

# PyMOL: Intermediate

Using the command line

Keeping a command log

Commands, Selections and Settings

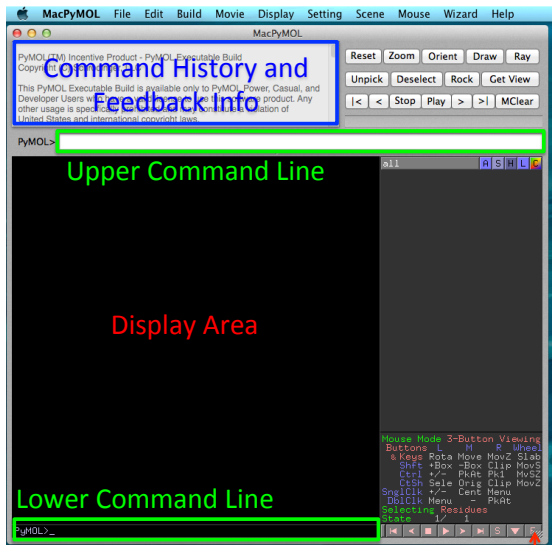
Defining default settings

Writing and executing simple scripts

# The PyMOL Interface: Command Line (CL)

Use **up** and **down** arrow to scroll through command history

**<ESC>** toggles display of feedback text in the display area (useful when working in full-screen mode)



Full screen on/off

# General Command Syntax

**command parameter1[, parameter2[, parameter3]]**



parameter1 is always required



square brackets denote optional Parameters

**<TAB>**

In the empty command line list of all commands recognized by the current version of PyMOL

**c<TAB>**

list of all commands that start with c

**command ? (e.g. show ?)**

Usage: show [ representation [, selection ]]

**help command (e.g. help show)**

DESCRIPTION

"show" turns on representations for objects and selections.

... With no arguments, "show" alone turns on lines for all bonds and nonbonded for all atoms in all molecular objects.

# Combining Different Representations: GUI

open file 3K8Y.pdb with PyMOL

Object Menu:

3K8Y: Hide: everything

3K8Y: Show: cartoon

3K8Y: C: spectrum: rainbow \*/CA

3K8Y: Show: organic : spheres

select these by left click,

sele: C: by element : CHNOS

select C-alpha atoms at either  
end of the protein chain

sele: L: residues

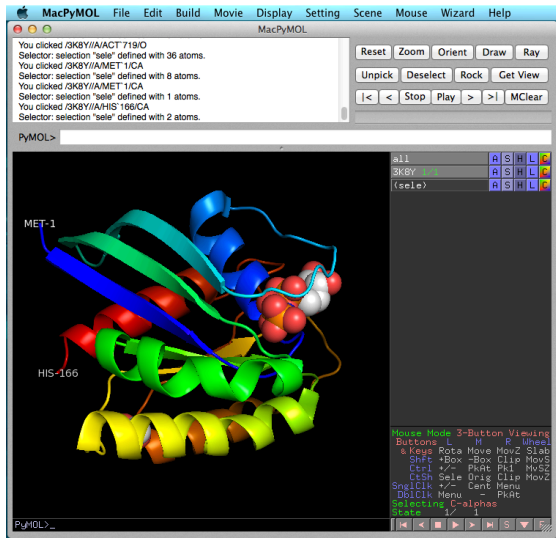
rotate so that both ends of the  
chain are clearly visible

For printing:

Display: Background: white

click on the “ray” button

File: save image: png: fig1a.png



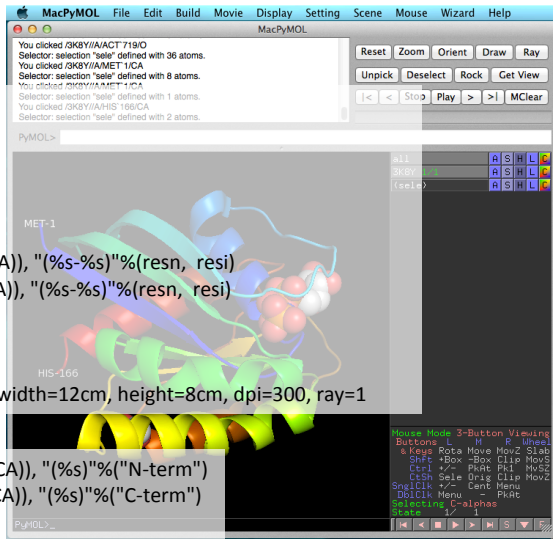


# Combining Different Representations:

## Command-line script to do the same:

```
load ~/pymol/pdb/3K8Y.pdb
hide all
show cartoon
util.chainbow polymer
show spheres, organic
util.cbaw organic
label (first (polymer and name CA)), "(%s-%s)"%(resn, resi)
label (last (polymer and name CA)), "(%s-%s)"%(resn, resi)
#rotate so that both ends of the
#chain are clearly visible
bg_color white
png ~/pymol/figures/fig1b.png, width=12cm, height=8cm, dpi=300, ray=1

label (first (polymer and name CA)), "(%s)"("N-term")
label (last (polymer and name CA)), "(%s)"("C-term")
set label_size, 30
set label_position,(-1.5,1,1)
```



DEMO, play along

## Two Types of Scripts:

### PyMOL scripts (extension .pml):

Use only PyMOL commands

Commands are either in PyMOL syntax:

`cartoon type, (selection)`

or PyMOL API syntax:

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

Can either be copy-pasted into the command line (whole or in segments) or by  
`@ path/scriptname.pml`  
and are executed immediately

**This course deals  
only with .pml scripts**

### Python scripts (extension .py)

Use the **Python programming language** and can access functionalities of Python libraries (NumPy, SciPy, ChemPy, cctbx, OpenBabel ...) and interact with external command-line driven programs, e.g. APBS, Caver, etc..

Always contain at least this line near top:

**`from pymol import cmd`**

additional similar lines indicate other dependencies, e.g.

`from cctbx import sgtbx, uctbx`

Commands only in the pymol API syntax:

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

**Plugins** are installed through the **Plugin manager** or are imported by

**`run path/scriptname.pml`**

They introduce new commands defined by `cmd.extend("cmd_name", python_function)` that can be accessed through the command line or in some cases from the GUI.

# PyMOL command line

More than 300 different commands in PyMOL

**Scripts and PlugIns further expand the repertoire**

More than ~~700~~ **1400** different setting variables modify the effects of these commands

**Only a fraction of these can be accessed through the GUI**

**Use of the command line allows:**

- much better **control of atom selections**
- access to **all commands** and their **parameters**
- **keeping a log** of applied commands and parameters
- **command sequences can be prepared as a text file** (script) and copied to the command line or called by other scripts
- adaptation and **re-use of scripts**
- automation

Nobody knows all these commands by heart

**Always keep the PyMOL Wiki at hand!**

[http://pymolwiki.org/index.php/Main\\_Page](http://pymolwiki.org/index.php/Main_Page)

Google PyMOL and <command> to find things

# The PyMOL Interface:

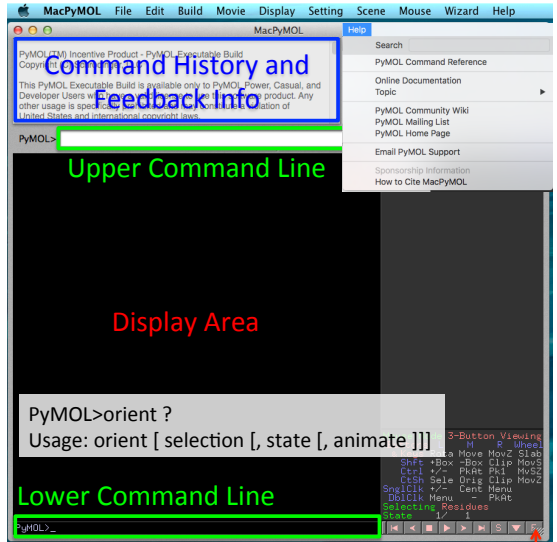
Enter **<TAB>** in empty CL  
for a list of all commands

**<TAB>** in non-empty CL:  
autocompletion  
e.g. "ori"<TAB> orient, origin

**help <command>**  
lists full information on that  
command

**<command> ?**  
to get a list of parameters

**<ESC>** toggles display of  
feedback text in the display  
area (useful when working in  
full-screen mode)



Full screen on/off

# Keep a Log

The screenshot shows the MacPyMOL application window. The 'File' menu is open, displaying options: Open..., Save Session, Save Session As..., Save Molecule..., Save Image As, Save Movie As, Log..., Resume..., Append..., Close Log, Run..., Quit, Reinitialize, and Close. A red bracket on the right side of the 'Log...' through 'Run...' options is labeled 'Keep a log'. The background shows a 3D molecular model. In the foreground, the 'Pymol01.log' file is open in a text viewer, showing a list of commands and their outputs. A red box highlights the text 'Pymol Tricks' and the instructions 'After opening Pymol, save log and open Logfile in log viewer' and 'The log will show all command that get executed by pymol, whether from command line'.

**MacPyMOL** File Edit Build Movie Display Setting Scene Mouse Wizard Help

COMPND 6 GTPASE HR...S, N-TERMINALLY PROCESSED;  
COMPND 7 ENGINEERE...  
ObjectMolecule: Read sec...  
Symmetry: Found 18 symm...  
CmdLoad: "/Users/Plueckf...  
"3KBV".  
PyMOL>viewport 500,500

PyMOL>

Open...  
Save Session  
Save Session As...  
Save Molecule...  
Save Image As  
Save Movie As  
Log...  
Resume...  
Append...  
Close Log  
Run...  
Quit  
Reinitialize  
Close

Keep a log

**Pymol Tricks**  
After opening Pymol, save log  
and open Logfile in log viewer  
The log will show all command  
that get executed by pymol,  
whether from command line

Pymol01.log

Hide Log List Clear Display Reload Ignore Sender Insert Marker Filter

SYSTEM LOG QUERIES  
All Messages  
DIAGNOSTIC AND USAGE 1...  
Diagnostic and Usage Mess...  
User Diagnostic Reports  
System Diagnostic Reports

FILES  
system.log  
~/Library/Logs  
~/Library/Logs  
Adobe  
AppleFileService  
CCC.0.log  
CCC.log  
CCC.stats  
DiagnosticReports  
EventMonitor  
FlashPlayerInstallManage...  
LKDC-setup.log  
ntfs-3g\_util.log  
slapconfig.log  
stackshot-syms.log  
stackshot.log  
Xsan  
/var/log

```
3KBV' 100|3KBV' 101|3KBV' 102| 7  
- cmd.select("sele", "(byresi(?sele) or byresi(_seeker))", enable=1)  
- cmd.delete("seeker")  
- cmd.select("seeker", "(3KBV' 198|3KBV' 199|3KBV' 200|3KBV' 201|3KBV' 202|  
3KBV' 203|3KBV' 204|3KBV' 205|3KBV' 206|3KBV' 207|3KBV' 208)")  
- cmd.select("sele", "((byresi(?sele) or byresi(_seeker))", enable=1)  
- cmd.delete("seeker")  
cmd.disable('sele')  
- cmd.select("seeker", "(3KBV' 389|3KBV' 390|3KBV' 391|3KBV' 392|3KBV' 393|  
3KBV' 394)")  
cmd.select('sele', 'none')  
- cmd.select("sele", "((byresi(?sele) or byresi(_seeker))", enable=1)  
- cmd.delete("seeker")  
cmd.show("spheres", "sele")  
- cmd.select("seeker", "(3KBV' 619|3KBV' 620|3KBV' 621|3KBV' 622|3KBV' 623|  
3KBV' 624)")  
- cmd.select("sele", "((byresi(?sele) or byresi(_seeker))", enable=1)  
- cmd.delete("seeker")  
cmd.show("spheres", "sele")  
- cmd.select("seeker", "(3KBV' 929|3KBV' 930|3KBV' 931|3KBV' 932|3KBV' 933|  
3KBV' 934)")  
- cmd.select("sele", "((byresi(?sele) or byresi(_seeker))", enable=1)  
- cmd.delete("seeker")  
cmd.show("spheres", "sele")  
select sele, resn Cys  
cmd.disable('sele')  
cmd.set('cartoon_side_chain_helper', '1', '1', quiet=0)  
cmd.select('sele', 'none')  
cmd.select('sele', "byresi((((sele) or byresi((3KBV' 537))) and not  
((byresi((3KBV' 537))) and byresi(sele))))", enable=1)  
cmd.select('sele', "byresi((((sele) or byresi((3KBV' 17))) and not  
((byresi((3KBV' 17))) and byresi(sele))))", enable=1)  
cmd.select('sele', "byresi((((sele) or byresi((3KBV' 38))) and not  
((byresi((3KBV' 38))) and byresi(sele))))", enable=1)  
cmd.disable('sele')  
select sele, resn Cys  
cmd.enable('sele')  
cmd.reset(object="3KBV")  
cmd.h_add("3KBV");cmd.sort("3KBV extend 1")
```

Size: 2 KB  
Earlier | Later | Now

## Example

DEMO, play along

```
load filename [, object]
hide [ representation [, selection ]]
show [ representation [, selection ]]
```

### Command:

### Effect:

load ~/pymol/pdb/3K8Y.pdb

loads structure 3K8Y.pdb from folder pdb  
4K8Y is shown in "lines" representation

hide

nothing displayed

show cartoon, 3K8Y

4K8Y is shown in cartoon representation

show spheres, organic

ligands GNP and ACT are shown as spheres

hide spheres, resn ACT

ligand ACT is no longer visible

show surface, polymer

protein is shown in surface representation

#now use the mouse to orient the molecule into a pleasing view

```
color color [, selection]
png filename [, width [, height [, dpi [, ray]]]]
save filename [, selection [, state [, format]]]
```

**Command:**

**Effect:**

color white, polymer	the color of the protein is changed to white
png ~/pymol/figures/fig1a.png	save image to folder "figures"
hide surface, polymer	the surface is no longer shown, we can see that the cartoon representation also changed color, although it was not visible.
png ~/pymol/figures/fig1b.png	save image to folder "figures"
	size and resolution are as in the display window
save ~/pymol/fig1a.pse	save the PyMOL session to pymol folder



## Advanced Coloring: util.cba

color atoms by atom type: Oxygen red, nitrogen blue, sulfur yellow, hydrogen white, ...  
the color of the carbon atom can be varied

**util.cbax selection**  
**util.cnc selection**

command	carbon color
util.cbag	green
util.cbac	cyan
util.cbam	light magenta
util.cbay	yellow
util.cbas	salmon
util.cbaw	white/grey
util.cbab	slate
util.cbao	bright orange
util.cbap	purple
util.cbak	pink

**util.cba** colors atoms by atom type,  
carbon atoms by the color defined  
by the last letter in the command

**util.cnc** colors atoms by atom type,  
but does not alter the color of the  
carbon atoms

**util.chainbow object**

colors each chain in the object in a  
rainbow of colors, from Nterm:blue to  
Cterm: red

**util.cbc [object]**

colors each chain in a different color

**util.cbss("object", "helixcolor", "sheetcolor", "coilcolor")**

colors object by secondary structure, if the secondary structure of the object is poorly  
defined, use command **dss selection** to re-assign secondary structure

# Defining your own colors

The color names used by pymol are documented here:

[http://www.pymolwiki.org/index.php/Color\\_Values](http://www.pymolwiki.org/index.php/Color_Values)

You can list the colors used in a selection by this command:

```
iterate all, print color
```

You can define your own color names and associated colors by their RGB values

<b>set_color dbblue, [0.05 , 0.19 , 0.57]</b>	values between 0 and 1
<b>set_color dbblue, [13 ,48 , 146]</b>	values between 0 and 255

(Or: select menu settings/color, enter a new color name in the name field and adjust the colors with the sliders. This can also be used to adjust the colors used for different elements. However, if you write a script, you can reproduce your color scheme across different PyMol sessions)

**Read the PyMOL wiki on the "spectrum" command  
to see how you can generate and apply color gradients**

# Write the Command Sequence to a Text File

On the Mac, TextWrangler is a good free text editor

<http://www.barebones.com/products/textwrangler/>

```
load ~/pymol/pdb/3K8Y.pdb
hide all
show cartoon, 3K8Y
show spheres, resn GNP
show surface, polymer
color white, polymer
bg_color white
ray
png ~/pymol/figures/fig1c.png
hide surface, polymer
ray
png ~/pymol/figures/fig1d.png
save ~/pymol/fig1b.pse
```

Save as plain text file as

~/pymol/pml\_scripts/MyScript1.pml

## Running your script

Re-initialize or restart PyMOL and type:

**@~/pymol/pml\_scripts/MyScript1.pml**

Were the two figures and the PyMOL file saved to your pymol folder?

**Congratulations!**  
**You've successfully generated and executed**  
**a functioning PyMOL script.**

## Setting the File Path

**To keep PyMOL-related data together**, we have placed the “pymol” data folder in the home directory, and within this folder, subfolders called “pdb”, “figures”, “movies” and “pml\_scripts”.

**Find the file path to your home directory (~/)**

- this is where saved files are stored by default
- this is where PyMOL is looking for files to load

On my MacBook, this would be /Users/ahonegger  
Under Windows, this would be something like ...

To set the PyMOL default path to the pymol folder,  
type in the command line:

**cd ~/pymol**

## **.pymolrc**

If a PyMOL command file named “pymolrc” (visible) or “.pymolrc” (invisible) exists in your home directory, PyMOL will execute this file on start-up.

This is a convenient way to have Pymol always open with your preferred settings, e.g. default path, viewport size, background color, parameters modifying the representation, illumination etc.

Files ending on **.pml** or without suffix will be parsed as **PyMOL command files**.

Files ending on **.py** (or **.pym**) will be parsed as **python command files**.

If neither extension is used, PyMOL will judge based on the content of the file.

## Now we only need to specify the local path:

```
load pdb/3K8Y.pdb
hide all
show cartoon, 3K8Y
show spheres, resn GNP
show surface, polymer
color white, polymer
bg_color white
ray
png figures/fig1c.png
hide surface, polymer
ray
png figures/fig1d.png
save fig1b.pse
```

Delete all instances of  
~/pymol/ from the script file  
and run the script in PyMOL

**reinitialize**  
**@pml\_scripts/MyScript1.pml**

# Selections

Greatly expand on the selection capabilities of the GUI

**explicit selections:**

**select** (expression)

produces a **temporary selection object** named “sele”

**select sele,** (expression)

produces a **temporary selection object** named “sele”

**select name,** (expression)

produces a **named selection object** for further use

**implicit selections:**

**color red,** (expression)

colors the residues specified by the expression red without creating a selection object

**show cartoon,** (expression)

displays the specified residues as cartoon, no selection object



# Single word selectors

Single-Word Selector	Abbrev. Selector	Description
all	*	All atoms currently loaded into PyMOL
none		No atoms (empty selection)
hydro	h.	All hydrogen atoms currently loaded into PyMOL
hetatm	het	All atoms loaded from Protein Data Bank HETATM records
polymer	pol.	All atoms on the polymer (not het). Protein, DNA or RNA
visible	v.	All atoms in enabled objects with at least one visible representation
enabled		All atoms in enabled objects
backbone	bb.	Polymer backbone atoms
sidechain	sc.	Polymer non-backbone atoms
donors	don.	All potential hydrogen bond donors
acceptors	acc.	All potential hydrogen bond acceptors
solvent	sol.	All water molecules
organic	org.	All atoms in non-polymer organic compounds (e.g. ligands, buffers).
inorganic	ino.	All non-polymer inorganic atoms/ions.
bonded		All atoms making at least one bond
metals		All metal atoms/cations
guide		All protein CA and nucleic acid C4*/C4'
present	pr.	All atoms with defined coordinates in the current state (used e.g in creating movies)

## Properly Selectors: Selecting Atoms, Residues, Chains

One word Selector	Abbrev. Selector	Description
element	e.	<b>chemical element</b> , e.g. C, N, O, H, ..., Ca, Fe, Mg
name	n.	<b>atom name</b> , eg. <a href="#">select mainchain, n. N+CA+C+O</a>
resn	r.	<b>residue name</b> , e.g. <a href="#">select neg, r. Glu+Asp</a>
resi	i.	<b>residue number</b> e.g. <a href="#">select domain1, i. 33-126</a> if you have a negative residue number, a “\” is needed, e.g. <a href="#">select Ntag, i. \-5+\-4+\-3+\-2+\-1</a>
alt	alt	<b>alternative conformation</b> , e.g. <a href="#">""</a> , a, b, c
chain	c.	<b>chain identifier</b>
ss	ss	<b>secondary structure</b> , e.g. <a href="#">select allSTR, h+s+l+""</a>
id	id	<b>atom number</b>
b	b	<b>b-factor value</b> , e.g. <a href="#">select fuzzy, b &gt; 10</a>
q	q	<b>occupancy</b> , e.g. <a href="#">select lowOccupancy, q &lt; 0.5</a>

# Selection Macros

**Shorthand for selecting specific parts of a protein or nucleic acid chain**

Instead of

```
select name, pept1 and segi a1 and chain b and resi 142 and name ca
```

you can type

```
select name, /pept1/lig/b/142/ca
```

also for wildcards, ranges, multiple selections

```
select name, /pept1//b/142-163/n+ca+c+o
```

```
select name, /pept1//A/L* selects Leu and Lys
```

**select /entity/segment/chain/residue/atom**

Name des Objekts

leer lassen

Kette

Rest

Atom

If you click on an atom in PyMol, the feedback window shows the selection in this form:

You clicked /3K8Y//A/GLU`3/CA

Selector: selection "sele" defined with 9 atoms.

# Selection Macros

You need only the relevant part of the chain

e.g.

`show spheres, CYS/CA`

shows the Calpha atoms of all cysteins as spheres,

`show spheres, CYS/`

shows all atoms in Cystein residues as spheres

## **beginning with a slash:**

`/object-name/segi-identifier/chain-identifier/resi-identifier/name-identifier`

`/object-name/segi-identifier/chain-identifier/resi-identifier`

`/object-name/segi-identifier/chain-identifier`

`/object-name/segi-identifier`

`/object-name`

## **or not beginning with a slash:**

`resi-identifier/name-identifier`

`chain-identifier/resi-identifier/name-identifier`

`segi-identifier/chain-identifier/resi-identifier/name-identifier`

`object-name/segi-identifier/chain-identifier/resi-identifier/name-identifier`

# Selection-Algebra

## Selections can be combined by logical operators:

not <b>s1</b>	<b>!s1</b>	all atoms except those in selection s1
<b>s1</b> and <b>s2</b>	<b>s1 &amp; s2</b>	intersection, atoms that are both in s1 and in s2
<b>s1</b> or <b>s2</b>	<b>s1   s2</b>	union, atoms that are either part of s1 or s2

## Expansion of selections

byres <b>s1</b>	br. <b>s1</b>	expands sel. from atoms to residues
bymolecule <b>s1</b>	bm. <b>s1</b>	expands sel. to molecule
bychain <b>s1</b>	bc. <b>s1</b>	expands sel. to chain
byobject <b>s1</b>	bo. <b>s1</b>	expands sel. to object
bycell <b>s1</b>		expands sel. to unit cell
neighbor <b>s1</b>	nbr. <b>s1</b>	directly bonded to s1, excl. s1
bound_to <b>s1</b>	bto. <b>s1</b>	directly bonded to s1, incl. s1
first, last		first or last atom in selection

## Selection: Distance operators

**select s3, s1 within 5.0 of s2**

all atoms in s1 that are no farther than 5.0 Å from atoms in s2

**select s3, s1 near\_to 5.0 of s2** ((s1 within 5.0 of s2) and not s2))

all atoms in s1 that are no farther than 5.0 Å from a atoms in s2,  
excludes s2

**select s3, s1 beyond 5.0 of s2** ((s1 and not (s1 within 5.0 of s2))

all atoms in s1 that are farther than 5.0 Å away from atoms in s2

**select s3, s2 around 5.0** ((all within 5.0 of s2) and not s2))

all atoms within 5.0 Å of s2, excluding s2

**select s3, s2 expand 5.0** (all within 5.0 of s2)

all atoms in s2 or within 5.0 Å of s2

# Example: Determine Contact Residues

# load complex and separate its components

load pdb/S4K5B\_B-C.pdb, Cplx

extract darp, Cplx and chain B

extract lig, Cplx and chain C

delete Cplx

color green, darp

color cyan, lig

select C3, (darp within 3.6 of lig) or (lig within 3.6 of darp)

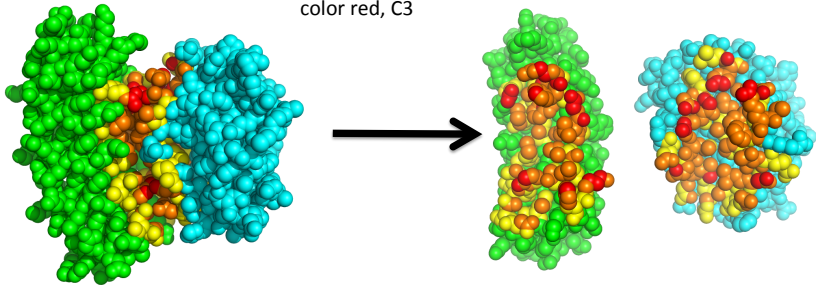
select C2, (darp within 5.0 of lig) or (lig within 5.0 of darp)

select C1, br. C2

color yellow, C1

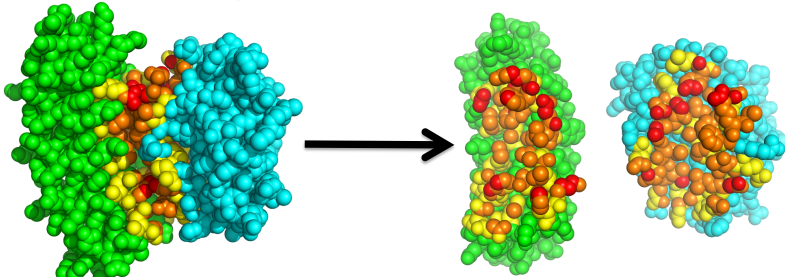
color orange, C2

color red, C3



# Listing Contact Residues

```
select C3, (darp within 3.6 of lig) or (lig within 3.6 of darp)
select C2, (darp within 5.0 of lig) or (lig within 5.0 of darp)
select C1, br. C2
```



**list contact atoms**

```
list=[]
iterate (C2),list.append((chain,resi,resn,name))
print list
```

**list contact residues**

```
list=[]
iterate (C1 and name CA),list.append((chain,resi,resn))
print list
```



## Setting the Orientation of a Molecule in a Script

```
get_view [ output [, quiet ] ]  
set_view view
```

Open fig1a.pse, rotate structure into a nice orientation and type:

**get\_view**

from the feedback window, copy-paste this to your script:

### cut below here and paste into script ###

set\_view (\

0.2857, -0.0944, 0.9536,\

0.0625, 0.9948, 0.0797,\

-0.9562, 0.0368, 0.2902,\

0.0000, 0.0000,-128.3235,\

-16.5101, 63.8420, -76.8616,\

103.0045, 153.6425, -20.0000 )

} Rotation Matrix

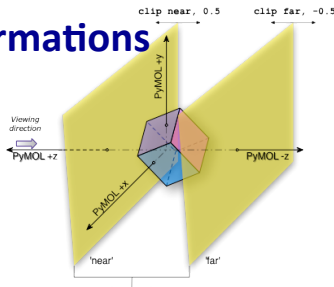
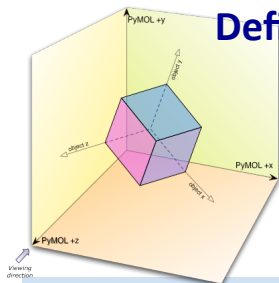
Origin in Camera Space

Origin in Coordinate Space

Back & Front Clipping Planes, Perspective

### cut above here and paste into script ###

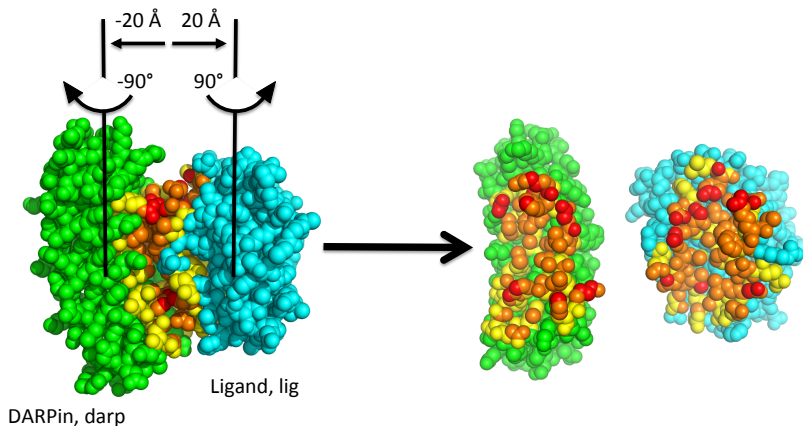
# Defined Transformations



```

reset [ object ]
turn axis, angle
move axis, distance
orient object-or-selection [, state]
center [ selection [, state [, origin [, animate ]]]]
zoom [ selection [,buffer [, state [, complete ]]]]
clip mode, distance [, selection [, state ]]
origin selection [, object [,position, [, state]]]
translate vector [,selection [,state [,camera [,object ]]]]
rotate axis, angle [,selection [,state [,camera [,object [,origin]]]]]
    
```

## Example: Looking at a Binding Interface



rotate y,  $-90$ , darp  
translate  $[-20,0,0]$ , darp  
rotate y,  $90$ , lig  
translate  $[20,0,0]$ , lig

# Measuring Distances

```
distance [ name [, selection1 [, selection2 [, cutoff [, mode ]]]]]
```

<b>name</b>	string: name of the distance object to create
<b>selection1</b>	string: first atom selection
<b>selection2</b>	string: second atom selection
<b>cutoff</b>	float: longest distance to show
<b>mode</b>	0: all interatomic distances 1: only bond distances 2: only show polar contact distances 3: like mode=0, but use <a href="#">distance exclusion setting</a> 4: distance between centroids ( <i>new in 1.8.2</i> )

## Simple H-bond detection:

dist name, sele1, sele2, mode=2

dependent on parameters:

set h\_bond\_cutoff\_center, 3.6

set h\_bond\_cutoff\_edge, 3.2

## More sophisticated H-Bond detection

```
load target.pdb,prot  
load docked_ligs.sdf,lig
```

```
# add hydrogens to protein
```

```
h_add prot
```

```
select don, (elem n,o and (neighbor hydro))  
select acc, (elem o or (elem n and not (neighbor hydro)))  
dist HBA, (lig and acc),(prot and don), 3.2  
dist HBD, (lig and don),(prot and acc), 3.2  
delete don  
delete acc  
hide (hydro)
```

```
hide labels,HBA  
hide labels,HBD
```

# Get information

get	get_extent	get_title
get_angle	get_position	get_type
get_area	get_property	get_version
get_bond	get_property_list	get_view
get_chains	get_renderer	get_viewport
get_dihedral	get_sasa_relative	
get_distance	get_symmetry	

**get\_angle [ atom1 [, atom2 [, atom3 [, state [, quiet ]]]]]**

**get\_dihedral [ atom1 [, atom2 [, atom3 [, atom4 [, state [, quiet ]]]]]]**

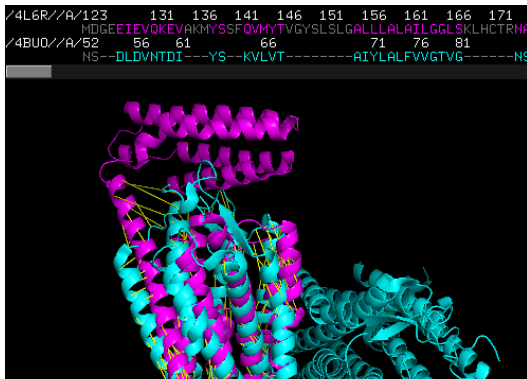
also: **phi\_psi [ selection [, quiet ]]** to get main chain torsion angles

**get\_area [ selection [, state [, load\_b [, quiet ]]]]** to get the surface area of an selection

**get\_sasa\_relative [ selection [, state [, vis [, var ]]]]** to get per-residue relative accessibility

## If align does not give reasonable results

In this alignment of the glucagon receptor (4L6R) to the rat neurotensin receptor 1 (4BUO) the sequence similarity was too low for a good sequence alignment, resulting in a bad residue pairing for the structural alignment



**Other alignment methods exist and can be used through the command line:**

**"cealign", "align", "super", "pair\_fit" or "fit"** , invoked with defined atom selections for better control over the alignment process

# Least Squares Superposition of two Structures

PyMOL offers several different commands for sequence-based and sequence independent structural alignments:

fit, intra\_fit, pair\_fit, super, align, cealign, rms, rms\_cur, intra\_rms, intra\_rms\_cur, extra\_fit.py, super\_all.py, align\_all.py, talign.py

They differ in how they determine the atom pairs included in the fit, and how they treat outliers (parts of the molecule that do not fit well).

Each method will return an rmsd value (root mean squares deviation)  
However, the values you get depend on the method used!

**rmsd values are meaningless  
if you do not indicate exactly what method and  
what parameters you used!!!**



```
fit mobile, target [, mobile_state [, target_state [, quiet [, matchmaker [, cutoff [, cycles  
[, object ]]]]]]]]
```

**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.

- mobile = string: atom selection
- target = string: atom selection
- mobile\_state = integer: object state {default=0, all states}
- target\_state = integer: object state {default=0, all states}
- matchmaker = integer: how to match atom pairs {default: 0}
- -1: assume that atoms are stored in the identical order
- 0/1: match based on all atom identifiers (segi,chain,resn,resi,name,alt)
- 2: match based on ID
- 3: match based on rank
- 4: match based on index (same as -1 ?)
- cutoff = float: outlier rejection cutoff (only if cycles>0) {default: 2.0}
- cycles = integer: number of cycles in outlier rejection refinement {default: 0}
- object = string: name of alignment object to create {default: None}

**Fit, Rms, Rms\_Cur** are finicky and **only work when all atom identifiers match**: segi, chain, resn, name, alt. If they don't, you'll need to use the alter command to change the identifiers to make them match -- typically that means clearing out the SEGI field, renaming chains, and sometimes renumbering.

```
intra_fit (selection),state
```

**intra\_fit** fits **all states of an object** (e.g. NMR) to an atom selection in the specified state. It returns the rms values to python as an array.

```
extra_fit [ selection [, reference [, method ]]]
```

**extra\_fit** aligns **multiple objects** to one reference object. It can use any of PyMOL's pairwise alignment methods (align, super, cealign, fit...). More precisely it can use any function which takes arguments mobile and target, so it will for example also work with talign.

**Additional keyword arguments are passed to the used method, so you can for example adjust outlier cutoff or create an alignment object.**

rms, rms\_cur, intra\_rms, Intra\_rms\_cur

compute a RMS fit between two atom selections, but **do not transform the models after performing the fit.**

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

**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. So long as the atoms are stored in PyMOL with the same order internally, you can provide just two selections. Otherwise, you may need to specify each pair of atoms separately, two by two, as additional arguments to pair\_fit.

Useful if you want to fit, e.g. the ring systems of ligands.

**Examples:**

*# superimpose protA residues 10-25 and 33-46 to protB residues 22-37 and 41-54:*

```
pair_fit protA///10-25+33-46/CA, protB///22-37+41-54/CA
```

*# superimpose ligA atoms C1, C2, and C4 to ligB atoms C8, C4, and C10, respectively:*

```
pair_fit ligA///C1, ligB///C8, ligA///C2, ligB///C4, ligA///C3, ligB///C10
```



## Python Scripts offer additional Functionalities:

```
run py_scripts/colorbyrmsd.py
```

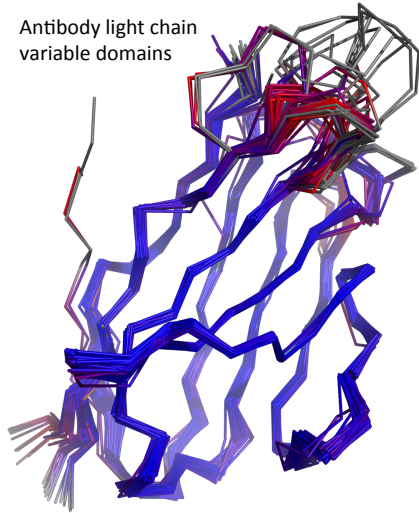
```
colorbyrmsd 4d3c, 2x7l  
colorbyrmsd 4ht1, 2x7l  
colorbyrmsd 5ds8, 2x7l  
colorbyrmsd 5dtf, 2x7l  
colorbyrmsd 5dub, 2x7l  
colorbyrmsd 4ma3, 2x7l  
colorbyrmsd 4o4y, 2x7l  
colorbyrmsd 4jo3, 2x7l  
colorbyrmsd 4jo4, 2x7l  
colorbyrmsd 4o51, 2x7l  
colorbyrmsd 4hbc, 2x7l  
colorbyrmsd 5i8o, 2x7l  
colorbyrmsd 4jo1, 2x7l
```

```
...
```

```
hide all
```

```
show ribbon
```

Antibody light chain  
variable domains



# PyMOL Settings

Style and quality of PyMOL representations are controlled by more than ~~600~~ **1400** different settings ...

# General Syntax for Settings

```
set name [, value [, selection [, state [, updates [, log [, quiet ]]]]]]
```



"set" is a command



the command "set" assigns a value to named variable



dependent on the setting, one or more additional parameters are required for boolean variables (on/off or 0/1), no parameter means "on"  
some settings are global (default), others can be applied to a selection.

## set<TAB>

parser: matching commands:

set	set_dihedral	set_property
set_atom_property	set_geometry	set_symmetry
set_bond	set_key	set_title
set_color	set_name	set_view

## set <TAB>

list of all settings set by "set" recognized by the current version of PyMOL

## set ?

Usage: set name [, value [, selection [, state [, updates [, log [, quiet ]]]]]]

# Settings define the Style of the Figure

set antialias = 1

set ambient=0.3

set direct=1.0

set ray\_trace\_mode=1

set stick\_radius = 0.2

set cartoon\_tube\_radius, 0.2

set cartoon\_fancy\_helices=1

set cartoon\_cylindrical\_helices=0

set cartoon\_flat\_sheets = 1.0

set cartoon\_smooth\_loops = 0

set cartoon\_highlight\_color =grey50

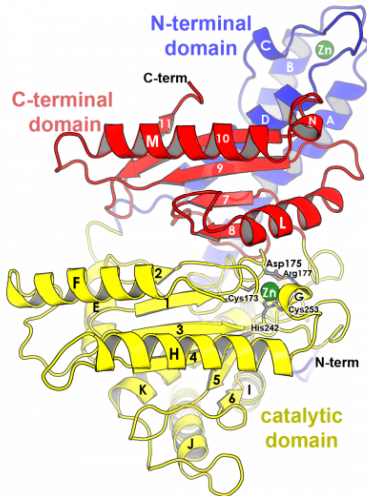
bg\_color white

set\_color maarine= [0.3, 0.8, 1.0]

set\_color gray=[0.8,0.8,0.8]

set\_color green=[0.0,0.5,0.0]

copy



PLoS\_script1.pml

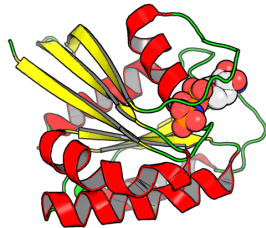
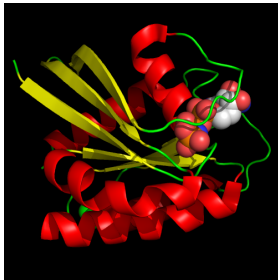


```
set antialias = 1
set ambient=0.3
set direct=1.0

set ray_trace_mode=1

paste set stick_radius = 0.2
      set cartoon_tube_radius, 0.2
      set cartoon_fancy_helices=1
      set cartoon_cylindrical_helices=0
      set cartoon_flat_sheets = 1.0
      set cartoon_smooth_loops = 0
      set cartoon_highlight_color =grey50

bg_color white
```

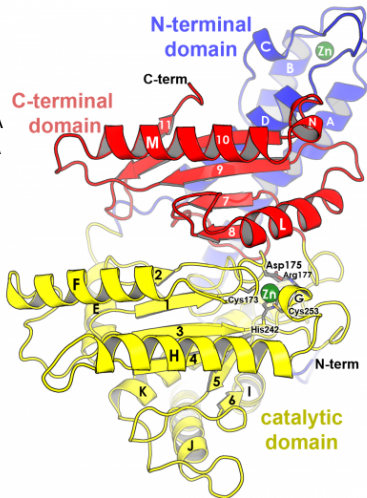


## These commands define what's shown

```
load csos3_18o_nobreak.pdb, csos3
hide all
show cartoon
show sticks, (resid 173 or resid 175 or resid 177 \
    or resid 242 or resid 253) and not (name n \
    or name c or name o)
show spheres, elem ZN
```

## and how it's colored

color gray50, elem C  
color green, elem ZN  
color blue, resid 38:147 and name ca  
color yellow, resid 148:397 and name ca  
color red, resid 398:514 and name ca



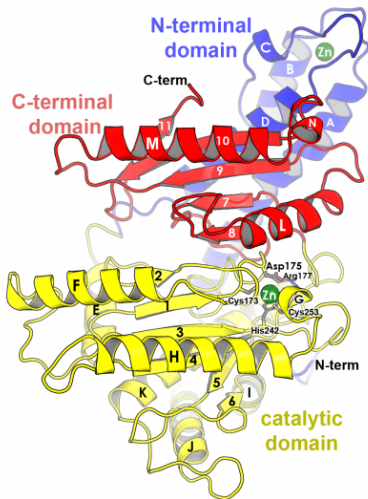
Carbonic anhydrase, CsoS3,  
from *Halothiobacillus neapolitanus*.  
(PDB ID 2G13)

## These commands define the orientation

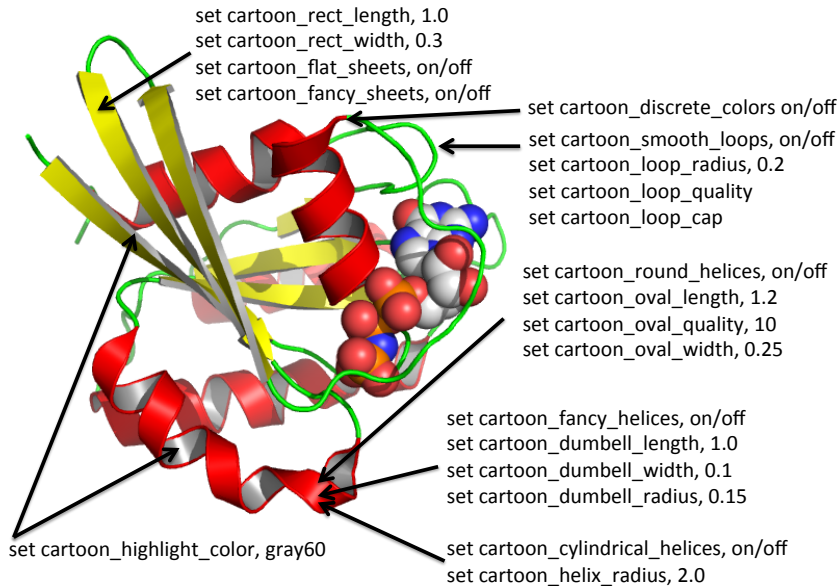
```
set_view (\
    0.091340274, -0.606698275, 0.789650559,\
    -0.991202235, -0.131515890, 0.013612081,\
    0.095602803, -0.783963323, -0.613382638,\
    0.001799395, 0.001679182, -246.492980957,\
    12.976243019, 41.245639801, 62.928291321,\
    187.538497925, 249.492980957, 0.000000000 )
turn y, 3.5
```

```
# and generate the figure
viewport 1200,1500
ray
png csos3-left.png
```

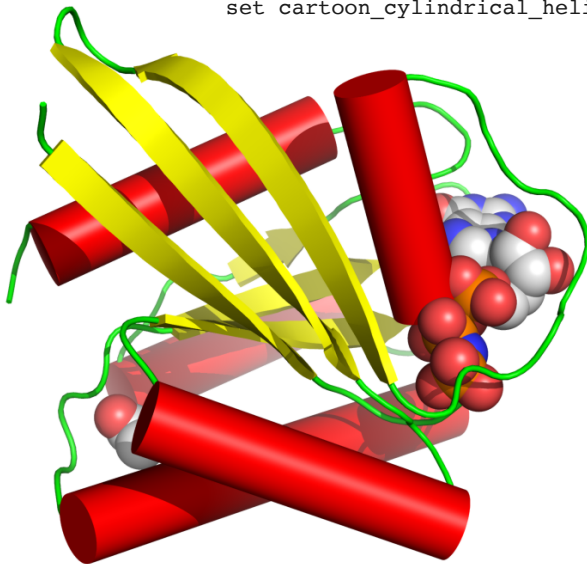
# labels were added in a generic graphics program  
# e.g. Photoshop, Illustrator ...



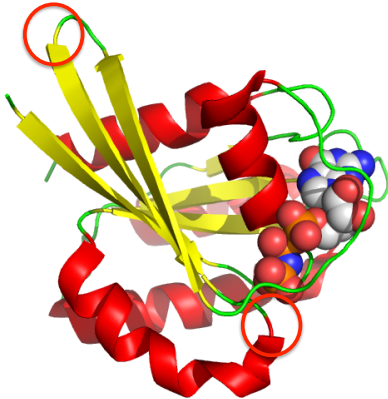
## Settings: Cartoon representation



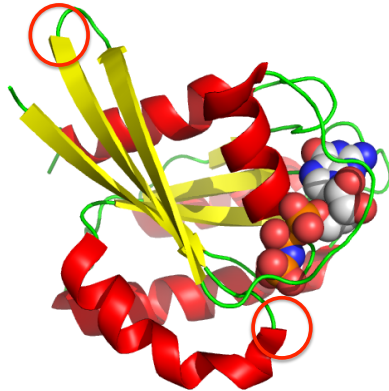
```
set cartoon_cylindrical_helices, on
```



## Settings: cartoon\_discrete\_colors



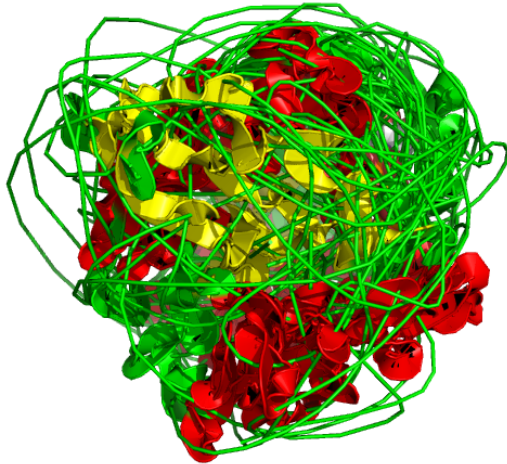
set cartoon\_discrete\_colors , off



set cartoon\_discrete\_colors , on

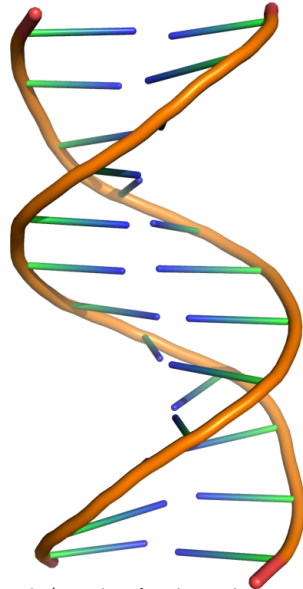
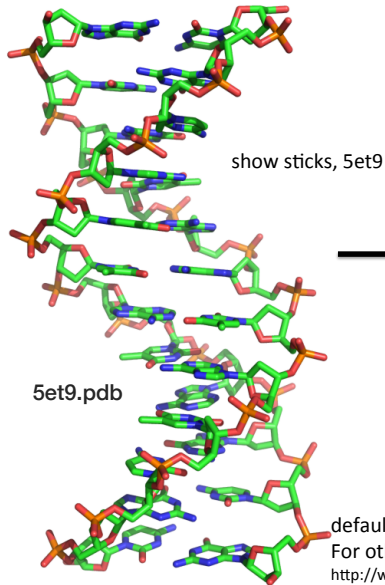
Stops secondary structure colors from bleeding into the coil areas

## When things going haywire



“set cartoon\_trace, 1” seems to confuse PyMOL if the structure contains more than just Calpha atoms

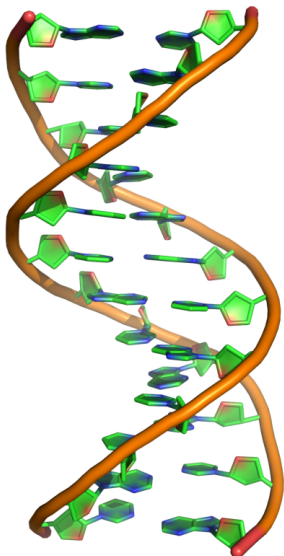
# Nucleic Acid Cartoons



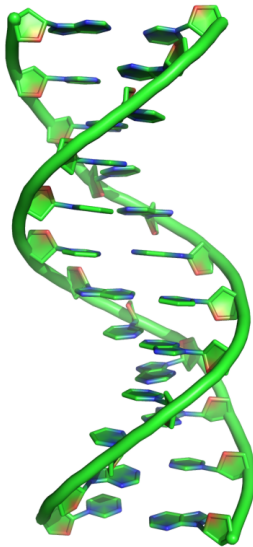
default cartoon view  
For other options, see:  
[http://www.pymolwiki.org/index.php/Examples\\_of\\_nucleic\\_acid\\_cartoons](http://www.pymolwiki.org/index.php/Examples_of_nucleic_acid_cartoons)



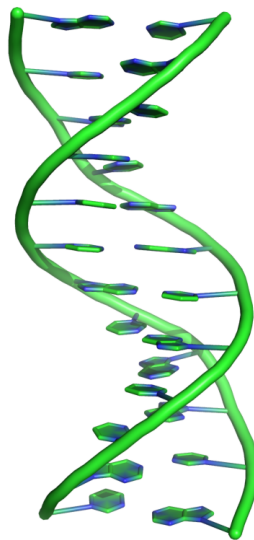
## Nucleic Acid Cartoons



set cartoon\_ring\_mode, 3



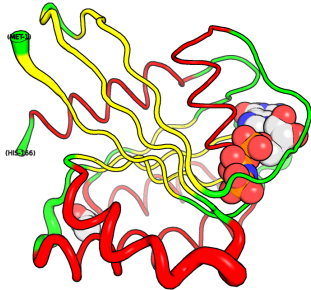
set cartoon\_nucleic\_acid\_mode, 3



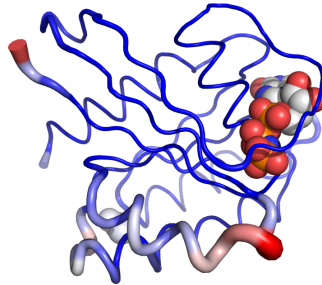
set cartoon\_ring\_finder, 2

## cartoon putty

Show which parts of the structure are more flexible in the crystal (b-factor)

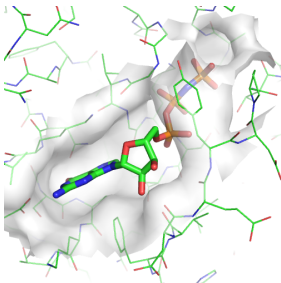


show cartoon  
cartoon putty  
unset cartoon\_smooth\_loops  
unset cartoon\_flat\_sheets



spectrum b, blue\_white\_red, minimum=20, maximum=40  
as cartoon  
cartoon putty

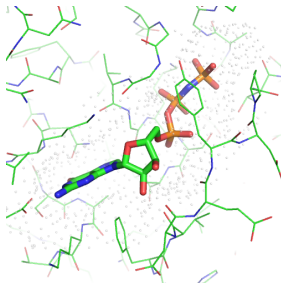
# Surface Settings



set surface\_type, 0

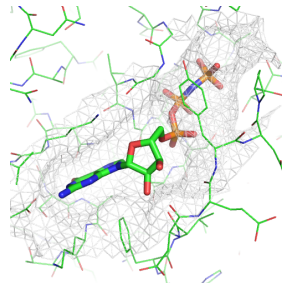
# smooth surface, made  
transparent by

set transparency, 0.5



set surface\_type, 1

# dot surface



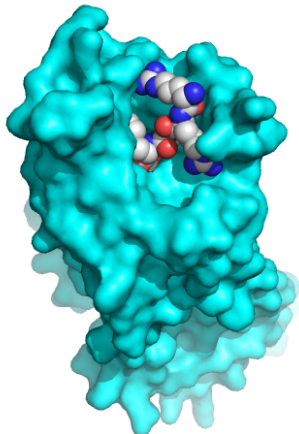
set surface\_type, 2

# mesh surface

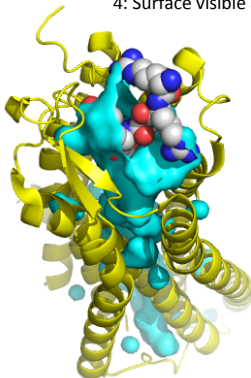
# Surface settings

## set surface\_mode, int

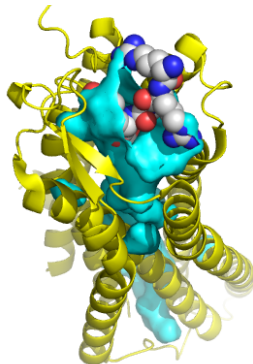
- 0: Default mode, surfacing with respect to flags.
- 1: Surface everything, including HET and hydrogens
- 2: Surface only heavy atoms
- 3: Surface only visible
- 4: Surface visible and heavy



set surface\_cavity\_mode, 0

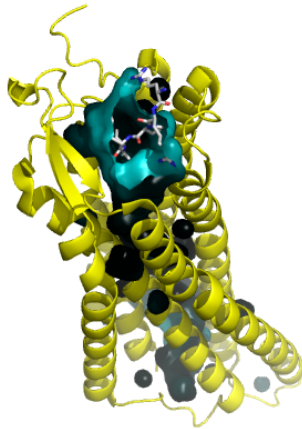
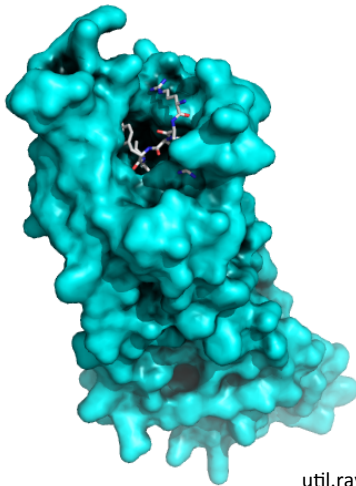


set surface\_cavity\_mode, 1



set surface\_cavity\_mode, 2  
set surface\_cavity\_radius, -3  
set surface\_cavity\_cutoff, 7

## Shading the Surface



`util.ray_shadows('occlusion2')`

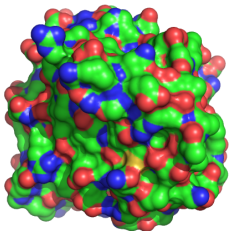
highlights pockets and cavities by depth-dependent shadowing

## Showing a solid clipping plane

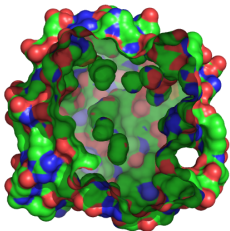
### Pymol Tricks

Normally, if the near clipping plane cuts a surface, the surface is shown as an open shell.

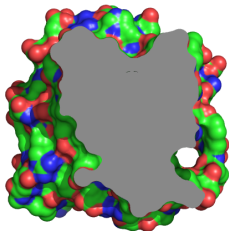
By turning interior lighting off and assigning a fixed color to the interior, in the ray-traced image, the cut appears closed by the clipping plane.



hide all  
show surface

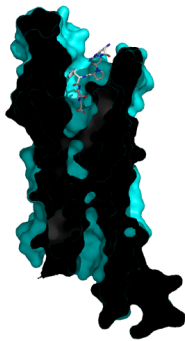
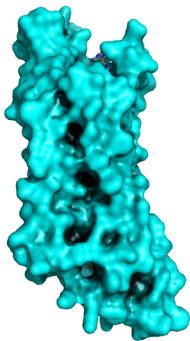


clip near, -20

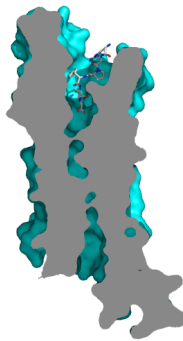


set two\_sided\_lighting, off  
set ray\_interior\_color, grey70

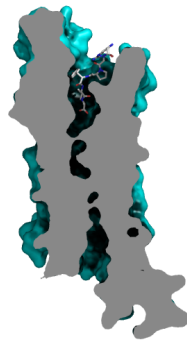
# Solid Clipping Plane



clip near, -30



set two\_sided\_lighting, off  
set ray\_interior\_color, grey70

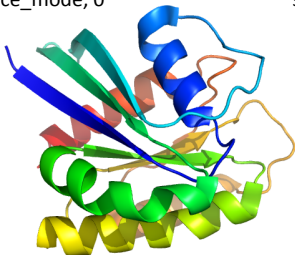


util.ray\_shadows('occlusion2')

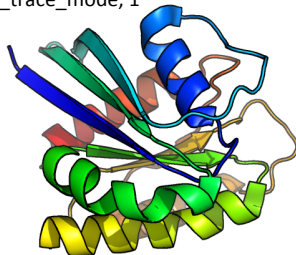
util.ray\_shadows('occlusion2')

# Ray\_trace\_mode

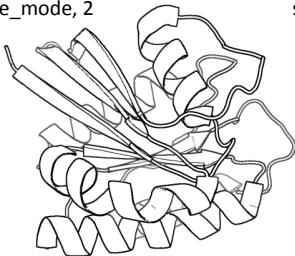
set ray\_trace\_mode, 0



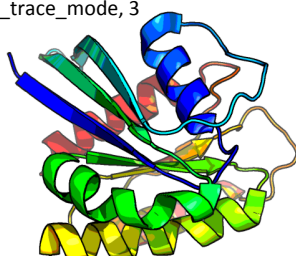
set ray\_trace\_mode, 1



set ray\_trace\_mode, 2



set ray\_trace\_mode, 3



set ray\_trace\_gain, 0.0 - sets thickness of outline, set ray\_trace\_disco\_factor, 1 to clean up

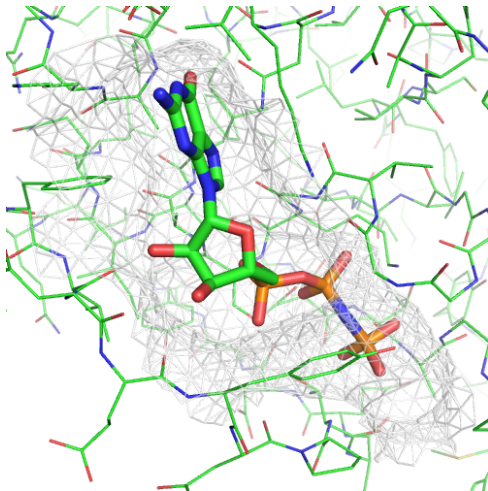


# Examples from the PyMOL Gallery

adapted to 3K8Y

*<http://www.pymolwiki.org/index.php/Gallery>*

# Binding Pocket

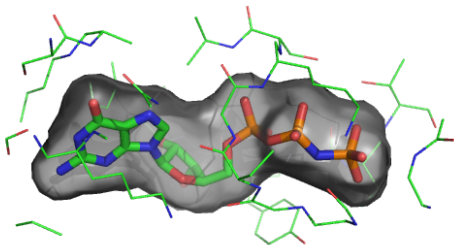


@pml\_scripts/AHo\_BindingPocket.pml

```
#load your molecule, extract prot, lig
load pdb/3K8Y.pdb, tmp
extract lig, resn GNP
extract prot, polymer
delete tmp

#representation
hide all
show lines, prot
show surface, prot within 8 of lig
show sticks, lig
#coloring, prot & lig in default color
bg_color white
set surface_color, white
#orientation (correct as needed)
orient lig
# special settings for this representation
set surface_carve_cutoff, 4.5
set surface_carve_selection, lig
set surface_carve_normal_cutoff, -0.1
set two_sided_lighting
set transparency, 0.5
set surface_type, 2
unset ray_shadows
#render image and save
ray
png figures/BindingPocket.png
save examples/AHo_BindingPocket.pse
```

# Ray-Normal-Based Transparency



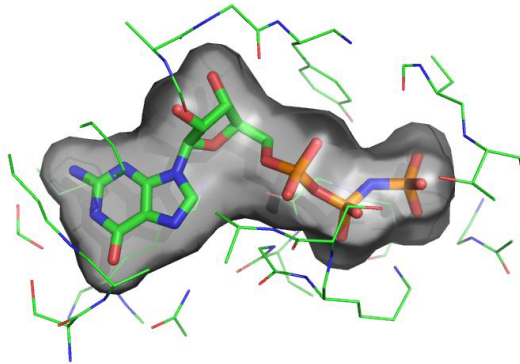
```
#load your molecule, extract prot, lig
load pdb/3K8Y.pdb, tmp
extract lig, resn GNP
extract prot, polymer
delete tmp

#representation
hide all
show surface, lig
show sticks, lig
show lines, prot within 5 of lig

#coloring
bg_color white
set surface_color, grey
# set the view (correct as needed)
orient lig
# special settings for this representation
set surface_mode, 3
set transparency_mode, 1
set transparency, 0.5
set ray_transparency_oblique
set ray_transparency_oblique_power, 8
set ray_transparency_contrast, 7
#render image and save
ray
png figures/RayNormal.png
save examples/AHo_RayNormal.pse
```

@pml\_scripts/AHo\_RayNormal.pml

# Make a Movie



#continued from last slide

# animate

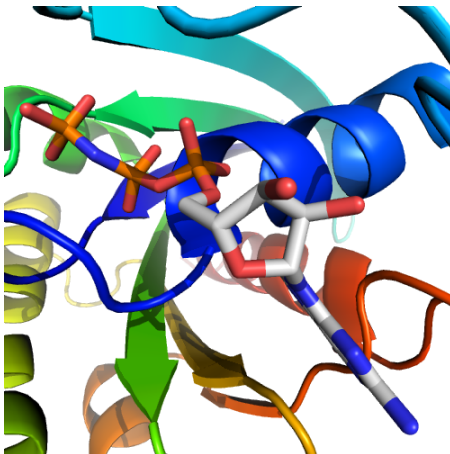
```
set cache_frames, 1  
set ray_trace_frames, 1  
mset 1x120  
movie.roll 1, 120, 1, x  
mplay
```

save movie as:  
as image sequence  
or Quicktime movie

Get Quicktime Pro 7 from:

<http://www.id.uzh.ch/dl/sw/angebote/grafik/QuickTimePro.html>

# Cool Perspective

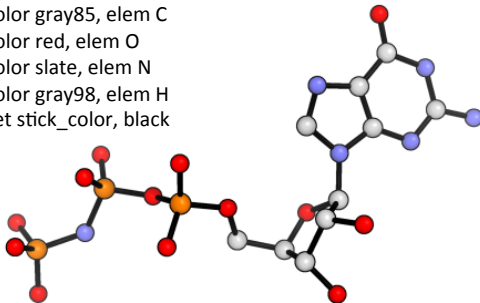


```
#load your molecule, extract protein and ligand
load pdb/3K8Y.pdb, tmp
extract lig, resn GNP
extract prot, polymer
delete tmp
#representation
hide all
show cartoon, prot
show sticks, lig
#coloring
bg_color white
spectrum count, rainbow, prot, byres=1
util.cbaw lig
#correct orientation and zoom factor as needed
zoom lig
# special settings for this representation
set field_of_view, 60
#render image and save
ray
png figures/CoolPerspective.png
save examples/AHo_CoolPerspective.pse
```

@pml\_scripts/AHo\_CoolPerspective.pml

# Stylized Ball-and-Stick

```
load pdb/3K8Y.pdb, tmp # special settings
extract lig, resn GNP    set stick_radius, .07
extract prot, polymer    set sphere_scale, .18
delete tmp               set sphere_scale, .13, elem H
#representation         set bg_rgb=[1, 1, 1]
hide everything           set stick_quality, 50
show sticks, Lig         set sphere_quality, 4
show spheres, Lig        set ray_trace_mode, 1
#coloring                set ray_texture, 2
color gray85, elem C      set antialias, 3
color red, elem O         set ambient, 0.5
color slate, elem N       set spec_count, 5
color gray98, elem H      set shininess, 50
set stick_color, black    set specular, 1
                          set reflect, .1
                          set dash_gap, 0
                          set dash_color, black
                          set dash_gap, .15
                          set dash_length, .05
                          set dash_round_ends, 0
                          set dash_radius, .05
#orientation
zoom lig
orient lig
#render image and save
ray
png figures/Ball-and-Sticks.png
save examples/AHo_Ball-and-Sticks.pse
```

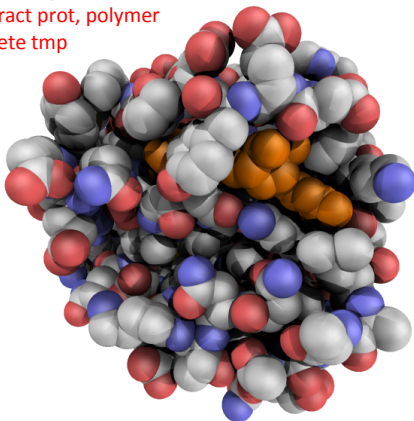


@pml\_scripts/AHo\_Ball-and-Sticks.pml

# QuteMol Style

<http://qutemol.sourceforge.net>

```
#load your molecule, extract prot, lig  
load 3K8Y.pdb, tmp  
extract lig, resn GNP  
extract prot, polymer  
delete tmp
```

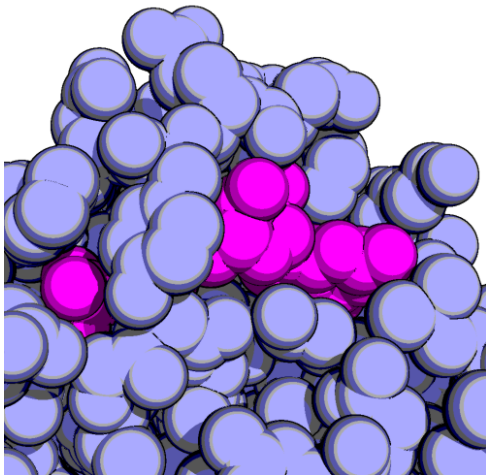


@pml\_scripts/AHo\_QuteMol.pml

```
# representation  
hide all  
as spheres  
#coloring  
bg_color white  
set_color oxygen, [1.0,0.4,0.4]  
set_color nitrogen, [0.5,0.5,1.0]  
util.cbaw  
color orange, resn GNP  
# special settings for this representation  
set light_count, 8  
set spec_count, 1  
set shininess, 10  
set specular, 0.25  
set ambient, 0  
set direct, 0  
set reflect, 1.5  
set ray_shadow_decay_factor, 0.1  
set ray_shadow_decay_range, 2  
unset depth_cue  
set field_of_view, 60  
#render image and save  
ray  
png figures/QuteMol.png  
save examples/AHo_QuteMol.pse
```

# Goodsell-like Representation

<http://www.rcsb.org/pdb/101/motm.do?momID=184>



```
#load your molecule, extract prot, lig
load pdb/3K8Y.pdb, tmp
extract lig, resn GNP
extract prot, polymer
delete tmp

#representation
as spheres

#coloring
bg_color white
color lightblue, prot
color magenta, lig

# set the view (correct as needed)
orient all within 8 of lig

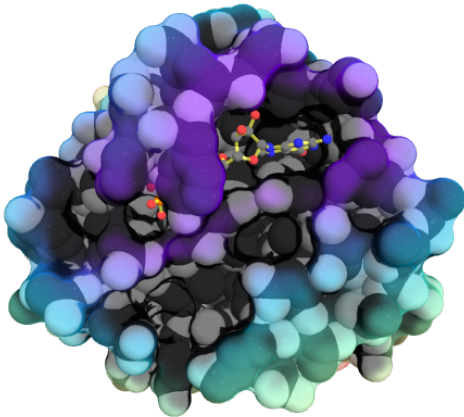
# special settings for this representation
unset specular
set ray_trace_gain, 0
set ray_trace_mode, 3
set ray_trace_color, black
unset depth_cue

# render image and save
ray
png figures/GoodsellLike.png
save examples/AHo_GoodsellLike.pse
```

@pml\_scripts/AHo\_GoodsellLike.pml



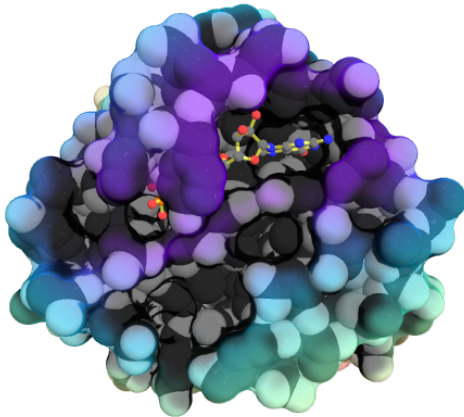
# Complex Stylized Protein



```
...  
# representation  
hide all  
preset.ball_and_stick("lig")  
show spheres, prot  
set sphere_scale, 0.99, prot  
show surface, prot  
  
# coloring  
bg_color white  
set_bond stick_color, 0xffff44, lig  
set_bond stick_transparency, 0.35, lig  
color grey, lig and e. C  
ramp_new pRamp, lig, selection=prot, \  
    range=[5,30], color=rainbow  
set surface_color, pRamp, prot  
color white, prot  
color grey30, prot and e. C  
disable pRamp
```

# continued on next slide

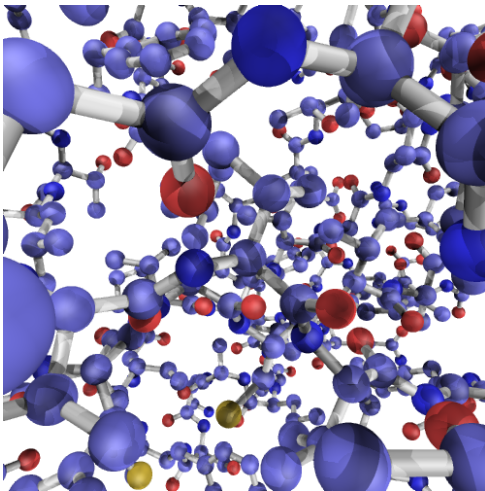
# Complex Stylized Protein



@pml\_scripts/AHo\_StylizedProtein.pml

```
# special settings for this representation
set ray_transparency_contrast, 0.20
set ray_transparency_oblique, 1.0
set ray_transparency_oblique_power, 20
set surface_quality, 2
set light_count, 5
set ambient_occlusion_mode, 1
set ambient_occlusion_scale, 50
set ambient, 0.40
set transparency, 0.50
set spec_power, 1200
set spec_reflect, 0.20
set ray_opaque_background, 0
set ray_shadow, 0
set field_of_view, 60
# orientation
zoom complete=1
# render image and save
ray
png figures/StylizedProtein.png
save examples/AHo_StylizedProtein.pse
```

# Ball-and-Stick



@pml\_scripts/AHo\_Ball\_and\_Stick2.pml

```
load pdb/3K8Y.pdb, tmp  
extract lig, resn GNP  
extract prot, polymer  
delete tmp
```

```
#representation
```

```
hide all  
show spheres  
show sticks
```

```
#coloring
```

```
bg_color white  
util.cbab  
set stick_ball_color, atomic  
set_bond stick_color, white, all, all
```

```
# special settings for this representation
```

```
set stick_radius, 0.4, (all)  
set sphere_scale, 0.3, (all)  
set_bond stick_radius, -0.14, all, all  
set light_count, 8  
set spec_count, 1  
set shininess, 10  
set specular, 0.25  
set ambient, 0  
set direct, 0  
set reflect, 1.5  
set ray_shadow_decay_factor, 0.1  
set ray_shadow_decay_range, 2  
unset depth_cue  
set field_of_view, 60
```

```
# render and save
```

```
ray
```

```
png figures/Ball_and_Stick2.png  
save examples/AHo_Ball_and_Stick2.pse
```

# Color by Distance from Origin

```
diff_len = lambda x,y : math.sqrt((x[0]-y[0])*(x[0]-y[0]) + (x[1]-y[1])*(x[1]-y[1]) + (x[2]-y[2])*(x[2]-y[2]))
```

```
#load your molecule, extract prot, lig
```

```
load pdb/3K8Y.pdb, tmp
```

```
extract lig, resn GNP
```

```
extract prot, polymer
```

```
delete tmp
```

```
#representation
```

```
as surface, prot
```

```
as stick, lig
```

```
# create the pseudoatom at the origin
```

```
pseudoatom pOrig, pos=(0,0,0), label=origin
```

```
# these are special PyMOL variables that will hold the
```

```
# coordinates of the atoms and the pseudoatom
```

```
stored.origCoord = []
```

```
stored.distCoord = []
```

```
# copy the coordinates into those special variables
```

```
iterate_state 1, pOrig, stored.origCoord.append((x,y,z))
```

```
iterate_state 1, prot, stored.distCoord.append((x,y,z))
```

```
# extend origCoord to be the same length as the other
```

```
stored.origCoord *= len(stored.distCoord)
```

```
# calculate the distances
```

```
newB = map(lambda x,y: diff_len(x,y), stored.distCoord, stored.origCoord)
```

```
# put them into the b-factor of the protein
```

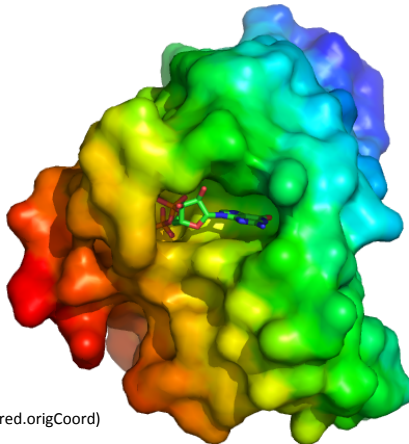
```
alter prot, b=newB.pop(0)
```

```
# color by rainbow_rev or any other palette listed in "help spectrum"
```

```
spectrum b, rainbow_rev, prot
```

```
bg_color white
```

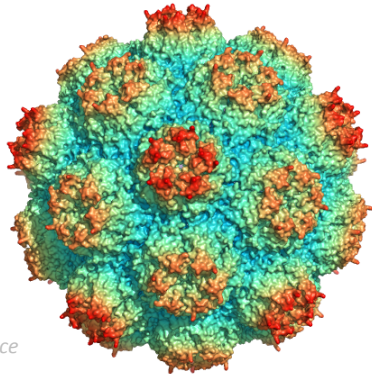
```
...
```



@pml\_scripts/AHo\_ColorDist.pml

# Virus Capsid

```
...  
# create a pseudoatom at the origin-- we will  
# measure the distance from this point  
pseudoatom pOrig, pos=(0,0,0), label=origin  
# load and build the capsid  
load pdb/2xpj.pdb1.gz  
split_states 2xpj  
delete 2xpj  
# show all 60 subunits it as a surface  
# this will take a few minutes to calculate  
as surface  
# create a new color ramp, measuring the distance  
# from pOrig to 1hug, colored as rainbow  
ramp_new proximityRamp, pOrig, selection=(2xpj*), range=[110,160], color=rainbow  
# set the surface color to the ramp coloring  
set surface_color, proximityRamp, (2xpj*)  
# some older PyMOLs need this recoloring/rebuilding  
recolor  
bg_color white  
disable proximityRamp  
# render image and save  
...
```



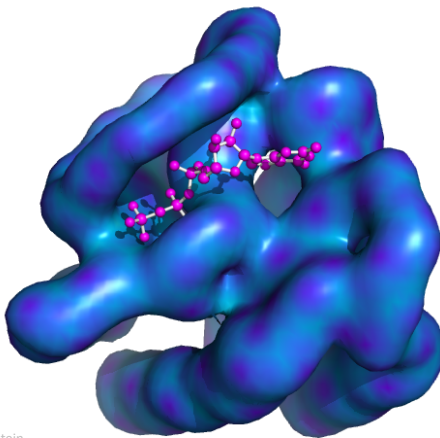
@pml\_scripts/AHo\_VirusCapsid.pml

# Smooth Pseudo-Surface with Ligand

...

```
# Ligand as ball and stick
bg_color white
hide lines
show sticks, lig
show spheres, lig
color magenta, lig
set_bond stick_radius, 0.13, lig
set_sphere_scale, 0.26, lig
set_bond stick_radius, 0.13, lig
set_bond stick_color, white, lig
set_sphere_scale, 0.26, lig

#protein pseudo-surface
# set the B-factors nice and high for smoothness
alter all, b=10
alter all, q=1
# 3.5 A map resolution
set gaussian_resolution, 8
# new gaussian map w/resolution=0.5 Ang on just the main chain
map_new map, gaussian, 1, n. C+O+N+CA, 5
# create a surface from the map
isosurface surf, map, 1.5
# color the protein by number
spectrum count, rainbow, prot
# now color the map based on the b-factors of the underlying protein
cmd.ramp_new("ramp", "prot", [0,10,10], "rainbow")
# set the surface color
cmd.set("surface_color", "ramp", "surf")
# hide the ramp and lines
disable ramp
...
```



#pml\_scripts//AHo\_SmoothSurfwLig.pml