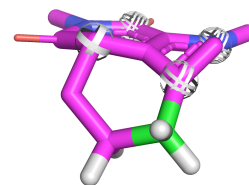
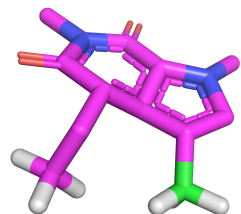
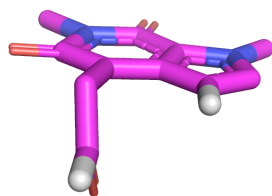
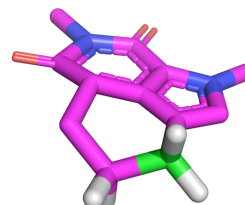
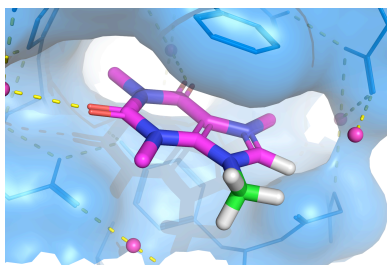
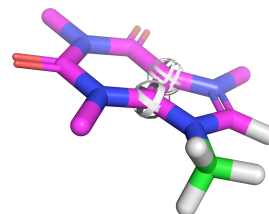
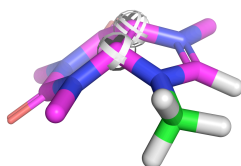
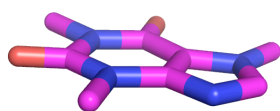


Molecular Editing and Modeling in PyMOL



Copyright Notice

This Tutorial is Copyright (C) Schrödinger. All Rights Reserved. Unauthorized reproduction or dissemination is prohibited under United States and international copyright laws. "Molecular Editing and Modeling in PyMOL" is a PyMOL Incentive Product created for the exclusive use of PyMOL Subscribers who sponsor the effort financially. Unrestricted distribution of this material could jeopardize the financial integrity of the PyMOL project, so please DO NOT POST THIS DOCUMENT IN AN INSECURE OR PUBLICLY ACCESSIBLE LOCATION. Current PyMOL Subscribers may only distribute copies of this tutorial internally to users within their organization who are covered by the scope of the subscription. If you come into possession of an inappropriate copy of this document, please either delete it or contact sales@schrodinger.com to purchase an appropriate PyMOL Subscription.

About This Booklet

This is a follow-along guide for the *Molecular Editing and Modeling in PyMOL* classroom tutorial taught by Schrödinger. It covers using PyMOL's Builder, creating and editing molecules, cleaning modified structures with the MMFF94 forcefield, protein mutagenesis, and molecular morphing.

This tutorial was created for PyMOL version 1.2 or greater running under Windows, Mac, or Linux.

Requirements

Before starting this tutorial, please be sure that you have the following.

- PyMOL version 1.2 or greater
- A 3-button wheel mouse
- Tutorial-specific data files found in the **MolecularEditing** subfolder of the **PyMOLTutorials** archive supplied by Schrödinger. This archive is typically provided as a compressed "ZIP" file, which should be extracted on to your Desktop

Read and understand, or attend the Introduction to PyMOL classroom tutorial taught by Schrödinger.

Be sure that your keyboard Caps Lock key is turned off when using PyMOL.

Intended Audience

This tutorial is intended for users who have completed the one-hour Introduction to PyMOL course or are familiar enough with PyMOL to type commands, read and run simple scripts, and easily navigate with the mouse.

Table of Contents

Small Molecule Building	4
The Builder	4
Advanced Molecular Building & Structure Editing	7
Small Molecule Cleanup	11
Editing the Caffeine Receptor	11
Protein Mutagenesis and	14
Main Chain Adjustments	14
The Mutagenesis Wizard	14
Animating a Molecular Morph	16
Next Steps	22
Other Courses	22
Joining the PyMOL–Users Mailing List	22
Visiting the PyMOLWiki Community Web Site	22
Accessing the Official Documentation Site	22

Small Molecule Building

The Builder

PyMOL comes with a geometry- and chemistry-aware molecular builder. To load the builder, please click on the **Builder** button in the Upper Control Window (far right). Once the builder loads, the Upper Control Window looks like Figure SMB.1. Please take a moment to review the annotations and structure of the Builder.

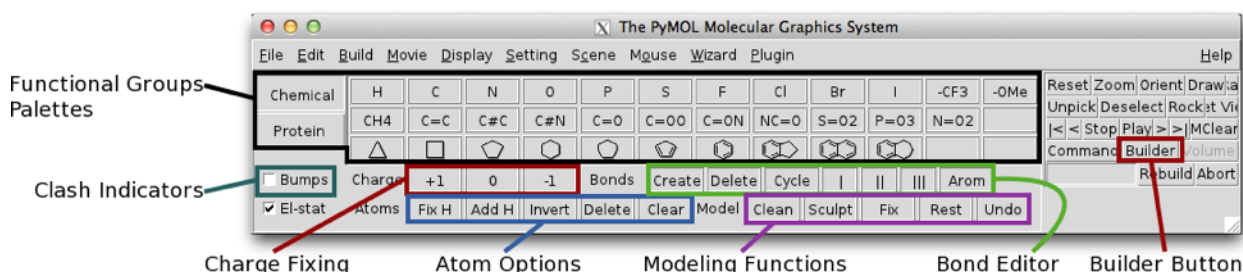


Figure SMB.1. The annotated Builder. To show this window, click on the Builder button in the Upper Control Window (far right or image). Here, the “Chemical” palette of functional groups is shown. Clicking on “Protein” will enable the protein-based functional groups and amino acids buttons.

The buttons in the Builder are usually linked to wizards. That means when we click a button in the Builder, PyMOL will then issue a message in the upper left hand corner of the Viewer Window. We respond to this message with an action, like clicking on an atom or bond.

The main features of the Builder are as follows. The “Functional Group Palettes” contain commonly used functional groups for building small organic molecules (Chemical palette) and for building protein chains (Protein palette). Please click on both the Chemical and Protein tabs to see which functional groups PyMOL makes available. The “Clash Indicators” are visual cues used in sculpting. The Bumps indicator shows small disks when a steric clash is detected. The “El-stat” checkbox controls whether or not electrostatic effects are considered when cleaning. “Charge Fixing” allows us to click on an atom and assign a formal charge to it. The “Atom Options” buttons allow us easy access to fixing and adding hydrogens as well as inverting, deleting and clearing any atom. “Modeling Functions” are used to clean up the structure and test for steric clashes. The “Bond Editor” allows us to create, delete, cycle valence and set valence on bonds.

Let’s learn how the Builder works by creating a caffeine molecule from scratch. If you haven’t already, please launch PyMOL.

Try it—Creating a Caffeine Molecule from Scratch

For reference, the 2D structure of caffeine is shown in Figure SMB.2.

Here is how we make the caffeine molecule. We will start by creating a new object from an indanyl group. We then replace the indanyl carbons with nitrogens at the appropriate locations according to the reference figure. Next, we add the carbonyl oxygens to the six-membered ring. We then fix their bond valences. Next, we add the three methyl groups. Then, we change necessary valences to match the caffeine structure in Figure SMB.2. Last, we clean up the structure.

As we guide you through this process please see Figure SMB.2 for the

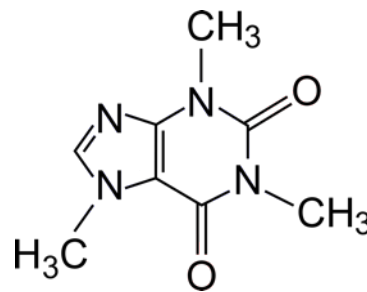



Figure SMB.2: The caffeine molecule. We will build this de novo using PyMOL’s molecular Builder.

final reference structure and Figure SMB.3 for step-wise fragments toward the final structure.

Place the Indanyl Group as a New Object


If the Chemical groups are not shown, please click on the Chemical palette to display them. Please locate the indane functional group button: . You can see group names by passing your mouse cursor over each functional group. When you do this the chemical name pops up. Once you've found indane, please click on it.

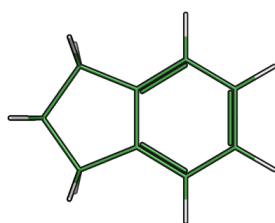
In the upper left of the Display Window, PyMOL displays, "Pick locations to attach indandyl," and in the lower right there are menu options for creating a new object, merging this with previous objects or quitting the wizard. Please press **Create as New Object**. PyMOL creates the indanyl object and names it **obj01**. Please press **Done**. Please rename "obj01" to "caffeine" by selecting **A > Action > Rename Object** for obj01. Using your mouse, please orient the molecule so that the five-membered ring is on the left as in Figure SMB.2.

Swap-in Four Nitrogens

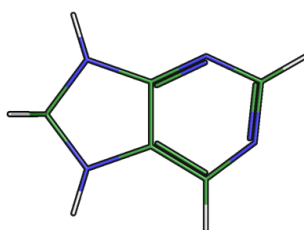
Next, let's swap out the carbons for the four nitrogens. Please find the nitrogens (N) in the reference structure. At present in our PyMOL session those atoms are carbons and need to be replaced by nitrogens. Please click on the N in the Builder. Now, when we click on an atom it will be replaced by Nitrogen. Please replace the four carbons in the indanyl ring according to the structure in Figure SMB.2 by clicking directly on the atoms to change. When the four nitrogens are in place, your molecule should look like panel two from Figure SMB3. Notice that PyMOL allows you to build fragments with incorrect chemical assignments. In the third panel, we have five bonds being made to one carbon atom. It is up to the user to clean up these spurious bonds. We will fix this in the coming steps, before minimizing the structure.

Add Two Carbonyl Carbons & Set Their Valence

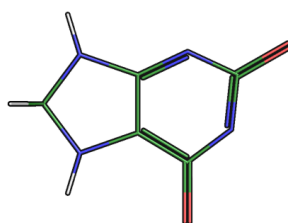
Now, let's add the two carbonyl groups. To do this we, will replace the two hydrogen atoms with oxygen atoms. Then we update the valence on the oxygens to be a double bond. Please click on **O** in the Builder to activate replacement by oxygen atoms. Find the hydrogens in the reference structure to replace, and in PyMOL replace the two hydrogens by clicking on them. Now, let's change the bond valences. Click the double-bond button  in the Builder. We can now click on a bond to change it to a double bond. Please click on the bonds being made to the new oxygens. You should now have a molecule that looks like panel 3 of Figure SMB.3.



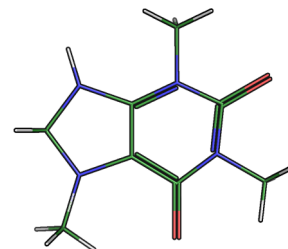
Step 1: Indanyl as a new object.



Step 2: Four nitrogens swapped in.



Step 3: Oxygens added and valences set to double.



Step 4: Methyl groups added.

Figure SMB3. Fragment-based steps in creating the Caffeine molecule *de novo*.

Add Methyl Groups

Now we need to add the three methyl groups. To do this, please click on the methyl (CH4) button from the Builder. We may now place methyl groups. Using the reference structure as a guide, select the three nitrogen atoms onto which we add the methyl groups.

Set Remaining Valences

We are almost done. Now, using the valence buttons, please change the bond valences in the rings to match those of Figure SMB.2. Once this is done, we just need to cleanup the structure.

Clean the Structure

Please click **A > Action > Clean** for the caffeine molecule. PyMOL now uses the MMFF to perform energy minimization. Your final molecule should look like Figure SMB.4. Please save this session on your **Desktop** as caffeine.pse.

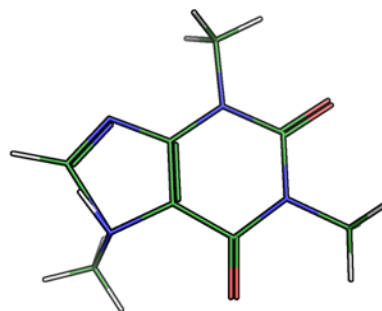


Figure SMB.4: Caffeine molecule generated from scratch.

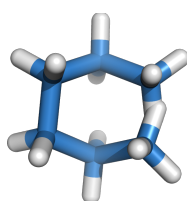
Common Building Tasks—Building Rings

It is often necessary to build ring systems. Here, we build a simple cyclohexyl ring system to show you how it can be done in general. Please delete the caffeine molecule we just built. Please select the methyl (CH4) group from the Builder. Click **Create as New Object** in the Viewer Window. PyMOL creates the lone methyl group and names it obj01. Please rename this to “ring” by selecting **A > Action > Rename Object** for obj01 and typing the new name.

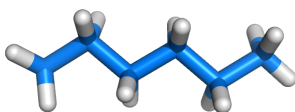
Add Six Methyl Groups

Now, please add five more methyl groups in a non-branching fashion, by repeatedly clicking on hydrogen atoms to replace with methyls. You might have something similar to Figure SMB5a-b, SMB5c will not work as we branched the chain so we would get a cyclopentyl ring instead of a cyclohexyl.

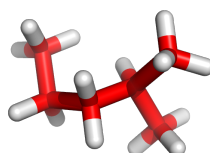
Once we have added the six necessary carbons through the repeated addition of methyl groups, we now need to create a bond between the two terminal carbons to close the ring.



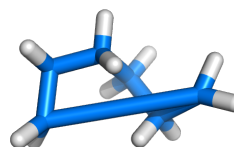
SMB5a: Ring-like structure built; right-most atoms are not bonded.



SMB5.b: Methyl groups added in an extended chain.



SMB5.c: Branch introduced; this will not close into a cyclohexyl ring.



SMB5.d: Bond created and shown between two terminal carbons. Cleaning will fix the structure.

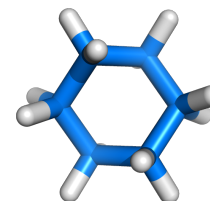


Figure SMB5e. Completed and minimized cyclohexane.

Figure SMB5a-e: (a-b) Possible hexanes which we may close to create a ring. (c) A branched chain will not close to a cyclohexyl ring. (d) Bond created between two terminal carbons. Once we clean this structure we will have a chemically and structurally valid cyclohexane. (e) The finished cyclohexane.

Close the Ring by Creating a New Bond

Please click the **Create** button in the Bond editor of the Builder. PyMOL asks you to click on the two atoms to bond. Doing this will create a bond, but it will not perform any minimization. Please click on the

two terminal atoms to bond them together. Bonds are similarly deleted. To delete a bond, click on the **Delete** button in the Builder. Then, click on the bond to remove it.

Clean the Final Structure

Now, that last bond is probably rather ridiculous. Let's clean up the structure by clicking on **A > Action > Clean** for the "ring" object. PyMOL cleans the ring structure. The result is a chemically valid cyclohexyl ring, as shown in Figure SMB5e.

Advanced Molecular Building & Structure Editing

Selecting vs Picking

When we prepare visualizations in PyMOL we *select* atoms in which we have interest. When we edit, however, we *pick* atoms in which we have interest. Just as the default selection has a name, "(sele)", so do default picked atoms, "pk1", "pk2", "pk3" and "pk4". Picking atoms is easy, just click on them using the left mouse button while *in editing mode*. If you pick an atom a globe appears over it, as in Figure SMB5.5. The picked atoms typically define geometric constraints around which we can modify atoms. For example if we imagined a hinge being placed at PK1 and PK4, in Figure SMB5.5, we could swing the bottom four atoms around that hinge region.

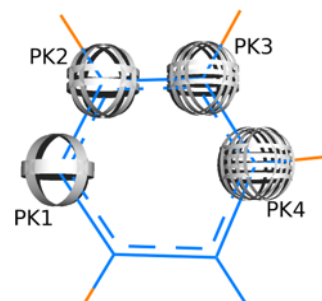


Figure SMB5.5: Picked atoms. Notice the different types of globes for picked atoms one through four.

The Structure Editing Session File

Please load the "editing_kinase.pse" file in the **PyMOLTutorials/MolecularEditing/sessions** directory. This is a session file prepared by Schrödinger for use in this tutorial. We have focused on the ligand binding pocket of this kinase and hid any unnecessary information. We will use this session to edit various parts of the ligand and test our changes by cleaning up the structure. Once the session file loads your screen should look like Figure SMB5.6. In this session we have created named selections. We will zoom in on each named selection and perform the edit suggested by the selection's name. For example, if the selection name is "to_delete," we will zoom in on that atom and use the Builder to delete it. Let's begin.

Editing Atoms

Moving Atoms

Please activate the "(to_move)" selection by clicking on its name. Please now zoom on this selection by selecting its **A > Action > Zoom**. If some of the surface gets in your way, please just use the mouse to rotate it out of the way.

To move an atom in PyMOL, just CTRL-left-click on the atom and drag it to its new position. Be sure you're in Editing Mode, or this won't work. Please try dragging the atom defined by the "(to_move)" selection, using this technique.

Another convenient way to move an atom is to perform a click-and-a-half on the atom and drag it with the mouse. A click-and-a-half is a double-click but you don't release the mouse button. Try it. Move the atom using the click-and-a-half method, too.

Now that we have made a change to the structure of the ligand in the binding pocket, we can run the MMFF on it to minimize the complex. Please choose **A > Action > Clean** for the "(pocket)" selection.

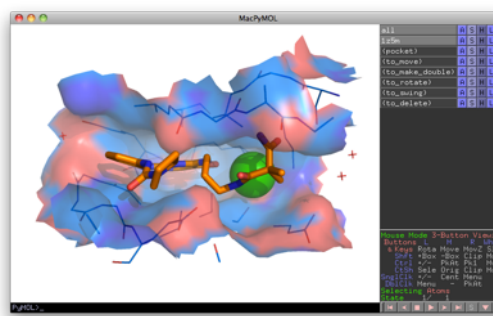


Figure SMB5.6: PyMOL immediately editing_kinase session file loaded.

PyMOL calculates the new atom positions. After a few moments the screen will update. Try moving the atom to a new location and clean again. Let's move on to picking atoms.

Please turn on the “(to_swing)” selection. Please zoom in on this selection, again by using its **A > Action > Zoom**. Again, if any surface is in the way of your view of the atoms, please use the mouse to rotate it out of the way. Please pick the two atoms indicated by the “(to_swing)” selection. We will use these picked atoms to swing the hanging fragment. Read on.

Hinge Movements

Picking two atoms with the mouse will define a hinge from which we can swing atoms. Take a look at Figure SMB7.

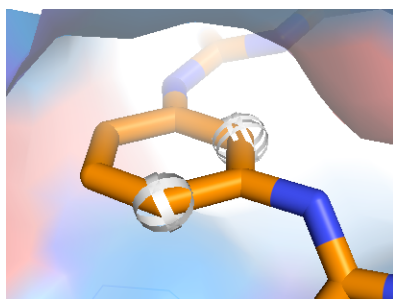


Figure SMB7a. Hinge defined by selecting two atoms. The substructures can be moved independently.

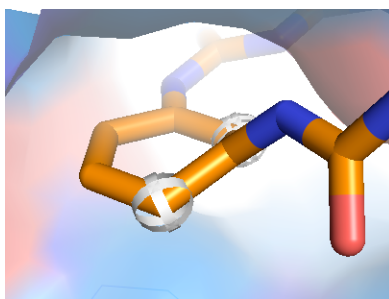


Figure SMB7b. Right-side of the hinge region is swung up, via a click-and-a-half and dragging upwards.

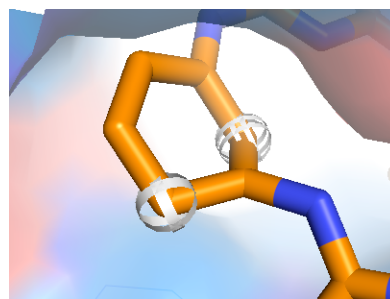


Figure SMB7c. Left-side of the hinge swung up.

Figure SMB7. The globes corresponding to Pk1 and Pk2 are shown. These two atoms define one hinge region about which two fragments may swing. We can swing the fragment to the right of the line that connects the two globes or the fragment on the left of that line.

Now that we have picked two atoms, we have defined a hinge region. We can swing the hinge region using a click-and-a-half and drag of the mouse. Please try that now. Just swing the hinge region by a few tens of degrees. Try cleaning up the pocket again—choose **A > Action > Clean** for the “(pocket)” selection.

Editing Bonds

Please turn on the “(to_rotate)” selection and zoom on it.

Picking Bonds

The first step in any of working with bonds is to *pick* a bond. This identifies to PyMOL the bond you wish to modify. To *pick* a bond, you simply CTRL-right click on it (CTRL-two-finger-click on a MacBook). If you have successfully picked a bond, PyMOL will draw a white cuff around the bond as in Figure SMB6. The bond in Figure SMB6 was chosen on its right side. We can also select the left side of the bond just by clicking to the left of the half of the bond.

Carefully pick the right-hand side of the bond being made to the two atoms in the “(to_rotate)” selection. Now, try picking the left-hand side of the very same bond. Notice that the cuff moves.

Rotating Bond Torsions

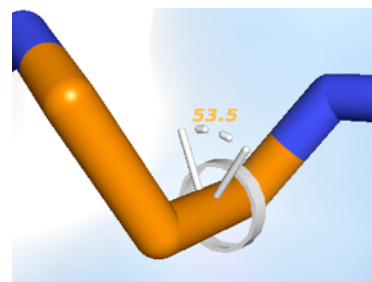


Figure SMB6. A picked bond has a white cuff around it. Many times a picked bond will also display its torsion angle as this one does.

Once you've picked a bond with CTRL-right click, you can rotate the bond by holding down the CTRL key and dragging with the right mouse button depressed. Picking a bond defines two fragments: the fragment to the left of the bond and the fragment to the right of the bond. You can also rotate those fragments by holding CTRL while dragging them with the *left*-mouse button depressed. Please take a moment to investigate Figure 29. If you make a mistake, CTRL-Z will undo recent conformational changes.

Please zoom out enough so that you can see the end of ligand. Now, please select the right-hand side of the indicated bond. Now, please rotate the bond torsion by holding down the CTRL key on your keyboard and dragging up and down with the right-mouse button depressed. The fragment rotates in place.

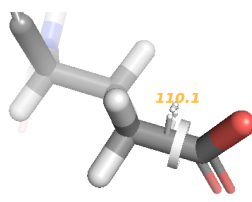
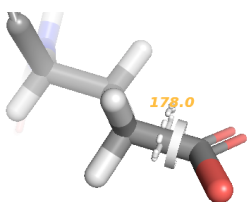


Figure 29: Rotating torsion bonds. Right side of the bond is selected (left image); and molecule is modified by rotating the torsion bond with CTRL-right mouse button dragging (right image). Notice that the white cuff is on the right half of the bond. This indicates that the right side of the bond will be rotated. The left side is unchanged. Compare this to Figure 30.

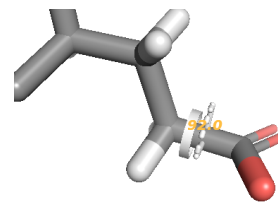
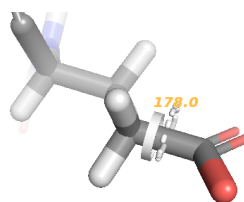


Figure 30: Similar to Figure 29, but left side of the bond is selected (left image), and the white cuff indicates this fact, as it is located on the left half of the bond. (Right image) Molecule shown after slight rotation. The right side of the molecule is unchanged.

Changing Bond Valences

Please activate and zoom in on the selection “(to_make_double)”.

One of the problem with PDB files is that they do not contain bond valences for ligands. As a consequence, it is not always possible to display correct chemistry for ligands. PyMOL may not always recognize the correct hydrogen-bonding interactions between proteins and ligands until the valences have been properly specified.

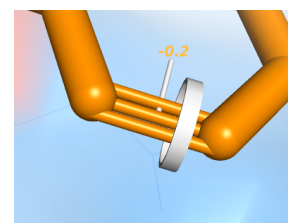
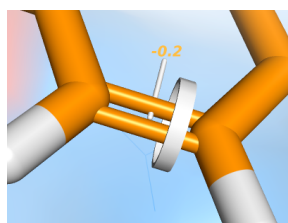
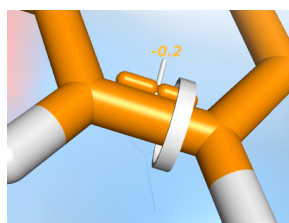
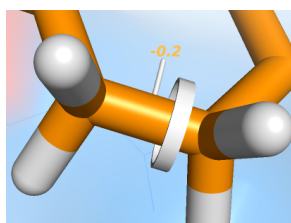


Figure SMB7a: The bond is picked. The original single bond is shown.


Figure SMB7b: After pressing CTRL-W once; an aromatic or 1.5 bond is created.

Figure SMB7c: After pressing CTRL-W twice; a double bond is created.

Figure SMB7d: After pressing CTRL-W three times; a triple bond is created.

Figure SMB7: Changing Bond Valences.

To edit bond valences please click on the Builder button in Upper Control Window. Next, pick the bond you wish to edit, with CTRL-right-click. You should see the familiar white cuff appear around the bond SMB7. Please cycle through the available bond valences by clicking on the Cycle button in the Builder area of the Upper Control Window, or by pressing CTRL-W on your keyboard. Alternatively, once your

bond is selected, you can just set the valence clicking on the single , double , triple , or aromatic  bond settings in the Bond Editor.

Once you have set the correct valence for that bond, you can CTRL-right-click to select the next bond to fix, continuing until all valences have been fixed. Note that PyMOL adds missing hydrogens to the atoms involved in the bond. If hydrogens are not desired, use **A > Actions > Remove Hydrogens** for the object, after correcting all of the bond valences.

If the Builder isn't shown, please click on the Builder button. Now, please choose the double bond icon from the Builder. PyMOL asks you to choose a bond to set as double. Please select the bond between the oxygen and carbon in the selection "(to_make_double)". Try changing a few bonds in the ring system next to the bond you just changed.

Creating and Deleting Bonds

Please activate and zoom in on the selection called "(to_delete)". The goal here is to remove the carbonyl bond. Then, we will replace it with a single bond and convert it back to its original state as a double bond.

Please select the bond to remove, by CTRL-right-clicking on it. Now, please delete that bond by pressing CTRL-D, or by pressing the **Delete** button in the Bond Editor. PyMOL deletes the bond.

Let's re-create that bond. Please click on the **Create** button in the Builder's Bond Editor. Now, select the two atoms from which we just removed the bond. PyMOL creates the bond for us and assumes the bond is a single bond. Please update the valence on the bond to double. There is another way to create a bond in PyMOL. Select the two atoms and then press CTRL-T on your keyboard. This creates the bond.

Small Molecule Cleanup

We learned how to stitch together small molecules. Now, we can take a look at fine-tuning the process of cleaning up the neighborhood in which we are editing. Again, the “cleanup” function in PyMOL prepares input to the external MMFF program. Due to computational intractability MMFF only allows us to operate on structures with fewer than 1000 atoms. This section will teach you how to properly limit your editing neighborhoods and submit calculations to MMFF.

Editing the Caffeine Receptor

Please load the caffeine receptor session file we use throughout this section. The file is called “caffeine_editing.pse” and is located in the **PyMOLTutorials/MolecularEditing/sessions** directory. Once loaded your screen should look like Figure C.1. This file is PDB 1L5Q’s chain B. The session defines the selection named “caffeine” as the first full caffeine molecule. We zoomed in on that molecule and saved the session.

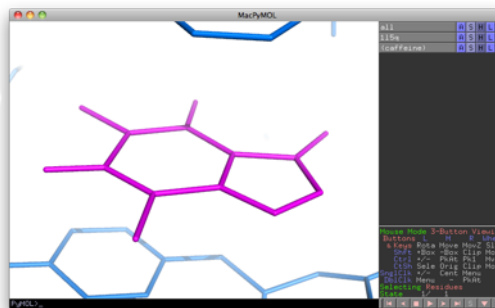


Figure C.1: Caffeine receptor session loaded in PyMOL.



Getting Ready to Edit

Before we consider more advanced editing let’s make sure our session is ready for the MMFF program. MMFF will only take input that has proper chemistry. MMFF is less strict about atoms locations, though. For example, if you make 42 bonds to a single carbon atom, MMFF will silently fail. However, if you move a single carbon atom 42 Angstroms from its original position while it’s still bound to its neighbors, MMFF will calculate the new position of the atom.

Try it—Preparing the View

Please click on the Builder tab. Notice the bonding in the caffeine molecule. Let’s recolor the protein by atom type so we can determine proper chemistry. Please click on 1L5Q’s **C > Color > By Element** and then choose the first menu option. The atoms are now colored by their element type.

Try it—Fixing Valences

MMFF will not accept those aromatic bonds inside the rings. We need to match the bonds as they are in Figure SMB2-copy. Please click on the Single Bond  in the Bond Editor. Convert all the aromatic bonds to single bonds, by clicking on the bonds in the five- and six-membered rings. Caffeine has two double bonds, please add those. Click on the Double Bond  in the Bond Editor. Then, just click on the bonds to convert to double. Please click Done to close the Double Bond wizard.

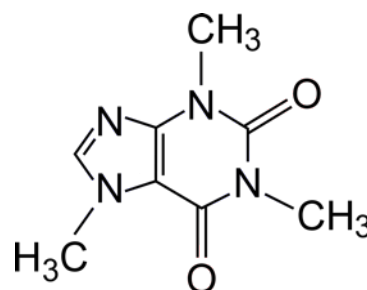


Figure SMB.2-copy: The caffeine molecule.

Try it—Preparing the Neighborhood

Our caffeine molecule is bound to a protein. If we just called MMFF on the caffeine molecule as it is now, MMFF would minimize the caffeine molecule by itself, without regard to the protein. So, we need to select a neighborhood of atoms around the caffeine to pass to MMFF. To do this, we will duplicate the caffeine selection and then expand it to a 5 Angstrom neighborhood.

Please duplicate the caffeine selection by choosing its **A > Action > Duplicate**. The selection, sel01, is created. Please expand this selection to consist of all atoms within 5 Angstroms of the sel01 by choosing it’s **A > Action > Modify > Expand > By 5 A**. The sel01 selection, originally the caffeine molecule, is

modified to consist of all atoms within a 5 Angstroms of caffeine. This is our neighborhood. Please take a second to review how we created this neighborhood, as we will be repeating this process every time we make an addition or deletion to our caffeine molecule. Now, please show the sel01 selection as a surface.

Try it—Solvent Interactions

Currently, we cannot see any solvent atoms. To fix this, we will duplicate the neighborhood selection and restrict it to just solvent atoms. We will then add hydrogens to them and view them as lines. Please duplicate the sel01, by selecting its **A > Action > Duplicate**. PyMOL creates a duplicate called sel02. Please now restrict it to just solvent atoms by clicking, **A > Action > Modify > Restrict > To Solvent**. Please click on 1L5Q and then **A > Action > Hydrogens > Add**. PyMOL generates hydrogen atoms for the protein. Now, please show the solvent as lines.

Restricting Atom Movement

Cleaning is the task of subjecting atoms to a forcefield in order to minimize molecular strain. Atoms are repositioned upon cleaning; however, this is not always desirable as we may want to restrict reference atoms from moving at all or dampen their movement with harmonic restraints. Using the **Fixed Atoms** and **Restrained Atoms** wizards, PyMOL gives us the ability to control which atoms are free to move, are fixed, or are harmonically restrained. Take a moment to find the “Fix” and “Rest” buttons in the Builder. Clicking these buttons start the wizards.

Fixing Atoms

Fixing an atom forces it to be stationary. Unfixed atoms may move about, but the fixed ones have preserved coordinates. This is useful if we add a fragment and want to minimize the fragment with respect to the currently loaded structure. Let's try that here. First we will add a methyl group to our caffeine and then fix everything but the methyl group. We will then minimize the new methyl group with respect to the rest of the structure.

Try it—Add a Fragment and Update the Neighborhood

Please add a methyl group to caffeine. We added the green-colored group in Figure C.2. Because we just added atoms to the caffeine selection, we need to update the selection to include our new atoms. To update the selection, please click on its **A > Action > Modify > Complete > Molecules**. This extends the selection to complete the organic molecule. Now, please recreate the neighborhood as we did above, by first duplicating (caffeine) and then modifying the new selection by selecting its **A > Action > Modify > Expand > By 5 A**.

Try it—Fixing Atoms

Now, let's fix all atoms except the new methyl group. Please click on **Fix** in the Builder menu. “Fixed Atoms” appears in the Viewer Window. You can fix All, None or just alpha carbons. You can toggle fixed/unfixed by just clicking on an atom. Let's try it. Please click the **All** button under the “Fixed Atoms” menu. Now, let's unfix the methyl. Please click the central carbon of the new methyl. Now in the Fixed Atoms menu, click **Less**. PyMOL takes your one-clicked atom and unfixes it's immediate neighbors. PyMOL unfixed the nitrogen at which we added our methyl. Please refix that by clicking on it.

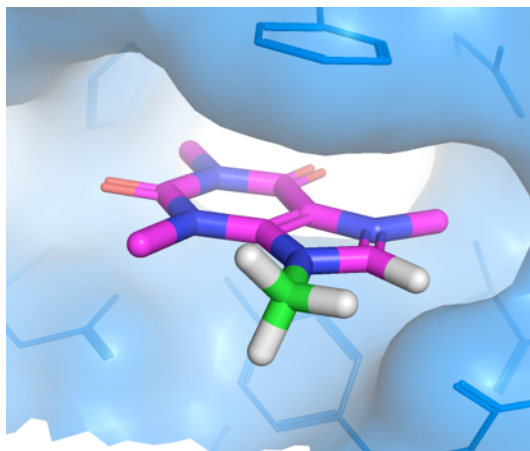


Figure C.2. Caffeine augmented with a methyl group. We will “fix” the atoms core caffeine molecule before cleaning. Then, we compare the results to “restrained” atomic positions.

Now, please clean the constrained neighborhood by selecting its **A > Action > Clean**. PyMOL cleans the new methyl. Here only the methyl could move, but the rest of the structure was taken into account when cleaning.

Restraining Atoms

PyMOL can also harmonically restrain atomic repositioning under cleanup. The Restrained Atoms wizard looks and acts nearly identical to the Atom Fixing wizard. To restrain atoms, click the **Rest** button positioned in the Builder to the right of the **Fix** button.

Hints

Don't forget, you can find polar contacts using **A > Action > Find > Polar Contacts** to help in editing. Make sure you re-find them after making changes.

Troubleshooting

If you prepare a neighborhood to clean but MMFF doesn't perform any action then there is incorrect chemistry in your neighborhood, or there are atoms of types that MMFF doesn't know about. For example, many metal ions are not allowed by MMFF. (A full specification of MMFF can be found in XYZ.) The workaround at present is to restrict your selection to ignore nearby metal ions.

Protein Mutagenesis and Main Chain Adjustments

PyMOL allows us to perform amino acid mutagenesis, change bond torsions, and make large-scale main chain adjustments. We demonstrate those features in this chapter.

The Mutagenesis Wizard

PyMOL provides the Mutagenesis Wizard to ease the process of making amino acid substitutions. Please restart PyMOL and load the file 1z5m.pdb in the **PyMOLTutorials/MolecularEditing/pdbs** directory. Let's take a look at using the Mutagenesis Wizard.

The Layout

Please load the mutagenesis wizard by clicking **Wizard > Mutagenesis** from the Upper Control window. PyMOL displays the wizard's menus and buttons at the lower right of the Viewer Window, as shown in Figure PM1. The top six menu items, colored blue, are actually pop-up menus. The first, "Mutate to ..." contains a list of amino acid substitutes as shown in Figure PM2. This menu is used to select replacement residues. The next menu items, "N-Cap" and "C-Cap" control how mutated residue will be capped at its N- and C-terminal ends, respectively. The "Hydrogens" submenu controls how PyMOL uses hydrogens during mutagenesis. The options are to use the current hydrogens (if present), add them and retain them, and remove them altogether. The "Show" submenu controls the visualization of the residue to be mutated. This just helps us more clearly distinguish the residue to be changed. The last menu option, "Backbone In/Dependent Rotamers" controls the rotamers PyMOL will use when placing the amino acid. You can use the backbone dependent rotamers, which come from the Dunbrack rotamer library or use backbone independent rotamers. The "Apply" button will make the substitution when you are ready. "Clear" will halt any current configuration and reset the wizard. "Done" closes the wizard. Knowing what options exist, let's take a look at using the wizard to make a residue swap.

Using the Mutagenesis Wizard

Please locate a residue in 1z5m that you would like to change. Using the mouse, zoom in on this residue and clip out the neighbors so we can clearly focus on just one residue.

Select the Target Residue and Choose its Replacement Residue

PyMOL asks us to pick a residue. Using your mouse, please click on any atom in the residue you would like to change. PyMOL identifies the entire residue. PyMOL also displays the selected residue in its original position, and also a white colored copy of the residue in its first rotameric position as shown in Figure PM3. At this point, please click on the first menu which should be "No Mutation". PyMOL brings up a list of possible amino acids substitutions. Please choose a residue type different from the one you chose.



Figure PM1: Mutagenesis wizard opened. The blue-colored (top 6) buttons are actually pop-up menus. The bottom three (grey colored) bars are buttons.



Figure PM2: Clicking on the first button, brings up this menu. We choose our alternate residue here.

Choose a Rotamer

By default PyMOL will load the rotamers for your chosen residue type. Steric clashes are shown as small colored disks. The greater the clash, the larger and redder the disk appears; similarly, the smaller the clash the smaller and greener the disk appears. See Figure PM3. The white colored tyrosine residue in Figure PM3 is exhibiting great steric clash.

Now we can choose the rotamer for the new residue. Please scan through the list of available rotamers for your residue by pressing the Left/Right arrow keys on your keyboard. Clicking on the sixth menu item, you can change whether PyMOL offers backbone-dependent rotamers or backbone independent rotamers.

Please sift through the rotamers and choose one with minimal clashing. Click “Apply” to make the change. Click “Done” to exit the wizard.

Cleanup the Neighborhood

We may have introduced significant steric clash through mutagenesis. PyMOL ships with an implementation of the Merck Molecular Force Field (MMFF) which we can use to cleanup the neighborhood of the mutated residue. First, select the residue with your mouse. (Hint: make sure you mouse is selecting “Residues” and click any atom in the residue you just mutated.) Now, expand this single residue selection to be the neighborhood of the mutated residue. You do this by clicking on the selection’s **A > Action > Modify > Expand > By 5A**. Now please clean this modified selection by choosing **A > Action > Clean** for the expanded selection. PyMOL cleans the selection and moves the atoms to their positions of lowest (local) energy.

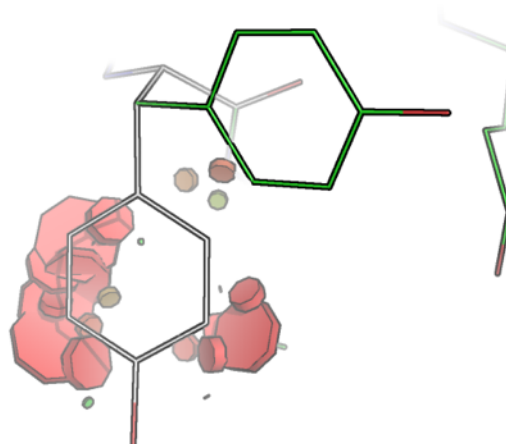


Figure PM3: Choosing rotameric position for residue. Notice the clash indicators shown as colored disks.

Animating a Molecular Morph

Many molecular systems undergo conformational changes as part of their natural function. In some cases, crystal structures can reveal distinct conformations for similar or even identical molecules under varied conditions. Molecular morphing can help the users to compare how two or more conformations relate to one another by creating a smooth and concerted 3D trajectory between them. It is important to understand that these artificial trajectories, commonly known as “molecular morphs” do not represent, and are not meant to suggest, actual physical pathways of inter-conversion. Rather, they are simply tools for illustration and comparison of complex conformational differences.

Molecular morphing in PyMOL follows the following steps.

1. create or load the first molecular structure
2. create or load the last molecular structure
3. align the first and last structures to each other
4. create the morphing input object from the aligned start and final states
5. call the morphing routine to create an output trajectory object

Morphing Between Two Conformations

Please quit PyMOL. In a plain-text editor, such as Microsoft Notepad, please open the **mtase_morph.pml** script in the **PyMOLTutorials/MolecularEditing/morphs** folder. A copy of **mtase_morph.pml** appears in Code 1. Here we walk you through the code, line by line.

Code 1: Methyltransferase morphing script.

```

1  # load start and final conformations
2  load ../pdbs/2z1b.pdb, startConf
3  load ../pdbs/2zvj.pdb, finalConf

4  # remove solvent atoms, they don't map 1-1 across structures
5  remove solvent

6  # align the two structures
7  align startConf, finalConf, object=aln

8  # make 2-state morph_in from state 1 of startConf and state 1 of finalConf
9  create morph_in, startConf in finalConf, 1, 1
10 create morph_in, finalConf in startConf, 1, 2

11 # import the rigimol module
12 from epymol import rigimol

13 # compute the morph and store the resulting
14 # trajectory in a new object called "morph_out"
15 rigimol.morph( "morph_in", "morph_out", refinement=2, async=1 )

```

Line 2 loads our starting object, and names it “startConf” instead of its default pdb file name “2zlb”. Line 3 loads the pdb file representing the final conformation, and names it “finalConf”. Line 5 removes the solvent atoms. While the structures of the two proteins have the same number of atoms, the waters in the pdb files don’t have a 1-1 correspondence so we remove them. Line 7 aligns the two proteins and then stores the alignment in the *alignment object* called “aln.” Line 9 creates the “morph_in” object by inserting state 1 of the startConf object into state 1 of the morph_in object. Line 10 adds finalConf’s 1st state into morph_in’s second state. So, morph_in is now a two-state object where state 1 is our start state and state 2 is our final state. Line 12 is a Python command that imports the module *rigimol* from the package *epymol*. Inside *rigimol* is the *morph* function, which we call in line 15. We pass the input structure, the output structure, the number of refinement steps, and a flag controlling responsiveness to the morph command.

Please find the icon for this script and drop it on the PyMOL icon. PyMOL launches and runs the script. When PyMOL prints the message “Morph: Refinement complete.” in the Upper Control Window, the morphing calculation is complete. When you see this message, please save your morph to a multi-state pdb file by typing, `save morph_out.pdb, morph_out, 0`. Now, again, quit PyMOL.

Making a Movie with the Morph Output

Please open the **mtase_movie.pml** script from the **PyMOLTutorials/MolecularEditing/morphs** directory into your simple text editor. The script is shown in Code 2. The goal of this movie is to visualize the morph. As the morph approaches the final conformation, to which a ligand is bound, we show the ligand in its binding pocket. Here we walk you through the script that creates this movie.

Line 3 loads our multi-state pdb file into PyMOL. This file was created when we ran the `mtase_morph.pml` script. Line 5 loads the original holo-structure. To remind ourselves that the holo-structure is being used temporarily just for its ligands and waters, we assign it the name “tmp.”

Next, on line 9, we copy the ligands from the tmp object to a new object that we call ligands. We copy from tmp’s first state and insert the object into ligands’ 30th state. We do this because we want the ligands and nearby waters to turn on when our movie reaches state 30. Line 11 is very much like line 9. However, in line 11 we pick up solvent atoms near the binding pocket, within 5 Ångstroms of the ligands. We are done with the tmp object, so we remove it in line 13.

Next, line 17 uses the `mset` command to program a movie. That command takes two types of arguments: a state-hold argument, which comes first, “1 x30” and a state-sweep argument, “1 -30” which follows. The state-hold argument, “1 x30” translates to “hold state 1 for 30 frames.” The state-sweep argument, “1 -30” translates to, “sweep through states 1 to 30 at one frame per state.” Line 17 continues with another state-hold argument, “30 x30” and another state sweep argument, “30 -1”. Our movie is now programmed to view state 1 for 30 frames. Then, sweep from state 1 to 30 over 30 frames. Next, it waits at state 30 for 30 frames before descending back down to state 1, taking 30 frames. Table 1 illustrates the usage of the `mset` command.

Command	Comment
<code>mset 1-30</code>	<i>Invalid</i> , no space after the 1.
<code>mset 1 -30</code>	Valid, sweep from state 1 to 30.
<code>mset 1x40</code>	<i>Invalid</i> , no space after the 1.
<code>mset 1 x40</code>	Valid, repeat state 1 for 40 frames.
<code>mset 10 -20 20 x30 20 -1</code>	Valid: sweep from state 10 to 20, then stay at state 20 for 30 frames, then sweep down to 1.

Table 1. Example usage of the `mset` command.

Please note the special syntax. There is a space before the “x” and “-” in the state-sweep argument, “1 -30” so, “1-30” will cause an error. “1 -30” is correct.

Lines 19–24 make the ligand the center of our attention for the entire movie. Specifically, line 19 starts the movie. Line 20 resets the view to the original position. Line 21 focuses in on the ligand and makes the center of the view the center of mass of the ligand. Line 22 shows the ligand as sticks. Line 23 zooms on the center just defined and then zooms out to a 12 Ångstroms sphere. Lastly, line 24 turns the scene for a slightly better view of the binding pocket.

Please run the **morph_movie.pml** script by dropping its icon onto the PyMOL icon. Once your movie is made, you can save it with **File→Save Movie As→MPEG** from the Upper Control Window.

Troubleshooting

When morphing fails, it will leave output for us to investigate called “rigimol.inp.” The most common cause of a morphing failure is that the two structures are incompatible. In order to successfully morph from a starting to a final conformation, you must have very high, if not perfect, sequence identity between the two structures.

Alternate coordinates will cause RigiMOL and `smooth` to fail. Please remove all atoms with alternate coordinates, leaving only one consistent set of atoms per structure.

Code 2: Making a movie from the methyltransferase morph trajectory

```

1  # Use a morph and holo structure to make a more biologically interesting movie
2  # Load the multi-state pdb file
3  load morph_out.pdb

4  # load the holo structure
5  load ../pdbs/2zvj.pdb, tmp

6  # we want to show the ligands bound to the protein in the final conformation, which is
7  # now state 30 of the morph_out object. So, we create a new object called ligand from
8  # the holo structure's ligand and put that into state 30 of the new object.
9  create ligand, tmp and organic, 1, 30

10 # do likewise for the waters nearby the binding pocket
11 create waters, tmp and (solvent within 5 of organic), 1, 30

12 # remove this object, we only needed the ligands and waters
13 delete tmp

14 # Use the mset command to make a movie. This starts in state 1 for 30 frames (1 x30),
15 # then transitions up to state 30 (1-30). Next, it stops at state 30 for 30 frames (30
16 # x30). Then, it goes back to state one, taking another 30 frames to do so (30 -1).
17 mset 1 x30 1 -30 30 x30 30 -1

18 # play the movie and setup the scene.
19 mplay
20 reset
21 orient ligand
22 show_as sticks, ligand
23 zoom center, 12
24 turn x, 90

25 # This is the command line version of File->Save Movie As->MPEG.
26 cmd.movie.produce("morph_movie.mpeg", quiet=0)

```

Morphing Large Structures

PyMOL provides another method for calculating morphs. This method simply calculates a linear combination of coordinates of the specified structures. If we opt to, we can refine the resulting morph using RigiMOL in “refinement” mode. This technique allows us to: morph large structures that RigiMOL alone could not handle; very quickly morph smaller structures; and easily make a morphs from multiple input conformations. The trade off for this added capability is that these morphs are visually less appealing; they’re based off simple linear combinations of coordinates not RigiMOL-like geometry-aware calculations.

Molecular morphing, following this technique in PyMOL follows the following steps:

1. create or load the first molecular structure
2. create or load the last molecular structure

3. (optional) align the first and last structures to each other
4. create a multi-state target object from N-copies of the first structure and N-copies of the second structure
5. call smooth to linearly combine the coordinates
6. (optional) refine the morph using RigiMOL in “refinement” mode.

Let's use this technique to create a morph. Please open the **linear_morph.pml** script from the **PyMOLTutorials/MolecularEditing/morphs** directory into a simple text editor, like Microsoft Notepad. The script is shown in Code 3. We will use the same two proteins as the last morph so we can compare the results and differences in speed.

Code 3: Linear Morphing and Refinement

```
1 # load start and final conformations
2 load ../pdbs/2z1b.pdb, startConf
3 load ../pdbs/2zvj.pdb, finalConf

4 # remove solvent atoms, they don't map 1-1 across structures
5 remove solvent

6 # align the two structures
7 align startConf, finalConf, object=aln

8 # make 100-state morph_in from state 1 of startConf and state 1 of finalConf
9 for x in range(51): cmd.create("m_out", "startConf", 1, x)
10 for x in range(51,101): cmd.create("m_out", "finalConf", 1, x)

11 # perform the linear smoothing across all states
12 smooth m_out, 1, 100, 1, 100

13 # import the rigimol module
14 from epymol import rigimol

15 # compute the morph and store the resulting
16 # trajectory in a new object called "morph_out"
17 rigimol.refine(2, "m_out")
```

Next Steps

Other Courses

Congratulations on finishing the Intermediate PyMOL tutorial. We hope you enjoyed the lesson and are motivated to use PyMOL for your molecular visualization, movie-making, scripting, and editing tasks. This isn't the end, though: Schrödinger has other courses available covering Moviemaking, and Intermediate PyMOL. Please contact us for more information on those courses via help@schrodinger.com.

Joining the PyMOL–Users Mailing List

As of Summer 2011, well over 1,500 PyMOL users subscribe to the pymol–users mailing list, which is where the community exchanges tips on how to use the software effectively and how to solve any problems that come up. To join the mailing list, please click on the **Mailing List** link that appears at the top of the PyMOL Home Page, (<http://www.pymol.org>). Or, to quickly search the mailing list archives for posts on a specific topic, click the **Mailing List Archive** link.

Visiting the PyMOLWiki Community Web Site

The community also runs a "Wiki" dynamic content site that aggregates information from the mailing list and provides other kinds of PyMOL–related documentation (<http://www.pymolwiki.org>). PyMOLWiki users can create their own pymol–related pages, on this site or elaborate upon useful information that is already posted. The site boasts over 1,300 users, over 7,000,000 page views, is typically accessed between 4,000–6,000 times a day, and has many useful scripts, plugins and tutorials free for download.

Accessing the Official Documentation Site

PyMOL subscribers access Schrödinger's Official Documentation site (<http://pymol.org/dsc>) using the subscription credentials shown from their most recent PyMOL receipt. This site contains the latest chapters from an updated PyMOL Manual, an assortment of narrated ScreenCasts, and plenty of reference information to help users on their way becoming PyMOL experts.