READ

7.1 Setting up for scripts: review of files, directories and path.

On the hard drive, information is arranged in files and folders, and each operating system has its own way of organizing and calling on files. As a reminder, files contain information such as lines of text or binary numbers making up an image or a software algorithm, and are organized throughout the hard drive within directories, also called folders. When you want to invoke a script, PyMol needs to know where it is! There are two fundamental ways to make sure PyMol knows where the script you want to use is located:

7.2 Give the full path

ForMac OSX or Unix/Linux system, full path to the script may look like:
/Users/DMC/Desktop/PyMol/myfirstscript.pml

On a Windows system it may be something like
D:\Data\My Datafiles\PMscripts\myfirstscript.pml

These long lines give the absolute path to the location of the script, and must be present at each invocation.

7.3 Set the default working directory to where scripts (and PDB files) are located.

We did something very similar when we gave the command **cd desktop** in a previous exercise, so that PyMol would automatically look onto the desktop for the necessary file(s). In some respect this is easier because once the path has been set with the **cd** command, the path does not have to be repeated again each time we want to run this, or another script at the same location.

READ

You should be (or become) familiar to the ways directories and paths are dealt with on your computer, which are essentially the Unix/Linux/OS X family or the Windows operating systems.

7.4 Text-only files: a review

INFO

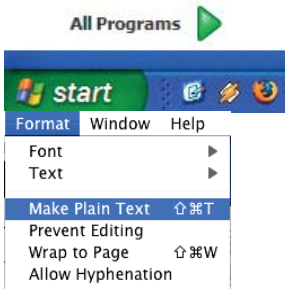
Although the following might appear trivial, it may be of vital importance!
A file containing text that is **bold**, underlined, or *italicized* it NOT a plain text file.

A plain text file:

- contains ONLY printable characters that can be typed from the simplest of keyboards: letters, numbers and symbols such as !, @, and $;
- is not specific to any particular word processor format and is displayed with the default font within any word processor;
- contains a “line break” and/or “end-of-line” code (specific to the operating system) that signifies that the line has terminated “here” and that a new line will begin “next” also known as carriage return (CR) and line feed (LF) respectively

7.5 Options to create text-only files:

- Windows: WordPad or NotePad should be free, standard text editors within the **Start > All Programs > Accessories** folder.



- Mac OS X:
  - The included **Applications > TextEdit** can be used only after the proper menu has been called: **Format > Make Plain Text**
  - **Applications > Classes > TextWrangler** is a freeware program that offers many options and will be used in this class.

`http://www.barebones.com/products/textwrangler/`

- General Unix/Linux/MacOSX

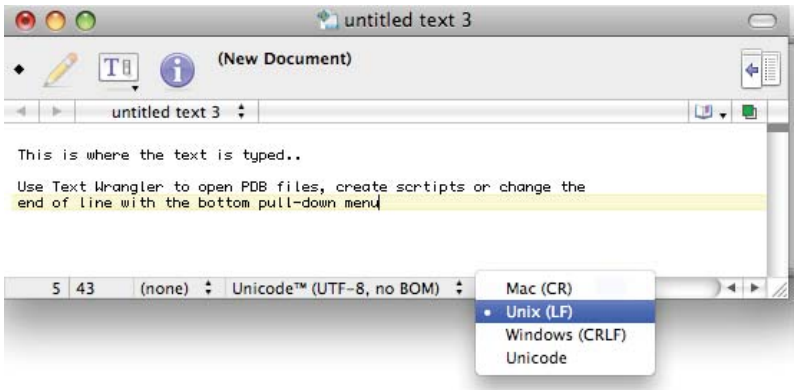
Any file created with vi, pico, nano, xedit, or a redirect command such as cat > myfile.txt is created as a plain text files.

7.6 Dealing with line breaks or end-of-lines problems:

A “hard-return” is coded within the text file. A “soft-return” is the apparent wrapping of a long text file to the next line, but in fact is a very long, single line.

Hard-returns are coded with an operating system-specific code. There are three options: Macintosh, Unix and DOS. Macintosh refers to Macintosh OS Classic (through Mac OS 9.x), Unix is for Unix, Linux, *and Mac OS X*. DOS/Windows is for the old Disk Operating System that has evolved into Windows.

With TextWrangler, one can switch between the 3 hard-returns with a single mouse option as seen on this image.



<CR> means carriage return, <LF> means line feed and (CRLF) is both of them.

✔ **READ -** If you import a file from another system or the web, it may be a good idea to verify it's end-of-line / hard-return status if it does not work as expected or not at all within PyMol. This even includes the simple action of *copy/paste* from a PDF document!

Note: See also: `http://en.wikipedia.org/wiki/Newline`

7.7 Creating a small script

✔ **TASK**

Preliminary:

√ - For this exercise the script will be saved on the **Desktop**.
√ - If PyMol is running, **Quit PyMol** and **restart** it again, to have a fresh start.
√ - Open either **TextEdit** or **TextWrangler** (both located under /Applications)
With TextEdit remember to use menu: **Format > Make Plain Text**
√ - With the text editor **create** and **save** a **new script** that we shall call **abc.pml**.
√ - **Save abc.pml** on the **desktop**.
√ - **Place a copy of 2BIW.pdb2 on the desktop.**

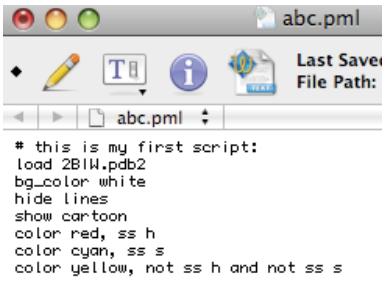
In a PyMol script comment lines which start with #.

You can omit typing anything after the comment sign # for these exercises. However, placing comments within your scripts is a very good practice for your future reference or for sharing your scripts. This can literally save hours of frustrations!

✔ **TASK**

Within abc.pml file type the following lines

 # this is my first script:
load 2BIW.pdb2
bg_color white
hide lines
show cartoon
color red, ss h
color cyan, ss s
color yellow, not ss h and not ss s



Note the use of the comma (,) after the name of the color.

`ss` signifies secondary structure
`h` signifies helix. (Note that the word "helix" would not be recognized.)
`s` signifies sheet. (Note that the word "sheet" would not be recognized.)
`not h and not s` are be the loops or turns.

7.8 Running a script

7.8.1 set the default directory

✔ **TASK**

In your new PyMol session, set the default directory by typing the `cd Desktop` command. Then verify that the path is correct with the command `pwd` (present working directory):

cd Desktop
pwd

Note: a copy of 2BIW.pdb2 should be present on the desktop for the script to function properly.

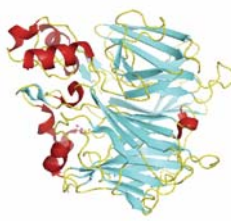
7.8.2 Run the script: @ command

Rule: a script is invoked with the @ command after its name within the line command.
example: **@abc.pml**

Note: either the top (External GUI) or bottom (Internal GUI) line command can be used

✔ **TASK**

Type **@abc.pml** within the line command.

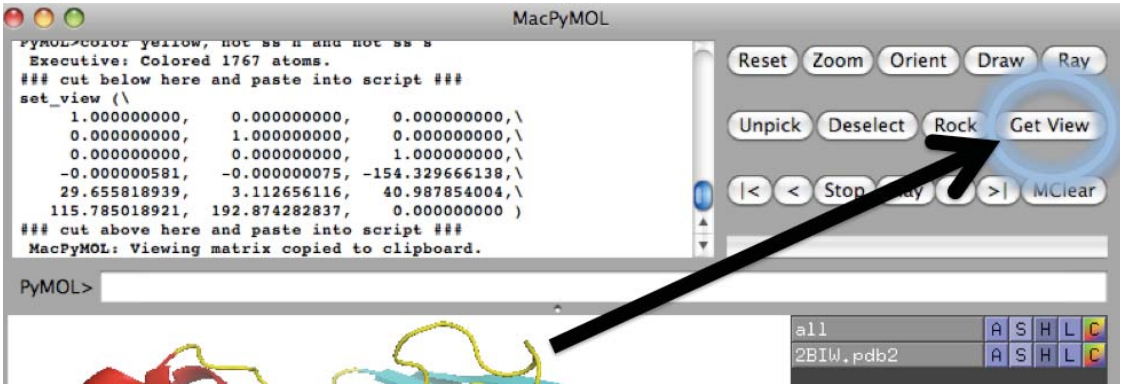


Activating the script will load the structure and create and modify the visual representation of the protein. The same image could have been obtained by either typing one by one each of the script commands within the line command, or by using some of the mouse options as we have done before. The advantage of scripts is that they are an easy way to create images in a reproducible fashion, but most importantly, it can be an easy way to remember how the images were created.

7.8.3 Orienting a molecule without the mouse

The image issued from the script was created with the default orientation that the molecule adopts when it is first opened. However, one important aspect of a structure illustration is the orientation, or even close up of the molecule(s).

Fortunately, PyMol offers a very nice way to return to a previously determined orientation and inscribe it within a script: the “Get View” button, located at the top right of the “External GUI.”



✔ **TASK**

√ - **Rotate** the **molecule** in an orientation you like
√ - **Click** on **Get View** button
The rotation matrix is shown on the PyMol text window, but is <u>also</u> placed within the clipboard.
√ - **Paste** the **matrix** at the **end** of the **abc.pml** script
√ - **Add** # in front of the load command to read: # **`load 2BIW.pdb2`**
√ - **Rotate** the molecule **again** in another orientation <u>or</u> you can just **type**

reset instead to go back to the default opening view.
√ - **Rerun** the **script** by typing **@abc.pml** on the line command.

The PDB file will not be reloaded since we commented out the line with the load command (#). The formatting and coloring of the protein will occur unnoticed since they are already in the cartoon state colored by our chosen secondary structure colors. But the orientation of the molecule will be restored.

Note: if the script appears NOT to function, make sure that the end-of-line is that appropriate to your operating system!

✔ **READ**

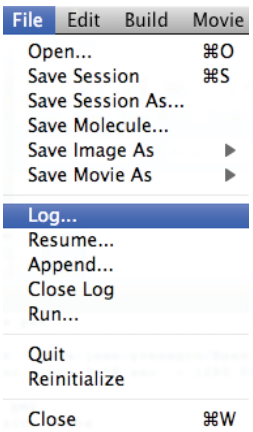
The PyMol way: this is a nice way to work with PyMol. Simply create and modify a script as you go, commenting out the loading of the PDB file(s) after the first run. It is a preferred way of working with PyMol that leaves the legacy of a finished script with complex commands. Therefore you can create your own repository of complex scripts, as a way to safeguard these commands for future use and reference. In addition, you can run the script again and again as you build it, line by line.

7.9 Automatic script making: Log menu

PyMol offers an easy way to make scripts: save a log file of typed line-commands

✔ **READ**

All commands are echoed on the Text panel together with any additional text triggered by the command. While it is possible to use Copy/Paste from this panel, it is much more convenient to have a separate file logging only the typed commands. Furthermore, this log file is *de facto* a script!



The logging can start after the **File > Log…** command is requested and a file name is given, e.g. **test.log**. The file can be saved anywhere on the hard drive, for example the desktop. The menus allow to close the current log, resume logging or append to any previously created plain text file.

✔ **TASK**

```
cd desktop
```

On your own create a log file while typing the commands on the right hand side on the PyMol line-command:

```
load 2BIW.pdb2
select hetatm
show stick
hide stick
show stick, sele
color yellow, sele
```

Reinitialize PyMol (**File > Reinitialize**) and run this test.log file with the following command: **cd desktop** **@test.log**

Note: if you want to run it from the **File > Run…** menu, you may need to **rename** the file with a .pml file extention e.g. **test.log.pml** otherwise the file cannot be opened by this method.

Practical note: previously typed command on the line-command can be recalled by using the "up-arrow" key.

8 *PyMol - Exercise U*: Select command, parameters, scripting, and subsets.

Understanding the content of this exercise can be beneficial to your PyMol skills!

✔ **READ**

The PyMol "select" line-command has the special property to create atom selections with a chosen name appearing within the Names Panel. This is a nice feature, but some confusion can arise depending on the words that are used.

For example, in the abc.pml script in the previous exercise, we used the command "color red, ss h" to color in red the secondary structures (ss) that are in helix (h) form. However the command "color red, ss helix" would not at this time have any effect, and PyMol might give us an error message. The reason is that "helix" has no meaning at this point.

Interestingly this command CAN work if "helix" is defined first.

"helix" can be defined when an atom selection is created with the "select" command. Any subsequent command can then contain the chosen word, and the commands will only apply to that atom selection, which can be a subset of a larger selection.

Note: When switching from Rasmol this can be confusing, as the command

"select helix" is a valid command on the Rasmol command line.

8.1 Defining subsets by selection: an illustrated example

Preliminary:
First let's assume that you just ran the abc.pml script above on a fresh start of PyMol. If not, simply either quit and restart PyMol or reinitialize it; then in the line-command area type `cd desktop` press return and then type `@abc.pml` to reach the same state as the previous exercise. In the script the helices are made red. Here we will make them green.

✔ TASK

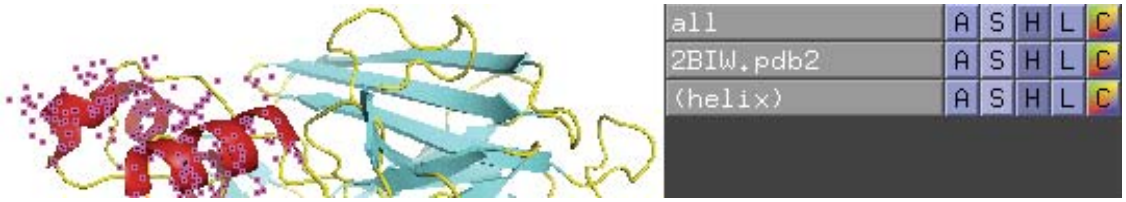
Within the line command type:

color green, helix

PyMOL> **color green, helix**
Selector-Error: Invalid Selection Name.
helix<--

Note the error echoed within the PyMol text window

Within the line command type: **select helix, ss h**	Two things will happen: a new name "(helix)" will appear within the Names Panel, and all the atoms participating in a helix structure will be selected within the Viewer and decorated with pink squares:



Within the line command type: **color green, helix**	Note: unlike the previous attempt, this time the command worked and the helices were colored green.

Note: The (helix) selection can be deleted with the line-command: delete helix. Alternatively the selection can be deleted with the "A" Actions menu "delete selection." The next menu allows to rename the selection.

READ

Important lesson: the word “helix” here could have been any other word such as “MyHelixSelection.”

…any word can do…

Using a “My” prefix before some of the selections can help avoid confusing words that are commands and words that are the names of created selections. This can be important when writing or deciphering other people’s scripts!

Another important difference to note is how a selection is shown in parentheses within the Names panel, while other objectsare not.

8.2 PyMol Selectors: keywords to make atom selections

Reminder: PDB (and other) 3D coordinate files are structured with certain fields arranged in columns. PyMol can detect and interpret some of these fields. For example, in the ATOM records, the “atom name column” is the column where the atoms are given a name, such as C, CA, CB, CG, or CD for example, corresponding to the asymmetric carbon, the alpha-, beta-, gamma- and delta-carbons of an amino acid respectively. Similarly, there is a column with the name of amino acids, and clearly more than one line of ATOM records make up for all the atoms of one amino acid. Finally, a chain and sequence numbers are also in column before the XYZ *Cartesian* coordinates.

```
ATOM   3752  N   GLN B 489      44.222   1.408  52.889  1.00 68.12           N
ATOM   3753  CA  GLN B 489      45.155   0.463  53.497  1.00 70.72           C
ATOM   3754  C   GLN B 489      45.787   1.050  54.759  1.00 71.44           C
ATOM   3755  O   GLN B 489      46.202   2.212  54.757  1.00 71.87           O
ATOM   3756  CB  GLN B 489      46.237   0.102  52.476  1.00 70.38           C
ATOM   3757  CG  GLN B 489      46.909  -1.223  52.738  1.00 72.85           C
ATOM   3758  CD  GLN B 489      47.621  -1.821  51.503  1.00 73.43           C
ATOM   3759  OE1 GLN B 489      47.815  -1.162  50.482  1.00 75.73           O
ATOM   3760  NE2 GLN B 489      48.014  -3.086  51.614  1.00 76.50           N
```

The complete list of selectors can be found on the PyMol manual (http://pymol.sourceforge.net/html/index.html) or it’s echo as the wikipedia version (http://www.pymolwiki.org/index.php/)

In summary, below are the most relevant selectors for working with macromolecules (proteins and nucleic acids) .

Each definition below is followed by an example. Remember: as we have seen before, the word after the word “select” is of your own creation. Adding “My” to the word may be a good reminder of that.

READ

Matching Property Selector	***Identifier and Example***
symbol	*chemical-symbol-list* list of 1- or 2-letter chemical symols from the periodic table `PyMol> select Mypolars , symbol o+n`

name	*atom-name-list* list of up to 4-letter codes for atoms in proteins or nucleic acids `PyMol> select Mycarbons , name ca+cb+cg+cd`

resn	*residue-name-list*
	list of 3-letter codes for amino acids
	`PyMol> select Myaas , resn asp+glu+asn+gln`
	
	list of up to 2-letter codes for nucleic acids
	`PyMol> select Mybases , resn a+g`

resi	*residue-identifier-list*
	list of up to 4-digit residue numbers
	`PyMol> select Myresidues , resi 1+2+20+8590`
	
	Residue-identifier-range
	`PyMol> select MyNterm , resi 1-25`

chain	*chain-identifier-list* list of single letter (rarely numbers) of the chain `PyMol> select MyChain , chain a`
ss	*Secondary-structure-type* list of single letters `PyMol> select Myalphas , ss hs+l+""`

8.3 Putting it together: a fancier script

Lets make a last script to wrap it up. Create a script file called abc2.pml with TextEdit or other means, and save it on the desktop.

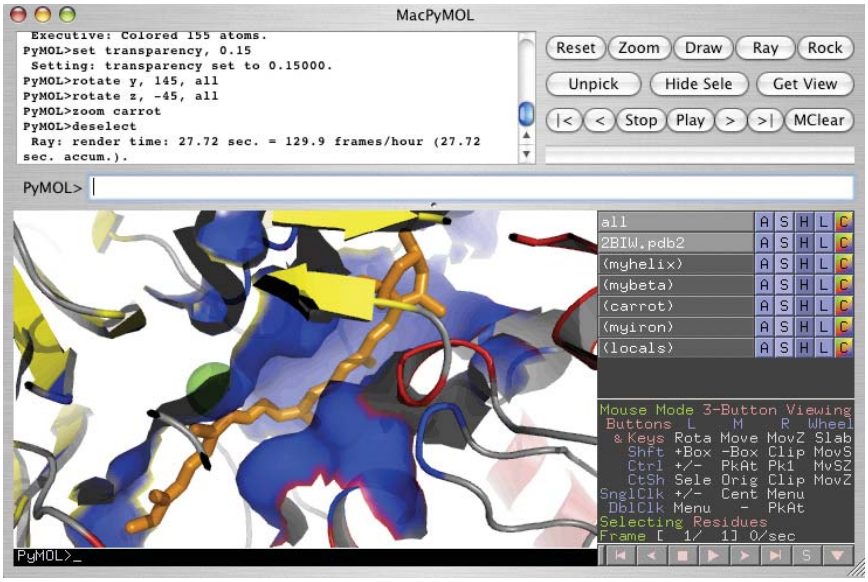
Within abc2.pml type the following script. *At this point you can omit typing the commented (#) lines*, but remember that placing comments within your scripts (annotation) can save (lots of) time and remove headaches in the future…

```
# my fancy script (assumes cd desktop for these exercises)
# first lets reset everything without need to quit PyMol
delete all
# set some parameters, even if some were done in previous
# white background:
bg_color white
# antialias: 1 for smooth images, 0 for jagged
set antialias = 1
# now reload the PDB file
load 2BIW.pdb2
# hide everything (all lines and so on)
hide everything
# show cartoon ribbon for the protein
show cartoon
# keep standard helix, strand and loops representations.
# other options would be: cartoon loop, cartoon rect,
# cartoon oval, and cartoon tube.
```

```
cartoon automatic
# make the helices with edges as in Molcript.
# 1 is on, 0 is off
set cartoon_fancy_helices=1
# color the inside of the helices in gray
set cartoon_highlight_color, gray
# color everything gray, and change other things later
color gray
# make selection of alpha helices and color it
select myhelix, ss h
color red, myhelix
# make selection of beta sheets and color it
select mybeta, ss s
color yellow, mybeta
# The PDB file contains a carotenoid ligand named 3ON
# it carrot when selecting it, show it as stick,
# and color it orange
select carrot, resn 3ON
show stick, carrot
color orange, carrot
# change the stick thickness from the default 0.25 to 0.40
set stick_radius = 0.40
# Inspection of the PDB file reveals there is also an iron
# select it and name it myiron, color it green and show it
# as a sphere. Then increase the sphere look quality.
select myiron, symbol FE
show sphere, myiron
color green, myiron
set sphere_quality = 8
# Select all full amino acids within 4 angstroms of the ligand
# and call this selection locals. Show them as a blue surface.
select locals, byres (chain B and carrot) around 4
show surface, locals
color blue, locals
# make the surface 15% transparent (85% opaque)
set transparency, 0.15
# Rotate about the Y axis to see the inside of the pocket
rotate y, 145, all
rotate z, -45, all
      # zoom in on the ligand
      zoom carrot
      # make sure nothing is selected to avoid the little pink
      # squares.
      deselect
      # end of this script – you could uncomment the next line to
      # make a fancy image:
      # ray
```

Within the PyMol line command **type @abc2.pml** to activate the script.

Your screen should be similar to this image, which is the ray-traced version.



8.4 Revisiting molecule rotation relative to each other

✔ **OPTIONAL** How can I rotate 2 molecules relative to each other?

There are 2 ways to accomplish this task: with the mouse or with line command.

Mouse method: reviewed with NMR, multistate structures exercise.

Line command method: learning by example.

The line command method is shown here by an example from the PyMol author.

1FJ1.pdb contains an antigenic fragment and its bound antibody in a single PDB file.

Once the PDB file is read in with the `load` command, the relevant protein chains are copied into independent objects and given a name (`anti` and `fab`.)

Some selection and coloring

```
load 1FJ1.pdb

# split PDB file

create anti=(chain F)
create fab=(chain A,B)

# delete original object
delete 1FJ1

# color objects
color green,fab
color pink,anti

# color interface
select inter = (byres ((fab
within 5 of anti)\
   or (anti within 5 of fab)))
```

is done to help visualize.
 	
`inter` is the name given to the interaction area.
`byres` helps select complete rather than partial amino acids.
When a line is long in a script, the charater \ indicates that the command continues on the next line.
 	
The command `orient` orients the molecule along the XYZ axes.
The command `origin` places
	`color yellow,inter`
	
	`# splay apart`
	`orient`
	`origin fab`
	`rotate y,60,fab`
	`origin anti`
	`rotate y,-60, anti`
	
	`# zoom interface region`
	`zoom inter`
	`show sph,inter`
	`disable inter`

Provided the PDB file 1FJ1.pdb (http://www.rcsb.org) is available, this script called split.pml can be run from the line command with @split.pml or called from the File>Run… menu cascade IF the filename extension is .pml.

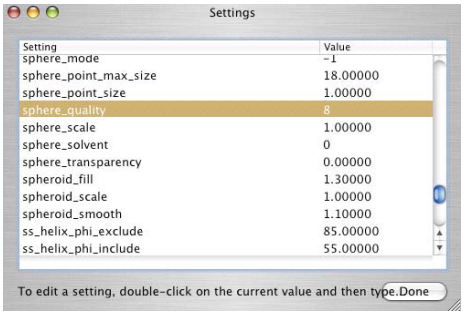
8.5 Where to go from here ?

Writing a script may seem tedious at first, but once it's written it can be saved, rerun, upgraded, placed on the web, and exchanged with other people via email.

Phylosophy: *Writing scripts is the PyMol way!*

***Where do I find the relevant variables such as sphere_quality*?**

The top menu cascade **Setting > Edit All…** opens a window that lists all the variables and their current value. The values can be changed interactively within this window, or changed in the script as we have just done. This image shows the current value for "sphere_quality after running the script: it has now the value 8, given by the script.



Learning by example: Web PyMol resources

Learning by example is what we have done so far, at a slow pace. However there are many web sites that offer PyMol help and scripts, often annotated with

the # comments to explain what is inside. Studying other's scripts is a way to learn some fancy options. Here are a few web sites to study at your own pace:

• Creating a eye-catching figure with Pymol: (a must see!)
`http://www.doe-mbi.ucla.edu/~sawaya/tutorials/Graphics/pymol.html`

• Image gallery with corresponding PyMol scripts
`http://www.chem.ucsb.edu/~molvisual/dna_biochem.html`

• PyMol tips/scripts (`http://www.rubor.de/bioinf/`):
`http://www.rubor.de/bioinf/pymol_tips.html`

• Introduction to Using PyMOL
`http://pymol.org/`

• Visualizing Protein Structures - A Practical Introduction to PyMOL
`http://www.ii.uib.no/~pal/teaching/mol305/files/pymol_intro.pdf`

• Brief PyMOL tutorial:
`http://www.mrc-lmb.cam.ac.uk/rlw/text/pymol_tutorial.htm`

• My PyMOL Script repository (*for advanced use*, including Python language)
`http://adelie.biochem.queensu.ca/~rlc/work/pymol/`

End of PyMol tutorial. Close or Quit all programs before leaving the classroom.