0.1 Systems for organized simulation

```
import sys
if 'src' not in sys.path: sys.path.append('src')
```

List of imports for all relevant systems.

```
from statistical_image import *
from ising_model import *
```

0.1.1 Specification

Extensions

- Encode this specification and the requirements of simulations as abstract base classes?
- Consider changing to `numba.typed.Dict` in the future if the API is guaranteed to be stable.

A system for simulation is a Numba `jitclass` that implements `state`, `state_names`, and `copy` functions. Given an instance `s` of a system class `System`, these functions should satisfy

```
id_systems = [
    s,
    s.copy(), # deep
    System(*s.state()),
    System(**{k: v for k, v in zip(s.state_names(), s.state())})
]
1 == len({t.state() for t in id_systems}) == len({t.state_names() for t in id_systems})
```

By default, we have

```
class System: # ...
    def copy(self):
        return self.__class__(*self.state())
```

In addition, different simulations may require more methods to be implemented.

0.1.2 Wang-Landau

A Wang-Landau simulation requires a `System` to have the variables `E`, `Ev`, and `sweep_steps`, and to implement the methods `energy_bins`, `energy`, `propose`, and `accept`.