Best Practices

Here are some general principles that have proven helpful for developing models.

Model Layout

A model should be contained in a folder named with lower-case letters and underscores, such as `thunder_cats`. Within that directory:

`README.md` describes the model, how to use it, and any other details. Github will automatically show this file to anyone visiting the directory.

`model.py` should contain the model class. If the file gets large, it may make sense to move the complex bits into other files, but this is the first place readers will look to figure out how the model works.

`server.py` should contain the visualization support, including the server class.

`run.py` is a Python script that will run the model when invoked via `mesa runserver`.

After the number of files grows beyond a half-dozen, try to use sub-folders to organize them. For example, if the visualization uses image files, put those in an `images` directory.

The Schelling model is a good example of a small well-packaged model.

It's easy to create a cookiecutter mesa model by running `mesa startproject`

Randomization

If your model involves some random choice, you can use the built-in `random` property that Mesa `Model` and `Agent` objects have. This works exactly like the built-in `random` library.

```
class AwesomeModel(Model):
  # ...

  def cool_method(self):
    interesting_number = self.random.random()
    print(interesting_number)

class AwesomeAgent(Agent):
  # ...
  def __init__(self, unique_id, model, ...):
    super().__init__(unique_id, model)
    # ...

  def my_method(self):
    random_number = self.random.randint(0, 100)
```

(The agent's random property is just a reference to its parent model's `random` property).

When a model object is created, its random property is automatically seeded with the current time. The seed determines the sequence of random numbers; if you instantiate a model with the same seed, you will get the same results. To allow you to set the seed, make sure your model has a `seed` argument in its constructor.

```
class AwesomeModel(Model):

  def __init__(self, seed=None):
    pass

  def cool_method(self):
    interesting_number = self.random.random()
    print(interesting_number)

>>> model0 = AwesomeModel(seed=0)
>>> model0._seed
0
>>> model0.cool_method()
0.8444218515250481
>>> model1 = AwesomeModel(seed=0)
>>> model1.cool_method()
0.8444218515250481
```