\_atom\_nums() (in module *dpdata.siesta.output*), 38  
 get\_atom\_perturb\_vector() (in module *dpdata.system*), 86  
 get\_atom\_types() (*dpdata.system.System* method), 80  
 get\_atom\_types() (in module *dpdata.siesta.aiMD\_output*), 37  
 get\_atom\_types() (in module *dpdata.siesta.output*), 38  
 get\_atoms() (in module *dpdata.lammps.lmp*), 7  
 get\_atype() (in module *dpdata.lammps.dump*), 7  
 get\_atype() (in module *dpdata.lammps.lmp*), 7  
 get\_block() (in module *dpdata.qe.scf*), 36  
 get\_cell() (in module *dpdata.qe.scf*), 36  
 get\_cell\_perturb\_matrix() (in module *dpdata.system*), 86  
 get\_cls\_name() (in module *dpdata.system*), 86  
 get\_coords() (in module *dpdata.qe.scf*), 36  
 get\_coordtype\_and\_scalefactor() (in module *dpdata.lammps.dump*), 7  
 get\_dumpbox() (in module *dpdata.lammps.dump*), 7  
 get\_energy() (in module *dpdata.qe.scf*), 36  
 get\_fhi\_aims\_block() (in module *dpdata.fhi\_aims.output*), 6  
 get\_force() (in module *dpdata.qe.scf*), 36  
 get\_formats() (*dpdata.format.Format* static method), 40  
 get\_frame() (in module *dpdata.qe.scf*), 36  
 get\_frames() (in module *dpdata.cp2k.output*), 5  
 get\_frames() (in module *dpdata.fhi\_aims.output*), 6  
 get\_frames() (in module *dpdata.pwmat.movement*), 36  
 get\_frames() (in module *dpdata.vasp.outcar*), 38

- [get\\_from\\_methods\(\)](#) (*dpdata.format.Format* static method), 40  
[get\\_info\(\)](#) (in module *dpdata.fhi\_aims.output*), 6  
[get\\_lmpbox\(\)](#) (in module *dpdata.lammps.lmp*), 7  
[get\\_log\\_block\\_generator\(\)](#) (*dpdata.cp2k.output.Cp2kSystems* method), 4  
[get\\_movement\\_block\(\)](#) (in module *dpdata.pwmat.movement*), 36  
[get\\_natoms\(\)](#) (*dpdata.system.System* method), 80  
[get\\_natoms\(\)](#) (in module *dpdata.lammps.dump*), 7  
[get\\_natoms\(\)](#) (in module *dpdata.lammps.lmp*), 7  
[get\\_natoms\\_vec\(\)](#) (in module *dpdata.lammps.dump*), 7  
[get\\_natoms\\_vec\(\)](#) (in module *dpdata.lammps.lmp*), 7  
[get\\_natomtypes\(\)](#) (in module *dpdata.lammps.dump*), 7  
[get\\_natomtypes\(\)](#) (in module *dpdata.lammps.lmp*), 7  
[get\\_nframes\(\)](#) (*dpdata.system.MultiSystems* method), 65  
[get\\_nframes\(\)](#) (*dpdata.system.System* method), 80  
[get\\_outcar\\_block\(\)](#) (in module *dpdata.vasp.outcar*), 38  
[get\\_plugin\(\)](#) (*dpdata.plugin.Plugin* method), 42  
[get\\_posi\(\)](#) (in module *dpdata.lammps.lmp*), 7  
[get\\_single\\_line\\_tail\(\)](#) (in module *dpdata.siesta.aiMD\_output*), 37  
[get\\_single\\_line\\_tail\(\)](#) (in module *dpdata.siesta.output*), 38  
[get\\_stress\(\)](#) (in module *dpdata.qe.scf*), 36  
[get\\_to\\_methods\(\)](#) (*dpdata.format.Format* static method), 40  
[get\\_varray\(\)](#) (in module *dpdata.vasp.xml*), 38  
[get\\_virial\(\)](#) (in module *dpdata.siesta.aiMD\_output*), 37  
[get\\_virial\(\)](#) (in module *dpdata.siesta.output*), 38  
[get\\_xyz\\_block\\_generator\(\)](#) (*dpdata.cp2k.output.Cp2kSystems* method), 4  
[GromacsGroFormat](#) (class in *dpdata.plugins.gromacs*), 19
- ## H
- [handle\\_single\\_log\\_frame\(\)](#) (*dpdata.cp2k.output.Cp2kSystems* method), 4  
[handle\\_single\\_xyz\\_frame\(\)](#) (*dpdata.cp2k.output.Cp2kSystems* method), 4  
[has\\_virial\(\)](#) (*dpdata.system.LabeledSystem* method), 52
- ## L
- [LabeledSystem](#) (class in *dpdata.system*), 42  
[LAMMPSDumpFormat](#) (class in *dpdata.plugins.lammps*), 20  
[LAMPSLmpFormat](#) (class in *dpdata.plugins.lammps*), 21  
[LengthConversion](#) (class in *dpdata.unit*), 87  
[ListFormat](#) (class in *dpdata.plugins.list*), 22  
[lmpbox2box\(\)](#) (in module *dpdata.lammps.lmp*), 7  
[load\(\)](#) (*dpdata.system.System* static method), 80  
[load\\_atom\\_names\(\)](#) (in module *dpdata.qe.traj*), 36  
[load\\_atom\\_types\(\)](#) (in module *dpdata.qe.traj*), 36  
[load\\_block\(\)](#) (in module *dpdata.qe.traj*), 36  
[load\\_cell\\_parameters\(\)](#) (in module *dpdata.qe.traj*), 36  
[load\\_celldm\(\)](#) (in module *dpdata.qe.traj*), 36  
[load\\_data\(\)](#) (in module *dpdata.qe.traj*), 36  
[load\\_energy\(\)](#) (in module *dpdata.qe.traj*), 37  
[load\\_file\(\)](#) (in module *dpdata.lammps.dump*), 7  
[load\\_format\(\)](#) (in module *dpdata.system*), 86  
[load\\_key\(\)](#) (in module *dpdata.qe.traj*), 37  
[load\\_param\\_file\(\)](#) (in module *dpdata.amber.mask*), 3  
[load\\_param\\_file\(\)](#) (in module *dpdata.qe.traj*), 37  
[load\\_systems\\_from\\_file\(\)](#) (*dpdata.system.MultiSystems* method), 65  
[load\\_type\(\)](#) (in module *dpdata.deepmd.raw*), 6
- ## M
- [make\\_sqm\\_in\(\)](#) (in module *dpdata.amber.sqm*), 4  
[map\\_atom\\_types\(\)](#) (*dpdata.system.System* method), 80  
[mass](#) (*dpdata.periodic\_table.Element* property), 41  
[module](#)  
     *dpdata*, 3  
     *dpdata.amber*, 3  
     *dpdata.amber.mask*, 3  
     *dpdata.amber.md*, 3  
     *dpdata.amber.sqm*, 4  
     *dpdata.cli*, 39  
     *dpdata.cp2k*, 4  
     *dpdata.cp2k.cell*, 4  
     *dpdata.cp2k.output*, 4  
     *dpdata.deepmd*, 5  
     *dpdata.deepmd.comp*, 5  
     *dpdata.deepmd.hdf5*, 5  
     *dpdata.deepmd.raw*, 6  
     *dpdata.fhi\_aims*, 6  
     *dpdata.fhi\_aims.output*, 6  
     *dpdata.format*, 39  
     *dpdata.gaussian*, 6  
     *dpdata.gaussian.log*, 6  
     *dpdata.gromacs*, 6  
     *dpdata.gromacs.gro*, 6  
     *dpdata.lammps*, 6  
     *dpdata.lammps.dump*, 6  
     *dpdata.lammps.lmp*, 7  
     *dpdata.periodic\_table*, 41  
     *dpdata.plugin*, 42  
     *dpdata.plugins*, 7  
     *dpdata.plugins.abacus*, 7

- dpdata.plugins.amber, 9
  - dpdata.plugins.ase, 11
  - dpdata.plugins.cp2k, 12
  - dpdata.plugins.deepmd, 13
  - dpdata.plugins.fhi\_aims, 17
  - dpdata.plugins.gaussian, 18
  - dpdata.plugins.gromacs, 19
  - dpdata.plugins.lammps, 20
  - dpdata.plugins.list, 22
  - dpdata.plugins.pwmat, 23
  - dpdata.plugins.pymatgen, 24
  - dpdata.plugins.qe, 26
  - dpdata.plugins.rdkit, 28
  - dpdata.plugins.siesta, 29
  - dpdata.plugins.vasp, 31
  - dpdata.plugins.xyz, 34
  - dpdata.pwmat, 35
  - dpdata.pwmat.atomconfig, 35
  - dpdata.pwmat.movement, 36
  - dpdata.pymatgen, 36
  - dpdata.pymatgen.molecule, 36
  - dpdata.qe, 36
  - dpdata.qe.scf, 36
  - dpdata.qe.traj, 36
  - dpdata.rdkit, 37
  - dpdata.rdkit.utils, 37
  - dpdata.siesta, 37
  - dpdata.siesta.aiMD\_output, 37
  - dpdata.siesta.output, 38
  - dpdata.system, 42
  - dpdata.unit, 86
  - dpdata.utils, 88
  - dpdata.vasp, 38
  - dpdata.vasp.outcar, 38
  - dpdata.vasp.poscar, 38
  - dpdata.vasp.xml, 38
  - mol\_to\_system\_data() (in module dpdata.rdkit.utils), 37
  - MolFormat (class in dpdata.plugins.rdkit), 28
  - MultiMode (dpdata.format.Format attribute), 40
  - MultiMode (dpdata.plugins.deepmd.DeepMDCompFormat attribute), 14
  - MultiMode (dpdata.plugins.deepmd.DeepMDRawFormat attribute), 16
  - MultiSystems (class in dpdata.system), 56
- ## N
- name (dpdata.periodic\_table.Element property), 41
  - nopbc (dpdata.system.System property), 80
  - NotImplemented (dpdata.format.Format.MultiModes attribute), 40
- ## O
- obtain\_frame() (in module dpdata.siesta.output), 38
  - obtain\_nframe() (in module dpdata.siesta.aiMD\_output), 37
- ## P
- parse\_sqm\_out() (in module dpdata.amber.sqm), 4
  - perturb() (dpdata.system.System method), 80
  - pick\_atom\_idx() (dpdata.system.LabeledSystem method), 52
  - pick\_atom\_idx() (dpdata.system.MultiSystems method), 65
  - pick\_atom\_idx() (dpdata.system.System method), 81
  - pick\_by\_amber\_mask() (dpdata.system.System method), 81
  - pick\_by\_amber\_mask() (in module dpdata.amber.mask), 3
  - Plugin (class in dpdata.plugin), 42
  - post() (dpdata.format.Format static method), 40
  - post\_funcs (dpdata.system.LabeledSystem attribute), 53
  - post\_funcs (dpdata.system.System attribute), 81
  - predict() (dpdata.system.MultiSystems method), 65
  - predict() (dpdata.system.System method), 81
  - PressureConversion (class in dpdata.unit), 87
  - PwmatAtomconfigFormat (class in dpdata.plugins.pwmat), 23
  - PwmatOutputFormat (class in dpdata.plugins.pwmat), 23
  - PyMatgenCSEFormat (class in dpdata.plugins.pymatgen), 24
  - PyMatgenMoleculeFormat (class in dpdata.plugins.pymatgen), 25
  - PyMatgenStructureFormat (class in dpdata.plugins.pymatgen), 26
- ## Q
- QECPWPSCFFFormat (class in dpdata.plugins.qe), 26
  - QECPTrajFormat (class in dpdata.plugins.qe), 27
  - QuipGapXYZFormat (class in dpdata.plugins.xyz), 34
- ## R
- radius (dpdata.periodic\_table.Element property), 41
  - read\_amber\_traj() (in module dpdata.amber.md), 3
  - register() (dpdata.format.Format static method), 40
  - register() (dpdata.plugin.Plugin method), 42
  - register\_from() (dpdata.format.Format static method), 41
  - register\_to() (dpdata.format.Format static method), 41
  - remove\_atom\_names() (dpdata.system.System method), 81
  - remove\_pbc() (dpdata.system.System method), 81
  - remove\_pbc() (in module dpdata.utils), 88
  - replace() (dpdata.system.System method), 82
  - replicate() (dpdata.system.System method), 82

- rot\_frame\_lower\_triangular() (*dpdata.system.LabeledSystem* method), 53
- rot\_frame\_lower\_triangular() (*dpdata.system.System* method), 82
- rot\_lower\_triangular() (*dpdata.system.LabeledSystem* method), 53
- rot\_lower\_triangular() (*dpdata.system.System* method), 82
- ## S
- safe\_get\_posi() (in module *dpdata.lammps.dump*), 7
- SdfFormat (class in *dpdata.plugins.rdkit*), 28
- set\_value() (*dpdata.unit.Conversion* method), 87
- shuffle() (*dpdata.system.LabeledSystem* method), 53
- shuffle() (*dpdata.system.System* method), 82
- SiestaAIMDOutputFormat (class in *dpdata.plugins.siesta*), 29
- SiestaOutputFormat (class in *dpdata.plugins.siesta*), 30
- sort\_atom\_names() (*dpdata.system.System* method), 82
- sort\_atom\_names() (in module *dpdata.utils*), 88
- sort\_atom\_types() (*dpdata.system.LabeledSystem* method), 53
- sort\_atom\_types() (*dpdata.system.System* method), 82
- split\_traj() (in module *dpdata.lammps.dump*), 7
- SQMInFormat (class in *dpdata.plugins.amber*), 10
- SQMOutFormat (class in *dpdata.plugins.amber*), 10
- sub\_system() (*dpdata.system.LabeledSystem* method), 53
- sub\_system() (*dpdata.system.System* method), 82
- System (class in *dpdata.system*), 69
- system\_data() (in module *dpdata.lammps.dump*), 7
- system\_data() (in module *dpdata.lammps.lmp*), 7
- system\_data\_to\_mol() (in module *dpdata.rdkit.utils*), 37
- system\_info() (in module *dpdata.pwmat.movement*), 36
- system\_info() (in module *dpdata.vasp.outcar*), 38
- ## T
- to() (*dpdata.system.MultiSystems* method), 66
- to() (*dpdata.system.System* method), 82
- to\_abacus\_lcao\_md() (*dpdata.system.LabeledSystem* method), 53
- to\_abacus\_lcao\_md() (*dpdata.system.MultiSystems* method), 66
- to\_abacus\_lcao\_md() (*dpdata.system.System* method), 82
- to\_abacus\_lcao\_scf() (*dpdata.system.LabeledSystem* method), 53
- to\_abacus\_lcao\_scf() (*dpdata.system.MultiSystems* method), 66
- to\_abacus\_lcao\_scf() (*dpdata.system.System* method), 83
- to\_abacus\_md() (*dpdata.system.LabeledSystem* method), 53
- to\_abacus\_md() (*dpdata.system.MultiSystems* method), 66
- to\_abacus\_md() (*dpdata.system.System* method), 83
- to\_abacus\_pw\_md() (*dpdata.system.LabeledSystem* method), 53
- to\_abacus\_pw\_md() (*dpdata.system.MultiSystems* method), 66
- to\_abacus\_pw\_md() (*dpdata.system.System* method), 83
- to\_abacus\_pw\_scf() (*dpdata.system.LabeledSystem* method), 53
- to\_abacus\_pw\_scf() (*dpdata.system.MultiSystems* method), 66
- to\_abacus\_pw\_scf() (*dpdata.system.System* method), 83
- to\_abacus\_scf() (*dpdata.system.LabeledSystem* method), 53
- to\_abacus\_scf() (*dpdata.system.MultiSystems* method), 66
- to\_abacus\_scf() (*dpdata.system.System* method), 83
- to\_amber\_md() (*dpdata.system.LabeledSystem* method), 53
- to\_amber\_md() (*dpdata.system.MultiSystems* method), 66
- to\_amber\_md() (*dpdata.system.System* method), 83
- to\_ase\_structure() (*dpdata.system.LabeledSystem* method), 53
- to\_ase\_structure() (*dpdata.system.MultiSystems* method), 66
- to\_ase\_structure() (*dpdata.system.System* method), 83
- to\_atomconfig() (*dpdata.system.LabeledSystem* method), 53
- to\_atomconfig() (*dpdata.system.MultiSystems* method), 66
- to\_atomconfig() (*dpdata.system.System* method), 83
- to\_bond\_order\_system() (*dpdata.format.Format* method), 41
- to\_bond\_order\_system() (*dpdata.plugins.rdkit.MolFormat* method), 28
- to\_bond\_order\_system() (*dpdata.plugins.rdkit.SdfFormat* method), 29
- to\_contcar() (*dpdata.system.LabeledSystem* method), 53
- to\_contcar() (*dpdata.system.MultiSystems* method), 66
- to\_contcar() (*dpdata.system.System* method), 83
- to\_cp2k\_aimd\_output() (*dpdata.system.LabeledSystem* method), 54
- to\_cp2k\_aimd\_output() (*dpdata.system.MultiSystems* method), 66
- to\_cp2k\_aimd\_output() (*dpdata.system.System* method), 83
- to\_cp2k\_output() (*dpdata.system.LabeledSystem*

- method), 54
- to\_cp2k\_output() (dpdata.system.MultiSystems method), 66
- to\_cp2k\_output() (dpdata.system.System method), 83
- to\_deepmd() (dpdata.system.LabeledSystem method), 54
- to\_deepmd() (dpdata.system.MultiSystems method), 66
- to\_deepmd() (dpdata.system.System method), 83
- to\_deepmd\_comp() (dpdata.system.LabeledSystem method), 54
- to\_deepmd\_comp() (dpdata.system.MultiSystems method), 66
- to\_deepmd\_comp() (dpdata.system.System method), 83
- to\_deepmd\_hdf5() (dpdata.system.LabeledSystem method), 54
- to\_deepmd\_hdf5() (dpdata.system.MultiSystems method), 66
- to\_deepmd\_hdf5() (dpdata.system.System method), 83
- to\_deepmd\_npy() (dpdata.system.LabeledSystem method), 54
- to\_deepmd\_npy() (dpdata.system.MultiSystems method), 66
- to\_deepmd\_npy() (dpdata.system.System method), 83
- to\_deepmd\_raw() (dpdata.system.LabeledSystem method), 54
- to\_deepmd\_raw() (dpdata.system.MultiSystems method), 66
- to\_deepmd\_raw() (dpdata.system.System method), 83
- to\_dump() (dpdata.system.LabeledSystem method), 54
- to\_dump() (dpdata.system.MultiSystems method), 67
- to\_dump() (dpdata.system.System method), 83
- to\_fhi\_aims\_md() (dpdata.system.LabeledSystem method), 54
- to\_fhi\_aims\_md() (dpdata.system.MultiSystems method), 67
- to\_fhi\_aims\_md() (dpdata.system.System method), 83
- to\_fhi\_aims\_output() (dpdata.system.LabeledSystem method), 54
- to\_fhi\_aims\_output() (dpdata.system.MultiSystems method), 67
- to\_fhi\_aims\_output() (dpdata.system.System method), 83
- to\_fhi\_aims\_scf() (dpdata.system.LabeledSystem method), 54
- to\_fhi\_aims\_scf() (dpdata.system.MultiSystems method), 67
- to\_fhi\_aims\_scf() (dpdata.system.System method), 84
- to\_finalconfig() (dpdata.system.LabeledSystem method), 54
- to\_finalconfig() (dpdata.system.MultiSystems method), 67
- to\_finalconfig() (dpdata.system.System method), 84
- to\_fmt\_obj() (dpdata.system.LabeledSystem method), 54
- to\_fmt\_obj() (dpdata.system.MultiSystems method), 67
- to\_fmt\_obj() (dpdata.system.System method), 84
- to\_gaussian\_log() (dpdata.system.LabeledSystem method), 54
- to\_gaussian\_log() (dpdata.system.MultiSystems method), 67
- to\_gaussian\_log() (dpdata.system.System method), 84
- to\_gaussian\_md() (dpdata.system.LabeledSystem method), 54
- to\_gaussian\_md() (dpdata.system.MultiSystems method), 67
- to\_gaussian\_md() (dpdata.system.System method), 84
- to\_gro() (dpdata.system.LabeledSystem method), 54
- to\_gro() (dpdata.system.MultiSystems method), 67
- to\_gro() (dpdata.system.System method), 84
- to\_gromacs\_gro() (dpdata.system.LabeledSystem method), 54
- to\_gromacs\_gro() (dpdata.system.MultiSystems method), 67
- to\_gromacs\_gro() (dpdata.system.System method), 84
- to\_labeled\_system() (dpdata.format.Format method), 41
- to\_labeled\_system() (dpdata.plugins.ase.ASEStructureFormat method), 12
- to\_labeled\_system() (dpdata.plugins.pymatgen.PyMatgenCSEFormat method), 25
- to\_lammps\_dump() (dpdata.system.LabeledSystem method), 54
- to\_lammps\_dump() (dpdata.system.MultiSystems method), 67
- to\_lammps\_dump() (dpdata.system.System method), 84
- to\_lammps\_lmp() (dpdata.system.LabeledSystem method), 54
- to\_lammps\_lmp() (dpdata.system.MultiSystems method), 67
- to\_lammps\_lmp() (dpdata.system.System method), 84
- to\_list() (dpdata.system.LabeledSystem method), 54
- to\_list() (dpdata.system.MultiSystems method), 67
- to\_list() (dpdata.system.System method), 84
- to\_lmp() (dpdata.system.LabeledSystem method), 55
- to\_lmp() (dpdata.system.MultiSystems method), 67
- to\_lmp() (dpdata.system.System method), 84
- to\_mlmd() (dpdata.system.LabeledSystem method), 55
- to\_mlmd() (dpdata.system.MultiSystems method), 67
- to\_mlmd() (dpdata.system.System method), 84
- to\_mol() (dpdata.system.LabeledSystem method), 55
- to\_mol() (dpdata.system.MultiSystems method), 67
- to\_mol() (dpdata.system.System method), 84
- to\_mol\_file() (dpdata.system.LabeledSystem method), 55
- to\_mol\_file() (dpdata.system.MultiSystems method), 67

- to\_mol\_file() (*dpdata.system.System* method), 84
- to\_movement() (*dpdata.system.LabeledSystem* method), 55
- to\_movement() (*dpdata.system.MultiSystems* method), 67
- to\_movement() (*dpdata.system.System* method), 84
- to\_multi\_systems() (*dpdata.format.Format* method), 41
- to\_multi\_systems() (*dpdata.plugins.deepmd.DeepMDHDF5Format* method), 15
- to\_outcar() (*dpdata.system.LabeledSystem* method), 55
- to\_outcar() (*dpdata.system.MultiSystems* method), 67
- to\_outcar() (*dpdata.system.System* method), 84
- to\_poscar() (*dpdata.system.LabeledSystem* method), 55
- to\_poscar() (*dpdata.system.MultiSystems* method), 67
- to\_poscar() (*dpdata.system.System* method), 84
- to\_pwmat\_atomconfig() (*dpdata.system.LabeledSystem* method), 55
- to\_pwmat\_atomconfig() (*dpdata.system.MultiSystems* method), 68
- to\_pwmat\_atomconfig() (*dpdata.system.System* method), 84
- to\_pwmat\_finalconfig() (*dpdata.system.LabeledSystem* method), 55
- to\_pwmat\_finalconfig() (*dpdata.system.MultiSystems* method), 68
- to\_pwmat\_finalconfig() (*dpdata.system.System* method), 84
- to\_pwmat\_mlmd() (*dpdata.system.LabeledSystem* method), 55
- to\_pwmat\_mlmd() (*dpdata.system.MultiSystems* method), 68
- to\_pwmat\_mlmd() (*dpdata.system.System* method), 84
- to\_pwmat\_movement() (*dpdata.system.LabeledSystem* method), 55
- to\_pwmat\_movement() (*dpdata.system.MultiSystems* method), 68
- to\_pwmat\_movement() (*dpdata.system.System* method), 85
- to\_pwmat\_output() (*dpdata.system.LabeledSystem* method), 55
- to\_pwmat\_output() (*dpdata.system.MultiSystems* method), 68
- to\_pwmat\_output() (*dpdata.system.System* method), 85
- to\_pymatgen\_ComputedStructureEntry() (*dpdata.system.LabeledSystem* method), 55
- to\_pymatgen\_computedstructureentry() (*dpdata.system.LabeledSystem* method), 55
- to\_pymatgen\_ComputedStructureEntry() (*dpdata.system.MultiSystems* method), 68
- to\_pymatgen\_computedstructureentry() (*dpdata.system.MultiSystems* method), 68
- to\_pymatgen\_ComputedStructureEntry() (*dpdata.system.System* method), 85
- to\_pymatgen\_computedstructureentry() (*dpdata.system.System* method), 85
- to\_pymatgen\_molecule() (*dpdata.system.LabeledSystem* method), 55
- to\_pymatgen\_molecule() (*dpdata.system.MultiSystems* method), 68
- to\_pymatgen\_molecule() (*dpdata.system.System* method), 85
- to\_pymatgen\_structure() (*dpdata.system.LabeledSystem* method), 55
- to\_pymatgen\_structure() (*dpdata.system.MultiSystems* method), 68
- to\_pymatgen\_structure() (*dpdata.system.System* method), 85
- to\_qe\_cp\_traj() (*dpdata.system.LabeledSystem* method), 55
- to\_qe\_cp\_traj() (*dpdata.system.MultiSystems* method), 68
- to\_qe\_cp\_traj() (*dpdata.system.System* method), 85
- to\_qe\_pw\_scf() (*dpdata.system.LabeledSystem* method), 55
- to\_qe\_pw\_scf() (*dpdata.system.MultiSystems* method), 68
- to\_qe\_pw\_scf() (*dpdata.system.System* method), 85
- to\_quip\_gap\_xyz() (*dpdata.system.LabeledSystem* method), 55
- to\_quip\_gap\_xyz() (*dpdata.system.MultiSystems* method), 68
- to\_quip\_gap\_xyz() (*dpdata.system.System* method), 85
- to\_quip\_gap\_xyz\_file() (*dpdata.system.LabeledSystem* method), 56
- to\_quip\_gap\_xyz\_file() (*dpdata.system.MultiSystems* method), 68
- to\_quip\_gap\_xyz\_file() (*dpdata.system.System* method), 85
- to\_sdf() (*dpdata.system.LabeledSystem* method), 56
- to\_sdf() (*dpdata.system.MultiSystems* method), 68
- to\_sdf() (*dpdata.system.System* method), 85
- to\_sdf\_file() (*dpdata.system.LabeledSystem* method), 56
- to\_sdf\_file() (*dpdata.system.MultiSystems* method), 68
- to\_sdf\_file() (*dpdata.system.System* method), 85
- to\_siesta\_aimd\_output() (*dpdata.system.LabeledSystem* method), 56
- to\_siesta\_aimd\_output() (*dpdata.system.MultiSystems* method), 68
- to\_siesta\_aimd\_output() (*dpdata.system.System* method), 85
- to\_siesta\_output() (*dpdata.system.LabeledSystem* method), 56



to\_siesta\_output() (*dpdata.system.MultiSystems method*), 68  
 to\_siesta\_output() (*dpdata.system.System method*), 85  
 to\_sqm\_in() (*dpdata.system.LabeledSystem method*), 56  
 to\_sqm\_in() (*dpdata.system.MultiSystems method*), 68  
 to\_sqm\_in() (*dpdata.system.System method*), 85  
 to\_sqm\_out() (*dpdata.system.LabeledSystem method*), 56  
 to\_sqm\_out() (*dpdata.system.MultiSystems method*), 68  
 to\_sqm\_out() (*dpdata.system.System method*), 85  
 to\_system() (*dpdata.format.Format method*), 41  
 to\_system() (*dpdata.plugins.amber.SQMINFormat method*), 10  
 to\_system() (*dpdata.plugins.ase.ASEStructureFormat method*), 12  
 to\_system() (*dpdata.plugins.deepmd.DeePMDCompFormat method*), 14  
 to\_system() (*dpdata.plugins.deepmd.DeePMDHDF5Format method*), 15  
 to\_system() (*dpdata.plugins.deepmd.DeePMDRawFormat method*), 16  
 to\_system() (*dpdata.plugins.gromacs.GromacsGroFormat method*), 20  
 to\_system() (*dpdata.plugins.lammps.LAMMPSLmpFormat method*), 22  
 to\_system() (*dpdata.plugins.list.ListFormat method*), 22  
 to\_system() (*dpdata.plugins.pwmat.PwmatAtomconfigFormat method*), 23  
 to\_system() (*dpdata.plugins.pymatgen.PyMatgenMoleculeFormat method*), 25  
 to\_system() (*dpdata.plugins.pymatgen.PyMatgenStructureFormat method*), 26  
 to\_system() (*dpdata.plugins.vasp.VASPPoscarFormat method*), 32  
 to\_system() (*dpdata.plugins.vasp.VASPStringFormat method*), 33  
 to\_system() (*dpdata.plugins.xyz.XYZFormat method*), 35  
 to\_system\_data() (*in module dpdata.deepmd.comp*), 5  
 to\_system\_data() (*in module dpdata.deepmd.hdf5*), 5  
 to\_system\_data() (*in module dpdata.deepmd.raw*), 6  
 to\_system\_data() (*in module dpdata.gaussian.log*), 6  
 to\_system\_data() (*in module dpdata.lammps.lmp*), 7  
 to\_system\_data() (*in module dpdata.pwmat.atomconfig*), 35  
 to\_system\_data() (*in module dpdata.pymatgen.molecule*), 36  
 to\_system\_data() (*in module dpdata.qe.traj*), 37  
 to\_system\_data() (*in module dpdata.vasp.poscar*), 38  
 to\_system\_label() (*in module dpdata.qe.traj*), 37  
 to\_vasp\_contcar() (*dpdata.system.LabeledSystem method*), 56  
 to\_vasp\_contcar() (*dpdata.system.MultiSystems method*), 69  
 to\_vasp\_contcar() (*dpdata.system.System method*), 85  
 to\_vasp\_outcar() (*dpdata.system.LabeledSystem method*), 56  
 to\_vasp\_outcar() (*dpdata.system.MultiSystems method*), 69  
 to\_vasp\_outcar() (*dpdata.system.System method*), 85  
 to\_vasp\_poscar() (*dpdata.system.LabeledSystem method*), 56  
 to\_vasp\_poscar() (*dpdata.system.MultiSystems method*), 69  
 to\_vasp\_poscar() (*dpdata.system.System method*), 85  
 to\_vasp\_string() (*dpdata.system.LabeledSystem method*), 56  
 to\_vasp\_string() (*dpdata.system.MultiSystems method*), 69  
 to\_vasp\_string() (*dpdata.system.System method*), 86  
 to\_vasp\_xml() (*dpdata.system.LabeledSystem method*), 56  
 to\_vasp\_xml() (*dpdata.system.MultiSystems method*), 69  
 to\_vasp\_xml() (*dpdata.system.System method*), 86  
 to\_xml() (*dpdata.system.LabeledSystem method*), 56  
 to\_xml() (*dpdata.system.MultiSystems method*), 69  
 to\_xml() (*dpdata.system.System method*), 86  
 to\_xyz() (*dpdata.system.LabeledSystem method*), 56  
 to\_xyz() (*dpdata.system.MultiSystems method*), 69  
 to\_xyz() (*dpdata.system.System method*), 86

## U

uniq\_atom\_names() (*in module dpdata.utils*), 88  
 unique\_formula() (*dpdata.system.System property*), 86

## V

value() (*dpdata.unit.Conversion method*), 87  
 VASPOutcarFormat (*class in dpdata.plugins.vasp*), 31  
 VASPPoscarFormat (*class in dpdata.plugins.vasp*), 31  
 VASPStringFormat (*class in dpdata.plugins.vasp*), 32  
 VASPXMLFormat (*class in dpdata.plugins.vasp*), 33

## X

X (*dpdata.periodic\_table.Element property*), 41  
 XYZFormat (*class in dpdata.plugins.xyz*), 34

## Z

Z (*dpdata.periodic\_table.Element property*), 41