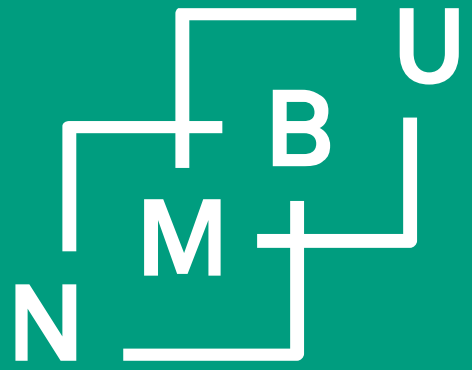


Norwegian University
of Life Sciences

Modelling the Ecosystem of Rossumøya

Group: BioSim T02 Abbas GeddeDahl

Ahmar Abbas

Gøran Sildnes Gedde-Dahl

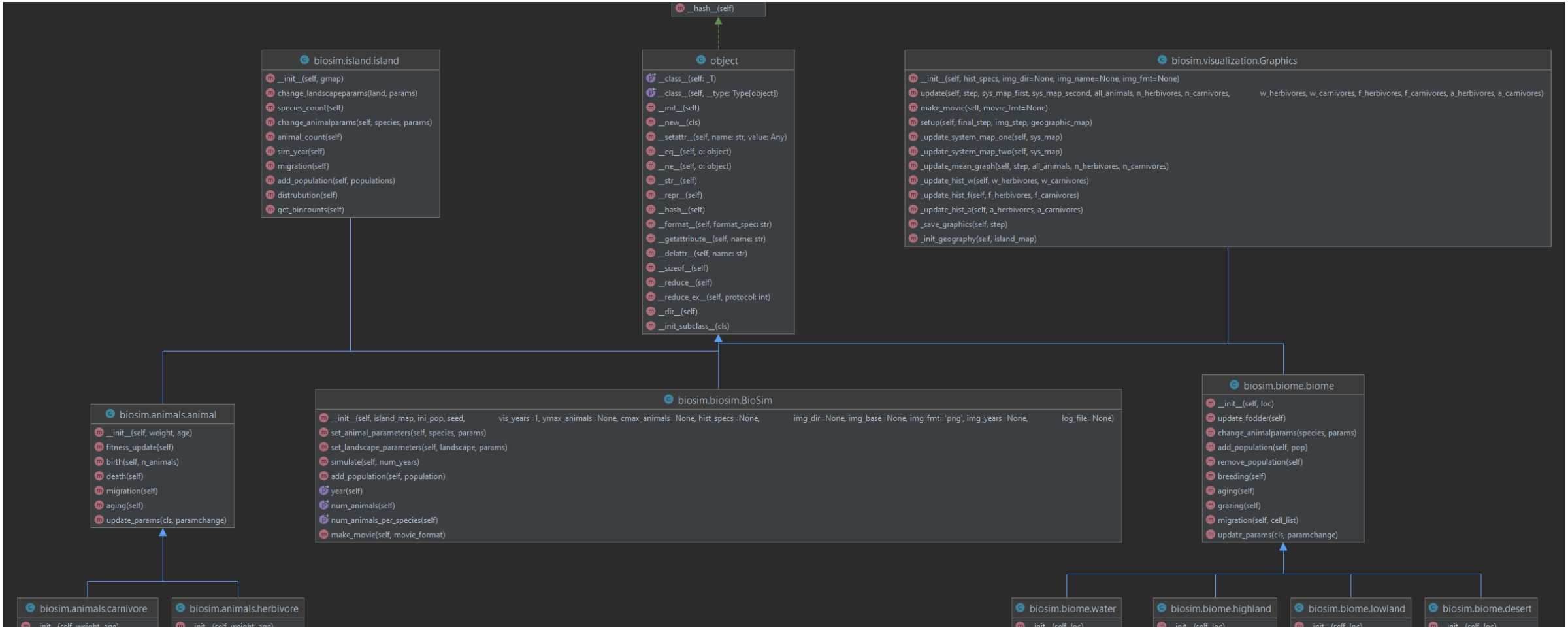
Agenda

- Design and Implementation
- Quality Assurance
- Documentation
- Improvements

Design and Implementation

Keep-it-simple!

Class Diagram

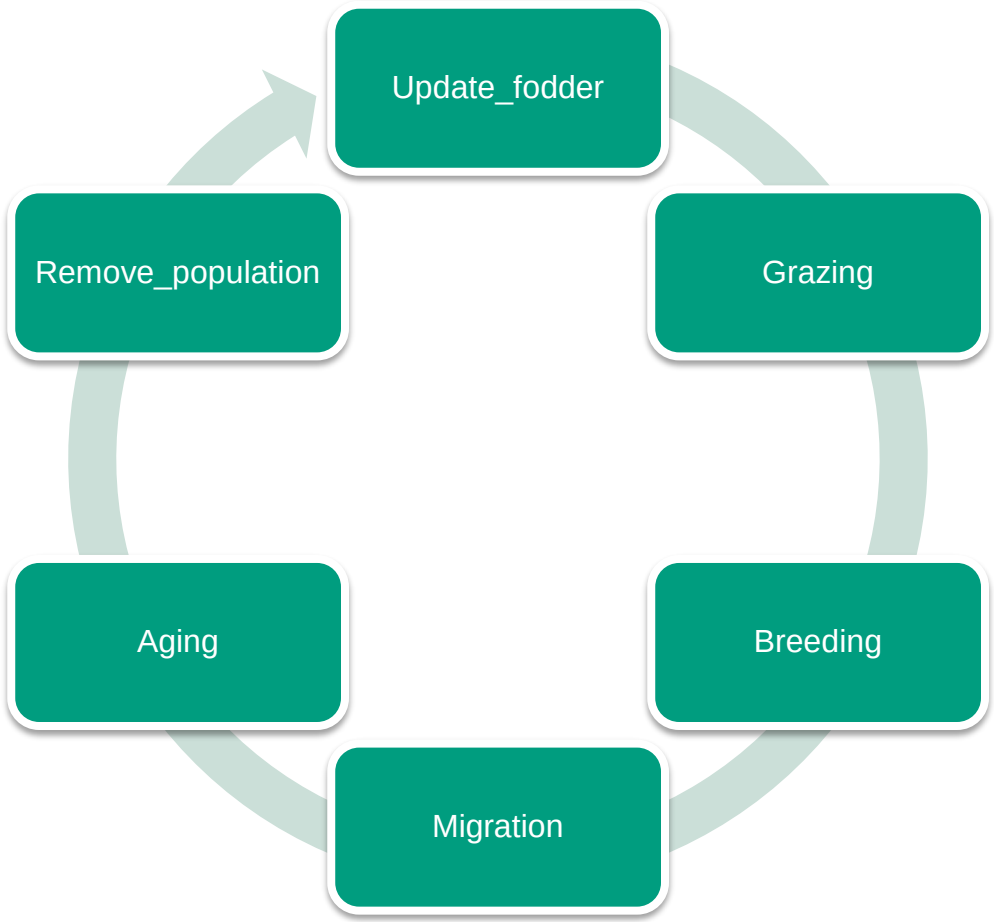


OOP implementation

 Inheritance	Herbivores, Carnivores → Animals Lowlands, Highlands, Water, Desert → Biome
 Polymorphism	Animal attributes → Herbivores attributes
 Encapsulation	Access modifier in Visualization
 Abstraction	Biosim.simulate() → island.sim_year()

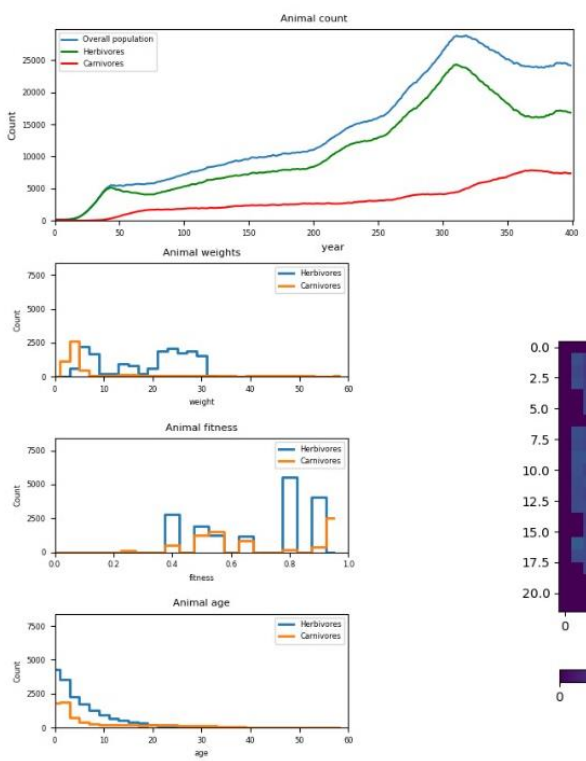
Annual Cycle

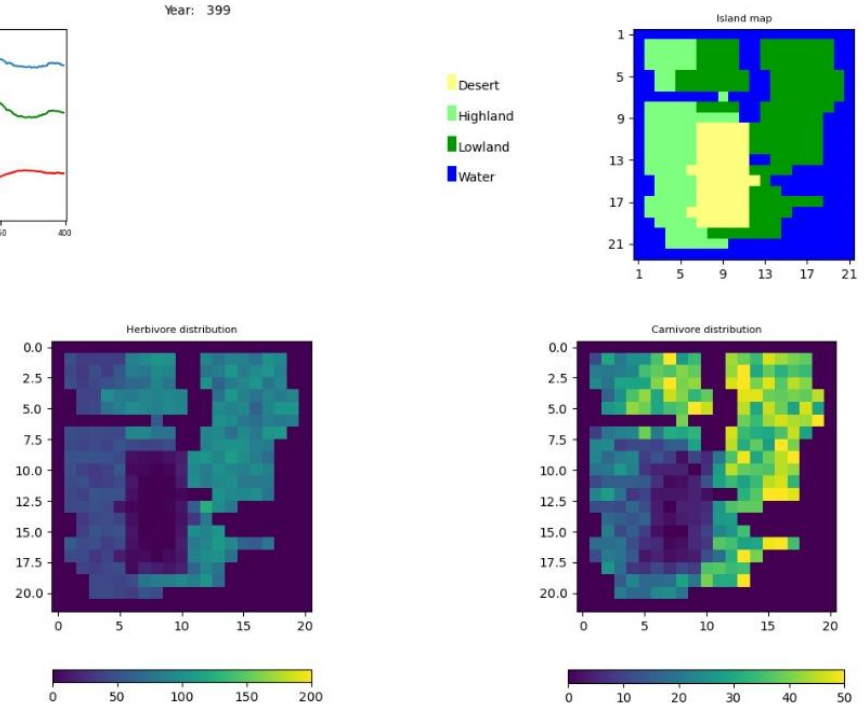
- Each function is called only once a year in each Island cell.
- All functions are implemented in Biome class except for migration.
- Actual implementation of migration is also in Biome class but a list of possible migration destinations is evaluated and passed through Island class.



Visualization and Logging

- Simple and informative layout
- Simulation level logging





log_file - Notepad

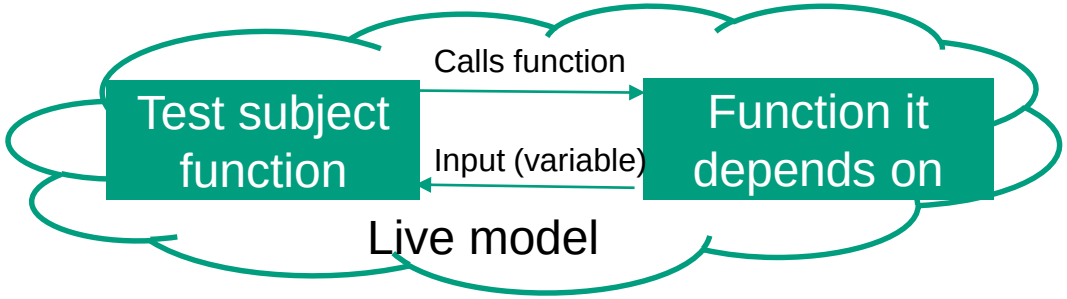
File Edit Format View Help

```
[INFO]biosim.simulate - 2021-06-16 11:18:56,826 - {'Year': 1, 'Total_Animals': 150, 'Animal_per_specie': {'Carnivore': 0, 'Herbivore': 150}}
[INFO]biosim.simulate - 2021-06-16 11:18:59,989 - {'Year': 2, 'Total_Animals': 135, 'Animal_per_specie': {'Carnivore': 0, 'Herbivore': 135}}
[INFO]biosim.simulate - 2021-06-16 11:19:03,470 - {'Year': 3, 'Total_Animals': 120, 'Animal_per_specie': {'Carnivore': 0, 'Herbivore': 120}}
[INFO]biosim.simulate - 2021-06-16 11:19:07,028 - {'Year': 4, 'Total_Animals': 113, 'Animal_per_specie': {'Carnivore': 0, 'Herbivore': 113}}
[INFO]biosim.simulate - 2021-06-16 11:19:10,856 - {'Year': 5, 'Total_Animals': 95, 'Animal_per_specie': {'Carnivore': 0, 'Herbivore': 95}}
[INFO]biosim.simulate - 2021-06-16 11:19:13,430 - {'Year': 6, 'Total_Animals': 94, 'Animal_per_specie': {'Carnivore': 0, 'Herbivore': 94}}
[INFO]biosim.simulate - 2021-06-16 11:19:15,914 - {'Year': 7, 'Total_Animals': 96, 'Animal_per_specie': {'Carnivore': 0, 'Herbivore': 96}}
[INFO]biosim.simulate - 2021-06-16 11:19:18,403 - {'Year': 8, 'Total_Animals': 94, 'Animal_per_specie': {'Carnivore': 0, 'Herbivore': 94}}
[INFO]biosim.simulate - 2021-06-16 11:19:20,918 - {'Year': 9, 'Total_Animals': 88, 'Animal_per_specie': {'Carnivore': 0, 'Herbivore': 88}}
[INFO]biosim.simulate - 2021-06-16 11:19:23,529 - {'Year': 10, 'Total_Animals': 92, 'Animal_per_specie': {'Carnivore': 0, 'Herbivore': 92}}
[INFO]biosim.simulate - 2021-06-16 11:19:26,171 - {'Year': 11, 'Total_Animals': 94, 'Animal_per_specie': {'Carnivore': 0, 'Herbivore': 94}}
[INFO]biosim.simulate - 2021-06-16 11:19:28,684 - {'Year': 12, 'Total_Animals': 106, 'Animal_per_specie': {'Carnivore': 0, 'Herbivore': 106}}
```

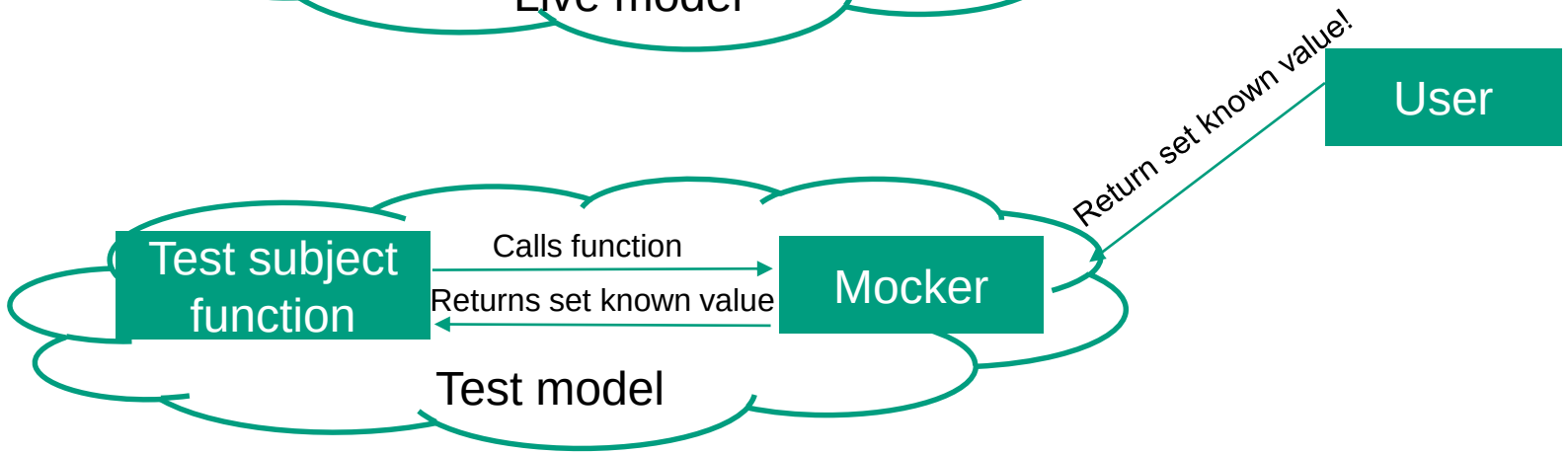
Quality Assurance

Quality over quantity always!

Testing

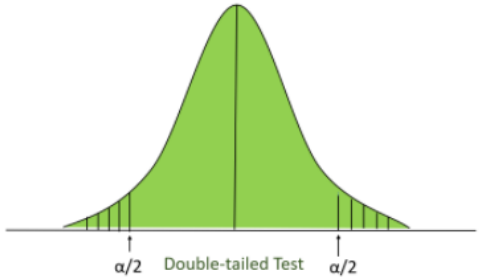


- Assertions
  - Object creation
  - Functions



- Mockers

- Statistical tests - Scipy stat binom_test function – two-tailed – null-hypothesis



$$p = \sum_{i \in \mathcal{I}} Pr(X = i) = \sum_{i \in \mathcal{I}} \binom{n}{i} p^i (1-p)^{n-i}$$

Testing Approach

- Unit testing

```python
def test_lowland_create():
    a = rd.randint(1, 50)
    b = rd.randint(1, 50)
    f_max = lowland.f_max
    cell = lowland((a, b))
    assert cell.f_max == f_max
    assert cell.habitable is True
```

- Statistical testing

```python
def test_stat_death():
    # Statistical test for probability of death,
    # checking that the probabilty of hypothesis correctness is more than 5%
    test_animals = [carnivore(age=2, weight=50) for _ in range(50)]
    p = test_animals[0].omega * (1 - test_animals[0].fitness)
    successes = [subject for subject in test_animals if subject.death() is True]
    p_hyp = stats.binom_test(len(successes), n=len(test_animals), p=p)
    assert p_hyp >= 0.05
```

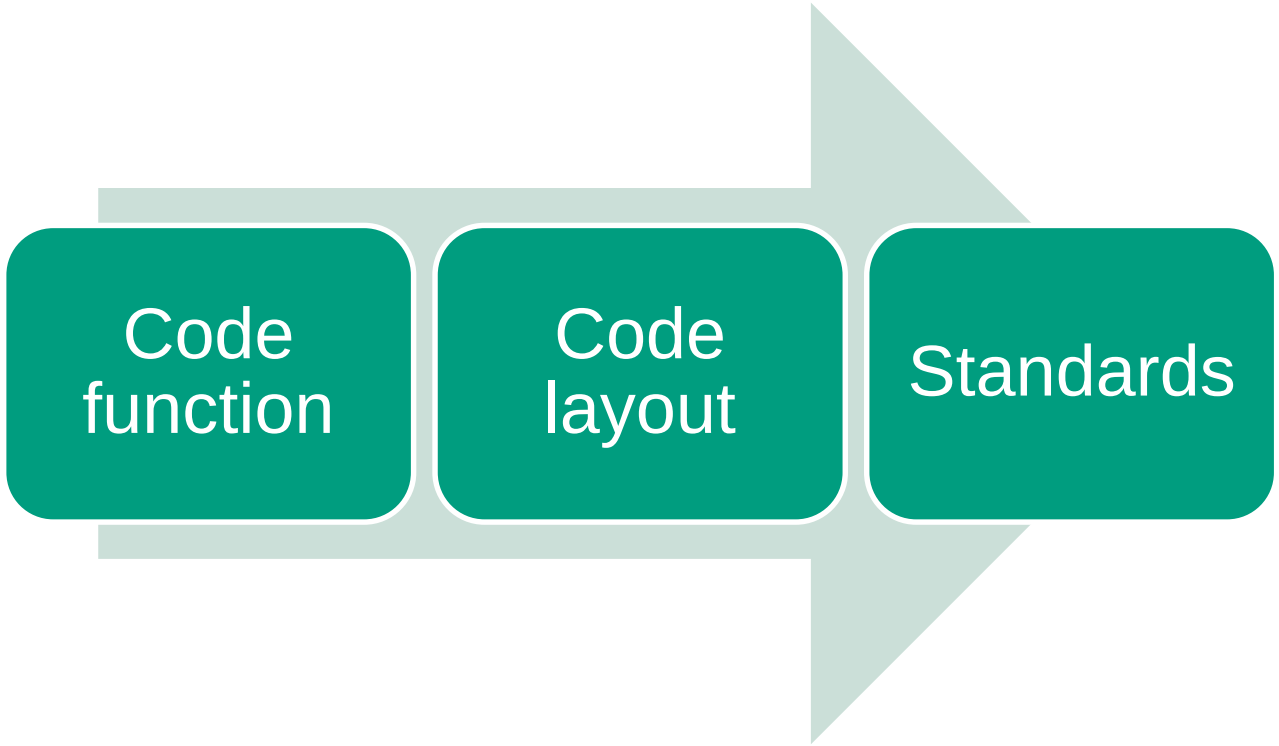
- Mockers

```python
def test_cell_procreation(mocker):
    # test no babies are born
    a = rd.randint(1, 50)
    b = rd.randint(1, 50)
    A = lowland((a, b))
    pop_size = 20
    pop = [{'species': 'Carnivore',
            'age': 5,
            'weight': 20}
           for _ in range(pop_size)]
    A.add_population(pop)
    mocker.patch('biosim.animals.animal.birth', return_value=None)
    A.breeding()

    assert len(A.carn) + len(A.herb) == pop_size
```

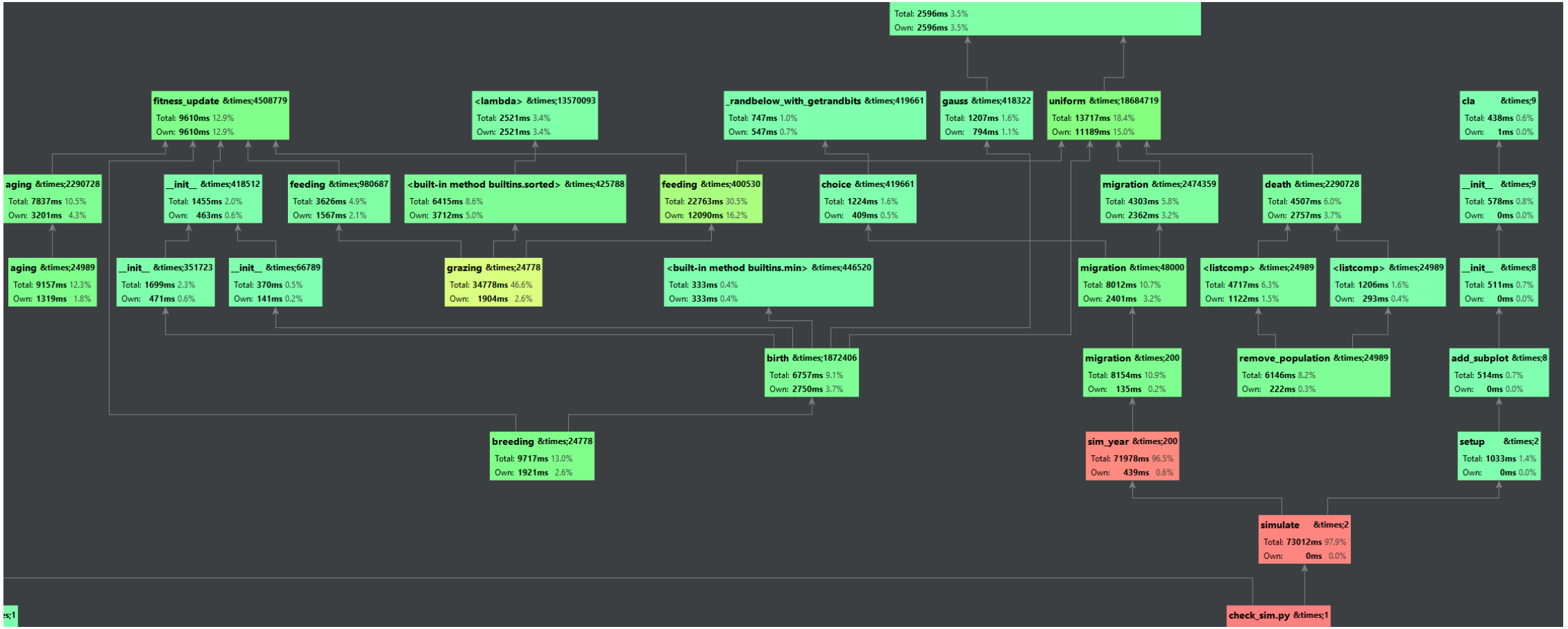
Results

```
Name                                                  Stmts   Miss  Cover
-------------------------------------------------------------------------
.tox\py38\Lib\site-packages\biosim\__init__.py            3      0   100%
.tox\py38\Lib\site-packages\biosim\animals.py           122      7    94%
.tox\py38\Lib\site-packages\biosim\biome.py             102     17    83%
.tox\py38\Lib\site-packages\biosim\biosim.py             64      6    91%
.tox\py38\Lib\site-packages\biosim\island.py            116     16    86%
.tox\py38\Lib\site-packages\biosim\visualization.py     266    174    35%
-------------------------------------------------------------------------
TOTAL                                                   673    220    67%



========================================================================== 61 passed in 11.68s ========
_____________________________________________________________________ summary _____________
  py38: commands succeeded
```

Gitlab automated testing



Profiling and optimization



Documentation

Short and to the point!

Documentation

- Param and return
- Mathematical formulas
- Cross-references
- Code examples
- ReStructured text – Pro's and cons

birth(*n_animals*) [source]

Calculating the probability of an individual animal giving birth, and whether it should happen or not. If the weight of the mother is less than the weight of the child + the standard deviation of birthweight, the birth will not occur. Otherwise, the probability of an animal giving birth is:

$$min(1, \gamma \times \Phi \times (N-1)$$

Where Φ is the animal fitness and N is the number of animals in the cell of the same species. The rest are constants from the animals' subclass

Parameters: **n_animals** – Integer representing the number of animals on the island.

Return child object/None: returning a child object if birth is given, or None if not.

add_population(*population*) [source]

Add a population to a given locations on the island. Runs the **island.island.add_population()** function.

This is an example of histogram specifications for creation of an object in the BioSim class. If image directory is provided, the standard value for image name and format is 'dv' and .png.

```
sim = BioSim(island_map, ini_herbs, seed=1, vis_years=1,
             hist_specs={'fitness': {'max': 1.0, 'delta': 0.05},
                         'age': {'max': 60.0, 'delta': 2},
                         'weight': {'max': 60, 'delta': 2}},
                         img_dir='results3', img_base='sample',
                         img_years=1, log_file='res.txt')
```

Improvements and Future work

Sky is the limit!

Improvements in our code - hindsight

Optimization

- Reduce number of for-loops
- Visualization.py – More efficient, readable and reusable code

Testing

- Island creation with map string – created as specified
- Mocker test for dying and birth on cell level
- Z-Tests – more tests on probability behaviour

Future work

- Animal genders
- Landscape changes over time
- Variation in regeneration of fodder
- Visualization – more advanced graphs and more information
- Application level logging – easier debugging
- Weather – Possibility for flood and drought in different cells

Questions?