Rossumøya

Modelling an ecosystem using python

Group 17: Peter Langdalen, 23.06.2020

The objectives of this Exam

- Develop a program that simulates the population dynamics of Rossumøya

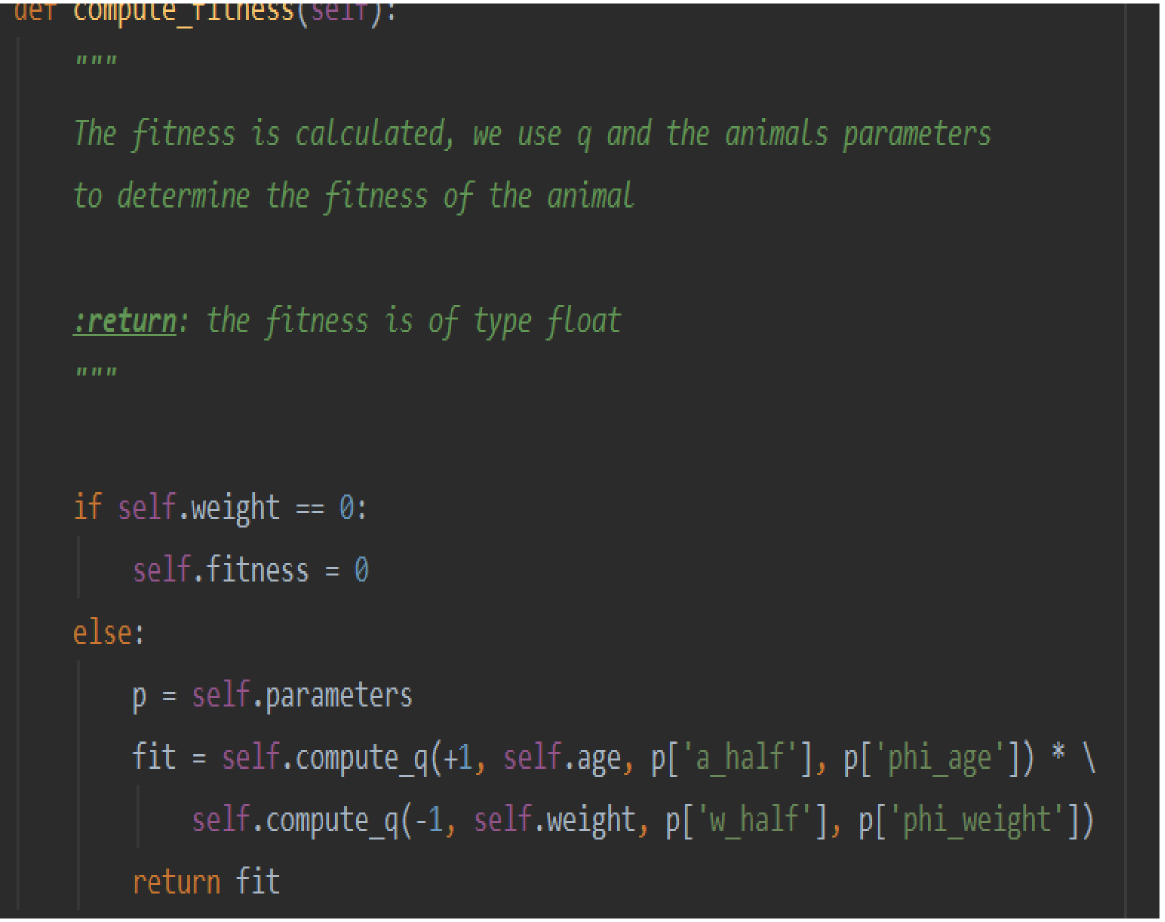
- Document the process

- Visualize

My approach

- Progression according to the project milestones
- Used solutions that i was familiar with
- Discussion with my peers

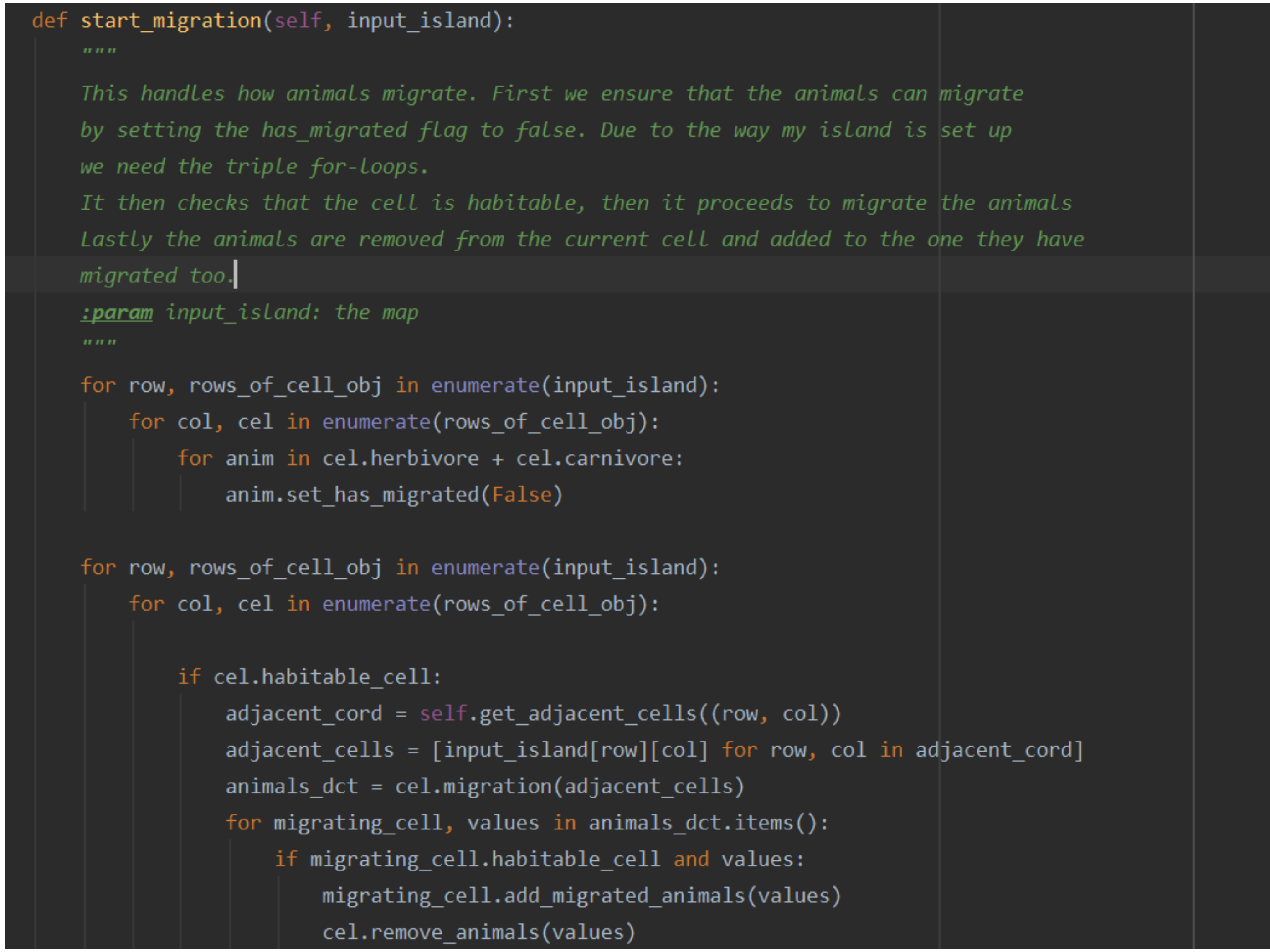
Code examples



```
def compute_fitness(self):
    """
    The fitness is calculated, we use q and the animals parameters
    to determine the fitness of the animal

    :return: the fitness is of type float
    """

    if self.weight == 0:
        self.fitness = 0
    else:
        p = self.parameters
        fit = self.compute_q(+1, self.age, p['a_half'], p['phi_age']) * \
            self.compute_q(-1, self.weight, p['w_half'], p['phi_weight'])
        return fit
```



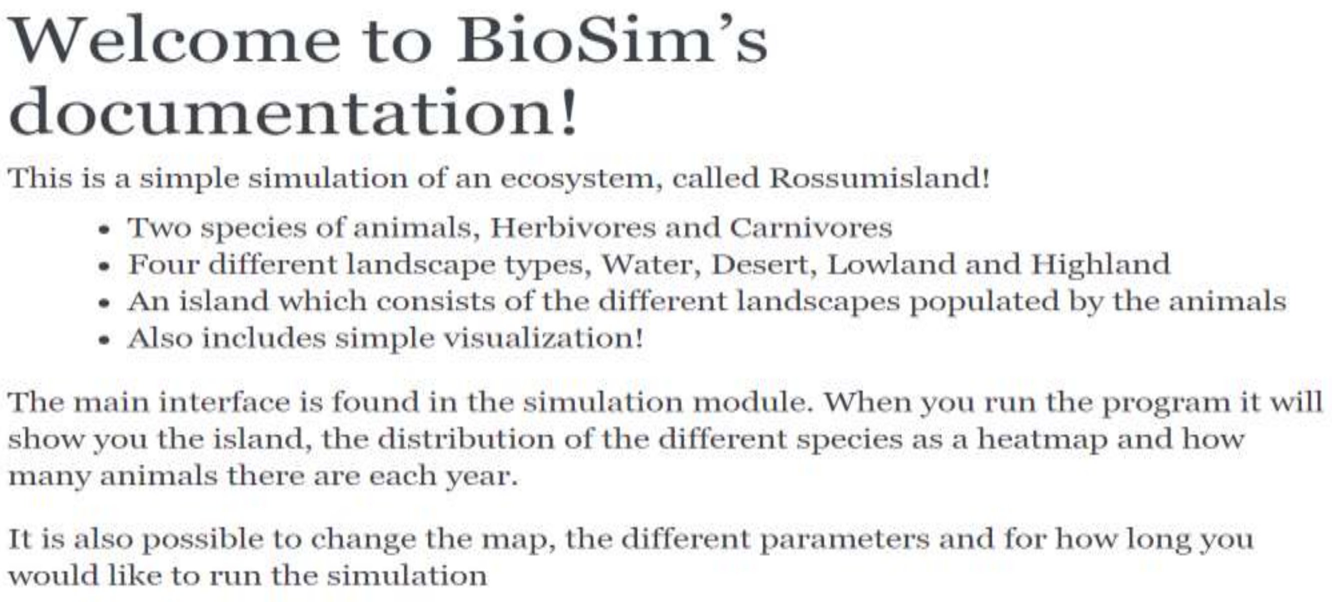
```
def start_migration(self, input_island):
    """
    This handles how animals migrate. First we ensure that the animals can migrate
    by setting the has_migrated flag to false. Due to the way my island is set up
    we need the triple for-loops.
    It then checks that the cell is habitable, then it proceeds to migrate the animals
    Lastly the animals are removed from the current cell and added to the one they have
    migrated too.
    :param input_island: the map
    """
    for row, rows_of_cell_obj in enumerate(input_island):
        for col, cel in enumerate(rows_of_cell_obj):
            for anim in cel.herbivore + cel.carnivore:
                anim.set_has_migrated(False)

    for row, rows_of_cell_obj in enumerate(input_island):
        for col, cel in enumerate(rows_of_cell_obj):

            if cel.habitable_cell:
                adjacent_cord = self.get_adjacent_cells((row, col))
                adjacent_cells = [input_island[row][col] for row, col in adjacent_cord]
                animals_dct = cel.migration(adjacent_cells)
                for migrating_cell, values in animals_dct.items():
                    if migrating_cell.habitable_cell and values:
                        migrating_cell.add_migrated_animals(values)
                        cel.remove_animals(values)
```

Program quality

- Different type of tests
- Documentation



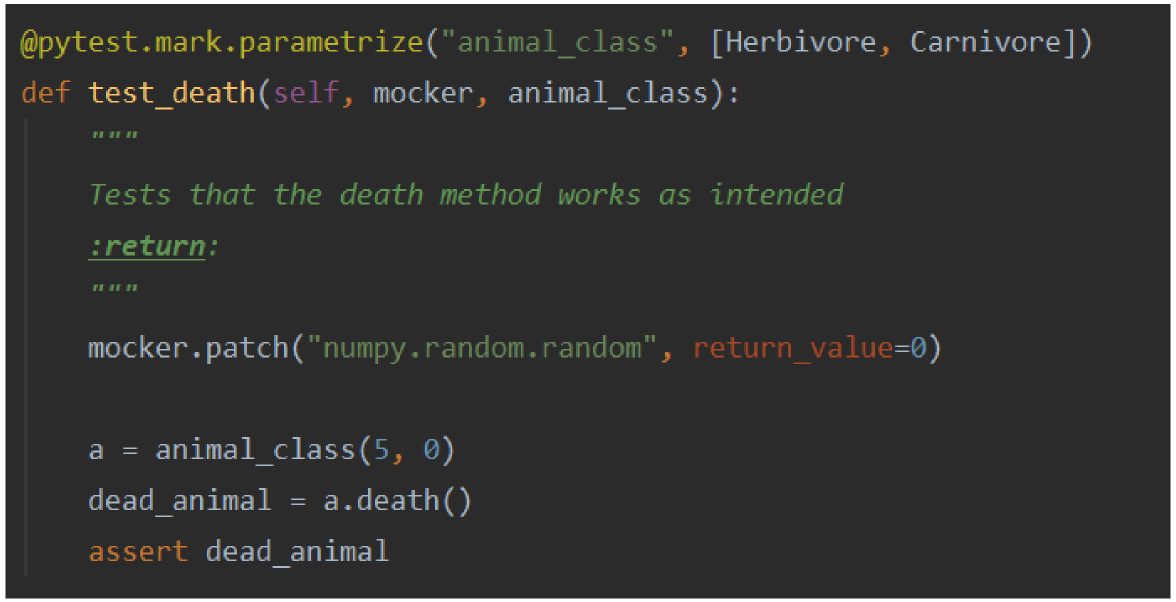
Welcome to BioSim's documentation!

This is a simple simulation of an ecosystem, called Rossumisland!

- Two species of animals, Herbivores and Carnivores
- Four different landscape types, Water, Desert, Lowland and Highland
- An island which consists of the different landscapes populated by the animals
- Also includes simple visualization!

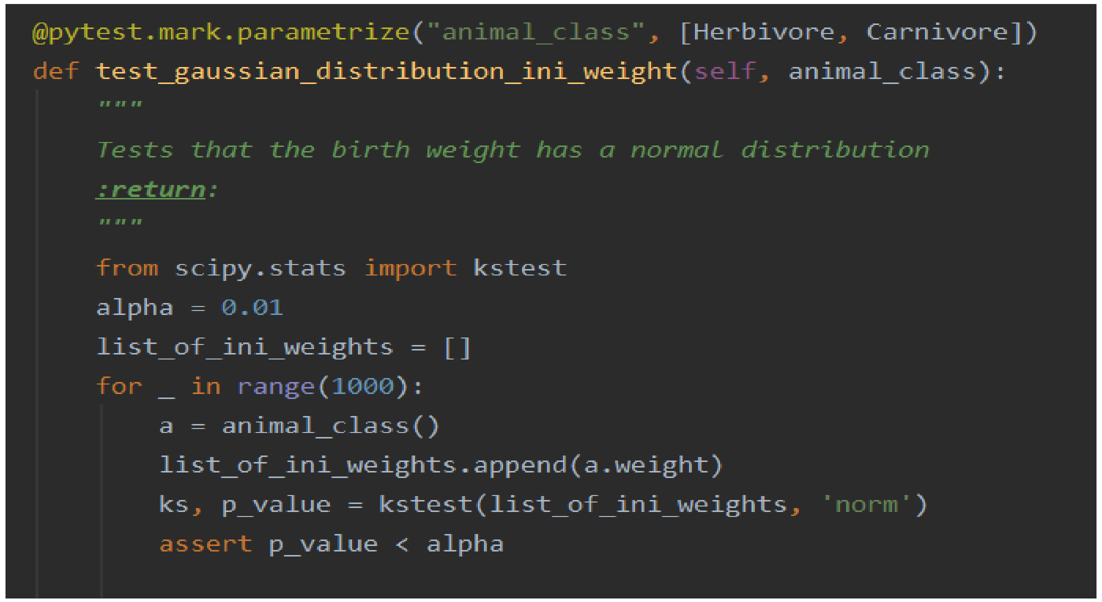
The main interface is found in the simulation module. When you run the program it will show you the island, the distribution of the different species as a heatmap and how many animals there are each year.

It is also possible to change the map, the different parameters and for how long you would like to run the simulation



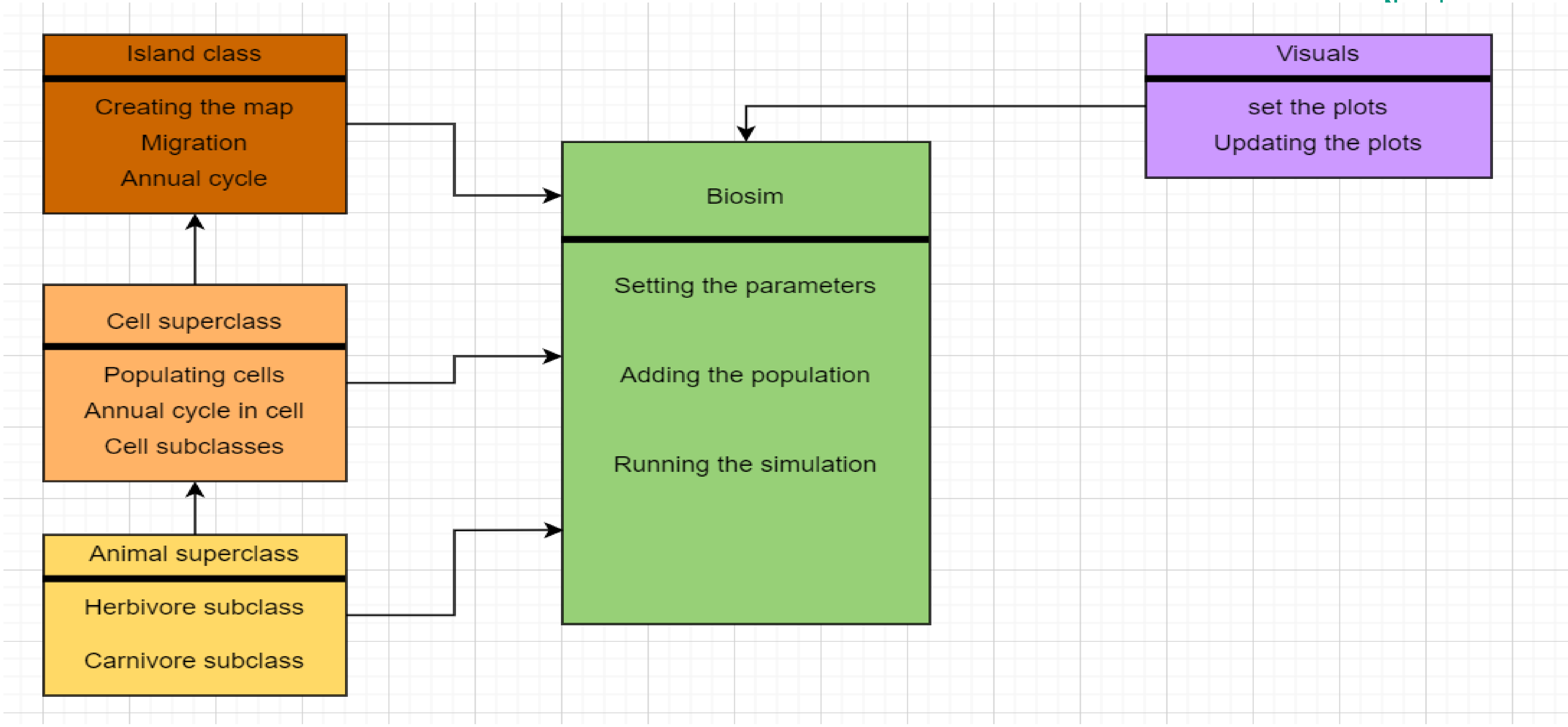
```
@pytest.mark.parametrize("animal_class", [Herbivore, Carnivore])
def test_death(self, mocker, animal_class):
    """
    Tests that the death method works as intended
    :return:
    """
    mocker.patch("numpy.random.random", return_value=0)

    a = animal_class(5, 0)
    dead_animal = a.death()
    assert dead_animal
```



```
@pytest.mark.parametrize("animal_class", [Herbivore, Carnivore])
def test_gaussian_distribution_ini_weight(self, animal_class):
    """
    Tests that the birth weight has a normal distribution
    :return:
    """
    from scipy.stats import kstest
    alpha = 0.01
    list_of_ini_weights = []
    for _ in range(1000):
        a = animal_class()
        list_of_ini_weights.append(a.weight)
        ks, p_value = kstest(list_of_ini_weights, 'norm')
        assert p_value < alpha
```

The general structure of the program



Challenges

- Properly managing time
- Dealing with bugs and problems
- Making good solutions, not just working ones

Future improvements

- Less dense methods and general reorganizing
- More elegant and clever solutions
- Making the program faster
- Implementing histograms