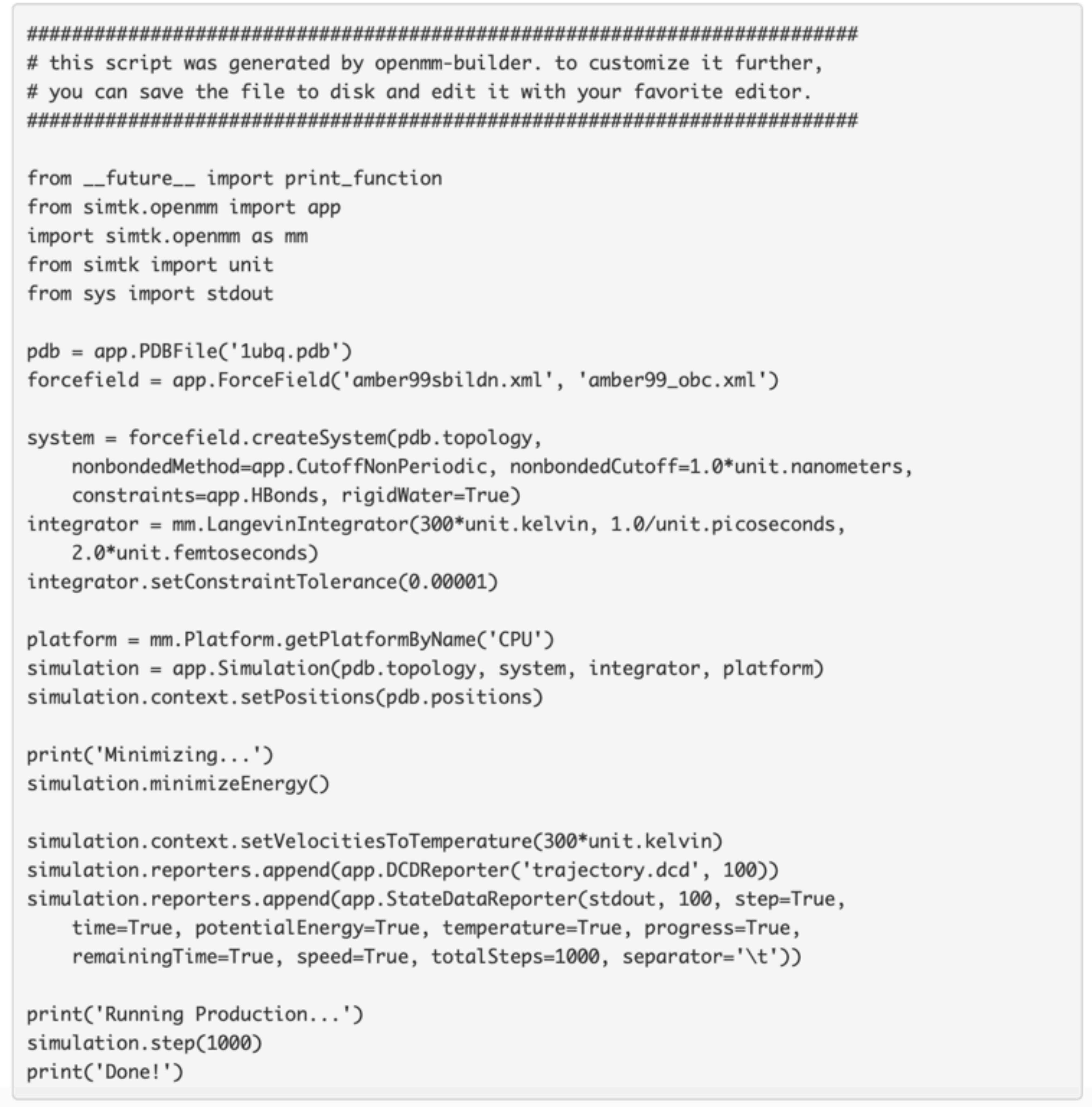
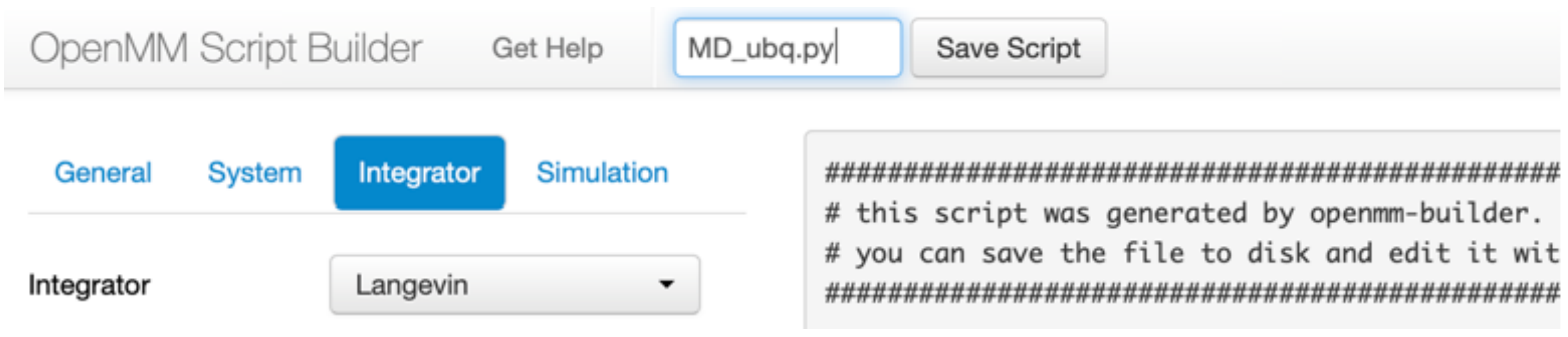
- Your script from the OpenMM script builder should look like the window on the right
- Create a directory called “OpenMM” on the Desktop.
- Save the script as “MD_ubq.py” and move it into the “OpenMM” directory.
- Copy “1ubq.pdb” from the VMD tutorial into the same “OpenMM” directory. The file can also be found on github.



```
##########################################################################
# this script was generated by openmm-builder. to customize it further,
# you can save the file to disk and edit it with your favorite editor.
##########################################################################

from __future__ import print_function
from simtk.openmm import app
import simtk.openmm as mm
from simtk import unit
from sys import stdout

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

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,
    2.0*unit.femtoseconds)
integrator.setConstraintTolerance(0.00001)

platform = mm.Platform.getPlatformByName('CPU')
simulation = app.Simulation(pdb.topology, system, integrator, platform)
simulation.context.setPositions(pdb.positions)

print('Minimizing...')
simulation.minimizeEnergy()

simulation.context.setVelocitiesToTemperature(300*unit.kelvin)
simulation.reporters.append(app.DCDReporter('trajectory.dcd', 100))
simulation.reporters.append(app.StateDataReporter(stdout, 100, step=True,
    time=True, potentialEnergy=True, temperature=True, progress=True,
    remainingTime=True, speed=True, totalSteps=1000, separator='\t'))

print('Running Production...')
simulation.step(1000)
print('Done!')
```

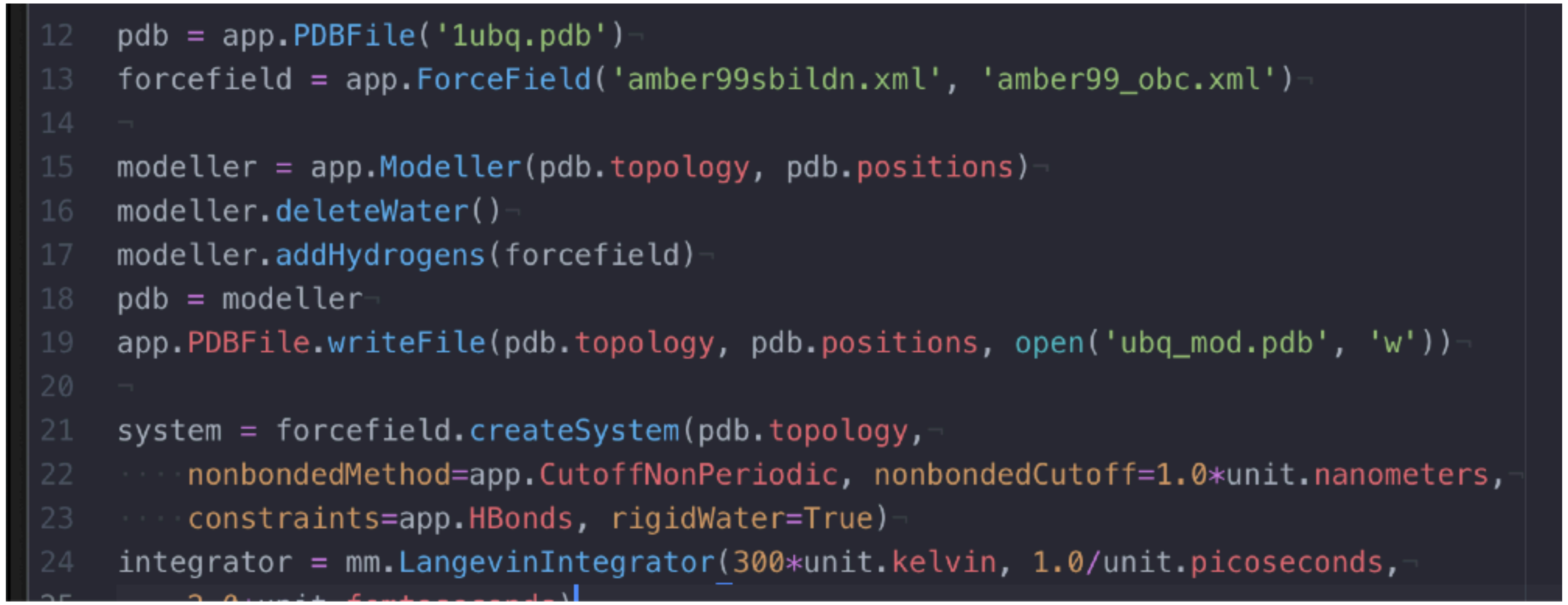


MD_ubq.py

Integrator: Langevin

```
###############################################
# this script was generated by openmm-builder.
# you can save the file to disk and edit it wit
###############################################
```

- 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,
    2.0*unit.femtoseconds)
```

The complete script can also be found on github.