```
###!/usr/local/bin/env python

# ============================================================================================
# GLOBAL IMPORTS
# ============================================================================================

# Non-scientific python packages needed for this protocol

import os
storage = os.path.join(os.getcwd(), 'test_storage.nc')
storage_checkpoint = os.path.join(os.getcwd(), 'test_storage_checkpoint.nc')
import sys
import timeit
from io import StringIO
import openmm
import openmmtools as mmtools
from openmmtools import testsystems
from simtk import unit
import mdtraj as md
# This is where replica exchange utilities are imported from Yank

from yank import mpi, analyze
from yank.multistate import MultiStateReporter, MultiStateSampler, ReplicaExchangeSampler,
ParallelTemperingSampler, SAMSSampler
from yank.multistate import ReplicaExchangeAnalyzer, SAMSAnalyzer
from yank.multistate.multistatereporter import _DictYamlLoader
from yank.utils import config_root_logger

# quiet down some citation spam
MultiStateSampler._global_citation_silence = True

# ============================================================================
# RUN REPLICA EXCHANGE
# ============================================================================

testsystem = testsystems.IdealGas()
        # Create thermodynamic state and save positions.

def run_replica_exchange(verbose=False, verbose_simulation=False):

    sampler_states = list()
    thermodynamic_states = list()

    # Define thermodynamic states.
    temperatures = [250.0, 300.0, 350.0, 400.0, 450.0] * unit.kelvin  # Temperatures.
    for temperature in temperatures:
        # Create thermodynamic state and save positions.
        system, positions = [testsystem.system, testsystem.positions]
        thermodynamic_state = mmtools.states.ThermodynamicState(system=system,
        temperature=temperature, pressure=1.0*unit.atmospheres)
        thermodynamic_states.append(thermodynamic_state)
        box_vectors = openmm.System().getDefaultPeriodicBoxVectors()
#        box_vectors = thermodynamic_state.system.default_box_vectors()
        sampler_states.append(mmtools.states.SamplerState(positions,box_vectors=box_vectors))
    # Create and configure simulation object.
    move = mmtools.mcmc.LangevinDynamicsMove(timestep=2.0*unit.femtoseconds,
                                             collision_rate=20.0/unit.picosecond,
                                             n_steps=100, reassign_velocities=True)
    simulation = ReplicaExchangeSampler(mcmc_moves=move, number_of_iterations=20)

    if os.path.exists(storage): os.remove(storage)
    reporter = MultiStateReporter(storage, checkpoint_interval=1)
    simulation.create(thermodynamic_states, sampler_states, reporter)
    config_root_logger(verbose_simulation)
```

```
    simulation.run()

    # Clean up.
    del simulation

    if verbose:
      print("PASSED.")

# ============================================================================
# MAIN
# ============================================================================

if __name__ == "__main__":
    # Configure logger.
    config_root_logger(False)

    start_time = timeit.default_timer()
    run_replica_exchange()
    stop_time = timeit.default_timer()
    print("Calculation time was: "+str(stop_time-start_time)+" seconds.")

    trajectory =
    analyze.extract_trajectory(testsystem,storage,nc_checkpoint_file=storage_checkpoint,replica_i
    ndex=1)
    trajectory.save_hdf5('test.h5')
    trajectory.save_pdb('test.pdb')
```