# Table of Contents

## Copyright and Usage Terms

Copyright...................................................................................................................................................1
Terms of Usage for the PyMOL Reference Manual.........................................................1
Trademarks..................................................................................................................................................1

## Reference

accept......................................................................................................................................................2
alias..........................................................................................................................................................2
align..........................................................................................................................................................2
alter............................................................................................................................................................3
alter state................................................................................................................................................3
attach.......................................................................................................................................................4
backward..................................................................................................................................................4
bg_color....................................................................................................................................................4
bond.........................................................................................................................................................5
button.......................................................................................................................................................5
cartoon.....................................................................................................................................................6
cd.............................................................................................................................................................6
center.......................................................................................................................................................6
clip...........................................................................................................................................................7
cls............................................................................................................................................................7
color..........................................................................................................................................................8
commands................................................................................................................................................8
copy.........................................................................................................................................................9
count atoms..........................................................................................................................................9
count frames.......................................................................................................................................9
count states..........................................................................................................................................10
create....................................................................................................................................................10
cycle valence.....................................................................................................................................10
decline...................................................................................................................................................11
delete.....................................................................................................................................................11
deprotect...............................................................................................................................................11
deselect...............................................................................................................................................12
disable..................................................................................................................................................12
distance...............................................................................................................................................13
do.........................................................................................................................................................13
dss..........................................................................................................................................................13
dummy....................................................................................................................................................14
edit..........................................................................................................................................................14
enable....................................................................................................................................................15
ending.....................................................................................................................................................15
extend......................................................................................................................................................15
feedback...............................................................................................................................................16
find_pairs............................................................................................................................................17
fit.............................................................................................................................................................17
flag..........................................................................................................................................................17
forward.................................................................................................................................................18
fragment...............................................................................................................................................18
<table>
<thead>
<tr>
<th>Reference</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>frame.</td>
<td>19</td>
</tr>
<tr>
<td>full_screen.</td>
<td>19</td>
</tr>
<tr>
<td>fuse.</td>
<td>19</td>
</tr>
<tr>
<td>get_area.</td>
<td>20</td>
</tr>
<tr>
<td>get_chains.</td>
<td>20</td>
</tr>
<tr>
<td>get_dihedral.</td>
<td>20</td>
</tr>
<tr>
<td>get_extent.</td>
<td>20</td>
</tr>
<tr>
<td>get_frame.</td>
<td>21</td>
</tr>
<tr>
<td>get_model.</td>
<td>21</td>
</tr>
<tr>
<td>get_names.</td>
<td>21</td>
</tr>
<tr>
<td>get_povray.</td>
<td>21</td>
</tr>
<tr>
<td>get_state.</td>
<td>22</td>
</tr>
<tr>
<td>get_title.</td>
<td>22</td>
</tr>
<tr>
<td>get_type.</td>
<td>22</td>
</tr>
<tr>
<td>get_view.</td>
<td>23</td>
</tr>
<tr>
<td>h_add.</td>
<td>23</td>
</tr>
<tr>
<td>h_fill.</td>
<td>24</td>
</tr>
<tr>
<td>help.</td>
<td>24</td>
</tr>
<tr>
<td>hide.</td>
<td>24</td>
</tr>
<tr>
<td>id_atom.</td>
<td>25</td>
</tr>
<tr>
<td>identify.</td>
<td>25</td>
</tr>
<tr>
<td>index.</td>
<td>25</td>
</tr>
<tr>
<td>indicate.</td>
<td>26</td>
</tr>
<tr>
<td>intra_fit.</td>
<td>26</td>
</tr>
<tr>
<td>intra_rms.</td>
<td>26</td>
</tr>
<tr>
<td>intra_rms_cur.</td>
<td>27</td>
</tr>
<tr>
<td>invert.</td>
<td>27</td>
</tr>
<tr>
<td>isodot.</td>
<td>28</td>
</tr>
<tr>
<td>isolevel.</td>
<td>28</td>
</tr>
<tr>
<td>isomesh.</td>
<td>28</td>
</tr>
<tr>
<td>isosurface.</td>
<td>29</td>
</tr>
<tr>
<td>iterate.</td>
<td>30</td>
</tr>
<tr>
<td>iterate_state.</td>
<td>30</td>
</tr>
<tr>
<td>keyword.</td>
<td>31</td>
</tr>
<tr>
<td>label.</td>
<td>31</td>
</tr>
<tr>
<td>load.</td>
<td>31</td>
</tr>
<tr>
<td>load_brick.</td>
<td>32</td>
</tr>
<tr>
<td>load_callback.</td>
<td>32</td>
</tr>
<tr>
<td>load_cgo.</td>
<td>32</td>
</tr>
<tr>
<td>load_map.</td>
<td>33</td>
</tr>
<tr>
<td>load_model.</td>
<td>33</td>
</tr>
<tr>
<td>load_object.</td>
<td>33</td>
</tr>
<tr>
<td>load_traj.</td>
<td>33</td>
</tr>
<tr>
<td>ls</td>
<td>34</td>
</tr>
<tr>
<td>map_double.</td>
<td>34</td>
</tr>
<tr>
<td>map_new.</td>
<td>34</td>
</tr>
<tr>
<td>map_set_border.</td>
<td>35</td>
</tr>
</tbody>
</table>
# Table of Contents

## Reference

- `sculpt_purge` ............................................................................................................... 52
- `select` ................................................................................................................................. 52
- `set` ................................................................................................................................ 53
- `set_color` ............................................................................................................................ 53
- `set_geometry` ...................................................................................................................... 53
- `set_key` ............................................................................................................................... 54
- `set_symmetry` ..................................................................................................................... 55
- `set_title` ............................................................................................................................. 55
- `set_view` ............................................................................................................................ 55
- `show` .................................................................................................................................. 56
- `smooth` ............................................................................................................................... 56
- `sort` .................................................................................................................................. 57
- `space` .................................................................................................................................. 57
- `spectrum` .............................................................................................................................. 57
- `spheroid` .............................................................................................................................. 58
- `splash` ................................................................................................................................ 58
- `stereo` .................................................................................................................................. 58
- `symexp` ................................................................................................................................. 59
- `sync` .................................................................................................................................. 59
- `system` ................................................................................................................................ 59
- `torsion` ................................................................................................................................ 60
- `translate` ............................................................................................................................. 60
- `turn` .................................................................................................................................... 61
- `unbond` ............................................................................................................................... 61
- `undo` .................................................................................................................................. 62
- `unmask` ................................................................................................................................ 62
- `unpick` .................................................................................................................................. 62
- `unset` .................................................................................................................................. 63
- `update` ................................................................................................................................ 63
- `view` ................................................................................................................................... 63
- `viewport` ............................................................................................................................... 64
- `wizard` .................................................................................................................................. 64
- `zoom` .................................................................................................................................... 65
Copyright and Usage Terms

Copyright


Terms of Usage for the PyMOL Reference Manual

This manual is NOT free. It is an Incentive Product created to help you use PyMOL while generating recurring sponsorship for the project. It is made available for evaluation via the "honor" system.

You may evaluate this Incentive Product for a continuous period of up to one year without obligation.

If you continue using this document beyond the end of the evaluation period, then you must become a sponsor of the project by purchasing a PyMOL license and maintenance subscription from DeLano Scientific LLC (http://www.delanoscientific.com).

If you are willing to sponsor the project today, then please don't wait a full year to start. The sooner your sponsorship comes in, the sooner we can apply it to improve the software and documentation!

Existing PyMOL maintenance subscribers may use this manual for no additional cost. However, subscribers who do not renew their subscription upon expiration must discontinue use of this and all other PyMOL Incentive Products. We have no direct means of enforcing this, but we ask, in recognition of our declared scientific mission, that you honor the trust we have placed in you.

PyMOL users who are unable to sponsor the project by purchasing a PyMOL license and maintenance subscription are welcome to use Open–Source versions of PyMOL and any free documentation that can be found on the internet.

Trademarks

PyMOL, DeLano Scientific, and the DeLano Scientific Logo are trademarks of DeLano Scientific LLC. Macintosh is a registered trademark of Apple Computer Inc., registered in the U.S. and other countries. Windows is a registered trademark of Microsoft Corporation in the U.S. and other countries. Linux is a trademark of Linus Torvalds. Unix is a trademark of The Open Group in the U.S. and other countries. MolScript is a trademark of Avatar Software AB. All other trademarks are the property of their respective owners.

Copyright © 2003 DeLano Scientific LLC. All rights reserved.
**alias**

**DESCRIPTION**

"alias" allows you to bind a commonly used command to a single PyMOL keyword.

**USAGE**

```
alias name, command-sequence
```

**PYMOL API**

```
cmd.alias(string name, string command)
```

**EXAMPLES**

```
alias go,load $TUT/1hpv.pdb; zoom 200/; show sticks, 200/ around 8
go
```

**NOTES**

For security reasons, new PyMOL commands created using "extend" are not saved or restored in sessions.

**SEE ALSO**

extend, api

---

**align**

**DESCRIPTION**

"align" performs a sequence alignment followed by a structural alignment, and then carries out zero or more cycles of refinement in order to reject structural outliers found during the fit.

**USAGE**

```
align (source), (target) [,cutoff [,cycles [,gap [,extend 
[,skip [,object [,matrix [, quiet ]]]]]]]])
```

**PYMOL API**

```
cmd.align( string source, string target, float cutoff=2.0, 
int cycles=2, float gap=-10.0, float extend=-0.5, 
float extend=-0.5, int skip=0, string object=None, 
string matrix="BLOSUM62", int quiet=1 )
```

**NOTE**
If object is not None, then align will create an object which indicates which atoms were paired between the two structures

EXAMPLES

align  prot1/\///CA, prot2, object=alignment

SEE ALSO

fit, rms, rms_cur, intra_rms, intra_rms_cur, pair_fit

---

**alter**

**DESCRIPTION**

"alter" changes one or more atomic properties over a selection using the python evaluator with a separate name space for each atom. The symbols defined in the name space are:

- name, resn, resi, chain, alt, elem, q, b, segi,
- type (ATOM,HETATM), partial_charge, formal_charge,
- text_type, numeric_type, ID

All strings must be explicitly quoted. This operation typically takes several seconds per thousand atoms altered.

**WARNING**: You should always issue a "sort" command on an object after modifying any property which might affect canonical atom ordering (names, chains, etc.). Failure to do so will confound subsequent "create" and "byres" operations.

**USAGE**

alter (selection),expression

**EXAMPLES**

alter (chain A),chain='B'
alter (all),resi=str(int(resi)+100)

sort

SEE ALSO

alter_state, iterate, iterate_state, sort

---

**alter_state**

**DESCRIPTION**

"alter_state" changes the atomic coordinates of a particular state using the python evaluator with a separate name space for each atom. The symbols defined in the name space are:

- x,y,z

**USAGE**
alter_state state, (selection), expression

EXAMPLES
alter_state 1, (all), x = x+5

SEE ALSO
iterate_state, alter, iterate

---

**attach**

DESCRIPTION

"attach" adds a single atom onto the picked atom.

USAGE

attach element, geometry, valence

PYMOL API

cmd.attach( element, geometry, valence )

NOTES

Im mature functionality. See code for details.

---

**backward**

DESCRIPTION

"backward" moves the movie back one frame.

USAGE

backward

PYMOL API

cmd.backward()

SEE ALSO

mset, forward, rewind

---

**bg_color**

DESCRIPTION

"bg_color" sets the background color

USAGE

attach
bond

DESCRIPTION

"bond" creates a new bond between two selections, each of which should contain one atom.

USAGE

bond [atom1,atom2 [,order]]

NOTES

The atoms must both be within the same object.

The default behavior is to create a bond between the (lb) and (rb) selections.

SEE ALSO

unbond, fuse, attach, replace, remove_picked

button

DESCRIPTION

"button" can be used to redefine what the mouse buttons do.

USAGE

button <button>,<modifier>,<action>

NOTES

button: L, M, R
modifiers: None, Shft, Ctrl, CtSh
actions: Rota, Move, MovZ, Clip, RotZ, ClpN, ClpF
lb, mb, rb, +lb, +lbX, -lbX, +mb, +rb, PkAt, PkBd, RotF, TorF, MovF, Orig, Cent
**cartoon**

**DESCRIPTION**

"cartoon" changes the default cartoon for a set of atoms.

**USAGE**

```python
cartoon type, (selection)
type = skip | automatic | loop | rectangle | oval | tube | arrow | dumbbell
```

**PYMOL API**

```python
cmd.cartoon(string type, string selection)
```

**EXAMPLES**

```python
cartoon rectangle, (chain A)
cartoon skip, (resi 145:156)
```

**NOTES**

the "automatic" mode utilizes ribbons according to the information in the PDB HELIX and SHEET records.

---

**cd**

**DESCRIPTION**

"cd" changes the current working directory.

**USAGE**

```bash
cd <path>
```

**SEE ALSO**

```
pwd, ls, system
```

---

**center**

**DESCRIPTION**

"center" translates the window, the clipping slab, and the origin to a point centered within the atom selection.

**USAGE**

```python
center [ selection [, state [, origin]]]
```

**EXAMPLES**

```python
center 145/
```
cmd.center( string selection, int state = 0, int origin = 1 )

NOTES

state = 0 (default) use all coordinate states
state = -1 use only coordinates for the current state
state > 0 use coordinates for a specific state

origin = 1 (default) move the origin
origin = 0 leave the origin unchanged

SEE ALSO

origin, orient, zoom

---

**clip**

DESCRIPTION

"clip" alters the near and far clipping planes

USAGE

clip {near|far|move|slab|atoms}, distance [,selection [,state ]]

EXAMPLES

clip near, -5          # moves near plane away from you by 5 A
clip far, 10           # moves far plane towards you by 10 A
clip move, -5          # moves the slab away from you by 5 A
clip slab, 20          # sets slab thickness to 20 A
clip slab, 10, resi 11 # clip 10 A slab about residue 11
clip atoms, 5, pept    # clip atoms in "pept" with a 5 A buffer
                        # about their current camera positions

PYMOL API

    cmd.clip( string mode, float distance, string selection = None)

SEE ALSO

zoom, reset

---

**cls**

DESCRIPTION

"cls" clears the output buffer.

USAGE

cls
color

DESCRIPTION

"color" changes the color of an object or an atom selection.

USAGE

color color-name
color color-name, object-name
color color-name, (selection)

PYMOL API

cmd.color( string color, string color-name )

EXAMPLES

color yellow, (name C*)

commands

COMMANDS

INPUT/OUTPUT load save delete quit
VIEW turn move clip rock
show hide enable disable
reset refresh rebuild
zoom origin orient
view get_view set_view
MOVIES mplay mstop mset mdo
mpng mmatrix frame
rewind middle ending
forward backward
IMAGING png mpng
RAY TRACING ray
MAPS isomesh isodot
DISPLAY cls viewport splash
SELECTIONS select mask
SETTINGS set button
ATOMS alter alter_state
EDITING create replace remove h_fill remove_picked
edit bond unbond h_add fuse
undo redo protect cycle_valence attach
FITTING fit rms rms_cur pair_fit
COLORS color set_color
HELP help commands
DISTANCES dist
STEREO stereo
SYMMETRY symexp
SCRIPTS @ run
LANGUAGE alias extend

Try "help <command-name>". Also see the following extra topics:

"movies", "keyboard", "mouse", "selections", "examples", "launching", "editing", and "api".

copy

DESCRIPTION

"copy" creates a new object that is an identical copy of an existing object.

USAGE

copy target, source

copy target = source # (DEPRECATED)

PYMOL API

cmd.copy(string target,string source)

SEE ALSO

create

count_atoms

DESCRIPTION

"count_atoms" returns a count of atoms in a selection.

USAGE

count_atoms (selection)

PYMOL API

cmd.count(string selection)

count_frames

DESCRIPTION

"count_frames" is an API-only function which returns the number of frames defined for the PyMOL movie.

PYMOL API

cmd.count_frames()

SEE ALSO

frame, count_states
count_states

DESCRIPTION

"count_states" is an API-only function which returns the number of states in the selection.

PYMOL API

    cmd.count_states(string selection="(all)")

SEE ALSO

    frame

create

DESCRIPTION

"create" creates a new molecule object from a selection. It can also be used to create states in an existing object.

NOTE: this command has not yet been thoroughly tested.

USAGE

    create name, (selection) [,source_state [,target_state ] ]
    create name = (selection) [,source_state [,target_state ] ]
        # (DEPRECATED)

    name = object to create (or modify)
    selection = atoms to include in the new object
    source_state (default: 0 - copy all states)
    target_state (default: 0)

PYMOL API

    cmd.create(string name, string selection, int state, int target_state)

NOTES

If the source and target states are zero (default), all states will be copied. Otherwise, only the indicated states will be copied.

SEE ALSO

    load, copy

cycle_valence

DESCRIPTION

"cycle_valence" cycles the valence on the currently selected bond.

USAGE

    count_states
cycle_valence [ h_fill ]

PYMOL API

    cmd.cycle_valence(int h_fill)

EXAMPLES

    cycle_valence
    cycle_valence 0

NOTES

    If the h_fill flag is true, hydrogens will be added or removed to
    satisfy valence requirements.

    This function is usually connected to the DELETE key and "CTRL-W".

SEE ALSO

    remove_picked, attach, replace, fuse, h_fill

---

**decline**

SECURITY FEATURE

---

**delete**

DESCRIPTION

    "delete" removes an object or a selection.

USAGE

    delete name
    delete all   # deletes all objects

    name = name of object or selection

PYMOL API

    cmd.delete (string name = object-or-selection-name )

SEE ALSO

    remove

---

**deprotect**

DESCRIPTION

    "deprotect" reverses the effect of the "protect" command.

USAGE

    decline
deselect

DESCRIPTION

"deselect" disables any and all visible selections

USAGE

deselect

PYMOL API

cmd.deselect()

disable

DESCRIPTION

"disable" disables display of an object and all currently visible representations.

USAGE

disable name
disable all

"name" is the name of an object or a named selection

PYMOL API

cmd.disable( string name )

EXAMPLE

disable my_object

SEE ALSO

show, hide, enable
**distance**

**DESCRIPTION**

"distance" creates a new distance object between two selections. It will display all distances within the cutoff.

**USAGE**

```python
distance
distance (selection1), (selection2)
distance name = (selection1), (selection1) [,cutoff [,mode] ]
```

- `name` = name of distance object
- `selection1,selection2` = atom selections
- `cutoff` = maximum distance to display
- `mode` = 0 (default)

**PYMOL API**

```python
cmd.distance( string name, string selection1, string selection2, string cutoff, string mode )
```

returns the average distance between all atoms/frames

**NOTES**

The distance wizard makes measuring distances easier than using the "dist" command for real-time operations.

"dist" alone will show distances between selections (lb) and (rb) created by left and right button atom picks. CTRL-SHIFT/ left-click on the first atom,  CTRL-SHIFT/ right-click on the second, then run "dist".

---

**do**

**DESCRIPTION**

"do" makes it possible for python programs to issue simple PyMOL commands as if they were entered on the command line.

**PYMOL API**

```python
cmd.do( commands )
```

**USAGE (PYTHON)**

```python
from pymol import cmd
cmd.do("load file.pdb")
```

---

**dss**

**DESCRIPTION**

"dss" defines secondary structure based on backbone geometry and hydrogen bonding patterns.
With PyMOL, heavy emphasis is placed on cartoon aesthetics, and so both hydrogen bonding patterns and backbone geometry are used in the assignment process. Depending upon the local context, helix and strand assignments are made based on geometry, hydrogen bonding, or both.

This command will generate results which differ slightly from DSSP and other programs. Most deviations occur in borderline or transition regions. Generally speaking, PyMOL is more strict, thus assigning fewer helix/sheet residues, except for partially distorted helices, which PyMOL tends to tolerate.

WARNING: This algorithm has not yet been rigorously validated.

USAGE

dss selection, state

state = state-index or 0 for all states

EXAMPLES

dss

NOTES

If you dislike one or more of the assignments made by dss, you can use the alter command to make changes (followed by "rebuild"). For example:

    alter 123-125/, ss='L'
    alter pk1, ss='S'
    alter 90/, ss='H'
    rebuild

dummy

DESCRIPTION

This is a dummy function which returns None.

edit

DESCRIPTION

"edit" picks an atom or bond for editing.

USAGE

    edit (selection) [ ,(selection) ]

PYMOL API

    cmd.edit( string selection [ ,string selection ] )

NOTES
If only one selection is provided, an atom is picked. If two selections are provided, the bond between them is picked (if one exists).

SEE ALSO

unpick, remove_picked, cycle_valence, torsion

---

**enable**

**DESCRIPTION**

"enable" enable display of an object and all currently visible representations.

**USAGE**

enable name
enable all

name = object or selection name

**PYMOL API**

cmd.enable( string object-name )

**EXAMPLE**

enable my_object

SEE ALSO

show, hide, disable

---

**ending**

**DESCRIPTION**

"ending" goes to the end of the movie.

**USAGE**

ending

**PYMOL API**

cmd.ending()

---

**extend**

**DESCRIPTION**

"extend" is an API-only function which binds a new external function as a command into the PyMOL scripting language.
PYMOL API

    cmd.extend(string name, function function)

PYTHON EXAMPLE

    def foo(moo=2): print moo
    cmd.extend('foo', foo)

    The following would now work within PyMOL:

    PyMOL>foo
    2
    PyMOL>foo 3
    3
    PyMOL>foo moo=5
    5
    PyMOL>foo ?
    Usage: foo [ moo ]

NOTES

    For security reasons, new PyMOL commands created using "extend" are
    not saved or restored in sessions.

SEE ALSO

    alias, api

feedback

DESCRIPTION

    "feedback" allows you to change the amount of information output by pymol.

USAGE

    feedback action, module, mask

    action is one of ['set', 'enable', 'disable']
    module is a space-separated list of strings or simply "all"
    mask is a space-separated list of strings or simply "everything"

NOTES:

    "feedback" alone will print a list of the available module choices

PYMOL API

    cmd.feedback(string action, string module, string mask)

EXAMPLES

    feedback enable, all, debugging
    feedback disable, selector, warnings actions
    feedback enable, main, blather

DEVELOPMENT TO DO

    Add a way of querying the current feedback settings.
Check C source code to make source correct modules are being used.
Check C source code to insure that all output is properly
Update Python API and C source code to use "quiet" parameter as well.

find_pairs

DESCRIPTION

"find_pairs" is currently undocumented.

fit

DESCRIPTION

"fit" superimposes the model in the first selection on to the model
in the second selection. Only matching atoms in both selections
will be used for the fit.

USAGE

fit (selection), (target−selection)

EXAMPLES

fit ( mutant and name ca ), ( wildtype and name ca )

SEE ALSO

rms, rms_cur, intra_fit, intra_rms, intra_rms_cur

flag

DESCRIPTION

"flag" sets the indicated flag for atoms in the selection and
clears the indicated flag for atoms not in the selection. This
is primarily useful for passing selection information into
Chempy models, which have a 32 bit attribute "flag" which holds
this information.

USAGE

flag flag, selection [ ,action ]

flag flag = selection [ ,action ] # (DEPRECATED)

action can be:
"reset" (default) sets flag on selection, clear it on other atoms
"set" sets the flag for selected atoms, leaves other atoms unchanged
"clear" clear the flag for selected atoms, leaves other atoms unchanged

PYMOL API

cmd.flag( int flag, string selection, string action="reset",
int indicate=0)
cmd.flag( string flag, string selection, string action="reset", int indicate=0)

EXAMPLES

    flag free, (resi 45 x; 6)

NOTE

    If the 'auto_indicate_flags' setting is true, then PyMOL will automatically create a selection called "indicate" which contains all atoms with that flag after applying the command.

RESERVED FLAGS

    Flags 0−7 are reserved for molecular modeling */
        focus      0 = Atoms of Interest (i.e. a ligand in an active site)
        free       1 = Free Atoms (free to move subject to a force-field)
        restrain   2 = Restrained Atoms (typically harmonically contrained)
        fix        3 = Fixed Atoms (no movement allowed)
        ignore     4 = Atoms which should not be part of any simulation
    Flags 8−15 are free for end users to manipulate
    Flags 16−23 are reserved for external GUIs and linked applications
    Flags 24−31 are reserved for PyMOL internal usage
        exfoliate 24 = Remove surface from atoms when surfacing
        ignore    25 = Ignore atoms altogether when surfacing

---

**forward**

**DESCRIPTION**

    "forward" moves the movie one frame forward.

**USAGE**

    forward

**PYMOL API**

    cmd.forward()

**SEE ALSO**

    mset, backward, rewind

---

**fragment**

**DESCRIPTION**

    "fragment" retrieves a 3D structure from the fragment library, which is currently pretty meager (just amino acids).

**USAGE**

    fragment name
**frame**

**DESCRIPTION**

"frame" sets the viewer to the indicated movie frame.

**USAGE**

frame frame-number

**PYMOL API**

cmd.frame( int frame_number )

**NOTES**

Frame numbers are 1-based

**SEE ALSO**

count_states

---

**full_screen**

**DESCRIPTION**

"full_screen" enables or disables PyMOL's full screen mode. This does not work well on all platforms.

**USAGE**

full_screen on
full_screen off

---

**fuse**

**DESCRIPTION**

"fuse" joins two objects into one by forming a bond. A copy of the object containing the first atom is moved so as to form an approximately resonable bond with the second, and is then merged with the first object.

**USAGE**

fuse (selection1), (selection2)

**PYMOL API**

cmd.fuse( string selection1="(lb)", string selection2="(lb)" )

**NOTES**

Each selection must include a single atom in each object. The atoms can both be hydrogens, in which case they are eliminated, or they can both be non-hydrogens, in which
case a bond is formed between the two atoms.

SEE ALSO

bond, unbond, attach, replace, fuse, remove_picked

---

**get_area**

PRE-RELEASE functionality - API will change

---

**get_chains**

PRE-RELEASE functionality - API will change

state is currently ignored

---

**get_dihedral**

DESCRIPTION

"get_dihedral" returns the dihedral angle between four atoms. By default, the coordinates used are from the current state, however an alternate state identifier can be provided.

By convention, positive dihedral angles are right-handed (looking down the atom2-atom3 axis).

USAGE

get_dihedral atom1, atom2, atom3, atom4 [,state ]

EXAMPLES

get_dihedral 4/n,4/c,4/ca,4/cb
get_dihedral 4/n,4/c,4/ca,4/cb,state=4

PYMOL API

cmd.get_dihedral(atom1,atom2,atom3,atom4,state=0)

---

**get_extent**

DESCRIPTION

"get_extent" returns the minimum and maximum XYZ coordinates of a selection as an array:

[ [ min-X , min-Y , min-Z ],[ max-X, max-Y , max-Z ]]

PYMOL API

cmd.get_extent(string selection="(all)", state=0 )
get_frame

DESCRIPTION

"get_frame" returns the current frame index (1-based)

PYMOL API

Frames refers to sequences of images in a movie. Sequential frames may contain identical molecular states, they may have one-to-one correspondence to molecular states (default), or they may have an arbitrary relationship, specific using the "mset" command.

SEE ALSO

get_state

get_model

DESCRIPTION

"get_model" returns a ChemPy "Indexed" format model from a selection.

PYMOL API

cmd.get_model(string selection [,int state] )

get_names

DESCRIPTION

"get_names" returns a list of object and/or selection names.

PYMOL API

cmd.get_names( [string: "objects"|"selections"|"all"] )

NOTES

The default behavior is to return only object names.

SEE ALSO

get_type, count_atoms, count_states

get_povray

DESCRIPTION

"get_povray" returns a tuple corresponding to strings for a PovRay input file.

PYMOL API

get_frame
**get_state**

**DESCRIPTION**

"get_state" returns the current state index (1-based)

**PYMOL API**

```
cmd.get_state()
```

**NOTES**

States refer to different geometric configurations which an object can assume. By default, states and movie frames have a one-to-one relationship. States can be visited in an arbitrary order to create frames. The "mset" command allows you to build a relationship between states and frames.

**SEE ALSO**

get_frame

---

**get_title**

**DESCRIPTION**

"get_title" retrieves a text string to the state of a particular object which will be displayed when the state is active.

**USAGE**

```
set_title object,state
```

**PYMOL API**

```
cmd.set_title(string object,int state,string text)
```

---

**get_type**

**DESCRIPTION**

"get_type" returns a string describing the named object or selection or the string "nonexistent" if the name is unknown.

**PYMOL API**

```
cmd.get_type(string object-name)
```

**NOTES**

Possible return values are
**get_view**

**DESCRIPTION**

"get_view" returns and optionally prints out the current view information in a format which can be embedded into a command script and can be used in subsequent calls to "set_view".

If a log file is currently open, get_view will not write the view matrix to the screen unless the "output" parameter is 2.

**USAGE**

get_view

**PYMOL API**

cmd.get_view(output=1,quiet=1)

my_view= cmd.get_view()

output:

0 = output matrix to screen
1 = don't output matrix to screen
2 = force output to screen even if log file is open

**API USAGE**

cmd.get_view(0) # zero option suppresses output (LEGACY approach)
cmd.get_view(quiet=1) # suppresses output using PyMOL's normal "quiet" parameter.

---

**h_add**

**DESCRIPTION**

"h_add" uses a primitive algorithm to add hydrogens onto a molecule.

**USAGE**

h_add (selection)

**PYMOL API**

cmd.h_add( string selection="(all)" )
**h_fill**

**DESCRIPTION**

"h_fill" removes and replaces hydrogens on the atom or bond picked for editing.

**USAGE**

h_fill

**PYMOL API**

cmd.h_fill()

**NOTES**

This is useful for fixing hydrogens after changing bond valences.

**SEE ALSO**

edit, cycle_valence, h_add

**help**

**DESCRIPTION**

"help" prints out the online help for a given command.

**USAGE**

help command

**hide**

**DESCRIPTION**

"hide" turns off atom and bond representations.

The available representations are:

- lines
- spheres
- mesh
- ribbon
- cartoon
- sticks
- dots
- surface
- labels
- nonbonded
- nb_spheres

**USAGE**

hide representation [,object]
hide representation [,,(selection)]
hide (selection)
PYMOL API

    cmd.hide( string representation="", string selection=""

EXAMPLES

    hide lines,all
    hide ribbon

SEE ALSO

    show, enable, disable

---

**id_atom**

DESCRIPTION

    "id_atom" returns the original source id of a single atom, or
    raises an exception if the atom does not exist or if the selection
    corresponds to multiple atoms.

PYMOL API

    list = cmd.id_atom(string selection)

---

**identify**

DESCRIPTION

    "identify" returns a list of atom IDs corresponding to the ID code
    of atoms in the selection.

PYMOL API

    list = cmd.identify(string selection="(all)",int mode=0)

NOTES

    mode 0: only return a list of identifiers (default)
    mode 1: return a list of tuples of the object name and the identifier

---

**index**

DESCRIPTION

    "index" returns a list of tuples corresponding to the
    object name and index of the atoms in the selection.

PYMOL API

    list = cmd.index(string selection="(all)")

NOTE
Atom indices are fragile and will change as atoms are added or deleted. Whenever possible, use integral atom identifiers instead of indices.

---

**indicate**

**DESCRIPTION**

"indicate" shows a visual representation of an atom selection.

**USAGE**

indicate (selection)

**PYMOL API**

cmd.count(string selection)

---

**intra_fit**

**DESCRIPTION**

"intra_fit" fits all states of an object to an atom selection in the specified state. It returns the rms values to python as an array.

**USAGE**

intra_fit (selection),state

**PYMOL API**

cmd.intra_fit( string selection, int state )

**EXAMPLES**

intra_fit ( name ca )

**PYTHON EXAMPLE**

```python
from pymol import cmd
rms = cmd.intra_fit("(name ca)",1)
```

**SEE ALSO**

fit, rms, rms_cur, intra_rms, intra_rms_cur, pair_fit

---

**intra_rms**

**DESCRIPTION**

"intra_rms" calculates rms fit values for all states of an object over an atom selection relative to the indicated state. Coordinates are left unchanged. The rms values are returned as a python array.
intra_rms_cur

DESCRIPTION

"intra_rms_cur" calculates rms values for all states of an object over an atom selection relative to the indicated state without performing any fitting. The rms values are returned as a python array.

PYMOL API

    cmd.intra_rms_cur( string selection, int state)

PYTHON EXAMPLE

    from pymol import cmd
    rms = cmd.intra_rms_cur("(name ca)",1)

SEE ALSO

    fit, rms, rms_cur, intra_fit, intra_rms_cur, pair_fit

invert

DESCRIPTION

"invert" inverts the stereo-chemistry of the atom currently picked for editing (pk1). Two additional atom selections must be provided in order to indicate which atoms remain stationary during the inversion process.

USAGE

    invert (selection1),(selection2)

PYMOL API

    cmd.api( string selection1="(lb)", string selection2="(lb)" )

NOTE

    The invert function is usually bound to CTRL-E in editing mode.
The default selections are (lb) and (rb), meaning that you can pick the atom to invert with CTRL-middle click and then pick the stationary atoms with CTRL-SHIFT/left-click and CTRL-SHIFT/right-click, then hit CTRL-E to invert the atom.

**isodot**

**DESCRIPTION**

"isodot" creates a dot isosurface object from a map object.

**USAGE**

```plaintext
isodot name = map, level [, (selection) [, buffer [, state ] ] ]
```

map = the name of the map object to use.

level = the contour level.

selection = an atom selection about which to display the mesh with an additional "buffer" (if provided).

**NOTES**

If the dot isosurface object already exists, then the new dots will be appended onto the object as a new state.

**SEE ALSO**

load, isomesh

**isolevel**

**DESCRIPTION**

"isolevel" changes the contour level of a isodot, isosurface, or isomesh object.

**USAGE**

```plaintext
isolevel name, level, state
```

**isomesh**

**DESCRIPTION**

"isomesh" creates a mesh isosurface object from a map object.

**USAGE**

```plaintext
isomesh name, map, level [, (selection) [, buffer [, state [, carve ]]]]
```

name = the name for the new mesh isosurface object.

map = the name of the map object to use for computing the mesh.
level = the contour level.

selection = an atom selection about which to display the mesh with
 an additional "buffer" (if provided).

state = the state into which the object should be loaded (default=1)
 (set state=0 to append new mesh as a new state)

carve = a radius about each atom in the selection for which to
 include density. If "carve" is not provided, then the whole
 brick is displayed.

NOTES

If the mesh object already exists, then the new mesh will be
 appended onto the object as a new state (unless you indicate a state).

state > 0: specific state
state = 0: all states
state = -1: current state

source_state > 0: specific state
source_state = 0: include all states starting with 0
source_state = -1: current state
source_state = -2: last state in map

SEE ALSO

isodot, load

--

isosurface

DESCRIPTION

"isosurface" creates a new surface object from a map object.

USAGE

isosurface name, map, level [, (selection) [, buffer [, state [, carve ]]]]

name = the name for the new mesh isosurface object.

map = the name of the map object to use for computing the mesh.

level = the contour level.

selection = an atom selection about which to display the mesh with
 an additional "buffer" (if provided).

state = the state into which the object should be loaded (default=1)
 (set state=0 to append new surface as a new state)

carve = a radius about each atom in the selection for which to
 include density. If "carve" is not provided, then the whole
 brick is displayed.

NOTES

If the surface object already exists, then the new surface will be
 appended onto the object as a new state (unless you indicate a state).
iterate

DESCRIPTION

"iterate" iterates over an expression with a separate name space for each atom. However, unlike the "alter" command, atomic properties can not be altered. Thus, "iterate" is more efficient than "alter".

It can be used to perform operations and aggregations using atomic selections, and store the results in any global object, such as the predefined "stored" object.

The local namespace for "iterate" contains the following names:

- name, resn, resi, chain, alt, elem,
- q, b, segi, and type (ATOM,HETATM),
- partial_charge, formal_charge,
- text_type, numeric_type, ID

All strings in the expression must be explicitly quoted. This operation typically takes a second per thousand atoms.

USAGE

iterate (selection),expression

EXAMPLES

stored.net_charge = 0
iterate (all),stored.net_charge = stored.net_charge + partial_charge

stored.names = []
iterate (all),stored.names.append(name)

SEE ALSO

iterate_state, alter, alter_state

iterate_state

DESCRIPTION

"iterate_state" is to "alter_state" as "iterate" is to "alter"

USAGE

iterate_state state,(selection),expression

EXAMPLES

stored.sum_x = 0.0

iterate
iterate 1, (all), stored.sum_x = stored.sum_x + x

SEE ALSO
iterate, alter, alter_state

---

**keyword**

dict() -> new empty dictionary.
dict(mapping) -> new dictionary initialized from a mapping object's
(key, value) pairs.
dict(seq) -> new dictionary initialized as if via:
    d = {}
    for k, v in seq:
        d[k] = v

---

**label**

DESCRIPTION

"label" labels one or more atoms properties over a selection using
the python evaluator with a separate name space for each atom. The
symbols defined in the name space are:

    name, resn, resi, chain, q, b, segi, type (ATOM, HETATM)
    formal_charge, partial_charge, numeric_type, text_type

All strings in the expression must be explicitly quoted. This
operation typically takes several seconds per thousand atoms
altered.

To clear labels, simply omit the expression or set it to "."

USAGE

    label (selection), expression

EXAMPLES

    label (chain A), chain
    label (n;ca), "%s-%s" % (resn, resi)
    label (resi 200), "%1.3f" % partial_charge

---

**load**

DESCRIPTION

"load" reads several file formats. The file extension is used to
determine the format. PDB files must end in ".pdb", MOL files must
end in ".mol", Macromodel files must end in ".mod", XPLOR
maps must end in ".xplor", CCP4 maps must end in ".ccp4",
Raster3D input (Molscript output) must end in ".r3d", PyMOL session
files must end in ".pse"

Pickled ChemPy models with a ".pkl" can also be directly read.
If an object is specified, then the file is loaded into that object. Otherwise, an object is created with the same name as the file prefix.

**USAGE**

```
load filename [,object [,state [,format [,finish [,discrete ]]]]]
```

**PYMOL API**

```
cmd.load( filename [,object [,state [,format [,finish [,discrete ]]]]]
```

**NOTES**

You can override the file extension by giving a format string:

- `'pdb'` : PDB
- `'mmod'` : Macromodel
- `'xyz'` : Tinker
- `'ccl'` : ChemDraw3D
- `'mol'` : MDL MOL-file
- `'sdf'` : MDL SD-file
- `'xplor'` : X-PLOR/CNS map
- `'ccp4'` : CCP4 map
- `'callback'` : PyMOL Callback object (PyOpenGL)
- `'cgo'` : compressed graphics object (list of floats)
- `'trj'` : AMBER trajectory (use load_traj command for more control)
- `'top'` : AMBER topology file
- `'rst'` : AMBER restart file
- `'cex'` : Metaphorics CEX format
- `'pse'` : PyMOL Session file

**SEE ALSO**

```
save
```

---

**load_brick**

Temporary routine for GAMESS-UK project.

---

**load_callback**

**DESCRIPTION**

"`load_callback`" is used to load a generic Python callback object. These objects are called every time the screen is updated and can be used to trigger OpenGL rendering calls (such as with PyOpenGL).

**PYMOL API**

```
cmd.load_callback(object,name,state,finish,discrete)
```

---

**load_cgo**

**DESCRIPTION**

"`load_cgo`" is used to load a compiled graphics object, which is actually a list of floating point numbers built using the constants in the `$PYMOL_PATH/modules/pymol/cgo.py` file.
load_map

Temporary routine for the Phenix project.

load_model

DESCRIPTION

"load_model" reads a ChemPy model into an object

PYMOL API

    cmd.load_model(model, object [,state [,finish [,discrete ]]])

load_object

DESCRIPTION

"load_object" is a general developer function for loading Python objects into PyMOL.

PYMOL API

    cmd.load_object(type,object,name,state=0,finish=1,discrete=0)

    type = one of the numeric cmd.loadable types
    object =
    name = object name (string)
    finish = perform (1) or defer (0) post-processing of structure after load
    discrete = treat each state as an independent, unrelated set of atoms

load_traj

DESCRIPTION

"load_traj" reads trajectory files (currently just AMBER files).
The file extension is used to determine the format.

AMBER files must end in ".trj"

USAGE

    load_traj filename [,object [,state [,format [,interval [,average ]
    [,start [,stop [,max [,selection [,image [,shift ]]]]]]]]]]]

PYMOL API

    cmd.load_traj(filename,object='',state=0,format='',interval=1,
NOTES

You must first load a corresponding topology file before attempting to load a trajectory file.

PyMOL does not know how to wrap the truncated octahedron used by Amber. You will need to use the "ptraj" program first to do this.

The average option is not a running average. To perform this type of average, use the "smooth" command after loading the trajectory file.

SEE ALSO

load

ls

DESCRIPTION

List contents of the current working directory.

USAGE

ls [pattern]
dir [pattern]

EXAMPLES

ls
ls *.pml

SEE ALSO

cd, pwd, system

map_double

DESCRIPTION

"map_double" resamples a map at twice the current resolution. The amount of memory required to store the map will increase eight-fold.

USAGE

map_double map_name, state

map_new

state > 0: do indicated state
state = 0: independent states in independent extents
state = -1: current state
state = -2: independent states in unified extent
state = -3: all states in one map

map_set_border

DESCRIPTION

"map_set_border" is a function (reqd by PDA) which allows you to set the
level on the edge points of a map

USAGE

map_set_border <name>,<level>

NOTES

unsupported.

SEE ALSO

load

mappend

DESCRIPTION

USAGE

mappend frame : command

PYMOL API

EXAMPLE

NOTES

SEE ALSO

mset, mplay, mstop

mask

DESCRIPTION

"mask" makes it impossible to select the indicated atoms using the mouse. This is useful when you are working with one molecule in front of another and wish to avoid accidentally selecting atoms in the background.

USAGE

mask (selection)

PYMOL API
cmd.mask( string selection="(all)" )

SEE ALSO

unmask, protect, deprotect, mouse

---

**mclear**

**DESCRIPTION**

"mclear" clears the movie frame image cache.

**USAGE**

mclear

**PYMOL API**

cmd.mclear()

---

**mdo**

**DESCRIPTION**

"mdo" sets up a command to be executed upon entry into the specified frame of the movie. These commands are usually created by a PyMOL utility program (such as util.mrock). Command can actually contain several commands separated by semicolons ';'

**USAGE**

mdo frame : command

**PYMOL API**

cmd.mdo( int frame, string command )

**EXAMPLE**

// Creates a single frame movie involving a rotation about X and Y
load test.pdb
mset 1
mdo 1, turn x,5; turn y,5;
mplay

**NOTES**

The "mset" command must first be used to define the movie before "mdo" statements will have any effect. Redefinition of the movie clears any existing mdo statements.

**SEE ALSO**

mset, mplay, mstop

---
**mdump**

**DESCRIPTION**

"mdump" dumps the current set of movie commands

**USAGE**

mdump

**PYMOL API**

`cmd.mdump()`

**SEE ALSO**

mplay, mset, mdo, mclear, mmatrix

---

**mem**

**DESCRIPTION**

"mem" Dumps current memory state to standard output. This is a debugging feature, not an official part of the API.

---

**meter_reset**

**DESCRIPTION**

"meter_reset" resets the frames per second counter

**USAGE**

meter_reset

---

**middle**

**DESCRIPTION**

"middle" goes to the middle of the movie.

**USAGE**

middle

**PYMOL API**

`cmd.middle()`
**mmatrix**

**DESCRIPTION**

"mmatrix" sets up a matrix to be used for the first frame of the movie.

**USAGE**

```
mmatrix {clear|store|recall}
```

**PYMOL API**

```
cmd.mmatrix( string action )
```

**EXAMPLES**

```
mmatrix store
```

---

**move**

**DESCRIPTION**

"move" translates the camera about one of the three primary axes.

**USAGE**

```
move axis,distance
```

**EXAMPLES**

```
move x,3
move y,-1
```

**PYMOL API**

```
cmd.move( string axis, float distance )
```

**SEE ALSO**

`turn`, `rotate`, `translate`, `zoom`, `center`, `clip`

---

**mplay**

**DESCRIPTION**

"mplay" starts the movie.

**USAGE**

```
mplay
```

**PYMOL API**

```
cmd.mplay()
```

**SEE ALSO**

`mmatrix`
**mpng**

**DESCRIPTION**

"mpng" writes a series of numbered movie frames to png files with the specified prefix. If the "ray_trace_frames" variable is non-zero, these frames will be ray-traced. This operation can take several hours for a long movie.

Be sure to disable "cache_frames" when issuing this operation on a long movie (>100 frames) to avoid running out of memory.

**USAGE**

mpng prefix [, first [, last]]

Options "first" and "last" can be used to specify an inclusive interval over which to render frames. Thus, you can write a smart Python program that will automatically distribute rendering over a cluster of workstations. If these options are left at zero, then the entire movie will be rendered.

**PYMOL API**

cmd.mpng( string prefix, int first=0, int last=0 )

---

**mset**

**DESCRIPTION**

"mset" sets up a relationship between molecular states and movie frames. This makes it possible to control which states are shown in which frame.

**USAGE**

mset specification

**PYMOL API**

cmd.mset( string specification )

**EXAMPLES**

mset 1         // simplest case, one state -> one frame
mset 1 x10     // ten frames, all corresponding to state 1
mset 1 x30 1 −15 15 x30 15 −1
    // more realistic example:
    // the first thirty frames are state 1
    // the next 15 frames pass through states 1-15
    // the next 30 frames are of state 15
    // the next 15 frames iterate back to state 1

**SEE ALSO**

mpng
mstop

DESCRIPTION

"mstop" stops the movie.

USAGE

mstop

PYMOL API

cmd.mstop()

SEE ALSO

mplay, mset, mdo, mclear, mmatrix

orient

DESCRIPTION

"orient" aligns the principal components of the atoms in the selection with the XYZ axes. The function is similar to the orient command in X-PLOR.

USAGE

orient object-or-selection [, state]
orient (selection)

PYMOL API

cmd.orient( string object-or-selection [, state = 0] )

NOTES

state = 0 (default) use all coordinate states
state = -1 use only coordinates for the current state
state > 0 use coordinates for a specific state

SEE ALSO

zoom, origin, reset

origin

DESCRIPTION

"origin" sets the center of rotation about a selection. If an object name is specified, it can be used to set the center of rotation for the object's TTT matrix.
USAGE

    origin selection [, object [,position, [, state]]]
    origin (selection)
    origin position=[1.0,2.0,3.0]

PYMOL API

    cmd.origin( string object-or-selection )

NOTES

    state = 0 (default) use all coordinate states
    state = -1 use only coordinates for the current state
    state > 0 use coordinates for a specific state

SEE ALSO

    zoom, orient, reset

---

**pair_fit**

DESCRIPTION

"pair_fit" fits a set of atom pairs between two models. Each atom in each pair must be specified individually, which can be tedious to enter manually. Script files are recommended when using this command.

USAGE

    pair_fit (selection), (selection), [ (selection), (selection) [ ... ] ]

SEE ALSO

    fit, rms, rms_cur, intra_fit, intra_rms, intra_rms_cur

---

**png**

DESCRIPTION

"png" writes a png format image file of the current image to disk.

USAGE

    png filename

PYMOL API

    cmd.png( string file )
**protect**

**DESCRIPTION**

"protect" protects a set of atoms from transformations performed using the editing features. This is most useful when you are modifying an internal portion of a chain or cycle and do not wish to affect the rest of the molecule.

**USAGE**

protect (selection)

**PYMOL API**

cmd.protect(string selection)

**SEE ALSO**

deprotect, mask, unmask, mouse, editing

**push_undo**

**DESCRIPTION**

"push_undo" stores the currently conformations of objects in the selection onto their individual kill rings.

**USAGE**

push_undo (all)

**SEE ALSO**

undo, redo

**pwd**

**DESCRIPTION**

Print current working directory.

**USAGE**

pwd

**SEE ALSO**

cd, ls, system
quit

DESCRIPTION
"quit" terminates the program.

USAGE
quit

PYMOL API
cmd.quit()

---

ray

DESCRIPTION
"ray" creates a ray-traced image of the current frame. This can take some time (up to several minutes, depending on image complexity).

USAGE
ray [width,height [,renderer [,angle [,shift ]]]]

angle and shift can be used to generate matched stereo pairs

EXAMPLES

ray
ray 1024,768
ray renderer=0

PYMOL API

cmd.ray(int width,int height,int renderer=-1,float shift=0)

NOTES
renderer = -1 is default (use value in ray_default_renderer)
renderer = 0 uses PyMOL's internal renderer
renderer = 1 uses PovRay's renderer. This is Unix-only
and you must have "x-povray" in your path. It utilizes two temporary files: "tmp_pymol.pov" and "tmp_pymol.png".

SEE ALSO

"help faster" for optimization tips with the builtin renderer.
"help povray" for how to use PovRay instead of PyMOL's built-in ray-tracing engine.
**read_mmodstr**

**DESCRIPTION**

"read_mmodstr" reads a macromodel format structure from a Python string.

**read_molstr**

**DESCRIPTION**

"read_molstr" reads an MDL MOL format file as a string

**PYMOL API ONLY**

```
cmd.read_molstr( string molstr, string name, int state=0,
  int finish=1, int discrete=1 )
```

**NOTES**

"state" is a 1-based state index for the object, or 0 to append.

"finish" is a flag (0 or 1) which can be set to zero to improve performance when loading large numbers of objects, but you must call "finish_object" when you are done.

"discrete" is a flag (0 or 1) which tells PyMOL that there will be no overlapping atoms in the file being loaded. "discrete" objects save memory but can not be edited.

**read_pdbstr**

**DESCRIPTION**

"read_pdbstr" in an API-only function which reads a pdb file from a Python string. This feature can be used to load or update structures into PyMOL without involving any temporary files.

**PYMOL API ONLY**

```
cmd.read_pdbstr( string pdb-content, string object name
  [ ,int state [ ,int finish [ ,int discrete ] ] ] )
```

**NOTES**

"state" is a 1-based state index for the object.

"finish" is a flag (0 or 1) which can be set to zero to improve performance when loading large numbers of objects, but you must call "finish_object" when you are done.

"discrete" is a flag (0 or 1) which tells PyMOL that there will be no overlapping atoms in the PDB files being loaded. "discrete" objects save memory but can not be edited.
**read_xplorstr**

**DESCRIPTION**

"read_xplorstr" is an API-only function which reads an XPLOR map from a Python string. This feature can be used to bypass temporary files.

**PYMOL API ONLY**

```python
cmd.read_xplorstr(string xplor-content, string object name [ , int state ] )
```

**NOTES**

"state" is a 1-based state index for the object.

---

**rebuild**

**DESCRIPTION**

"rebuild" forces PyMOL to recreate geometric objects in case any of them have gone out of sync.

**USAGE**

```python
rebuild [selection [, representation ]]
```

**PYMOL API**

```python
cmd.rebuild(string selection = 'all', string representation = 'everything')
```

**SEE ALSO**

refresh

---

**recolor**

**DESCRIPTION**

"rebuild" forces PyMOL to reapply colors to geometric objects in case any of them have gone out of sync.

**USAGE**

```python
recolor [selection [, representation ]]
```

**PYMOL API**

```python
cmd.recolor(string selection = 'all', string representation = 'everything')
```

**SEE ALSO**

recolor
**redo**

**DESCRIPTION**
"redo" reapplies the conformational change of the object currently being edited.

**USAGE**
redo

**SEE ALSO**
undo, push_undo

**refresh**

**DESCRIPTION**
"refresh" causes the scene to be refresh as soon as it is safe to do so.

**USAGE**
refresh

**PYPiOL API**
cmd.refresh()

**SEE ALSO**
rebuild

**reinitialize**

**DESCRIPTION**
"reinitialize" reinitializes PyMOL

**USAGE**
reinitialize

**remove**

**DESCRIPTION**
"remove" eliminates a selection of atoms from models.

**USAGE**
remove (selection)
PYMOL API

    cmd.remove( string selection )

EXAMPLES

    remove ( resi 124 )

SEE ALSO

    delete

---

**remove_picked**

DESCRIPTION

"remove_picked" removes the atom or bond currently picked for editing.

USAGE

    remove_picked [hydrogens]

PYMOL API

    cmd.remove_picked(integer hydrogens=1)

NOTES

This function is usually connected to the DELETE key and "CTRL-D".

By default, attached hydrogens will also be deleted unless hydrogen-flag is zero.

SEE ALSO

    attach, replace

---

**rename**

DESCRIPTION

"rename" creates new atom names which are unique within residues.

USAGE

CURRENT

    rename object-name [ ,force ]

    force = 0 or 1 (default: 0)

PROPOSED

    rename object-or-selection,force

PYMOL API

    remove_picked
CURRENT
   cmd.rename( string object-name, int force )

PROPOSED
   cmd.rename( string object-or-selection, int force )

NOTES
   To regerate only some atom names in a molecule, first clear them
   with an "alter (sele),name=''", then use "rename"

SEE ALSO
   alter

---

replace

DESCRIPTION
   "replace" replaces the picked atom with a new atom.

USAGE
   replace element, geometry, valence [,h_fill [,name ]]

PYMOL API
   cmd.replace(string element, int geometry, int valence,
             int h_fill = 1, string name = "" )

NOTES
   Immature functionality. See code for details.

SEE ALSO
   remove, attach, fuse, bond, unbond

---

reset

DESCRIPTION
   "reset" restores the rotation matrix to identity, sets the origin
   to the center of mass (approx.) and zooms the window and clipping
   planes to cover all objects.

USAGE
   reset

PYMOL API
   cmd.reset ( )
**rewind**

**DESCRIPTION**

"rewind" goes to the beginning of the movie.

**USAGE**

rewind

**PYMOL API**

cmd.rewind()

---

**rms**

**DESCRIPTION**

"rms" computes a RMS fit between two atom selections, but does not transform the models after performing the fit.

**USAGE**

rms (selection), (target−selection)

**EXAMPLES**

fit ( mutant and name ca ), ( wildtype and name ca )

**SEE ALSO**

fit, rms_cur, intra_fit, intra_rms, intra_rms_cur, pair_fit

---

**rms_cur**

**DESCRIPTION**

"rms_cur" computes the RMS difference between two atom selections without performing any fitting.

**USAGE**

rms_cur (selection), (selection)

**SEE ALSO**

fit, rms, intra_fit, intra_rms, intra_rms_cur, pair_fit

---

**rock**

**DESCRIPTION**

"rock" toggles Y axis rocking.
**rotate**

**DESCRIPTION**

"rotate" can be used to rotate the atomic coordinates of a molecular object. Behavior differs depending on whether or not the "object" parameter is specified.

If object is None, then rotate rotates the atomic coordinates according to the axes and angle for the selection and state provided. All representation geometries will need to be regenerated to reflect the new atomic coordinates.

If object is set to an object name, then selection and state are ignored and instead of translating the atomic coordinates, the object's representation display matrix is modified. This option is for use in animations only.

**USAGE**

rotate axis, angle [,selection [,state [,camera [,object [,origin]]]]]

**PYMOL API**

cmd.rotate(list−or−string axis, string selection = "all", int state = 0, int camera = 1, string object = None)

**EXAMPLES**

rotate x, 45, pept

**NOTES**

if state = 0, then only visible state(s) are affected.
if state = −1, then all states are affected.

---

**save**

**DESCRIPTION**

"save" writes selected atoms to a file. The file format is autodetected if the extension is ".pdb", ".pse", ".mol", ".mmod", or ".pkl"

Note that if the file extension ends in ".pse" (PyMOL Session), the complete PyMOL state is always saved to the file (the selection and state parameters are thus ignored).
save file [, (selection) [, state [, format]] ]

PYMOL API

    cmd.save(file, selection, state, format)

SEE ALSO

    load, get_model

scene

DESCRIPTION

"scene" makes it possible to save and restore multiple scenes within a single session. A scene consists of the view, all object activity information, all atom-wise visibility, color, representations, and the global frame index.

USAGE

    scene key [, action [, message [, view [, color [, active [, rep [, frame]]]]]]]
    scene *

    key can be any string
    action should be 'store' or 'recall' (default: 'recall')

    view: 1 or 0 controls whether the view is stored
    color: 1 or 0 controls whether colors are stored
    active: 1 or 0 controls whether activity is stored
    rep: 1 or 0 controls whether the representations are stored
    frame: 1 or 0 controls whether the frame is stored

PYMOL API

    cmd.scene(string key, string action, string-or-list message, int view,
             int color, int active, int rep, int frame)

EXAMPLES

    scene F1, store
    scene F2, store, This view shows you the critical hydrogen bond.

    scene F1
    scene F2

NOTES

    Scenes F1 through F12 are automatically bound to function keys provided that "set_key" hasn't been used to redefine the behaviour of the respective key.

SEE ALSO

    view, set_view, get_view

DEVELOPMENT TO DO

    Add support for save/restore of a certain global and
object-and-state specific settings, such as: state, surface_color, ribbon_color, stick_color, transparency, sphere_transparency, etc. This would probably best be done by defining a class of "scene" settings which are treated in this manner. The current workaround is to create separate objects which are enabled/disabled differentially.

**sculpt_activate**

undocumented.

**sculpt_deactivate**

undocumented

**sculpt_iterate**

undocumented.

**sculpt_purge**

undocumented

**select**

**DESCRIPTION**

"select" creates a named selection from an atom selection.

**USAGE**

```
select (selection)
select name, (selection)
select name = (selection)            # (DEPRECATED)
```

**PYMOL API**

```
cmd.select(string name, string selection)
```

**EXAMPLES**

```
select near, (ll expand 8)
select near, (ll expand 8)
select bb, (name ca,n,c,o )
```

**NOTES**

'help selections' for more information about selections.
**set**

**DESCRIPTION**

"set" changes one of the PyMOL state variables,

**USAGE**

    set name, value [,object-or-selection [,state ]]  
    set name = value          # (DEPRECATED)

**PYMOL API**

    cmd.set ( string name, string value,  
        string selection='', int state=0,  
        int quiet=0, int updates=1 )

**NOTES**

The default behavior (with a blank selection) changes the global settings database. If the selection is 'all', then the settings database in all individual objects will be changed. Likewise, for a given object, if state is zero, then the object database will be modified. Otherwise, the settings database for the indicated state within the object will be modified.

If a selection is provided, then all objects in the selection will be affected.

---

**set_color**

**DESCRIPTION**

"set_color" defines a new color with color indices (0.0−1.0)

**USAGE**

    set_color name, [ red=float, green=float, blue=float ]
    set_color name = [ red=float, green=float, blue=float ]  
        # (DEPRECATED)

**PYMOL API**

    cmd.set_color( string name, float-list rgb )

**EXAMPLES**

    set_color red = [ 1.0, 0.0, 0.0 ]

---

**set_geometry**

**DESCRIPTION**

"set_geometry" changes PyMOL's assumptions about the proper valence
and geometry of the picked atom.

**USAGE**

```
set_geometry geometry, valence
```

**PYMOL API**

```
cmd.set_geometry(int geometry, int valence)
```

**NOTES**

Immature functionality. See code for details.

**SEE ALSO**

remove, attach, fuse, bond, unbond

---

**set_key**

**DESCRIPTION**

"set_key" binds a specific python function to a key press.

**PYMOL API (ONLY)**

```
cmd.set_key(string key, function fn, tuple arg=(), dict kw={})
```

**PYTHON EXAMPLE**

```python
def color_blue(object):
    cmd.color("blue", object)

cmd.set_key('F1', make_it_blue, ("object1"))
    # would turn object1 blue when the F1 key is pressed and

cmd.set_key('F2', make_it_blue, ("object2"))
    # would turn object2 blue when the F2 key is pressed.

cmd.set_key('CTRL-C', cmd.zoom)
cmd.set_key('ALT-A', cmd.turn, ('x', 90))
```

**KEYS WHICH CAN BE REDEFINED**

F1 to F12
left, right, pgup, pgdn, home, insert
CTRL-A to CTRL-Z
ALT-0 to ALT-9, ALT-A to ALT-Z

**SEE ALSO**

button
**set_symmetry**

**DESCRIPTION**

"set_symmetry" can be used to define or redefine the crystal and spacegroup parameters for a molecule or map object.

**USAGE**

`set_symmetry selection, a, b, c, alpha, beta, gamma, spacegroup`

**PYMOL API**

`cmd.set_symmetry(string selection, float a, float b, float c, float alpha, float beta, float gamma, string spacegroup)`

**NOTES**

The new symmetry will be defined for every object referenced by the selection.

---

**set_title**

**DESCRIPTION**

"set_title" attaches a text string to the state of a particular object which can be displayed when the state is active. This is useful for display the energies of a set of conformers.

**USAGE**

`set_title object, state, text`

**PYMOL API**

`cmd.set_title(string object, int state, string text)`

---

**set_view**

**DESCRIPTION**

"set_view" sets viewing information for the current scene, including the rotation matrix, position, origin of rotation, clipping planes, and the orthoscopic flag.

**USAGE**

`set_view (...)` where ... is 18 floating point numbers

**PYMOL API**

`cmd.set_view(string-or-sequence view)`
**show**

**DESCRIPTION**

"show" turns on atom and bond representations.

The available representations are:

- lines
- spheres
- mesh
- ribbon
- cartoon
- sticks
- dots
- surface
- labels
- extent
- nonbonded nb_spheres

**USAGE**

- `show`
- `show representation [,object]`
- `show representation [,,(selection)]`
- `show (selection)`

**PYMOL API**

- `cmd.show( string representation="", string selection="" )`

**EXAMPLES**

- `show lines, (name ca or name c or name n)`
- `show ribbon`

**NOTES**

- "selection" can be an object name
- "show" alone will turn on lines for all bonds.

**SEE ALSO**

- `hide`, `enable`, `disable`

---

**smooth**

**DESCRIPTION**

"smooth" performs a window average over a series of states. This type of averaging is often used to suppress high-frequency vibrations in a molecular dynamics trajectory.

**USAGE**

- `smooth [selection [, passes [,window [,first [,last [,ends]]]]]]`

**SEE ALSO**

- `load_traj`

**ARGUMENTS**

- `ends (0 or 1) controls whether or not the end states are also smoothed using a weighted asymmetric window`

**NOTES**
This function is not memory efficient. For reasons of flexibility, it uses two additional copies of every atomic coordinate for the calculation. If you are memory-constrained in visualizing MD trajectories, then you may want to use an external tool such as Amber's ptraj to perform smoothing before loading coordinates into PyMOL.

**sort**

**DESCRIPTION**

"sort" reorders atoms in the structure. It usually only necessary to run this routine after an "alter" command which has modified the names of atom properties. Without an argument, sort will resort all atoms in all objects.

**USAGE**

```
sort [object]
```

**PYMOL API**

```
cmd.sort(string object)
```

**SEE ALSO**

alter

**space**

**DESCRIPTION**

"space" selects a color palette (or color space).

**USAGE**

```
space space-name
```

**PYMOL API**

```
cmd.space(string space_name)
```

**EXAMPLES**

```
space rgb
space cmyk
space pymol
```

**spectrum**

**DESCRIPTION**

"spectrum" colors atoms using a spectrum

**USAGE**

```
sort
```
**spheroid**

**DESCRIPTION**

"spheroid" averages trajectory frames together to create an ellipsoid-like approximation of the actual anisotropic motion exhibited by the atom over a series of trajectory frames.

**USAGE**

```python
spheroid object,average
average = number of states to average for each resulting spheroid state
```

**splash**

**DESCRIPTION**

"splash" shows the splash screen information.

**USAGE**

splash

**stereo**

**DESCRIPTION**

"stereo" activates or deactivates stereo mode.

**USAGE**

```python
stereo on
stereo off
stereo swap
stereo crosseye
stereo walleye
stereo quadbuffer
```

**NOTES**

quadbuffer is the default stereo mode if hardware stereo is available otherwise, crosseye is the default.

**PYMOL API**

```python
cmd.stereo(string state="on")
```
**symexp**

**DESCRIPTION**

"symexp" creates all symmetry related objects for the specified object that occurs within a cutoff about an atom selection. The new objects are labeled using the prefix provided along with their crystallographic symmetry operation and translation.

**USAGE**

symexp prefix = object, (selection), cutoff

**PYMOL API**

```
cmd.symexp( string prefix, string object, string selection, float cutoff)
```

**SEE ALSO**

load

---

**sync**

**DESCRIPTION**

"sync" is an API-only function which waits until all current commands have been executed before returning. A timeout can be used to insure that this command eventually returns.

**PYMOL API**

```
cmd.sync(float timeout=1.0, float poll=0.05)
```

**SEE ALSO**

frame

---

**system**

**DESCRIPTION**

"system" executes a command in a subshell under Unix or Windows.

**USAGE**

system command

**PYMOL API**

```
cmd.system(string command, int sync=1)
```

**NOTES**

async can only be specified from the Python level (not the command language) if async is 0 (default), then the result code from "system" is returned in r
if async is 1, then the command is run in a separate thread whose object is
returned

SEE ALSO

ls, cd, pwd

torsion

DESCRIPTION

"torsion" rotates the torsion on the bond currently
picked for editing. The rotated fragment will correspond
to the first atom specified when picking the bond (or the
nearest atom, if picked using the mouse).

USAGE

torsion angle

PYMOL API

cmd.torsion( float angle )

SEE ALSO

edit, unpick, remove_picked, cycle_valence

translate

DESCRIPTION

"translate" can be used to translate the atomic coordinates of a
molecular object. Behavior differs depending on whether or not the
"object" parameter is specified.

If object is None, then translate translates atomic coordinates
according to the vector provided for the selection and in the state
provided. All representation geometries will need to be
regenerated to reflect the new atomic coordinates.

If object is set to an object name, then selection and state are
ignored and instead of translating the atomic coordinates, the
object's overall representation display matrix is modified. This
option is for use in animations only.

The "camera" option controls whether the camera or the model's
axes are used to interpret the translation vector.

USAGE

translate vector [,selection [,state [,camera [,object ]]]]

PYMOL API

    cmd.translate(list vector, string selection = "all", int state = 0,
int camera = 1, string object = None

EXAMPLES

translate [1,0,0], name ca

NOTES

if state = 0, then only visible state(s) are affected.
if state = -1, then all states are affected.

---

**turn**

**DESCRIPTION**

"turn" rotates the camera about one of the three primary axes, centered at the origin.

**USAGE**

turn axis, angle

**EXAMPLES**

turn x,90
turn y,45

**PYMOL API**

cmd.turn( string axis, float angle )

**SEE ALSO**

move, rotate, translate, zoom, center, clip

---

**unbond**

**DESCRIPTION**

"unbond" removes all bonds between two selections.

**USAGE**

unbond atom1,atom2

**PYMOL API**

cmd.unbond(selection atom1="(lb)", selection atom2="(rb)")

**SEE ALSO**

bond, fuse, remove_picked, attach, detach, replace
**undo**

**DESCRIPTION**

"undo" restores the previous conformation of the object currently being edited.

**USAGE**

undo

**SEE ALSO**

redo, push_undo

---

**unmask**

**DESCRIPTION**

"unmask" reverses the effect of "mask" on the indicated atoms.

**PYMOL API**

```python
cmd.unmask( string selection="(all)" )
```

**USAGE**

unmask (selection)

**SEE ALSO**

mask, protect, deprotect, mouse

---

**unpick**

**DESCRIPTION**

"unpick" deletes the special "pk" atom selections (pk1, pk2, etc.) used in atom picking and molecular editing.

**USAGE**

unpick

**PYMOL API**

```python
cmd.unpick()
```

**SEE ALSO**

edit
unset

DESCRIPTION

"unset" undefines an object-specific or state-specific setting so that the global setting will be in effect.

USAGE

unset name [,selection [,state ]]

PYMOL API

    cmd.unset ( string name, string selection='all',
                int state=0, int quiet=0, int updates=1 )

update

DESCRIPTION

"update" transfers coordinates from one selection to another.

USAGE

update (target−selection),(source−selection)

EXAMPLES

update target,(variant)

NOTES

Currently, this applies across all pairs of states. Fine control will be added later.

SEE ALSO

load

view

DESCRIPTION

"view" makes it possible to save and restore viewpoints on a given scene within a single session.

USAGE

    view key[,action]
    view *

key can be any string
action should be 'store' or 'recall' (default: 'recall')

PYMOL API

    cmd.view(string key,string action)
**VIEWs**

Views F1 through F12 are automatically bound to function keys provided that "set_key" hasn't been used to redefine the behaviour of the respective key, and that a "scene" hasn't been defined for that key.

**EXAMPLES**

view 0,store
view 0

**SEE ALSO**

scene, set_view, get_view

---

**viewport**

**DESCRIPTION**

"viewport" changes the size of the viewing port (and thus the size of all png files subsequently output)

**USAGE**

viewport width, height

**PYMOL API**

cmd.viewport(int width, int height)

---

**wizard**

**DESCRIPTION**

"wizard" launches on of the built-in wizards. There are special Python scripts which work with PyMOL in order to obtain direct user interaction and easily perform complicated tasks.

**USAGE**

wizard name

**PYMOL API**

cmd.wizard(string name)

**EXAMPLE**

wizard distance  # launches the distance measurement wizard
**zoom**

**DESCRIPTION**

"zoom" scales and translates the window and the origin to cover the atom selection.

**USAGE**

```plaintext
zoom [ selection [,buffer [, state [, complete ]]]]
```

**EXAMPLES**

```plaintext
zoom
zoom complete=1
zoom (chain A)
zoom 142/
```

**PYMOL API**

```plaintext
cmd.zoom( string selection, float buffer=0.0,
         int state=0, int complete=0 )
```

**NOTES**

- `state = 0` (default) use all coordinate states
- `state = -1` use only coordinates for the current state
- `state > 0` use coordinates for a specific state

- `complete = 0` or `1`:
  - Normally the zoom command tries to guess an optimal zoom level for visualization, balancing closeness against occasional clipping of atoms out of the field of view. You can change this behavior by setting the complete option to `1`, which will guarantee that the atom positions for the entire selection will fit in the field of an orthoscopic view. To absolutely prevent clipping, you may also need to add a buffer (typically 2 Å) to account for the perspective transformation and for graphical representations which extend beyond the atom coordinates.

**SEE ALSO**

- origin, orient, center