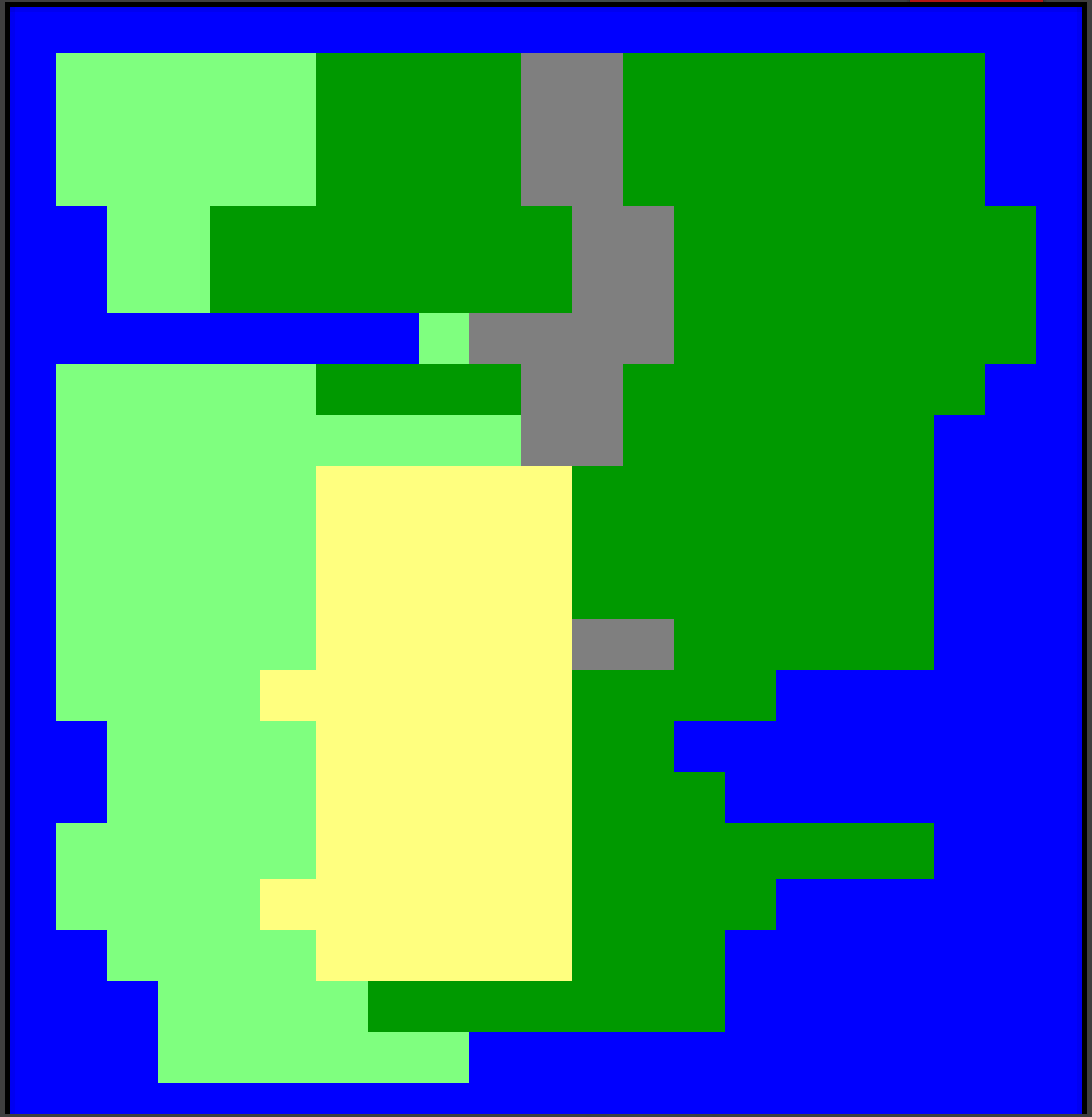
Modellering av økosystemet på Rossumøya

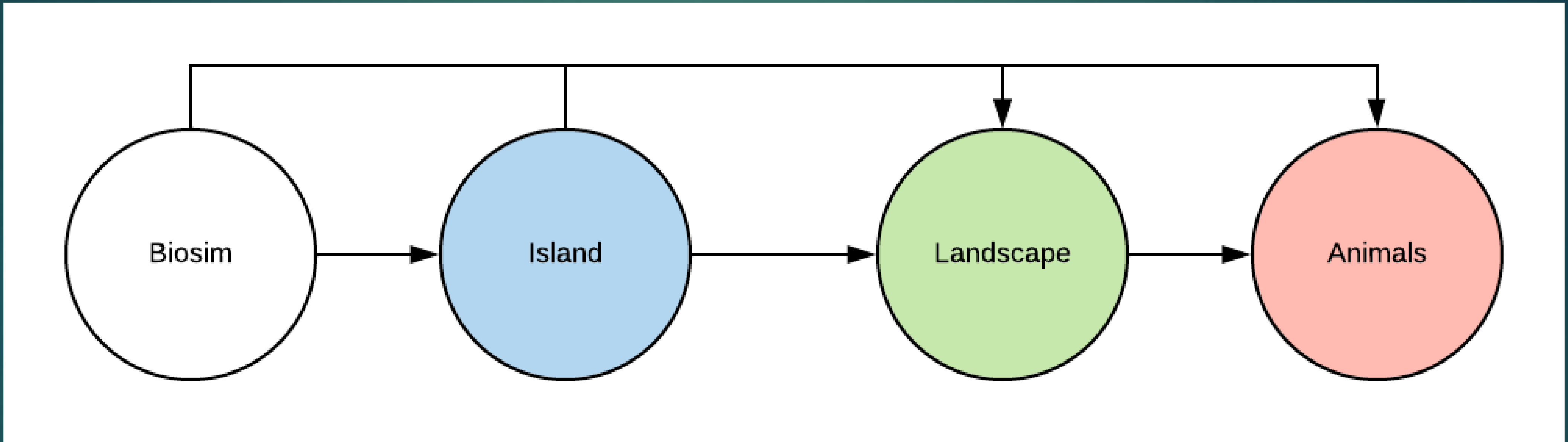
HELGE HELØ KLEMETSDAL & ADAM JULIUS OLOF KVIMAN

INF200 PROSJEKT

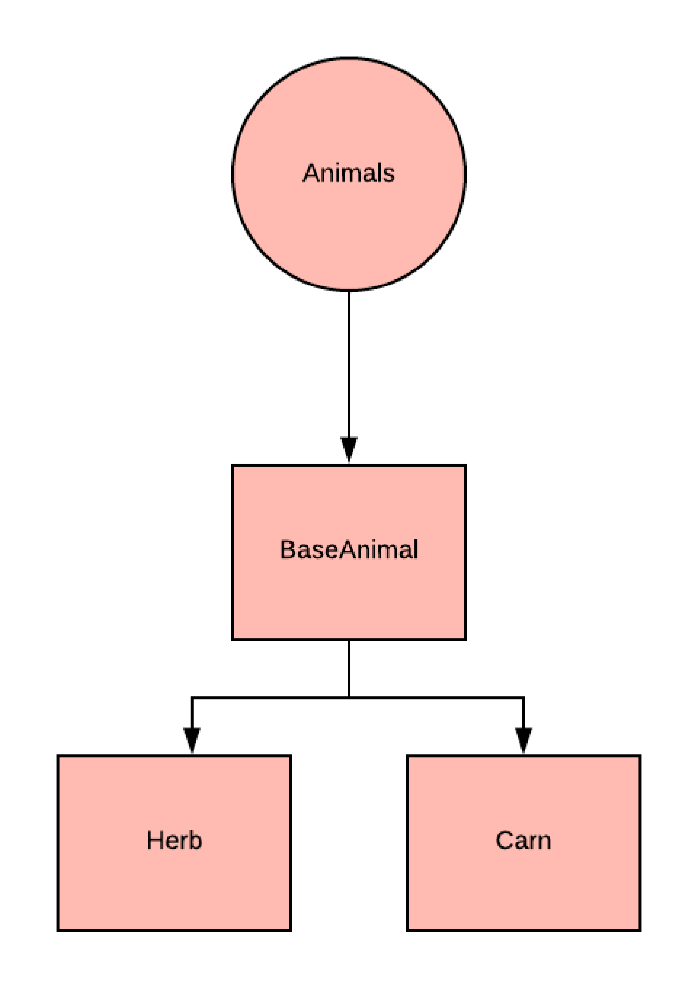
JANUAR 2020

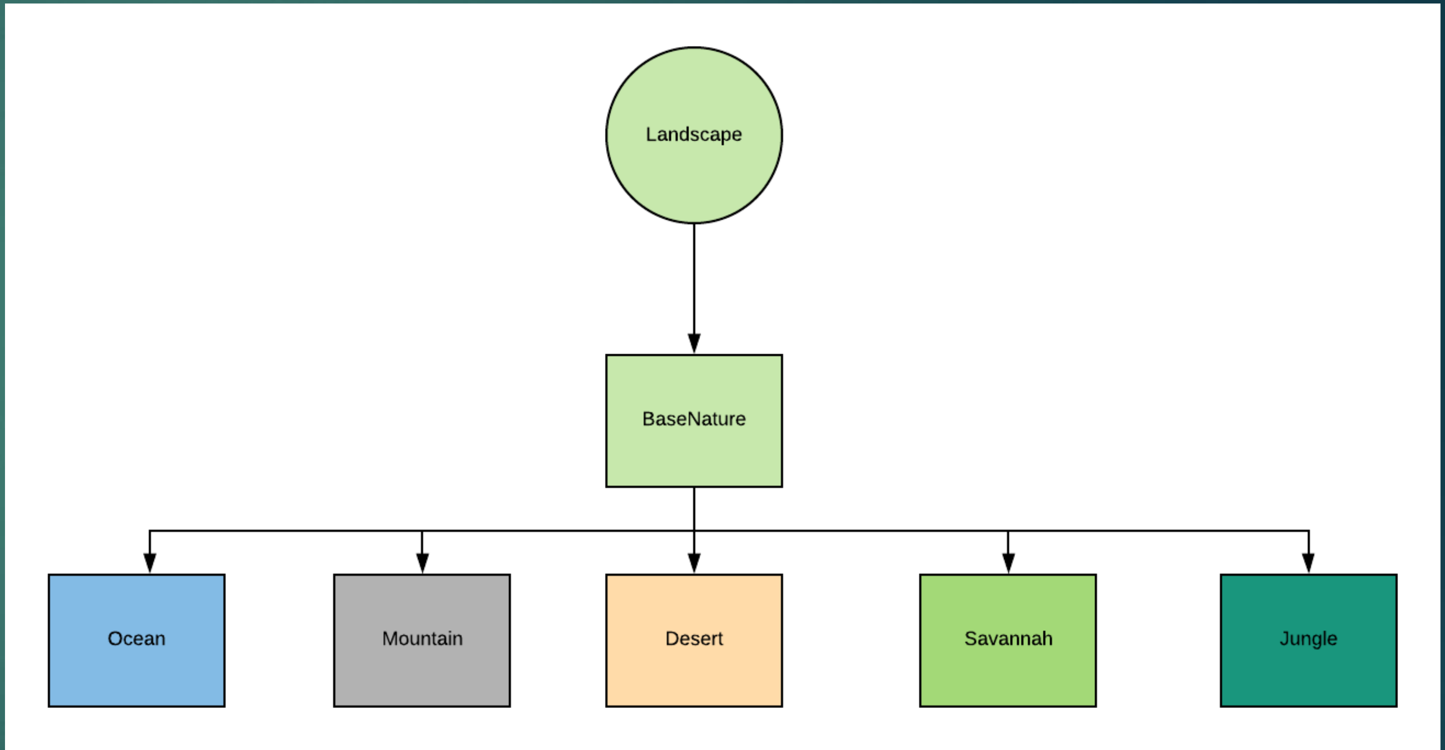


Struktur

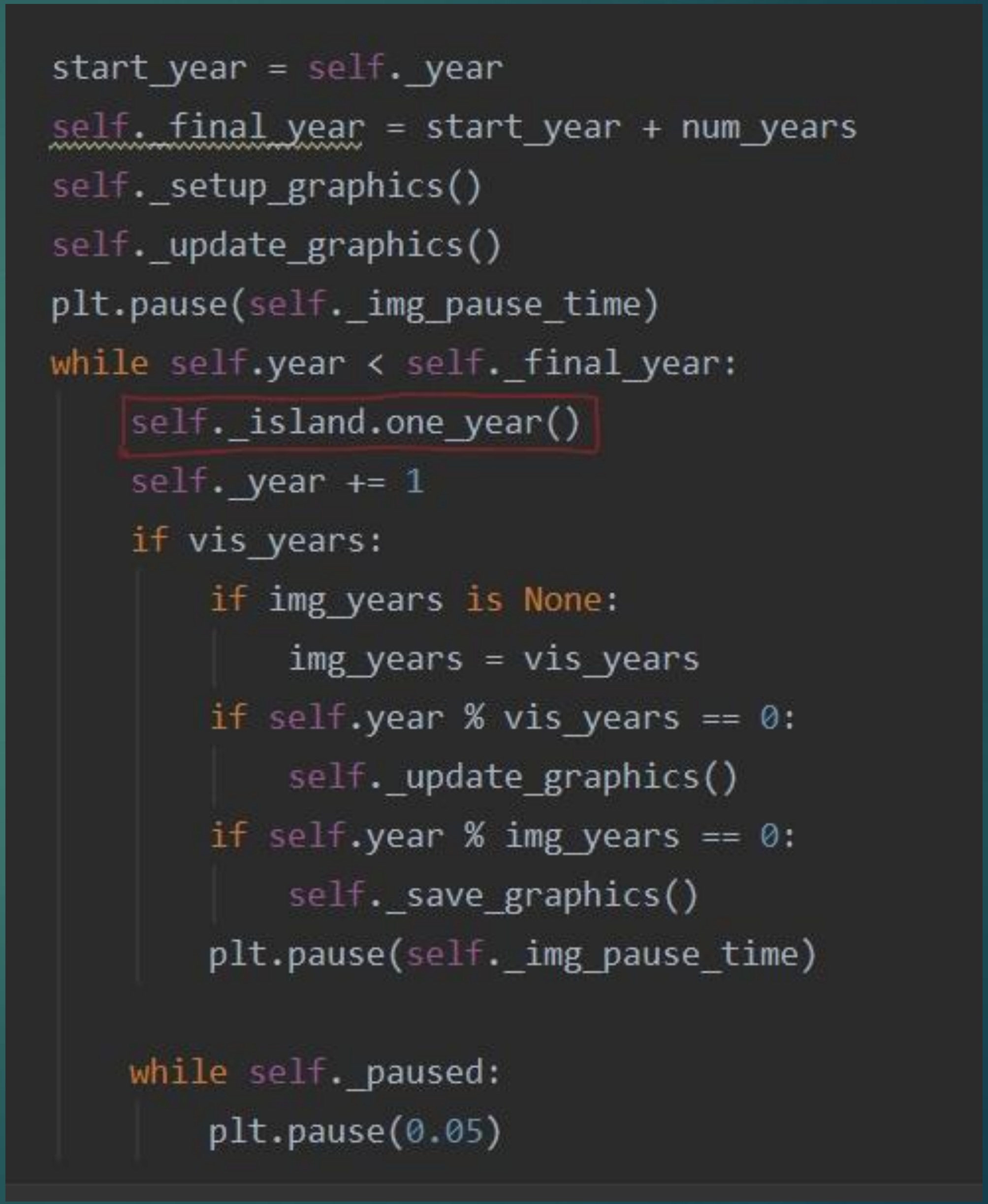


Klassestruktur



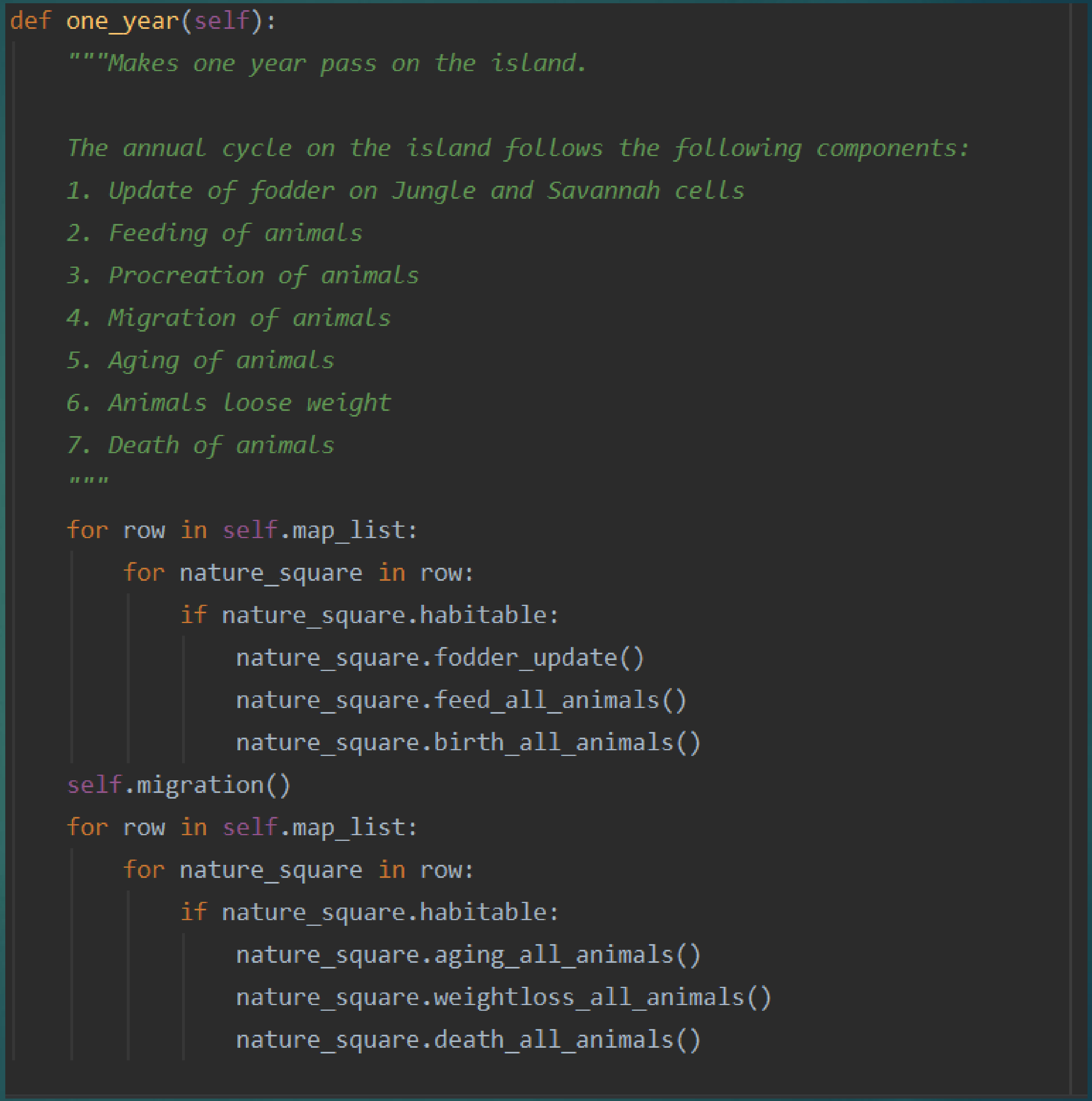


Biosim



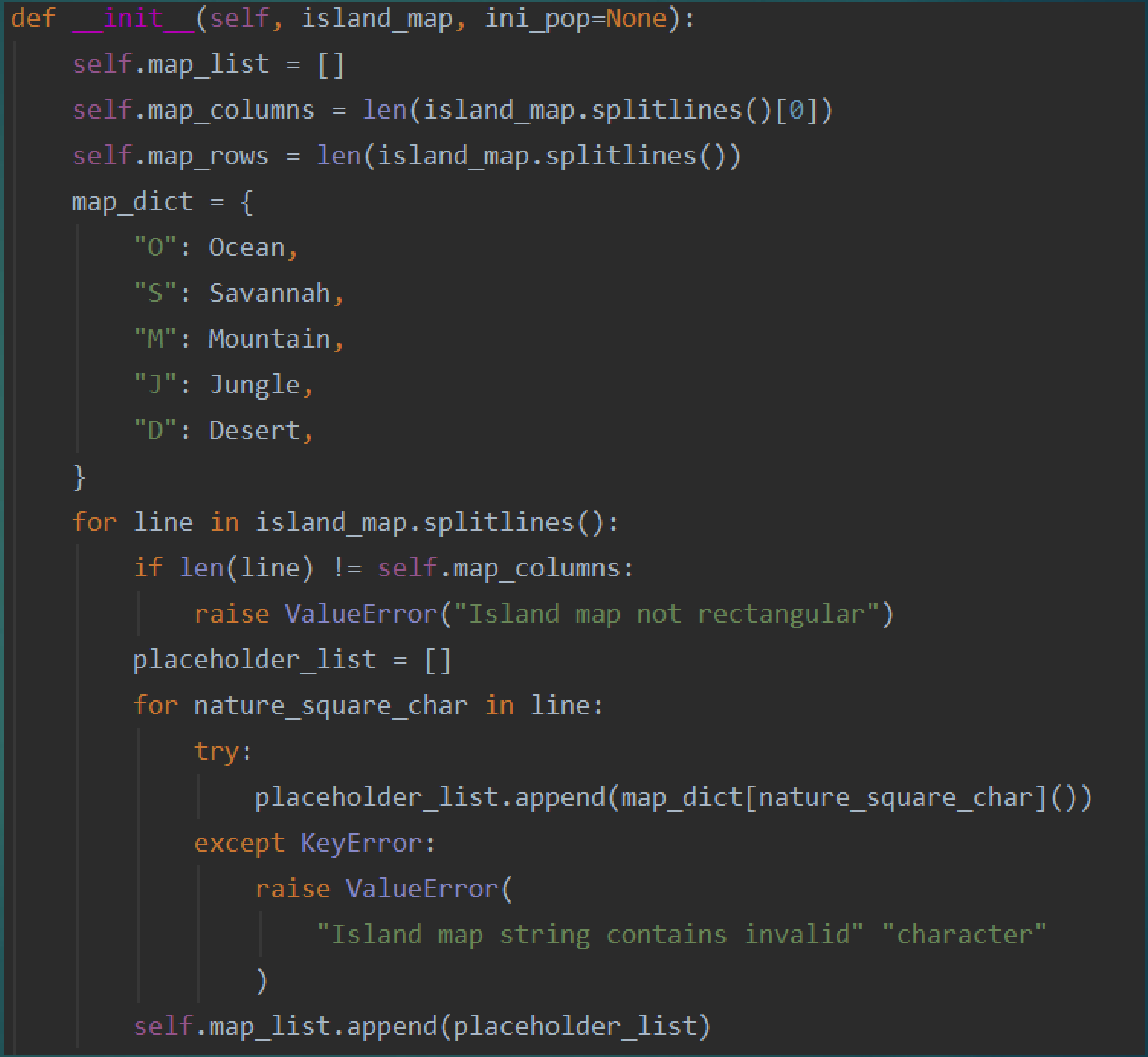
```python
start_year = self._year
self._final_year = start_year + num_years
self._setup_graphics()
self._update_graphics()
plt.pause(self._img_pause_time)
while self.year < self._final_year:
    self._island.one_year()
    self._year += 1
    if vis_years:
        if img_years is None:
            img_years = vis_years
        if self.year % vis_years == 0:
            self._update_graphics()
        if self.year % img_years == 0:
            self._save_graphics()
        plt.pause(self._img_pause_time)

    while self._paused:
        plt.pause(0.05)
```

Island:
Årlig syklus



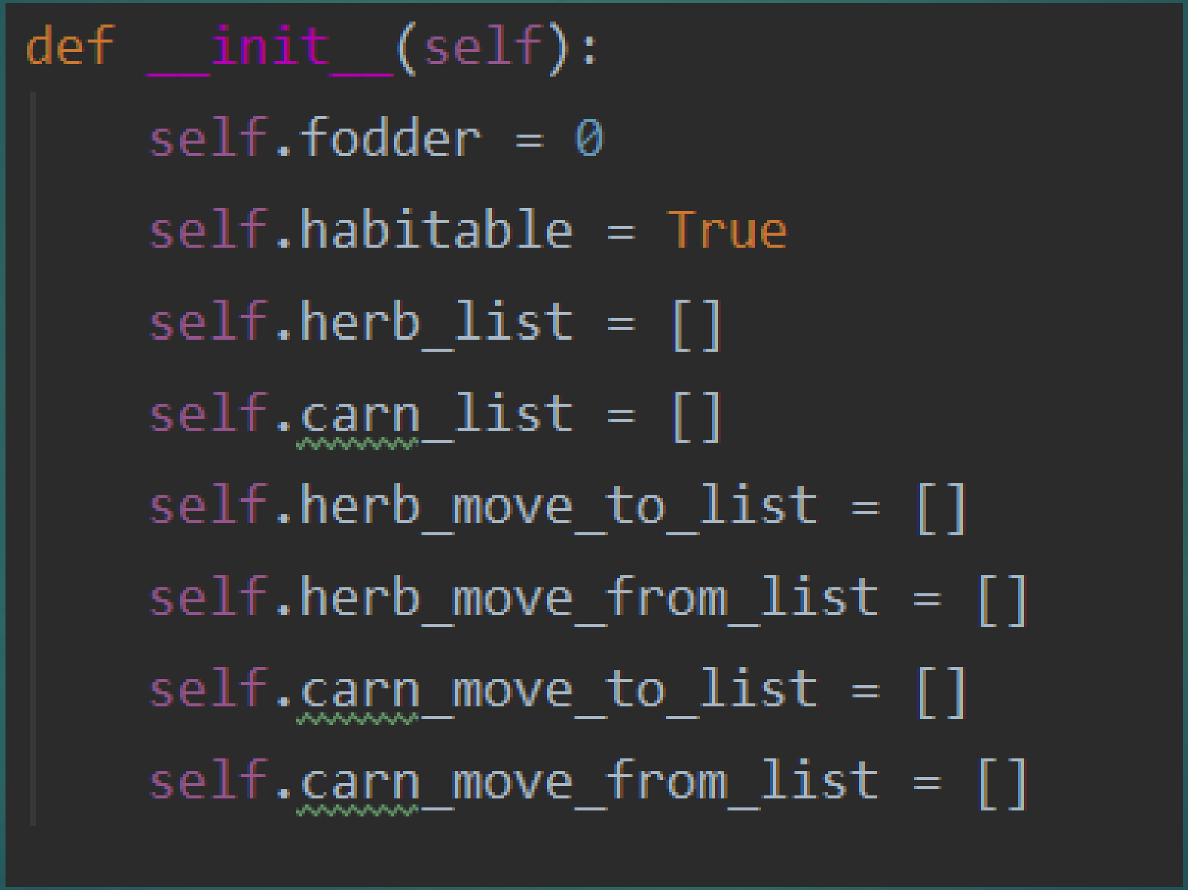
```python
def one_year(self):
    """Makes one year pass on the island.

    The annual cycle on the island follows the following components:
    1. Update of fodder on Jungle and Savannah cells
    2. Feeding of animals
    3. Procreation of animals
    4. Migration of animals
    5. Aging of animals
    6. Animals loose weight
    7. Death of animals
    """
    for row in self.map_list:
        for nature_square in row:
            if nature_square.habitable:
                nature_square.fodder_update()
                nature_square.feed_all_animals()
                nature_square.birth_all_animals()
    self.migration()
    for row in self.map_list:
        for nature_square in row:
            if nature_square.habitable:
                nature_square.aging_all_animals()
                nature_square.weightloss_all_animals()
                nature_square.death_all_animals()
```

Island: Map



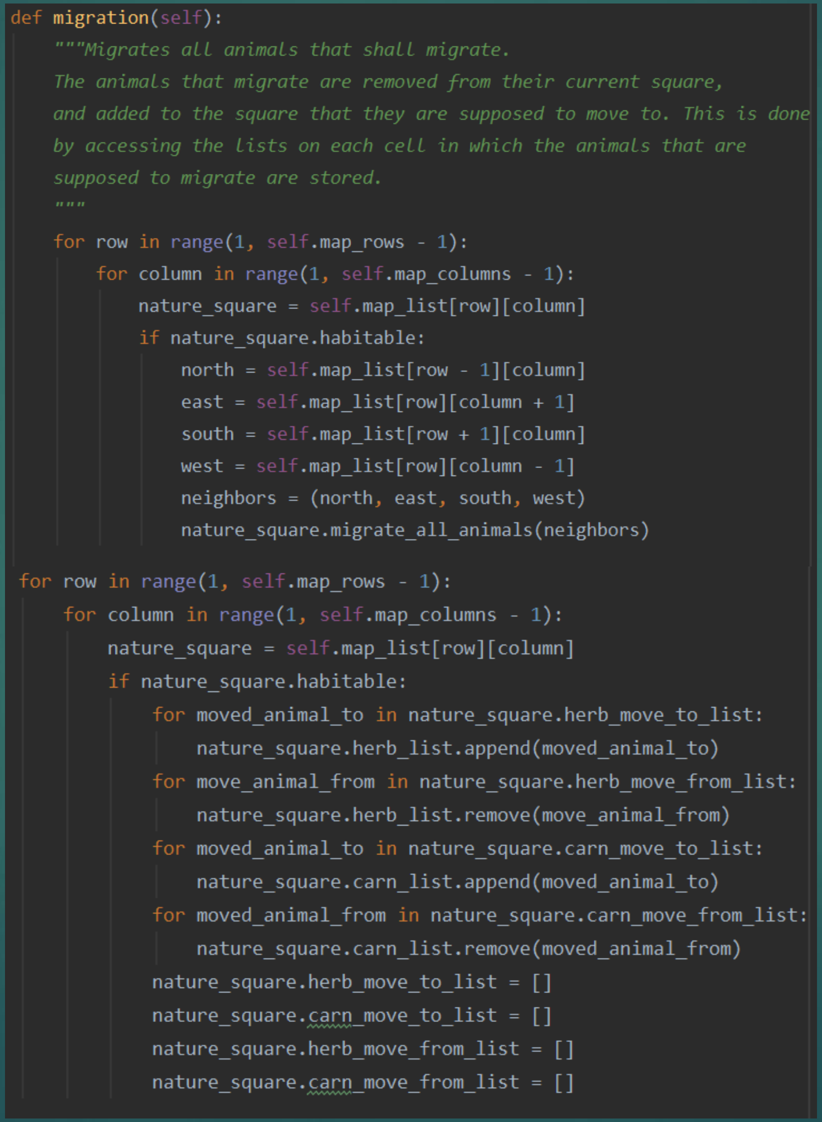
```python
def __init__(self, island_map, ini_pop=None):
    self.map_list = []
    self.map_columns = len(island_map.splitlines()[0])
    self.map_rows = len(island_map.splitlines())
    map_dict = {
        "O": Ocean,
        "S": Savannah,
        "M": Mountain,
        "J": Jungle,
        "D": Desert,
    }
    for line in island_map.splitlines():
        if len(line) != self.map_columns:
            raise ValueError("Island map not rectangular")
        placeholder_list = []
        for nature_square_char in line:
            try:
                placeholder_list.append(map_dict[nature_square_char]())
            except KeyError:
                raise ValueError(
                    "Island map string contains invalid" "character"
                )
        self.map_list.append(placeholder_list)
```

Landscape



```python
def __init__(self):
    self.fodder = 0
    self.habitable = True
    self.herb_list = []
    self.carn_list = []
    self.herb_move_to_list = []
    self.herb_move_from_list = []
    self.carn_move_to_list = []
    self.carn_move_from_list = []
```

Migration-Island

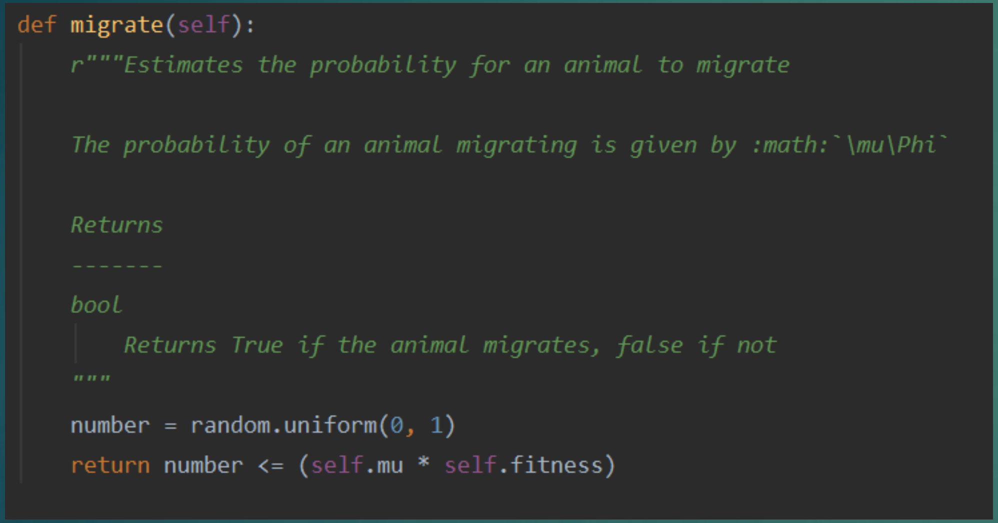


```python
def migration(self):
    """Migrates all animals that shall migrate.
    The animals that migrate are removed from their current square,
    and added to the square that they are supposed to move to. This is done
    by accessing the lists on each cell in which the animals that are
    supposed to migrate are stored.
    """
    for row in range(1, self.map_rows - 1):
        for column in range(1, self.map_columns - 1):
            nature_square = self.map_list[row][column]
            if nature_square.habitable:
                north = self.map_list[row - 1][column]
                east = self.map_list[row][column + 1]
                south = self.map_list[row + 1][column]
                west = self.map_list[row][column - 1]
                neighbors = (north, east, south, west)
                nature_square.migrate_all_animals(neighbors)

    for row in range(1, self.map_rows - 1):
        for column in range(1, self.map_columns - 1):
            nature_square = self.map_list[row][column]
            if nature_square.habitable:
                for moved_animal_to in nature_square.herb_move_to_list:
                    nature_square.herb_list.append(moved_animal_to)
                for move_animal_from in nature_square.herb_move_from_list:
                    nature_square.herb_list.remove(move_animal_from)
                for moved_animal_to in nature_square.carn_move_to_list:
                    nature_square.carn_list.append(moved_animal_to)
                for moved_animal_from in nature_square.carn_move_from_list:
                    nature_square.carn_list.remove(moved_animal_from)
                nature_square.herb_move_to_list = []
                nature_square.carn_move_to_list = []
                nature_square.herb_move_from_list = []
                nature_square.carn_move_from_list = []
```

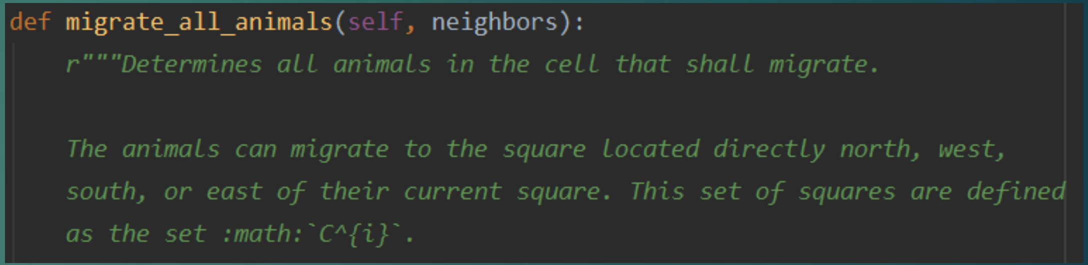
Migration- Landscape og animals

Animal

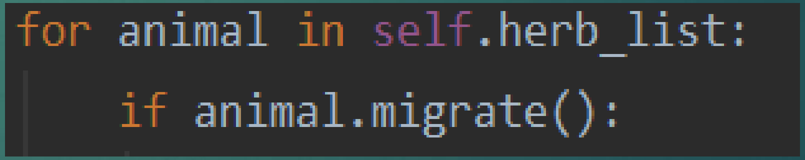
Landscape



```
def migrate(self):
    r"""Estimates the probability for an animal to migrate

    The probability of an animal migrating is given by :math:`\mu\Phi`

    Returns
    -------
    bool
        Returns True if the animal migrates, false if not
    """
    number = random.uniform(0, 1)
    return number <= (self.mu * self.fitness)
```

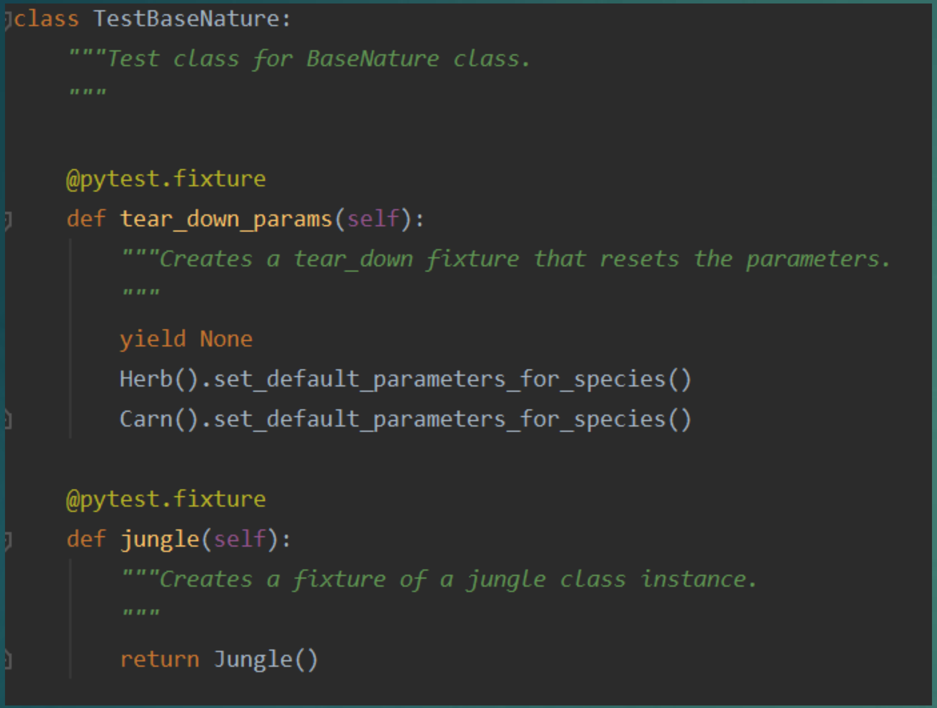


```
def migrate_all_animals(self, neighbors):
    r"""Determines all animals in the cell that shall migrate.

    The animals can migrate to the square located directly north, west,
    south, or east of their current square. This set of squares are defined
    as the set :math:`C^{i}`.
```

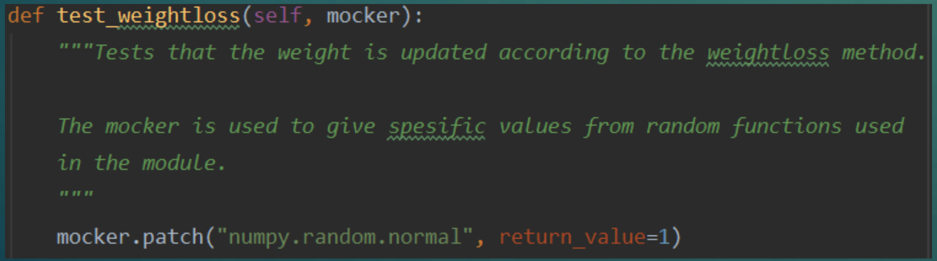


```
for animal in self.herb_list:
    if animal.migrate():
```

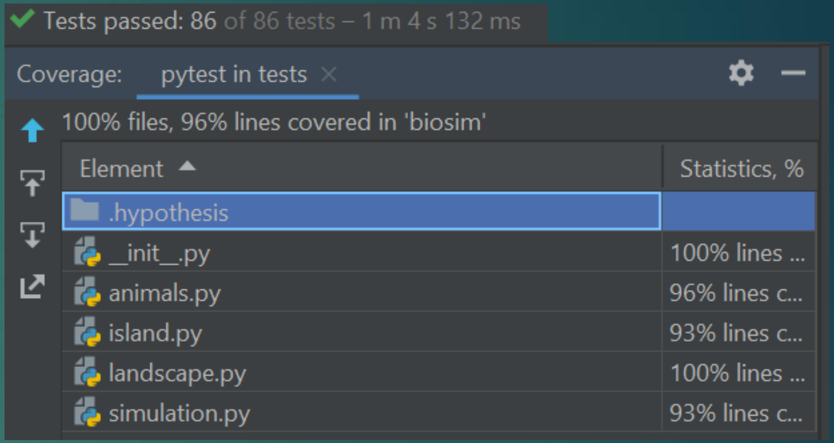
Testing



```
class TestBaseNature:
    """Test class for BaseNature class.
    """

    @pytest.fixture
    def tear_down_params(self):
        """Creates a tear_down fixture that resets the parameters.
        """
        yield None
        Herb().set_default_parameters_for_species()
        Carn().set_default_parameters_for_species()

    @pytest.fixture
    def jungle(self):
        """Creates a fixture of a jungle class instance.
        """
        return Jungle()
```



```
def test_weightloss(self, mocker):
    """Tests that the weight is updated according to the weightloss method.

    The mocker is used to give spesific values from random functions used
    in the module.
    """
    mocker.patch("numpy.random.normal", return_value=1)
```

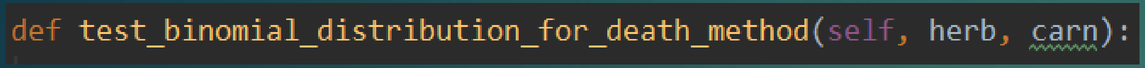


✓ Tests passed: 86 of 86 tests – 1 m 4 s 132 ms

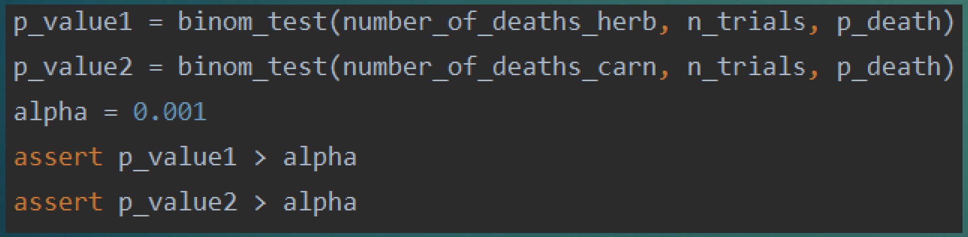
100% files, 96% lines covered in 'biosim'

Element	Statistics, %
.hypothesis	
__init__.py	100% lines ...
animals.py	96% lines c...
island.py	93% lines c...
landscape.py	100% lines ...
simulation.py	93% lines c...

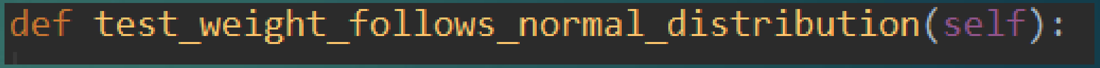
Statistiske tester



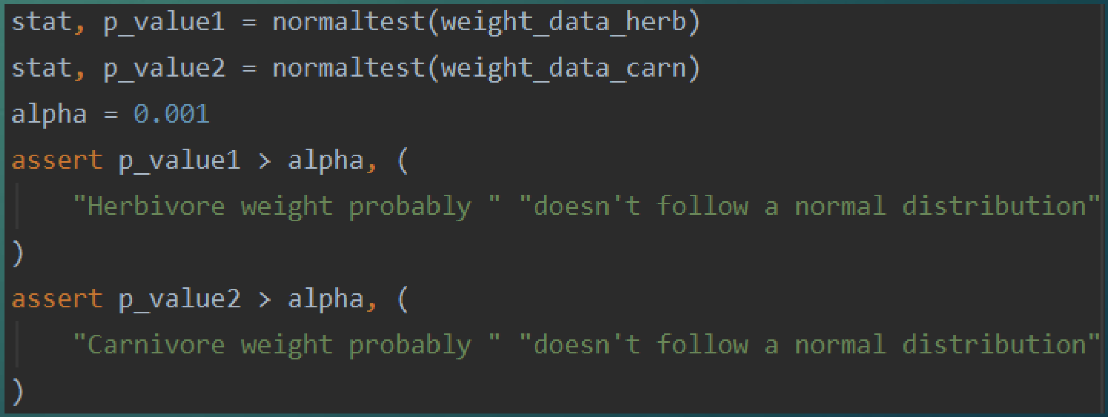
```python
def test_binomial_distribution_for_death_method(self, herb, carn):
```



```python
p_value1 = binom_test(number_of_deaths_herb, n_trials, p_death)
p_value2 = binom_test(number_of_deaths_carn, n_trials, p_death)
alpha = 0.001
assert p_value1 > alpha
assert p_value2 > alpha
```

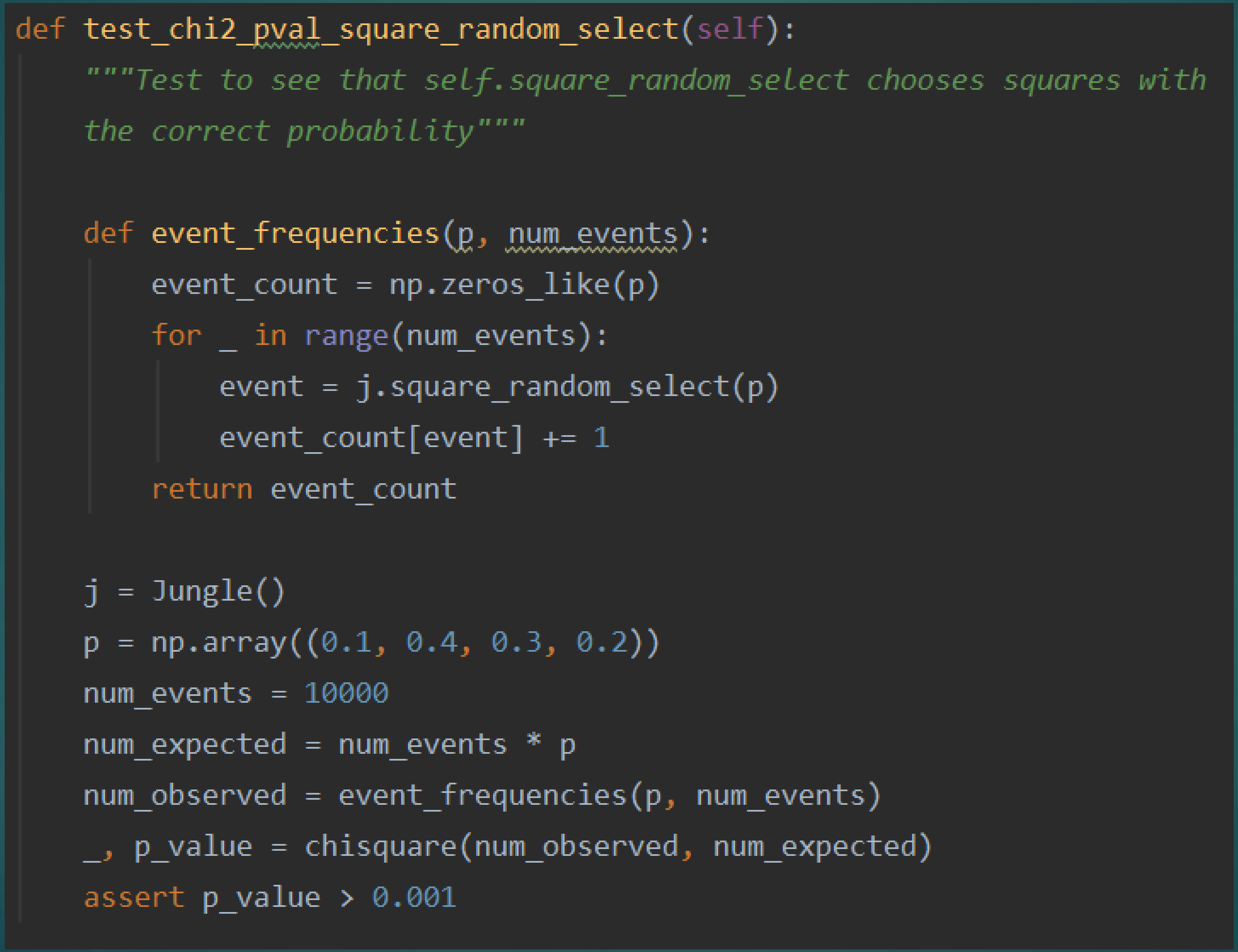


```python
def test_weight_follows_normal_distribution(self):
```



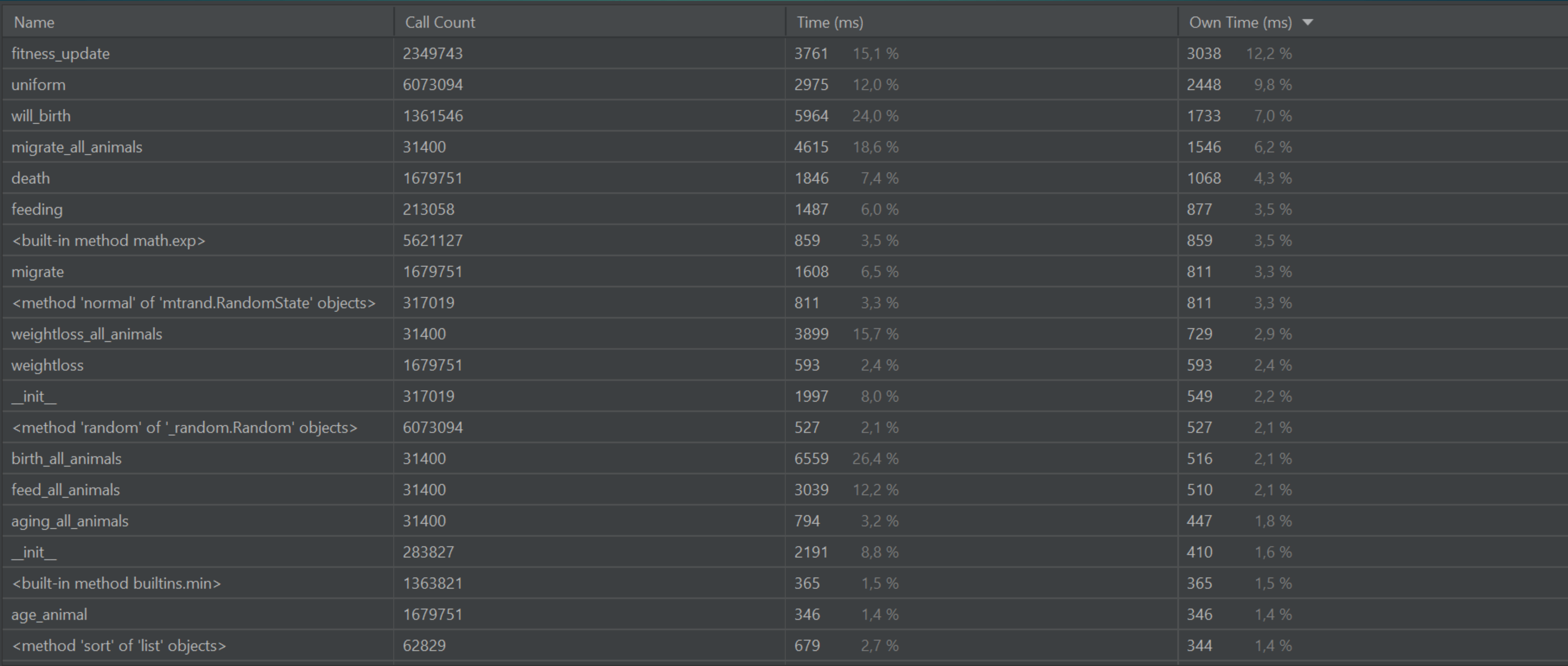
```python
stat, p_value1 = normaltest(weight_data_herb)
stat, p_value2 = normaltest(weight_data_carn)
alpha = 0.001
assert p_value1 > alpha, (
    "Herbivore weight probably " "doesn't follow a normal distribution"
)
assert p_value2 > alpha, (
    "Carnivore weight probably " "doesn't follow a normal distribution"
)
```

Statistiske tester



```
def test_chi2_pval_square_random_select(self):
    """Test to see that self.square_random_select chooses squares with
    the correct probability"""

    def event_frequencies(p, num_events):
        event_count = np.zeros_like(p)
        for _ in range(num_events):
            event = j.square_random_select(p)
            event_count[event] += 1
        return event_count

    j = Jungle()
    p = np.array((0.1, 0.4, 0.3, 0.2))
    num_events = 10000
    num_expected = num_events * p
    num_observed = event_frequencies(p, num_events)
    _, p_value = chisquare(num_observed, num_expected)
    assert p_value > 0.001
```

Profiling-check sim

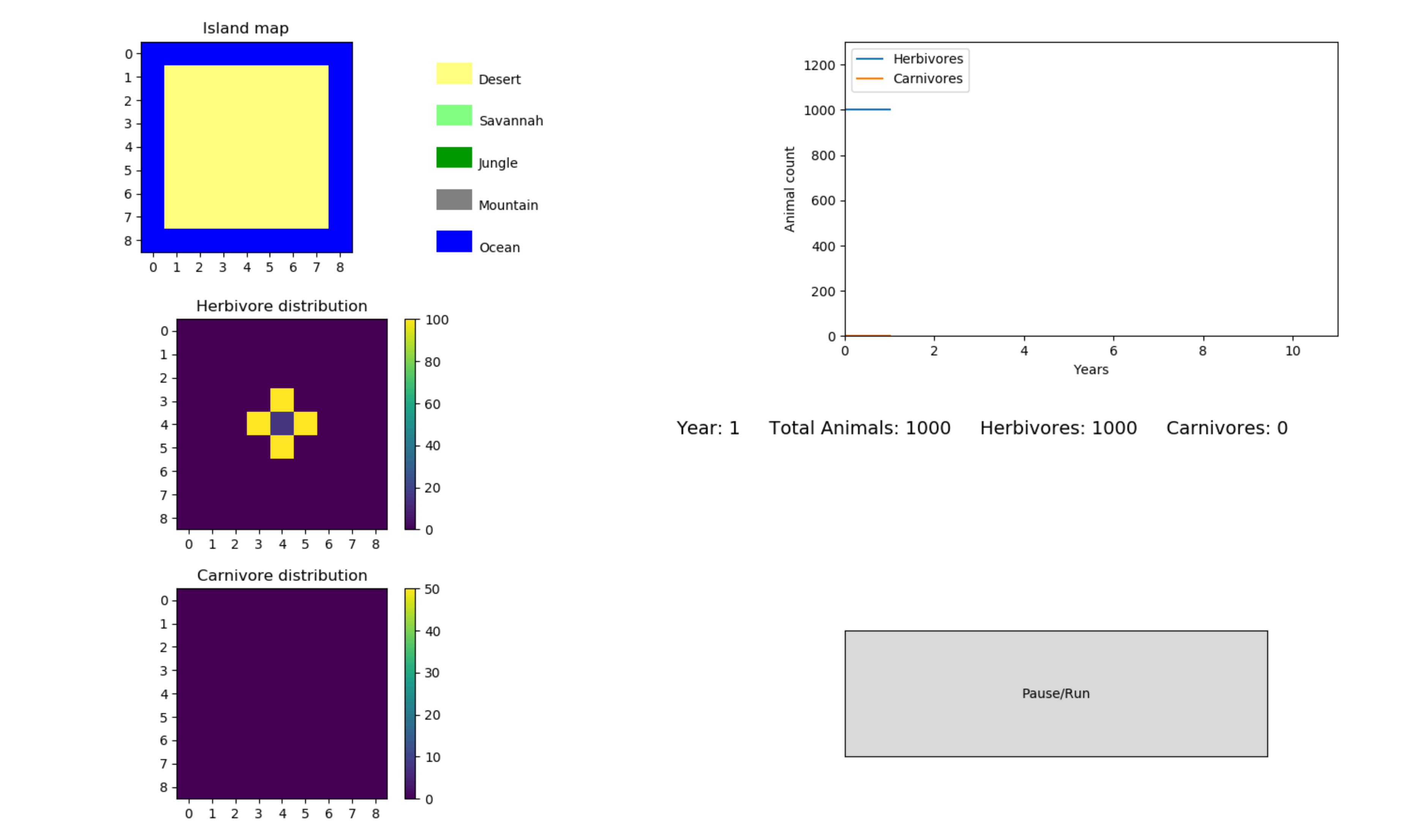


Name	Call Count	Time (ms)	Own Time (ms) ▼
fitness_update	2349743	3761 15,1 %	3038 12,2 %
uniform	6073094	2975 12,0 %	2448 9,8 %
will_birth	1361546	5964 24,0 %	1733 7,0 %
migrate_all_animals	31400	4615 18,6 %	1546 6,2 %
death	1679751	1846 7,4 %	1068 4,3 %
feeding	213058	1487 6,0 %	877 3,5 %
<built-in method math.exp>	5621127	859 3,5 %	859 3,5 %
migrate	1679751	1608 6,5 %	811 3,3 %
<method 'normal' of 'mtrand.RandomState' objects>	317019	811 3,3 %	811 3,3 %
weightloss_all_animals	31400	3899 15,7 %	729 2,9 %
weightloss	1679751	593 2,4 %	593 2,4 %
__init__	317019	1997 8,0 %	549 2,2 %
<method 'random' of '_random.Random' objects>	6073094	527 2,1 %	527 2,1 %
birth_all_animals	31400	6559 26,4 %	516 2,1 %
feed_all_animals	31400	3039 12,2 %	510 2,1 %
aging_all_animals	31400	794 3,2 %	447 1,8 %
__init__	283827	2191 8,8 %	410 1,6 %
<built-in method builtins.min>	1363821	365 1,5 %	365 1,5 %
age_animal	1679751	346 1,4 %	346 1,4 %
<method 'sort' of 'list' objects>	62829	679 2,7 %	344 1,4 %

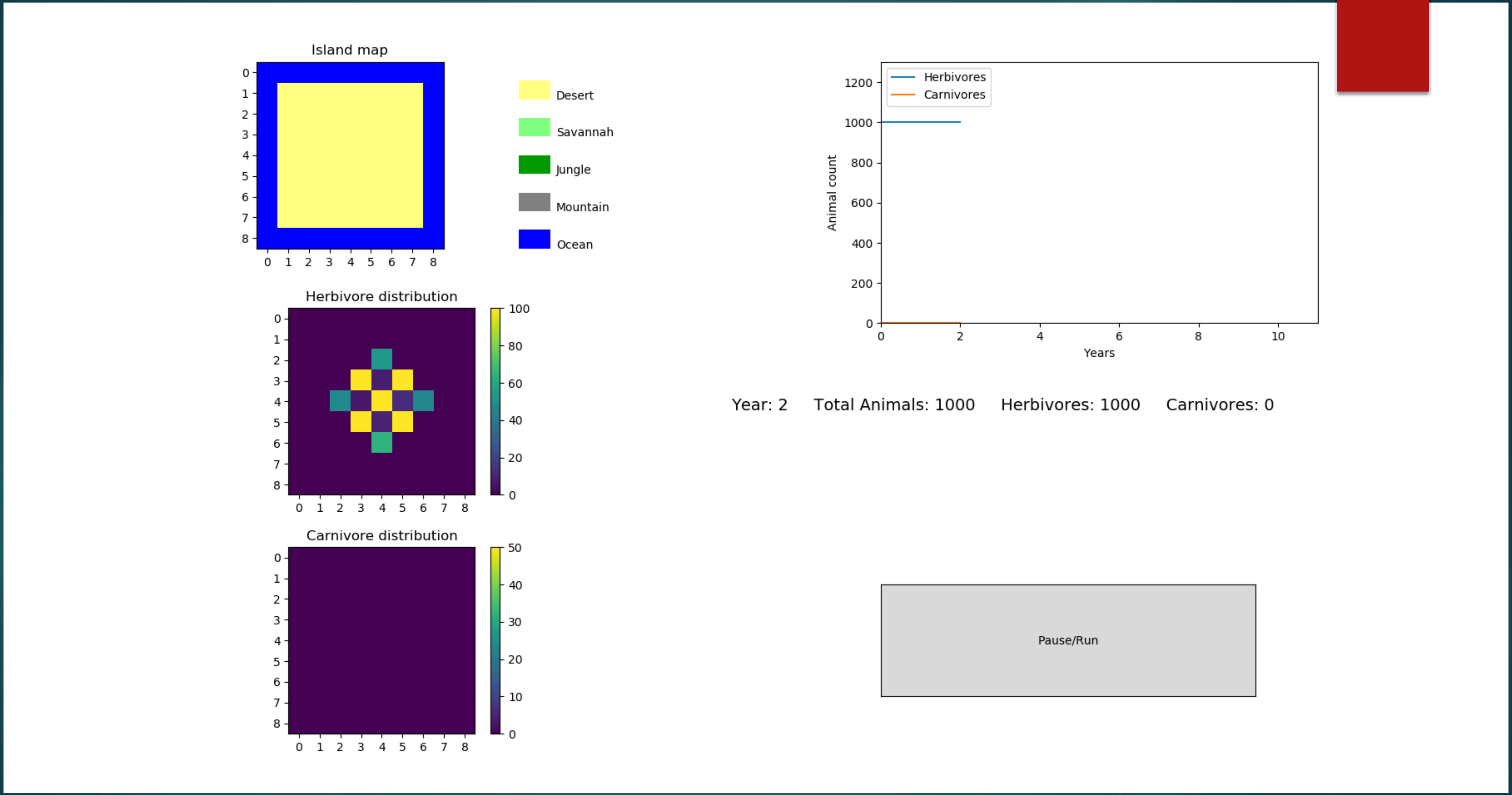
Problemer og forbedring

- C_max verdi Herbivore burde være mindre
- Pep-8 Eksempler
- Plotter alltid år 0 men sparer ikke til grafikken
- Image_years = 0 gir Zerodivisionerror

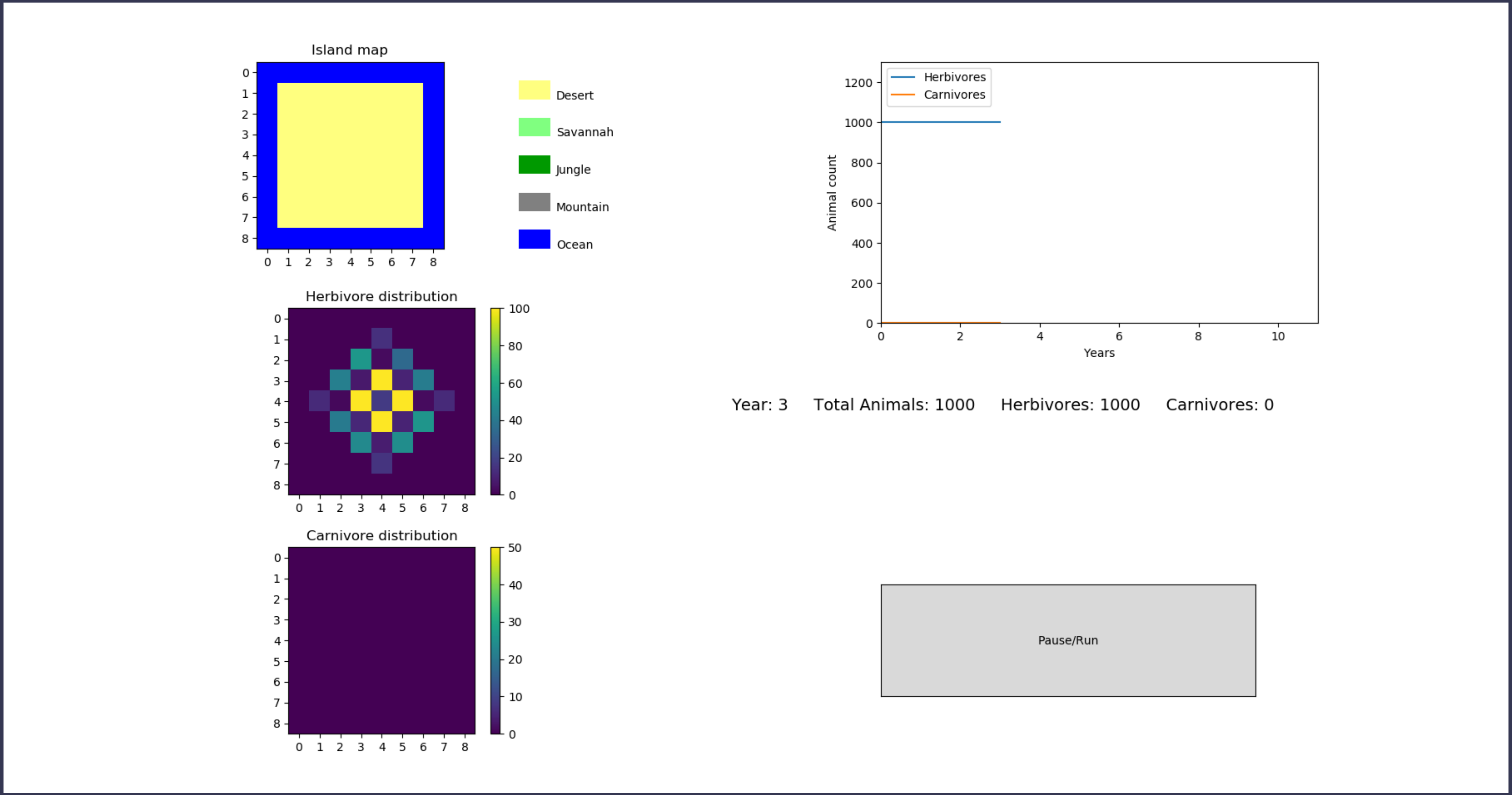
Eksempler: migration_only



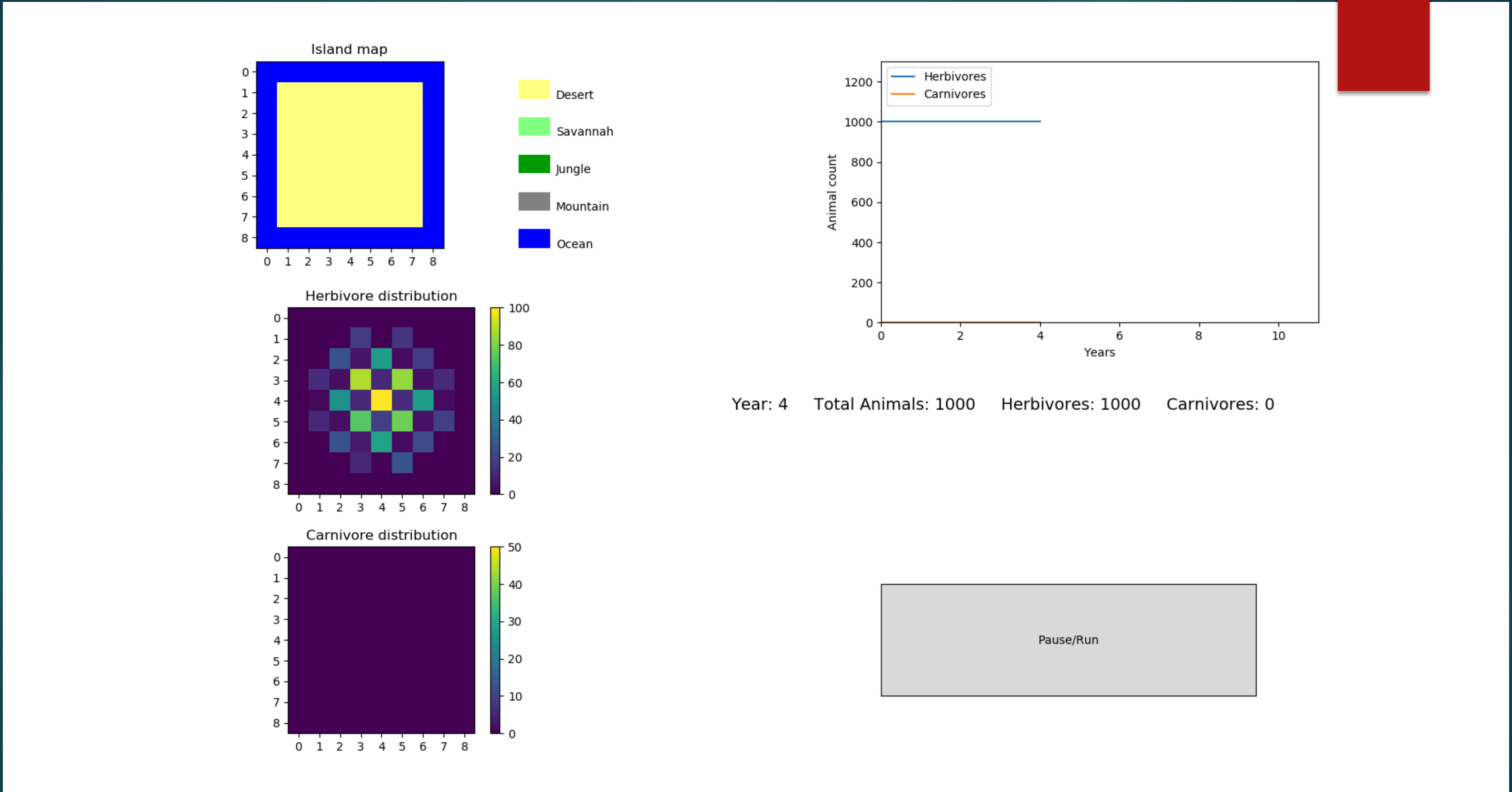
Year: 1   Total Animals: 1000   Herbivores: 1000   Carnivores: 0



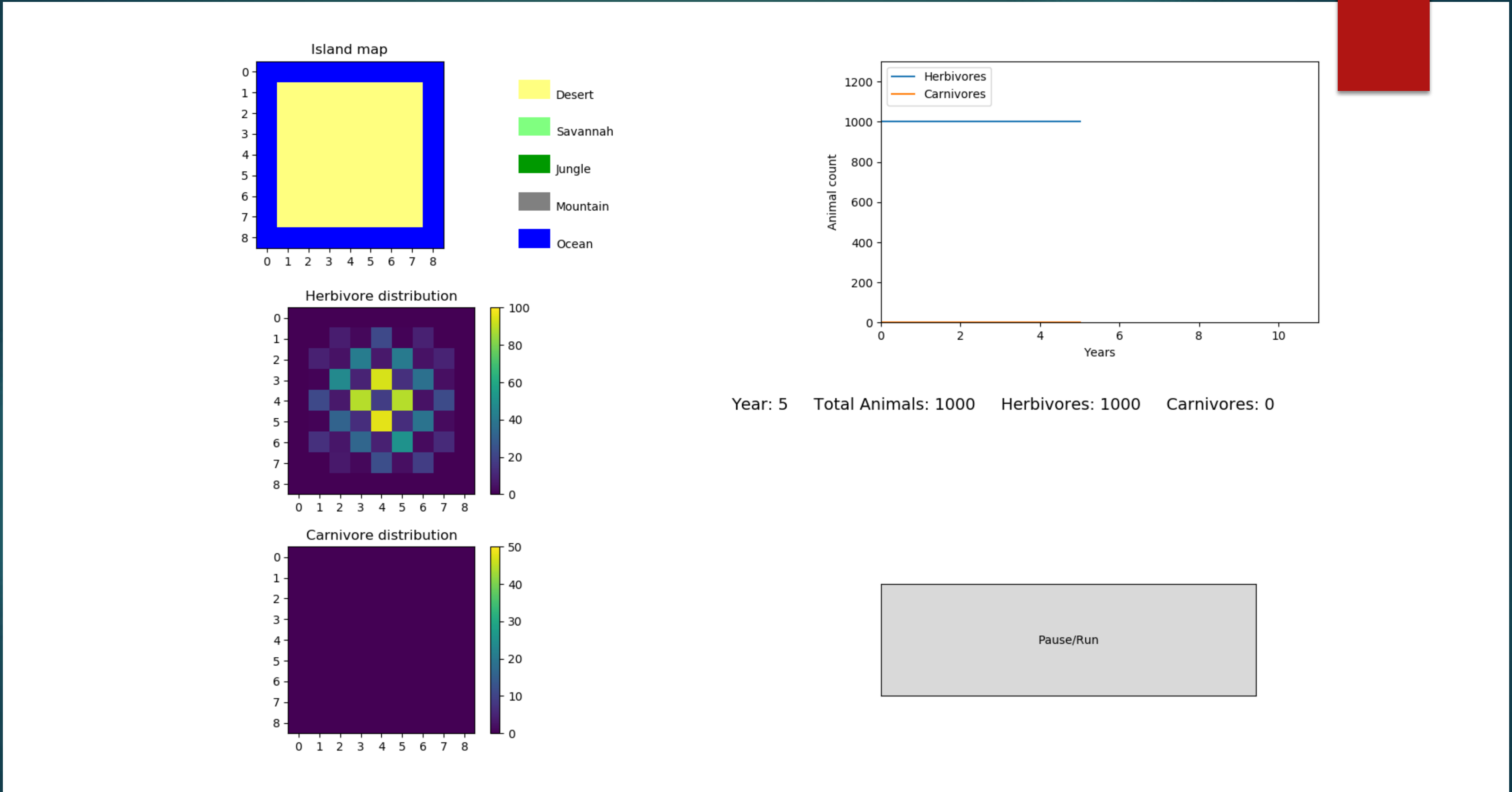
```
Year: 2     Total Animals: 1000     Herbivores: 1000     Carnivores: 0
```



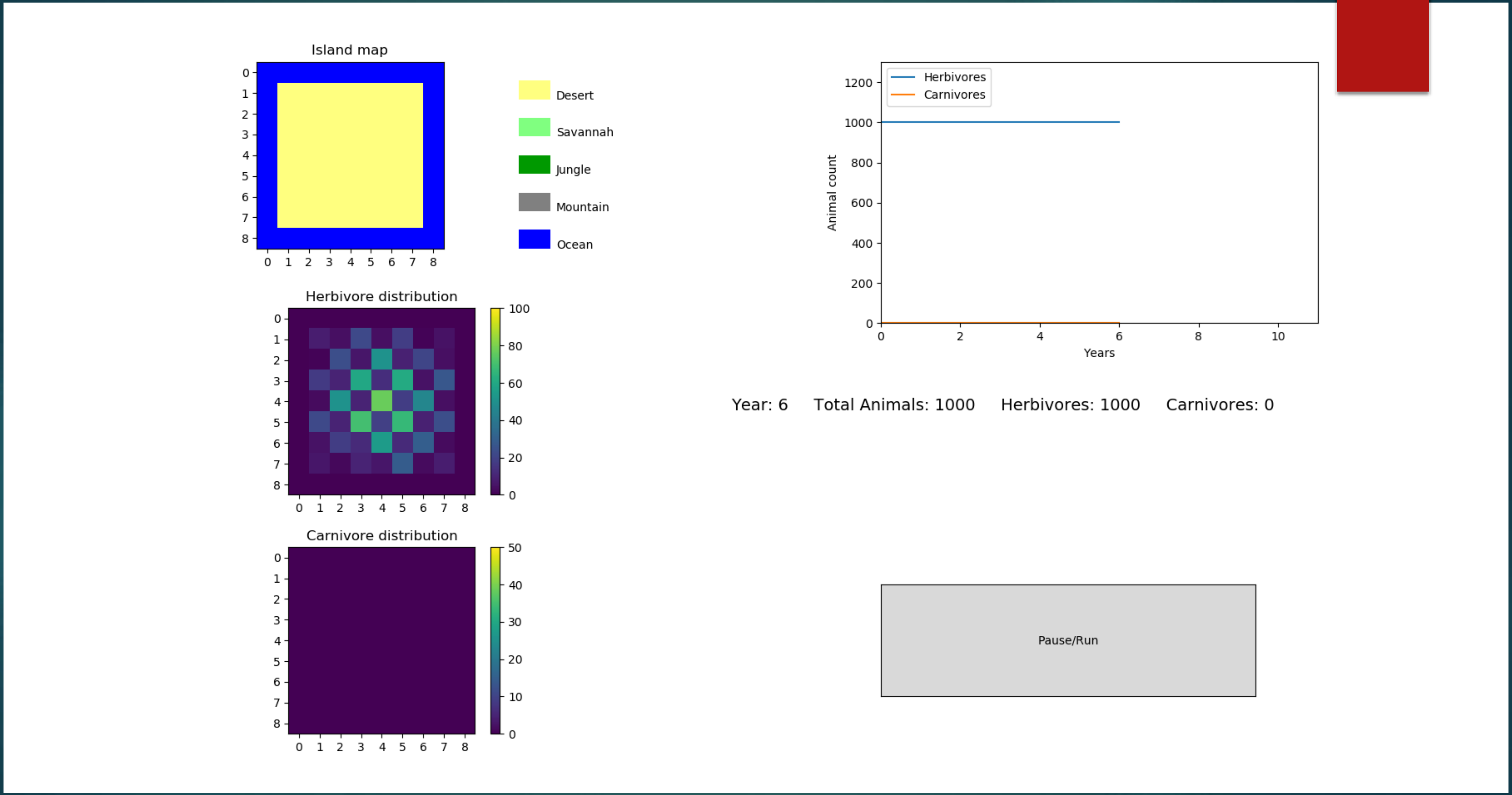
Year: 3 Total Animals: 1000 Herbivores: 1000 Carnivores: 0



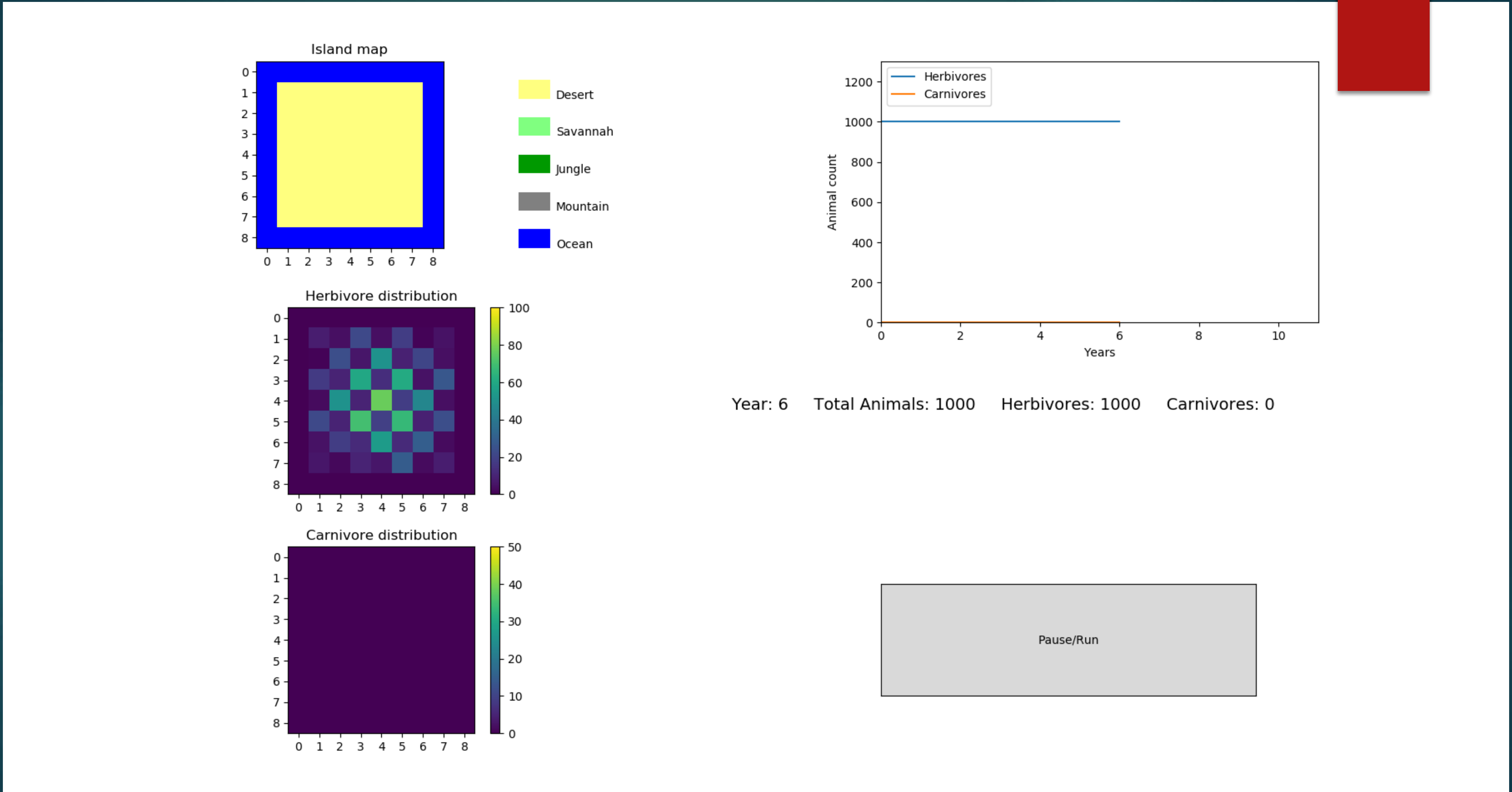
```
Year: 4    Total Animals: 1000    Herbivores: 1000    Carnivores: 0
```



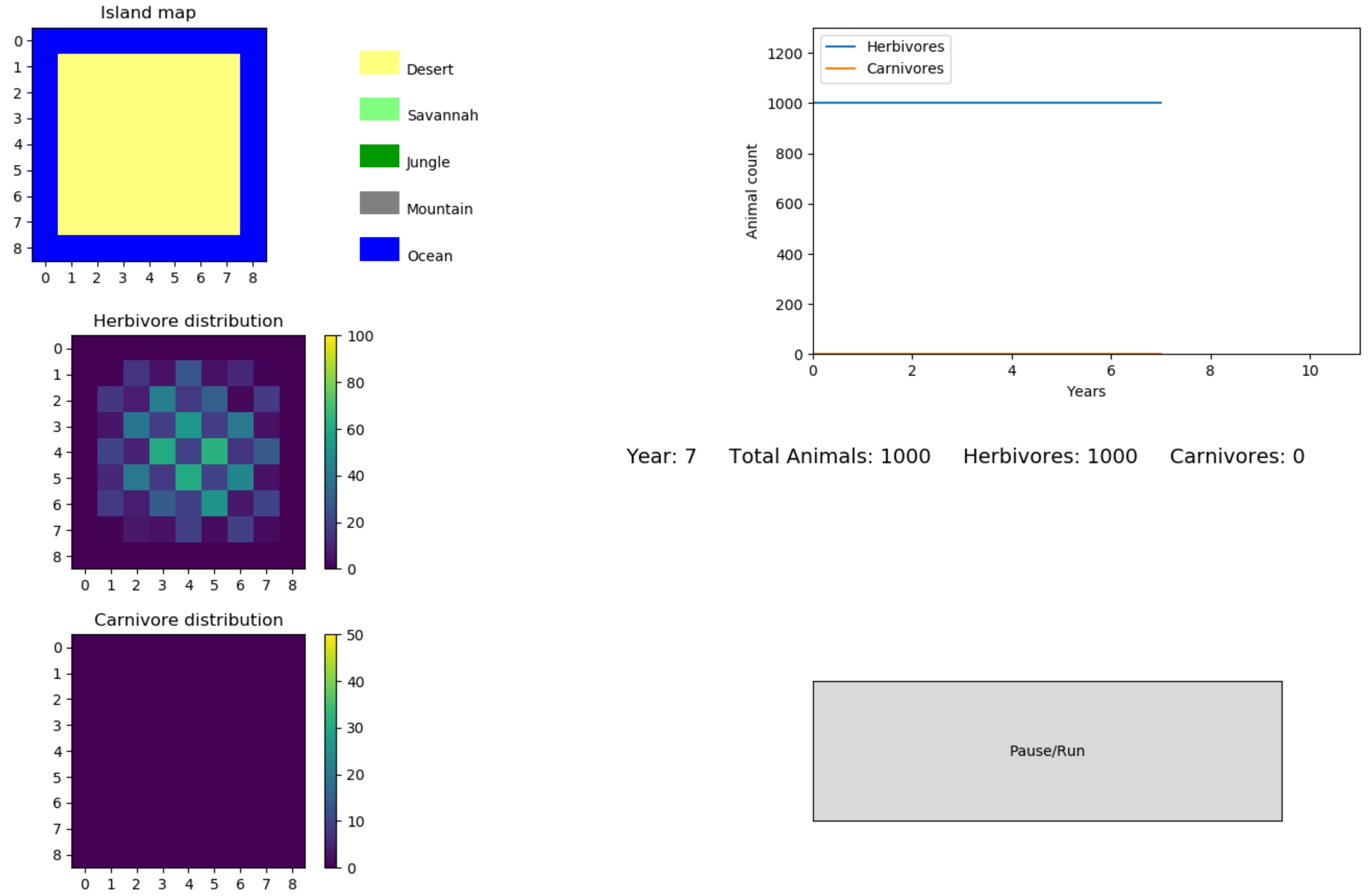
Year: 5 Total Animals: 1000 Herbivores: 1000 Carnivores: 0



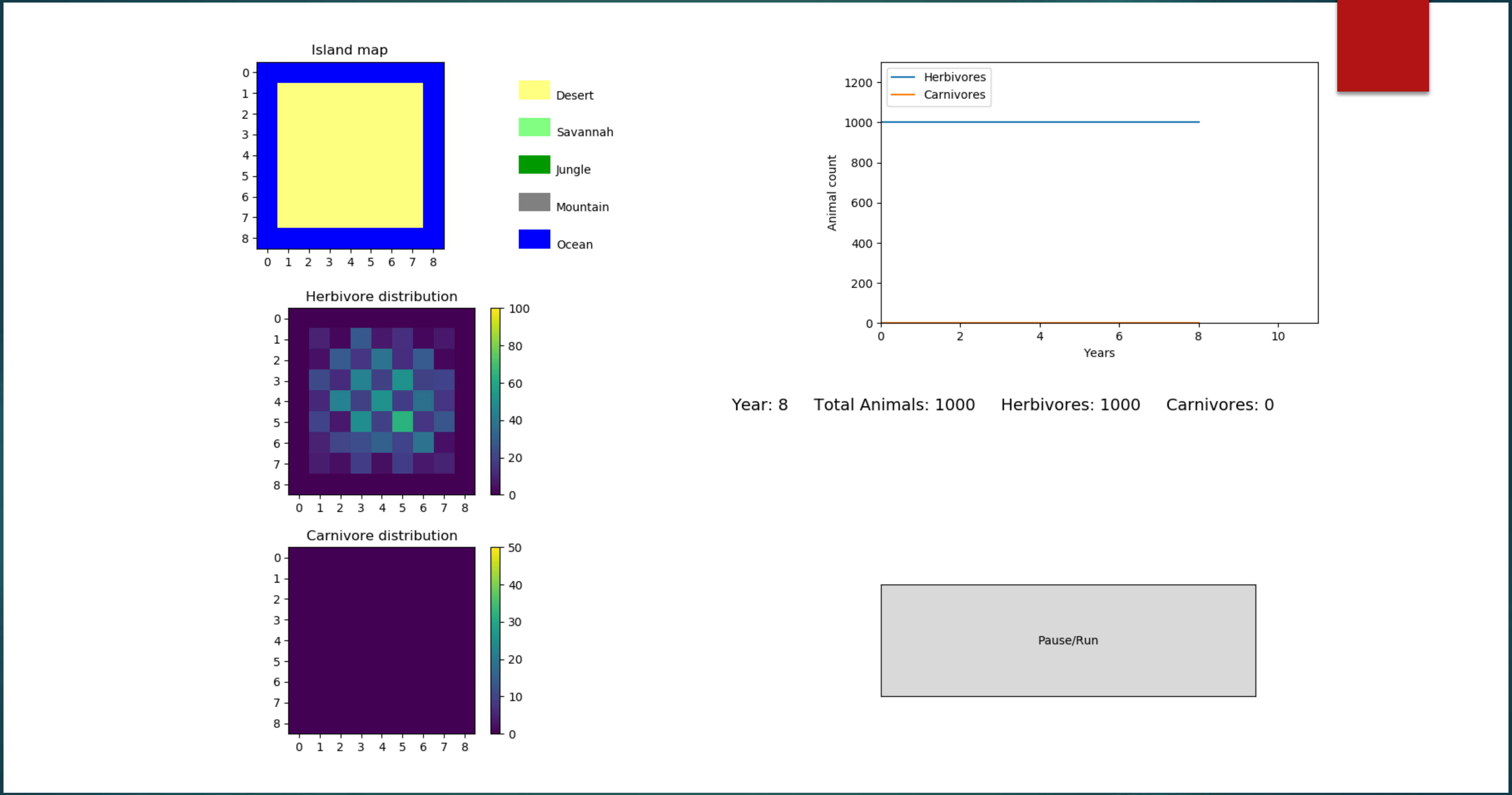
```
Year: 6    Total Animals: 1000    Herbivores: 1000    Carnivores: 0
```



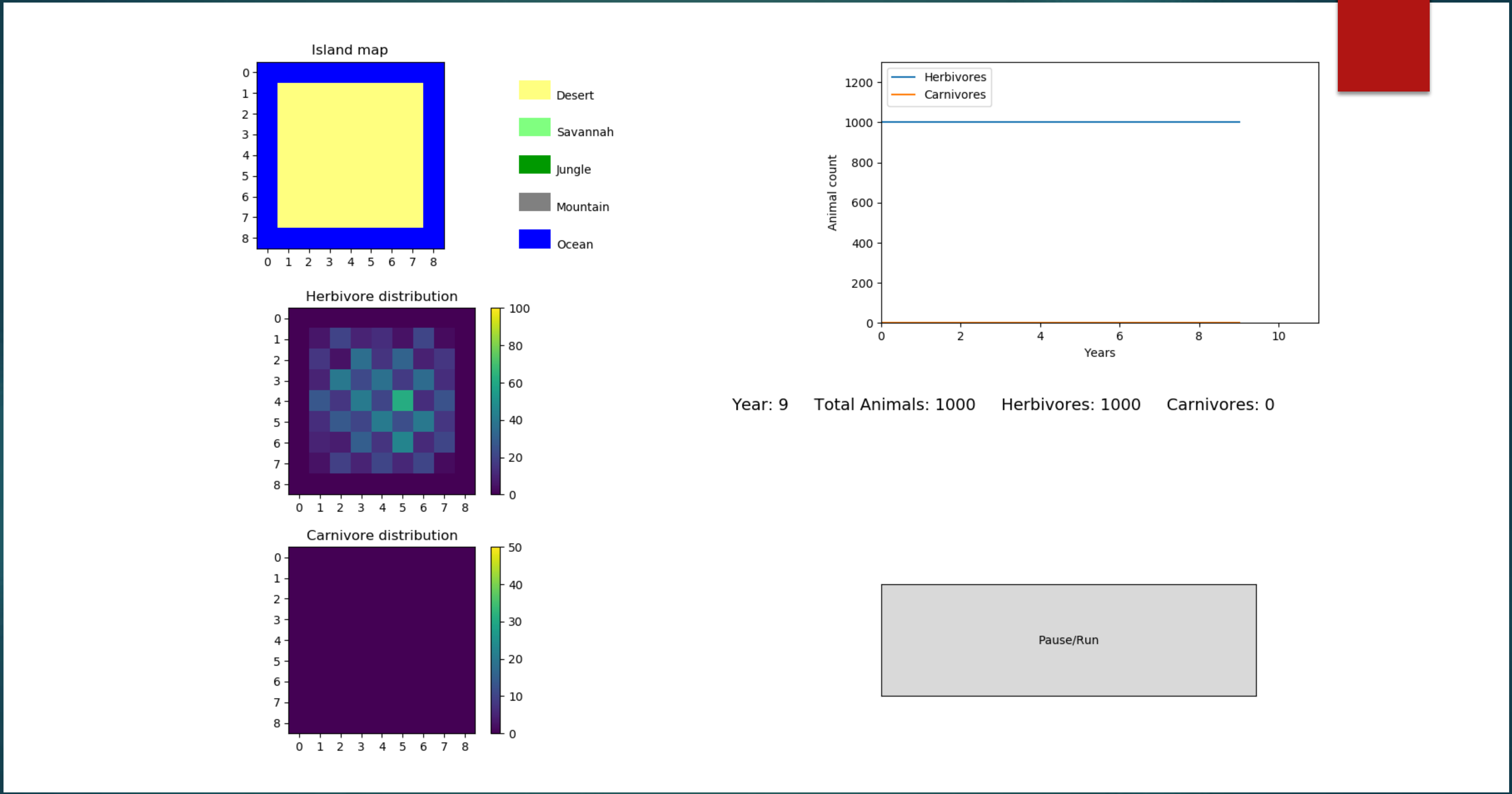
Year: 6 Total Animals: 1000 Herbivores: 1000 Carnivores: 0



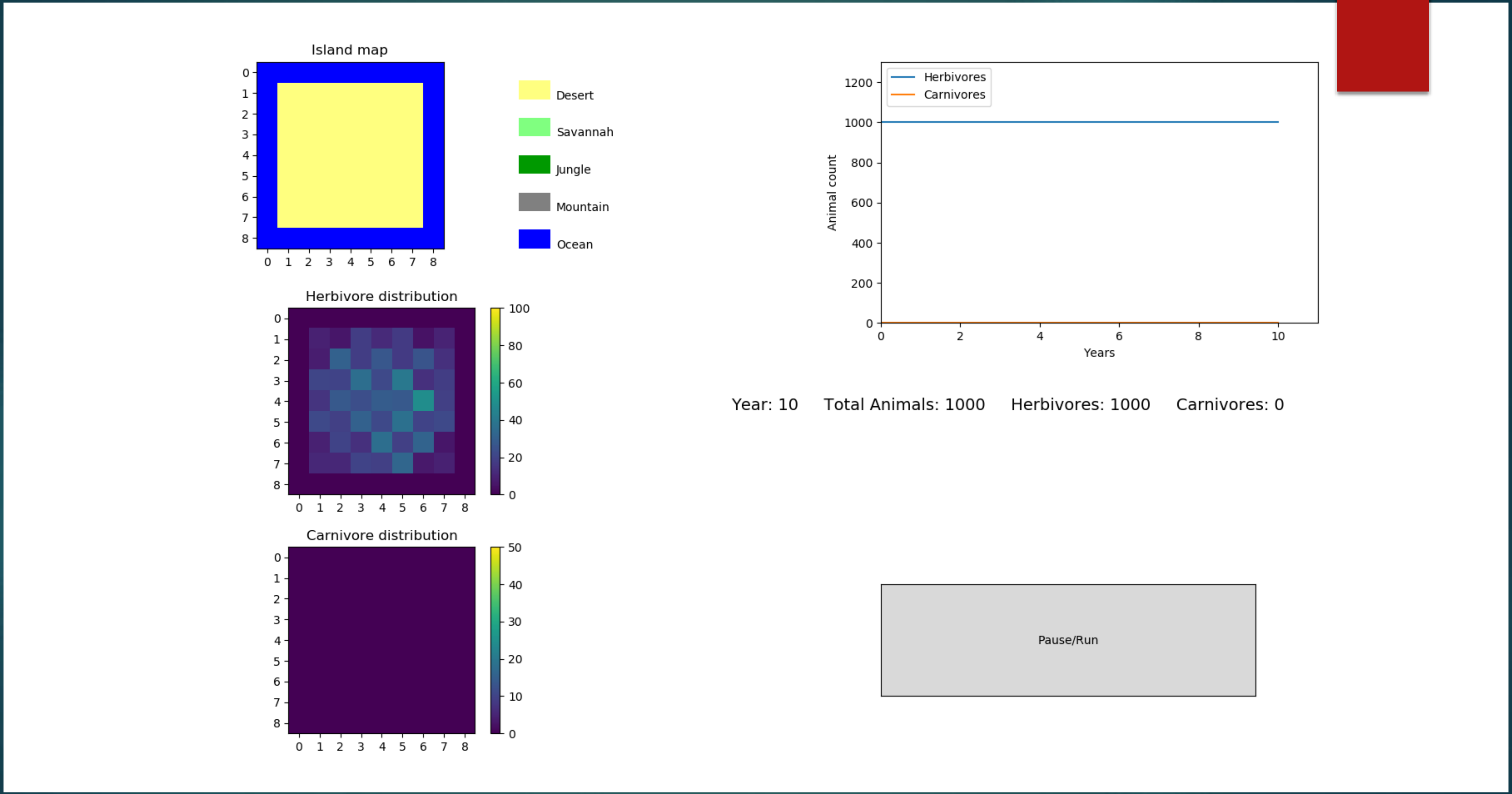
Year: 7 Total Animals: 1000 Herbivores: 1000 Carnivores: 0



```
Year: 8    Total Animals: 1000    Herbivores: 1000    Carnivores: 0
```

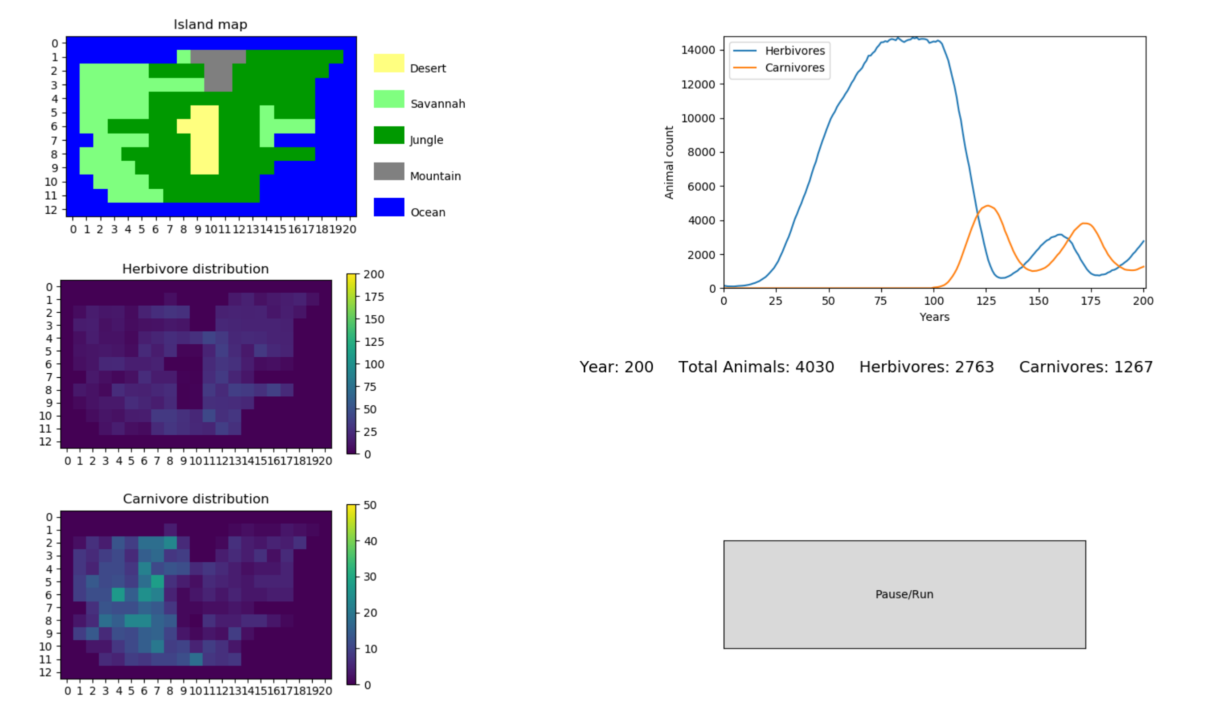


Year: 9 Total Animals: 1000 Herbivores: 1000 Carnivores: 0

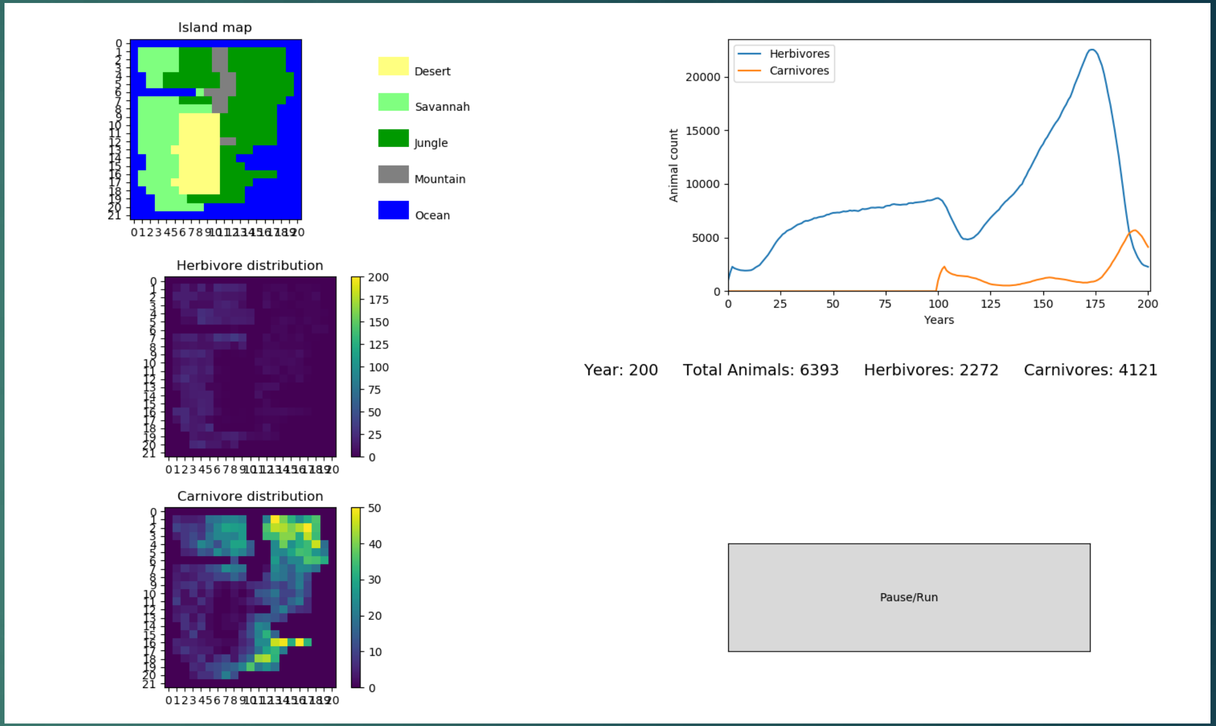


```
Year: 10     Total Animals: 1000     Herbivores: 1000     Carnivores: 0
```

Check_sim og Rossumøya



Year: 200 Total Animals: 4030 Herbivores: 2763 Carnivores: 1267



Year: 200 Total Animals: 6393 Herbivores: 2272 Carnivores: 4121