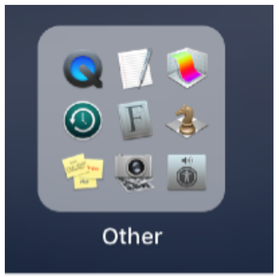
- We are not completely ready for the simulation because 1ubq.pdb
  - does not have hydrogen molecules
  - includes water, which does not help when we have an implicit solvent model
- OpenMM provides a package that does some simple modeling, including these required tasks.
- To make our script use this capability, open a text editor and add the five lines shown in the box

```
pdb = app.PDBFile('1ubq.pdb')
forcefield = app.ForceField('amber99sbildn.xml', 'amber99_obc.xml')

modeller = app.Modeller(pdb.topology, pdb.positions)
modeller.deleteWater()
modeller.addHydrogens(forcefield)
pdb = modeller
app.PDBFile.writeFile(pdb.topology, pdb.positions, open('ubq_mod.pdb', 'w'))

system = forcefield.createSystem(pdb.topology,
    nonbondedMethod=app.CutoffNonPeriodic, nonbondedCutoff=1.0*unit.nanometers,
    constraints=app.HBonds, rigidWater=True)
integrator = mm.LangevinIntegrator(300*unit.kelvin, 1.0/unit.picoseconds,
```

- To actually run OpenMM, you need to invoke a Python script.
- Python scripts are run from a command-line interface (cli). On a Mac, you can get to a cli by starting the Launchpad, clicking on the “Other” group, and starting the “Terminal” app.
- Once you start the terminal, enter the following commands
  - `cd ~/Desktop/OpenMM’, which changes your current directory to the directory where the files are
  - ‘ls’ lists the contents of the current directory. It isn’t 100% necessary but useful to check.
  - `python MD_ubq.py’ actually runs the script




```
(openmm) Minh-IIT-MBP2018:[~]: cd ~/Desktop/OpenMM
(openmm) Minh-IIT-MBP2018:[~/Desktop/OpenMM]: ls
1ubq.pdb  MD_ubq.py
(openmm) Minh-IIT-MBP2018:[~/Desktop/OpenMM]: python MD_ubq.py
```