0.1 The 2D Ising model

```
import numpy as np
import os, tempfile, pickle
from multiprocessing import Pool
```

```
class Ising:
    def __init__(self, n):
        self.n = n
        self.spins = np.sign(np.random.rand(n, n) - 0.5)
        self.E = self.energy()
        self.Ev = self.E
    def neighbors(self, i, j):
        return np.hstack([self.spins[:,j].take([i-1,i+1], mode='wrap'),
                          self.spins[i,:].take([j-1,j+1], mode='wrap')])
    def energy(self):
        return -0.5 * sum(np.sum(s * self.neighbors(i, j))
                          for (i, j), s in np.ndenumerate(self.spins))
    def propose(self):
        i, j = np.random.randint(self.n), np.random.randint(self.n)
        self.i, self.j = i, j
        dE = 2 * np.sum(self.spins[i, j] * self.neighbors(i, j))
        self.dE = dE
        self.Ev = self.E + dE
    def accept(self):
        self.spins[self.i, self.j] *= -1
        self.E = self.Ev
```

Note that this class-based approach adds some overhead. For speed, instances of `Ising` should be inlined into the `wanglandau`.

0.1.1 Simulation

```
isingn = 32
sys = Ising(isingn)
```

The Ising energies over the full range, with correct end bin. We remove the penultimate energies since $E = 2$ or $E_{\max} - 2$ cannot happen.

```
isingE0 = -2 * isingn**2
isingEf = 2 * isingn**2
isingΔE = 4
Es = np.arange(isingE0, isingEf + isingΔE + 1, isingΔE)
Es = np.delete(np.delete(Es, -3), 1)
```

```
psystems = parallel_systems(sys, Es, n = 16, k = 0.5, N = 50_000_000)
```

```
def parallel_wanglandau(subsystem): # Convenient form for `Pool.map`
    wl.urandom_reseed()
    results = wanglandau(*subsystem, M = 1_000_000, logging=False)
    print('*', end='', flush=True)
    return results
```

```
with Pool() as pool:
    wlresults = pool.map(parallel_wanglandau, psystems)
```

```
sEs, sS = stitch_results(wlresults)
```

```
for Es, S, H in wlresults:
    plt.plot(Es[:-1], S)
```

```
plt.plot(sEs[:-1], sS);
```

```
with tempfile.NamedTemporaryFile(mode='wb', prefix='wlresults-ising-', suffix='.pickle',
↪  dir='data', delete=False) as f:
    print(os.path.basename(f.name))
    pickle.dump(wlresults, f)
    pickle.dump(sEs, f)
    pickle.dump(sS, f)
```