

# PMV and Underlying Components

---

Michel F. Sanner

The Molecular Graphics Laboratory



The Scripps Research Institute  
La Jolla, California



SciPy'02

CalTech Pasadena CA. Sept. 2002



# SCHEDULE

---

- o Reusable Components
  - ✍ MolKit, DejaVu, ViewerFramework, ...
- o From building blocks to applications
  - ✍ PMV: a molecular visualization program
- o Extending PMV
  - ✍ ADT: setting up and running AutoDock
- o Conclusion





# Re-usable components

---

- o MolKit:
  - ✍ read/write/represent/manipulate molecules
- o Mslib:
  - ✍ molecular surface calculation
- o DejaVu:
  - ✍ General purpose 3D geometry viewer
- o ViewerFramework:
  - ✍ Visualization application template
- o Mslib, PyBabel, PyMead, SFF, Isocontour, ...

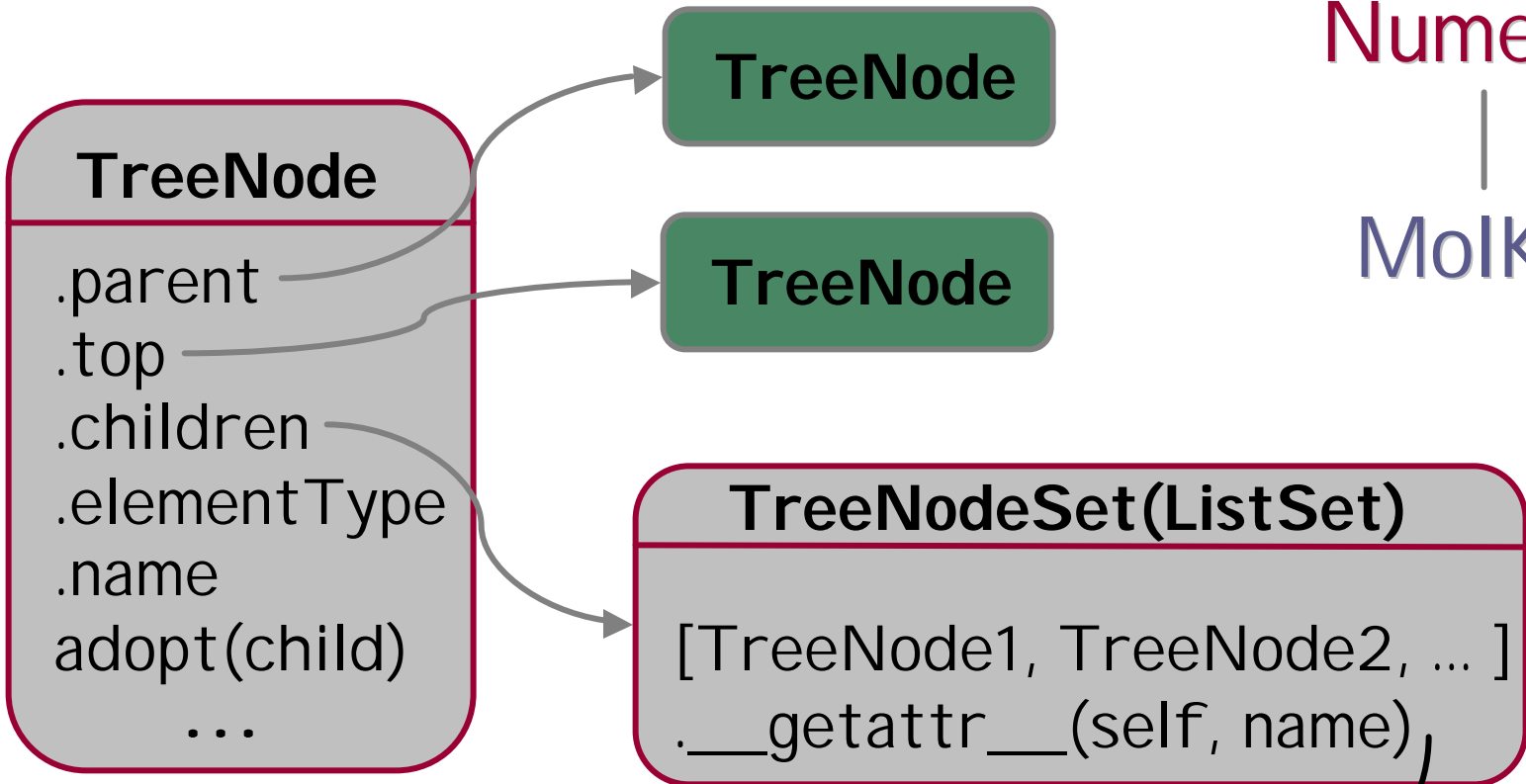
**(Sophie I. Coon, Michel F. Sanner and Art J. Olson, Re-usable components for structural bioinformatic, 9th Python Conference 2001)**





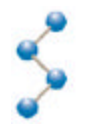
# MolKit

Numeric  
|  
MolKit

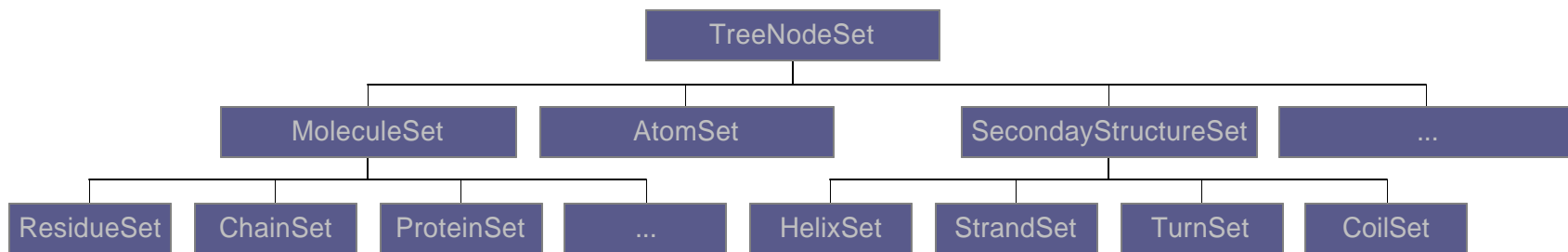
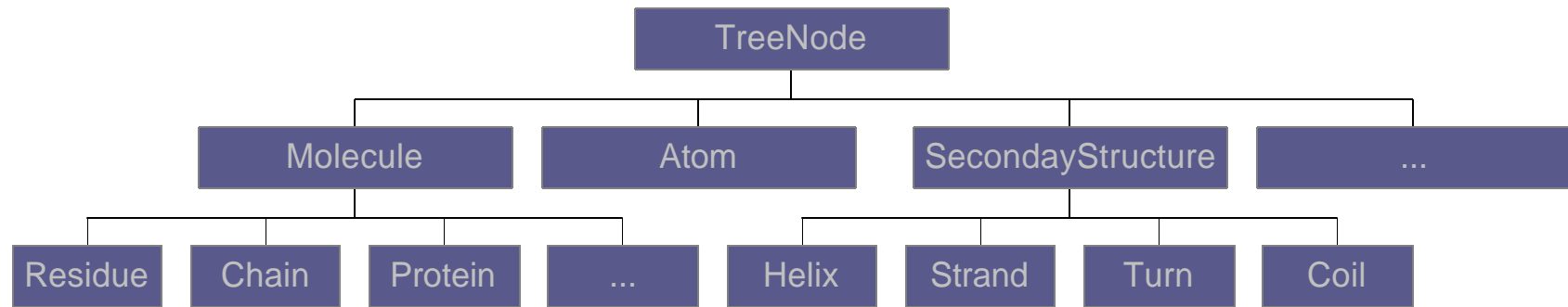


[ `TreeNode1.name`, `TreeNode2.name`, ... ]

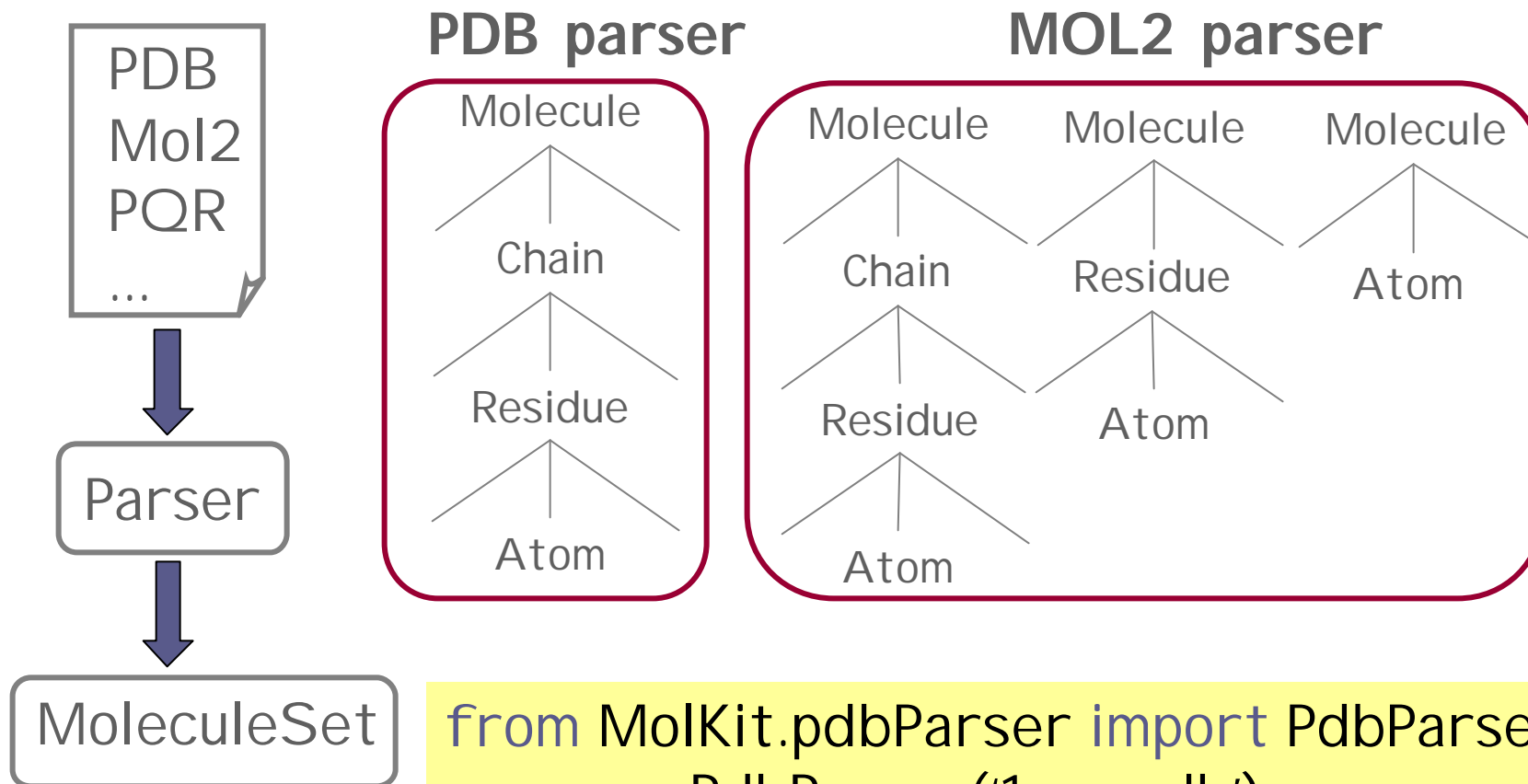
returns



# TreeNode and TreeNodeSet specialization



# MolKit



```
from MolKit.pdbParser import PdbParser
parser = PdbParser('1crn.pdb')
mols = parser.parse( )
```

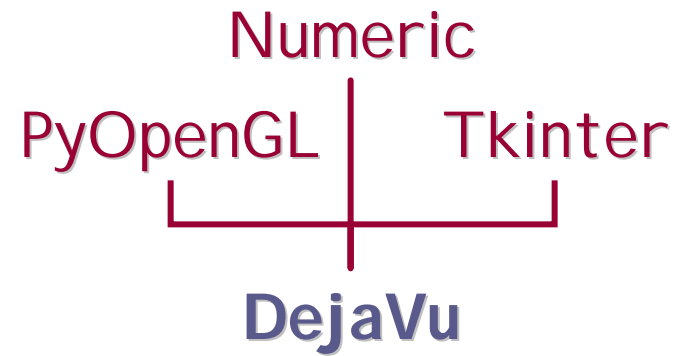
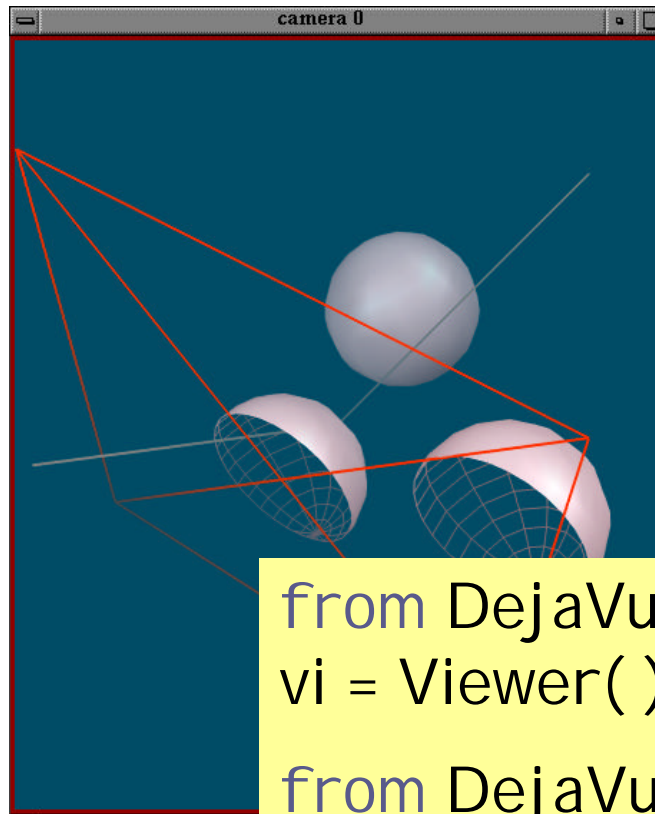
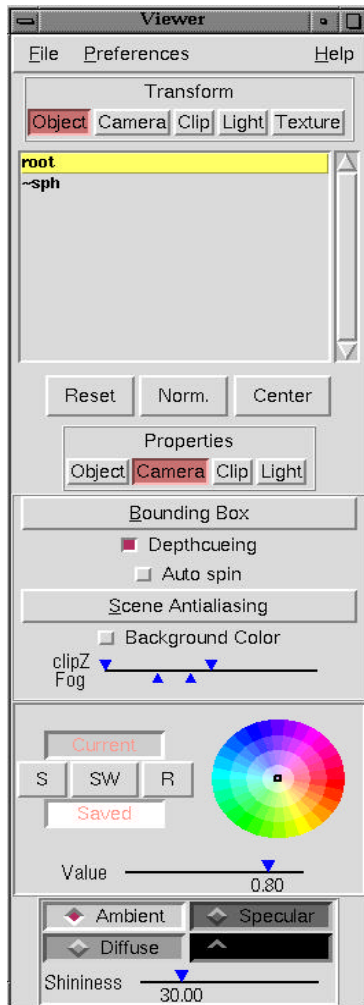


# Examples

```
>>> from MolKit import Read
>>> molecules = Read('./1crn.pdb') # Read returns a ProteinSet
>>> mol = molecules[0]
>>> print mol.chains.residues.name
>>> print mol.chains.residues.atoms[20:85].full_name()
>>> from MolKit.molecule import Atom
>>> allAtoms = mol.findType(Atom)
>>> set1 = allAtoms.get(lambda x: x.temperatureFactor >20)
>>> allResidues = allAtoms.parent.uniq()
>>> import Numeric
>>> for r in allResidues:
...     coords = r.atoms.coords
...     r.geomCenter = Numeric.sum(coords) / len(coords)
```



# DejaVu



```
from DejaVu import Viewer
vi = Viewer( )
```

```
from DejaVu.Spheres import Spheres
centers = [[0,0,0],[3,0,0],[0,3,0]]
s = Spheres('sph', centers = centers)
s.Set(quality=10)
vi.AddObject(s)
```

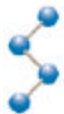




# Features

---

- o OpenGL Lighting and Material model
- o Arbitrary clipping planes
- o Multiple light sources
- o Material editor
- o DepthCueing (fog), global anti-aliasing
- o glScissors/magic lens
- o Object hierarchy with transformation and rendering properties inheritance
- o Multi-level picking
- o Extensible set of geometries





# Geometries

---

## Geom

- o PolyLine
- o Points
- o Spheres
- o Labels
- o Arc3D...

## IndexedGeoms

- o IndexedPolyLines
- o IndexedPolygons
- o TriangleStrip
- o QuadStrip
- o Cylinders ....





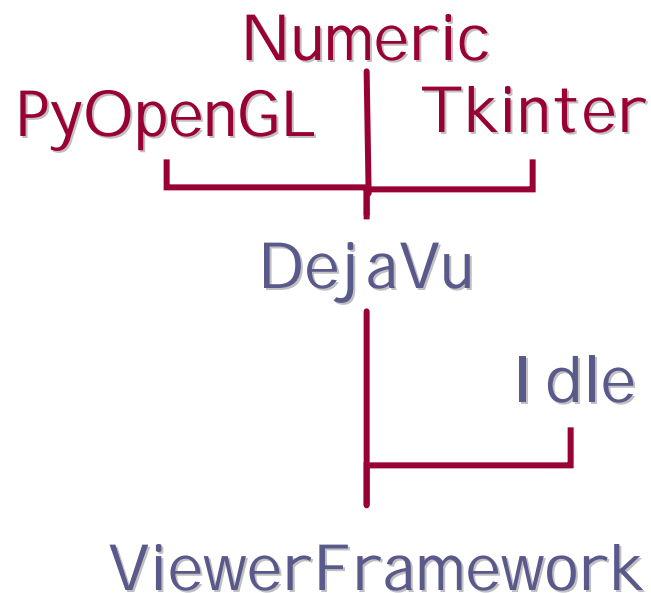
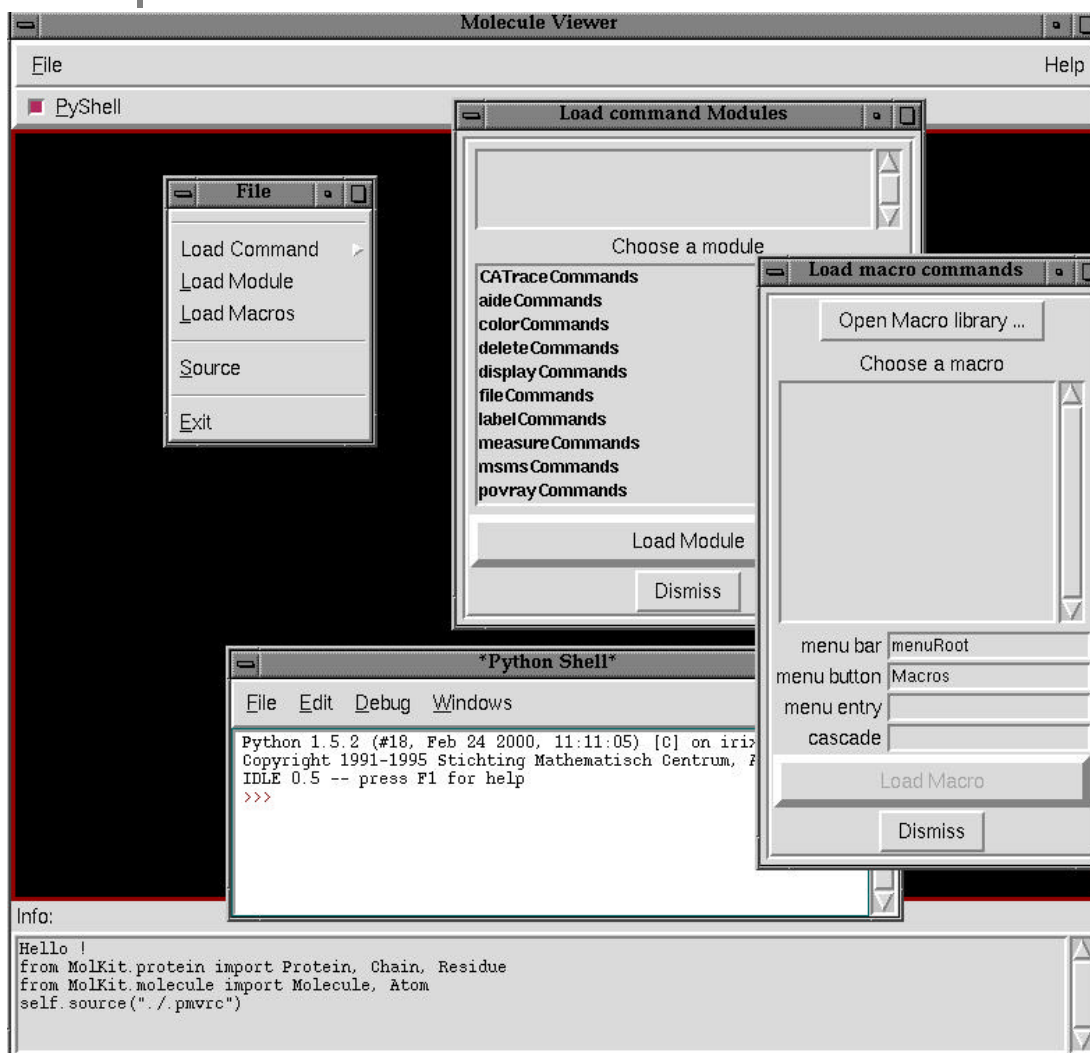
# Mslib

---

- o Python wrapper of MSMS
  - ✍ XYZR -> reduced Surface
    - > Analytical SES
    - > triangulated SES
- o Surface genus, areas
- o buried surface calculation



# ViewerFramework





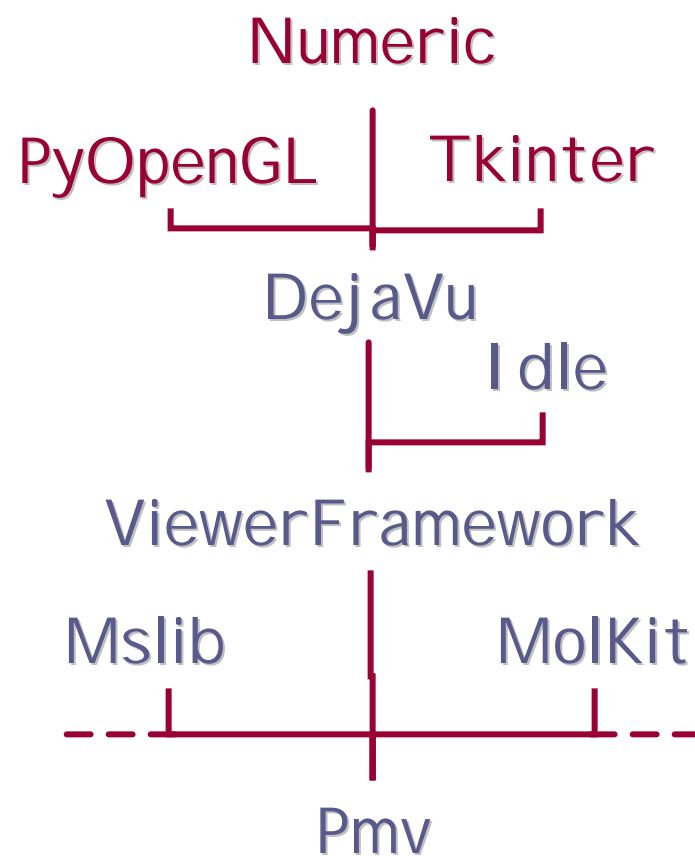
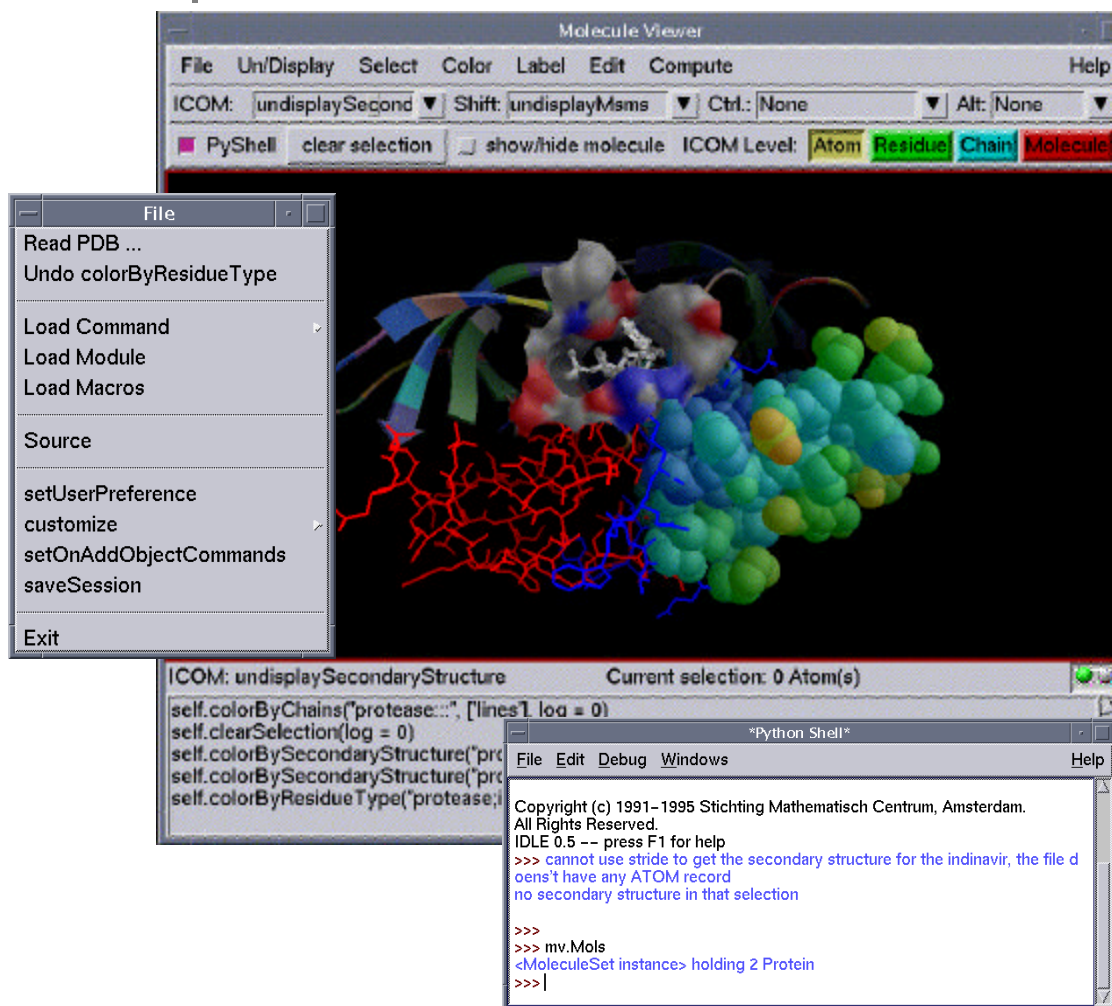
# Design features

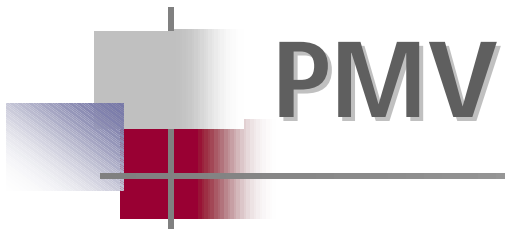
---

- o Dynamic loading of commands
- o Python shell for scripting
- o Dual interaction mode (GUI /Shell)
- o Support for command:
  - o development, logging, GUI , dependencies
- o Lightweight commands: Macros
- o Dynamic commands (introspection)
- o Extensible set of commands
- o Access to documentation

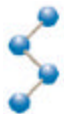


# PMV: From Building Blocks to applications





# DEMO





# PMV Capabilities

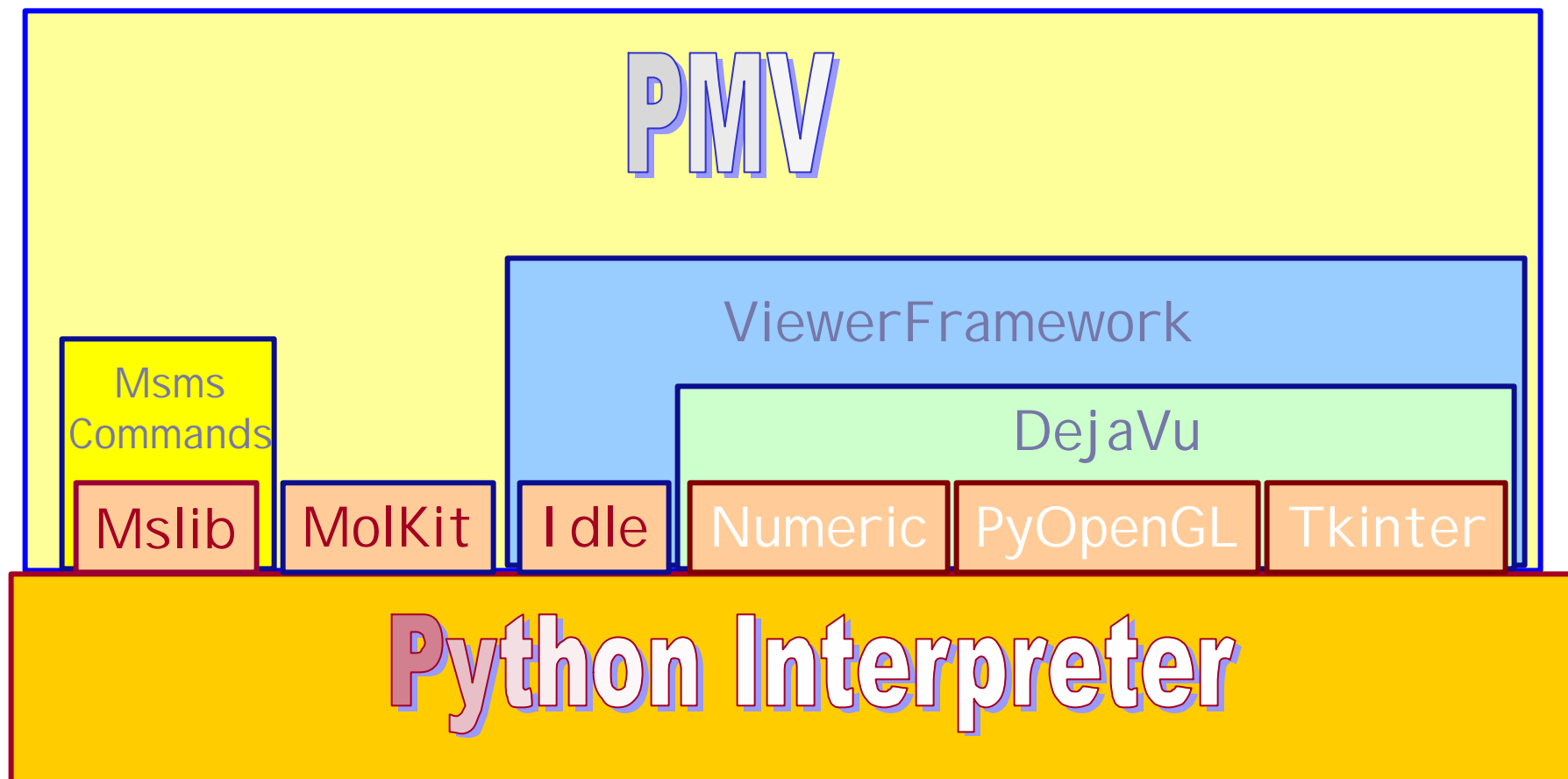
---

- o Display/compute:
  - ✍ Lines, CPK, Stick and Balls, Surfaces, Splines, Ribbons, H-bonds, bond-order, Gasteiger and Kollmann charges
- o Editing:
  - ✍ deleting atoms, adding hydrogens
- o Electrostatic potential
  - ✍ MEAD\*, APBS\*
- o Molecular Dynamics:

 ✍ Amber\*

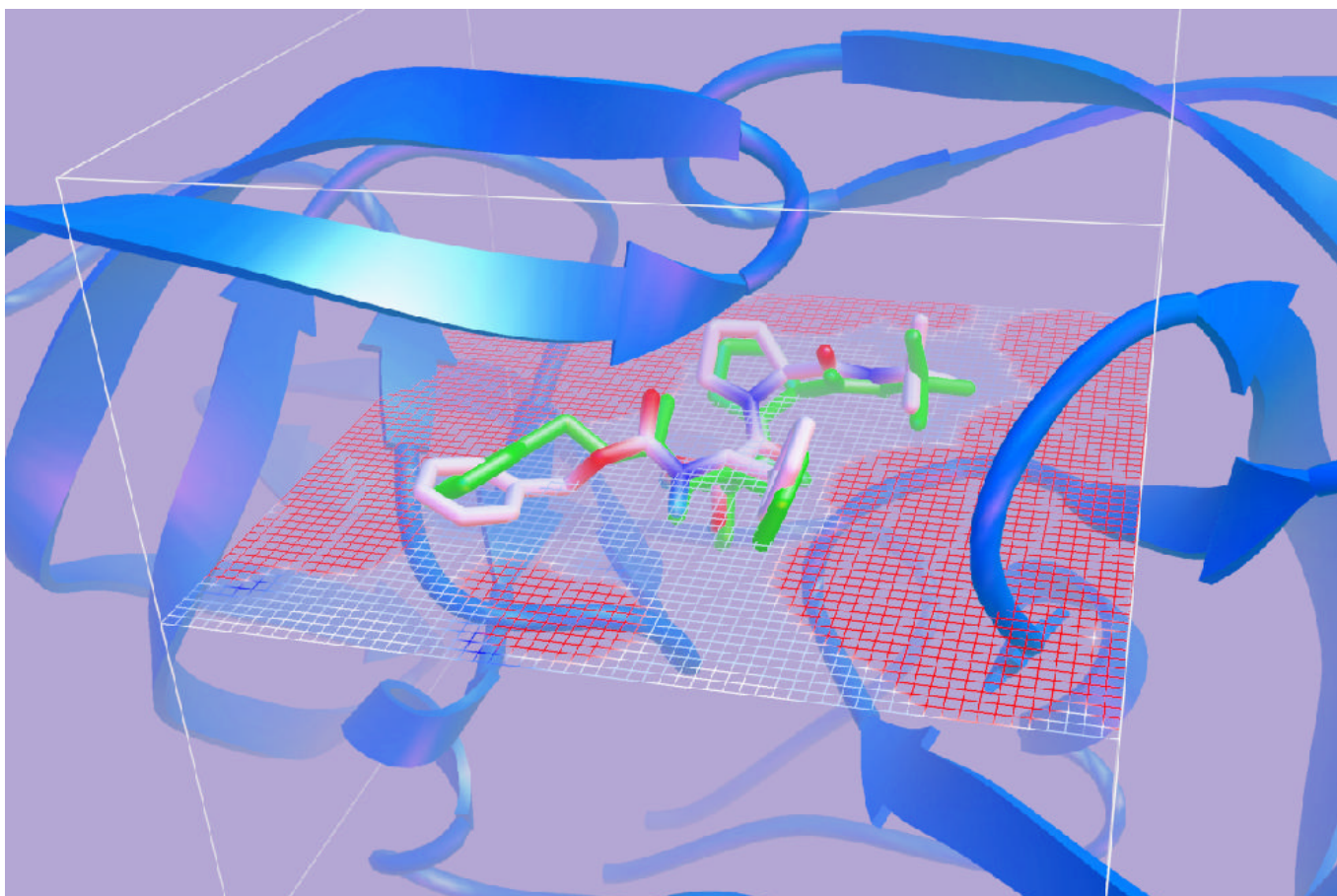


# PMV: Architecture



# AutoDock fundamentals

Automated docking of a flexible ligand to macromolecules using affinity grids





# AutoDock ToolKit (ADT)

---

- o AutoTors : ligand preparation
- o AutoGpf : grid definition
- o AutoDpf : docking parameters definition
- o AutoStart : job launching and monitoring
- o AutoAnalyze : docking results analysis

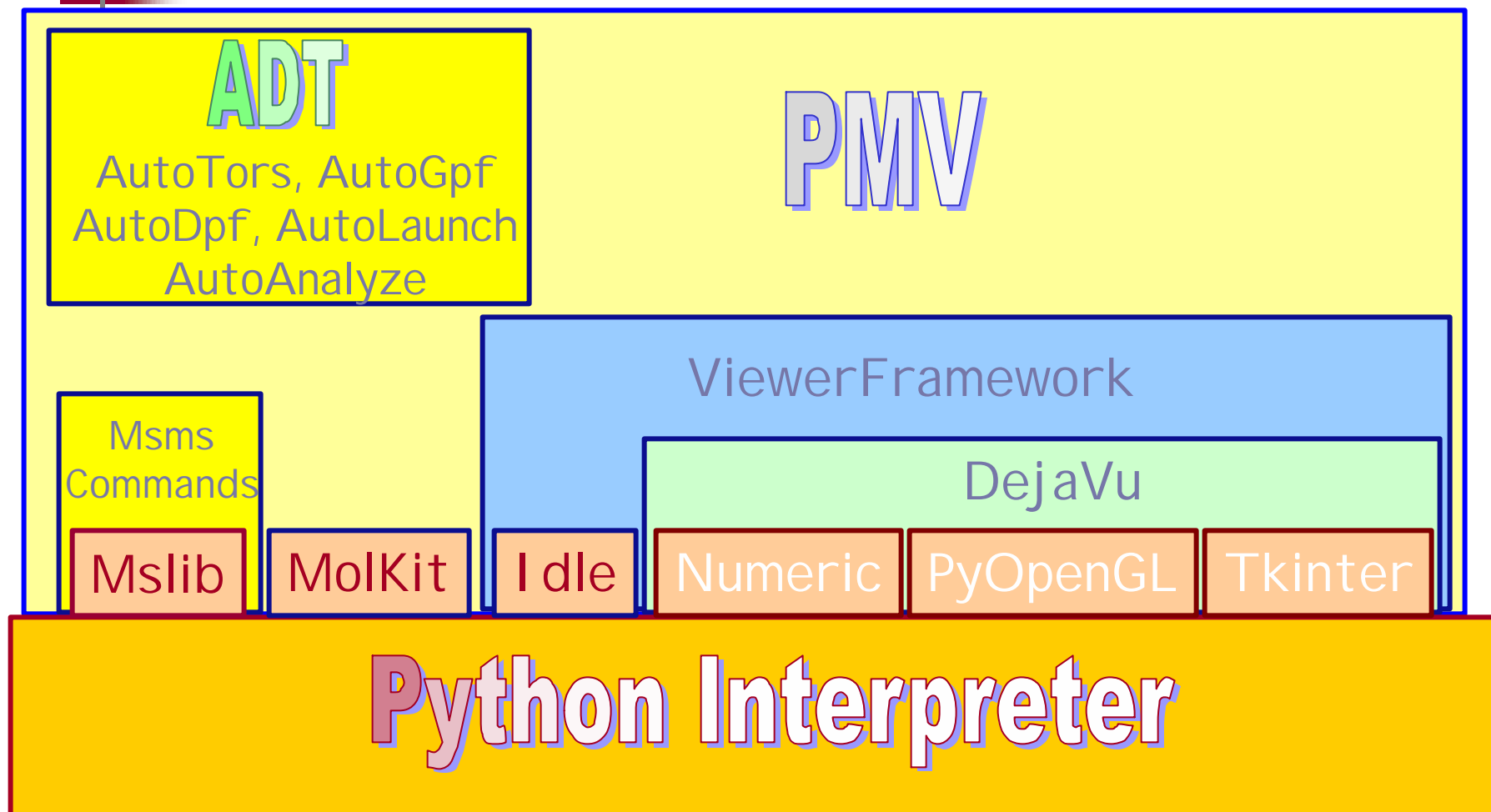


# ADT: extending PMV

The screenshot displays the PMV (PyMol Viewer) interface with the ADT (Automated Docking Tool) extension. The main window shows a 3D molecular model with a grid overlay. Several sub-windows are open, including 'Grid Options', 'AutoTors', 'AutoGpf', 'AutoDpf', and 'Start'. The 'Grid Options' window shows parameters for grid spacing and centering. The 'AutoTors' window lists options like 'Select Molecule' and 'ToggleTorsionActivity'. The 'AutoGpf' window shows options for 'Get values from a GPF' and 'Set Map Types'. The 'AutoDpf' window shows options for 'Get values from a DPF' and 'Set Docking Algorithm...'. The 'Start' window shows options for 'Start AutoGrid', 'Start AutoDock', and 'Process Manager'. The 'Analyze' window shows options for 'Read Docking Log' and 'Choose a Docked Conformation'.



# ADT: Architecture





# Conclusion

---

- **Flexible software built from components**
- **Take advantage of Python's advanced features**
- **Molecular manipulation environment**





# Acknowledgments

---

- o Molecular Graphics Lab. at TSRI
- o NIH (NBCR RR08605 and ITR EIA 0121282)
- o NSF (NPACI CA ACI 9619020)
- o Swiss National Science Foundation
- o Daniel Stoffler
- o Sophie Coon
- o Ruth Huey

Available at:

<http://www.scripps.edu/~sanner/python>

