NMBU

Norwegian University
of Life Sciences

BioSim T02

Ahmar Abbas
Gøran Sildnes Gedde-Dahl

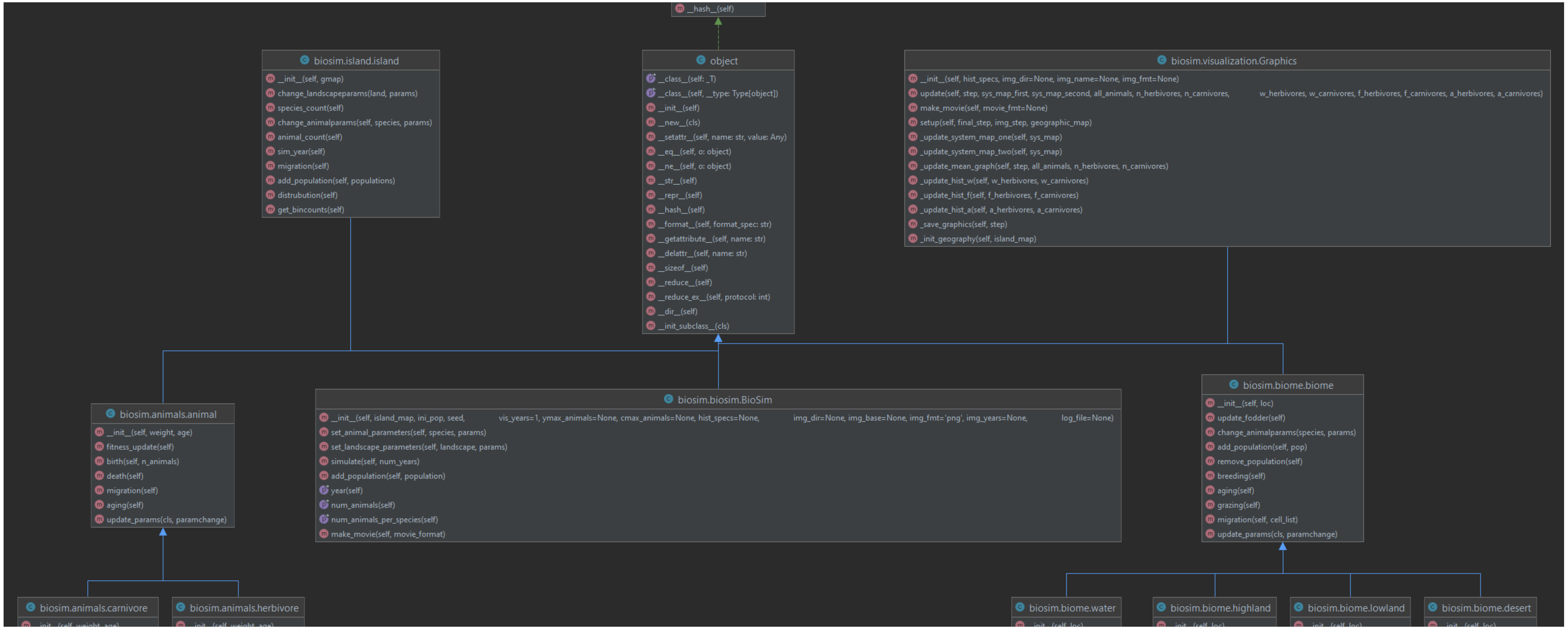
Agenda

- Design and Implementation
- Quality Assurance
- Documentation
- Improvements

Design and Implementation

Keep-it-simple!

Class Diagram

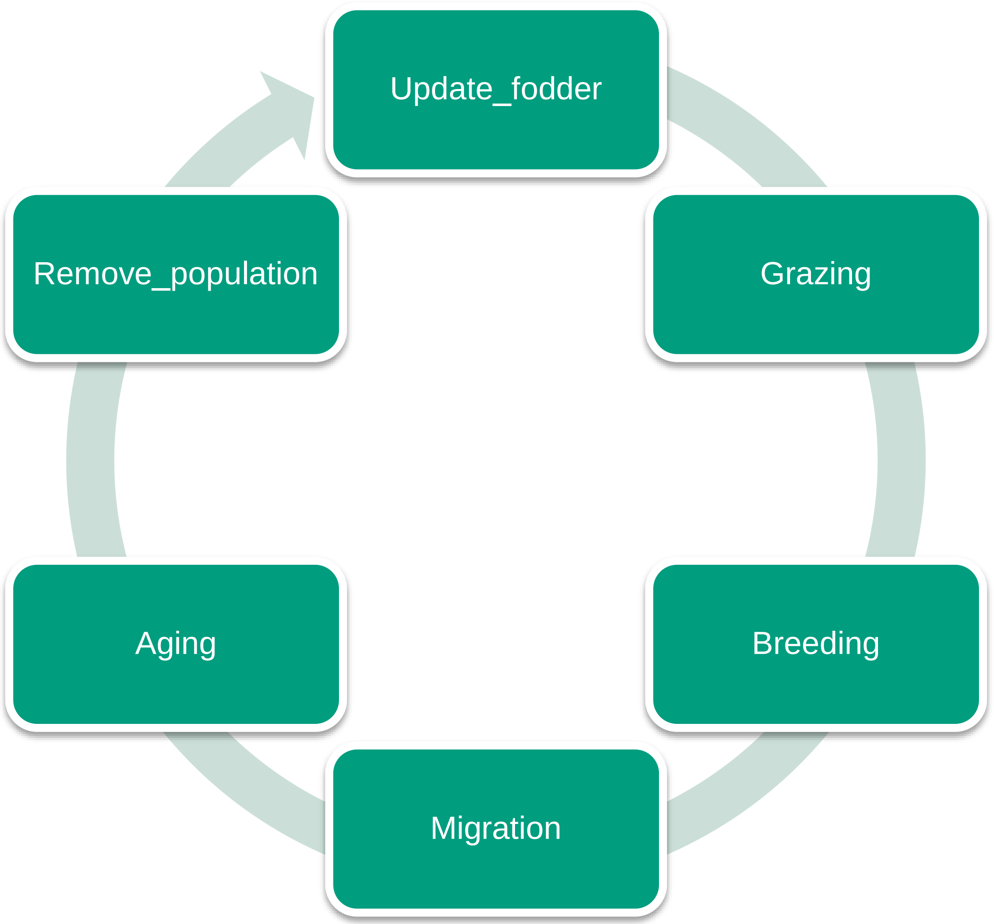

__hash__(self)
object
__class__(self: _T)
__class__(self, __type: Type[object])
__init__(self)
__new__(cls)
__setattr__(self, name: str, value: Any)
__eq__(self, o: object)
__ne__(self, o: object)
__str__(self)
__repr__(self)
__hash__(self)
__format__(self, format_spec: str)
__getattribute__(self, name: str)
__delattr__(self, name: str)
__sizeof__(self)
__reduce__(self)
__reduce_ex__(self, protocol: int)
__dir__(self)
__init_subclass__(cls)
biosim.island.island
__init__(self, gmap)
change_landscapeparams(land, params)
species_count(self)
change_animalparams(self, species, params)
animal_count(self)
sim_year(self)
migration(self)
add_population(self, populations)
distrubution(self)
get_bincounts(self)
biosim.visualization.Graphics
__init__(self, hist_specs, img_dir=None, img_name=None, img_fmt=None)
update(self, step, sys_map_first, sys_map_second, all_animals, n_herbivores, n_carnivores, w_herbivores, w_carnivores, f_herbivores, f_carnivores, a_herbivores, a_carnivores)
make_movie(self, movie_fmt=None)
setup(self, final_step, img_step, geographic_map)
_update_system_map_one(self, sys_map)
_update_system_map_two(self, sys_map)
_update_mean_graph(self, step, all_animals, n_herbivores, n_carnivores)
_update_hist_w(self, w_herbivores, w_carnivores)
_update_hist_f(self, f_herbivores, f_carnivores)
_update_hist_a(self, a_herbivores, a_carnivores)
_save_graphics(self, step)
_init_geography(self, island_map)
biosim.animals.animal
__init__(self, weight, age)
fitness_update(self)
birth(self, n_animals)
death(self)
migration(self)
aging(self)
update_params(cls, paramchange)
biosim.biosim.BioSim
__init__(self, island_map, ini_pop, seed, vis_years=1, ymax_animals=None, cmax_animals=None, hist_specs=None, img_dir=None, img_base=None, img_fmt='png', img_years=None, log_file=None)
set_animal_parameters(self, species, params)
set_landscape_parameters(self, landscape, params)
simulate(self, num_years)
add_population(self, population)
year(self)
num_animals(self)
num_animals_per_species(self)
make_movie(self, movie_format)
biosim.biome.biome
__init__(self, loc)
update_fodder(self)
change_animalparams(species, params)
add_population(self, pop)
remove_population(self)
breeding(self)
aging(self)
grazing(self)
migration(self, cell_list)
update_params(cls, paramchange)
biosim.animals.carnivore
biosim.animals.herbivore
biosim.biome.water
biosim.biome.highland
biosim.biome.lowland
biosim.biome.desert

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()

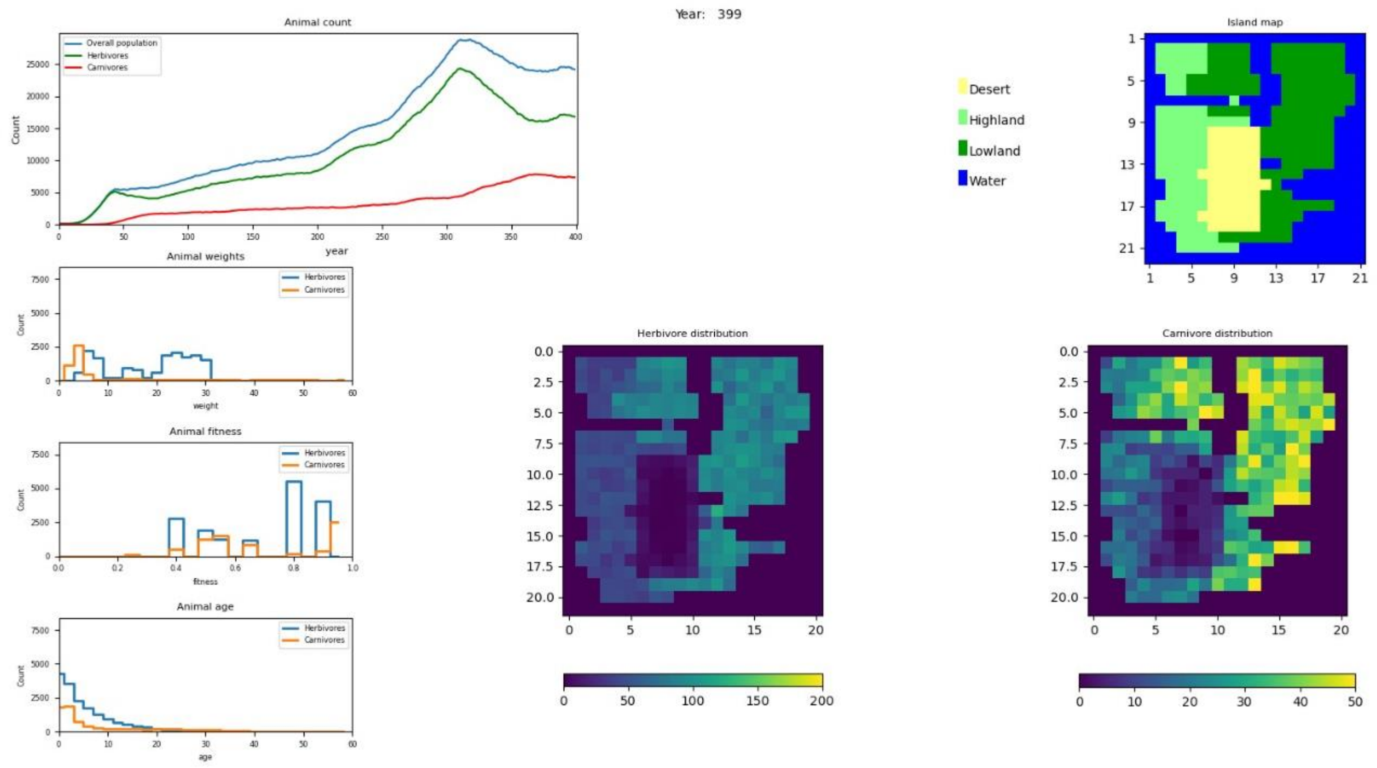
Annual Cycle

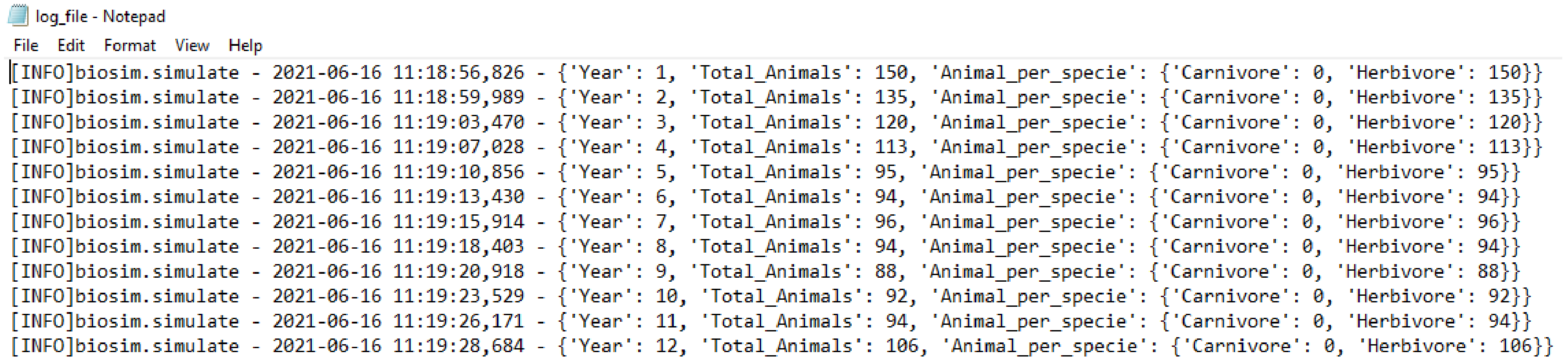
- Each function is called only once a year in each Island cell.
- All functions are implemented in Biome class except for migration.
- Actual implementation of migration is also in Biome class but a list of possible migration destinations is evaluated and passed through Island class.



Visualization and Logging

- Simple and informative layout
- Simulation level logging





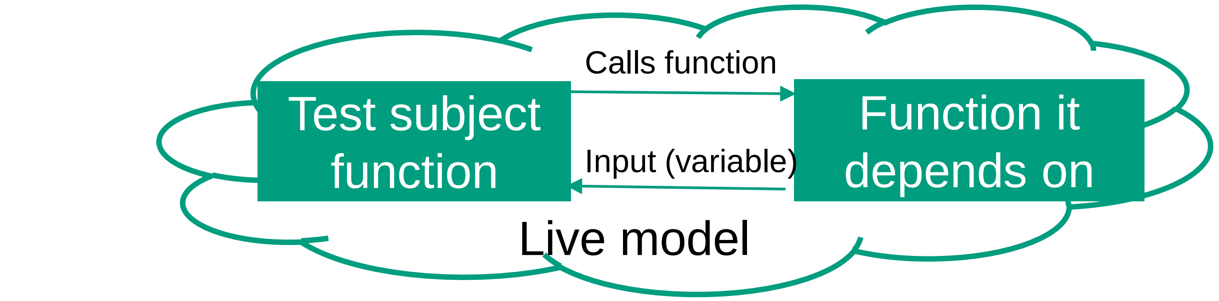
```
[INFO]biosim.simulate - 2021-06-16 11:18:56,826 - {'Year': 1, 'Total_Animals': 150, 'Animal_per_specie': {'Carnivore': 0, 'Herbivore': 150}}
[INFO]biosim.simulate - 2021-06-16 11:18:59,989 - {'Year': 2, 'Total_Animals': 135, 'Animal_per_specie': {'Carnivore': 0, 'Herbivore': 135}}
[INFO]biosim.simulate - 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}}
```

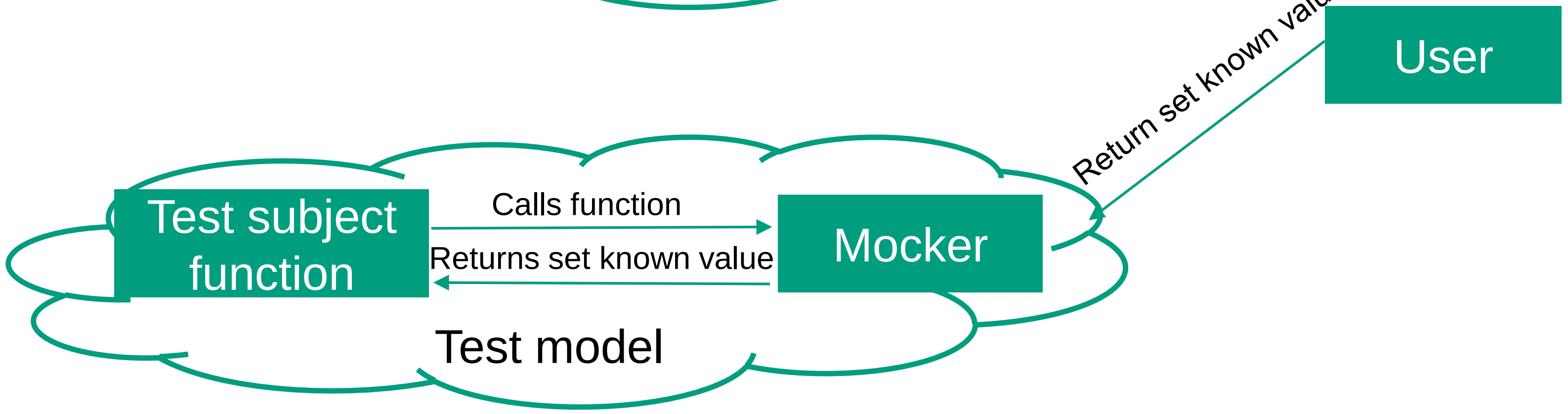
Quality Assurance

Quality over quantity always!

Testing

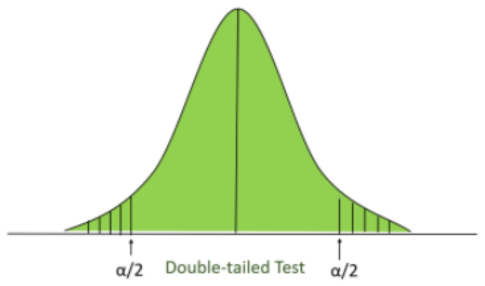
- Assertions
  - Object creation
  - Functions





- Mockers

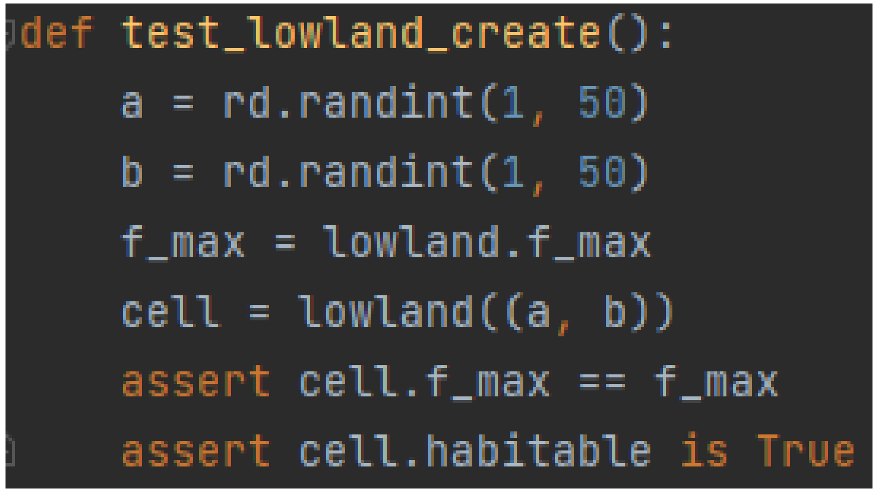
- Statistical tests - Scipy stat binom_test function – two-tailed – null-hypothesis



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

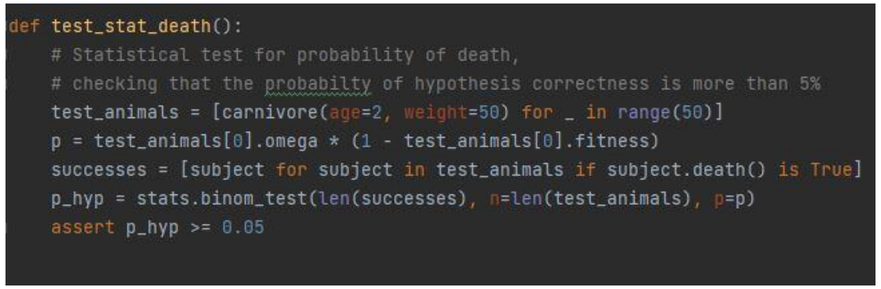
Testing Approach

- Unit testing



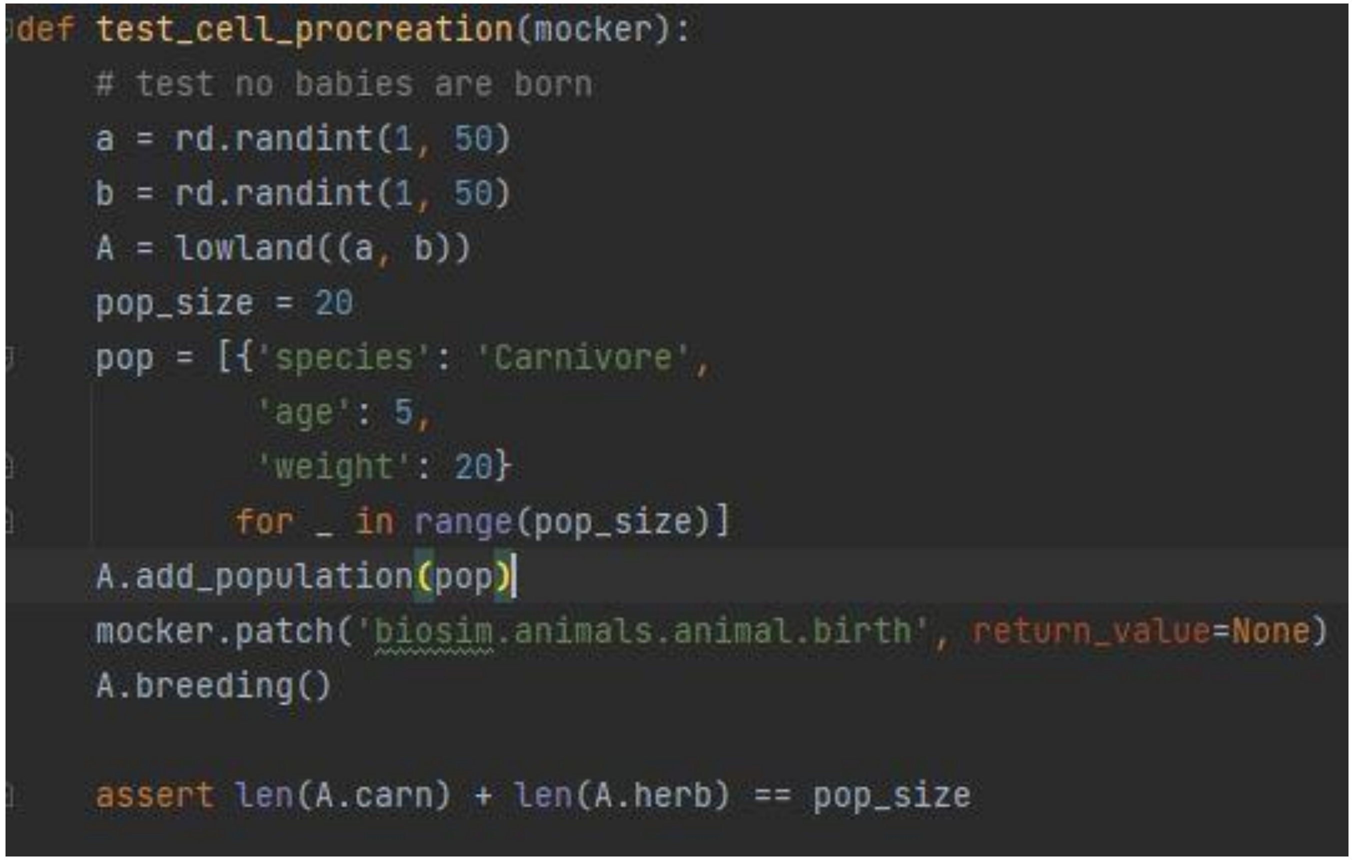
```
def test_lowland_create():
    a = rd.randint(1, 50)
    b = rd.randint(1, 50)
    f_max = lowland.f_max
    cell = lowland((a, b))
    assert cell.f_max == f_max
    assert cell.habitable is True
```

- Statistical testing



```
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
```

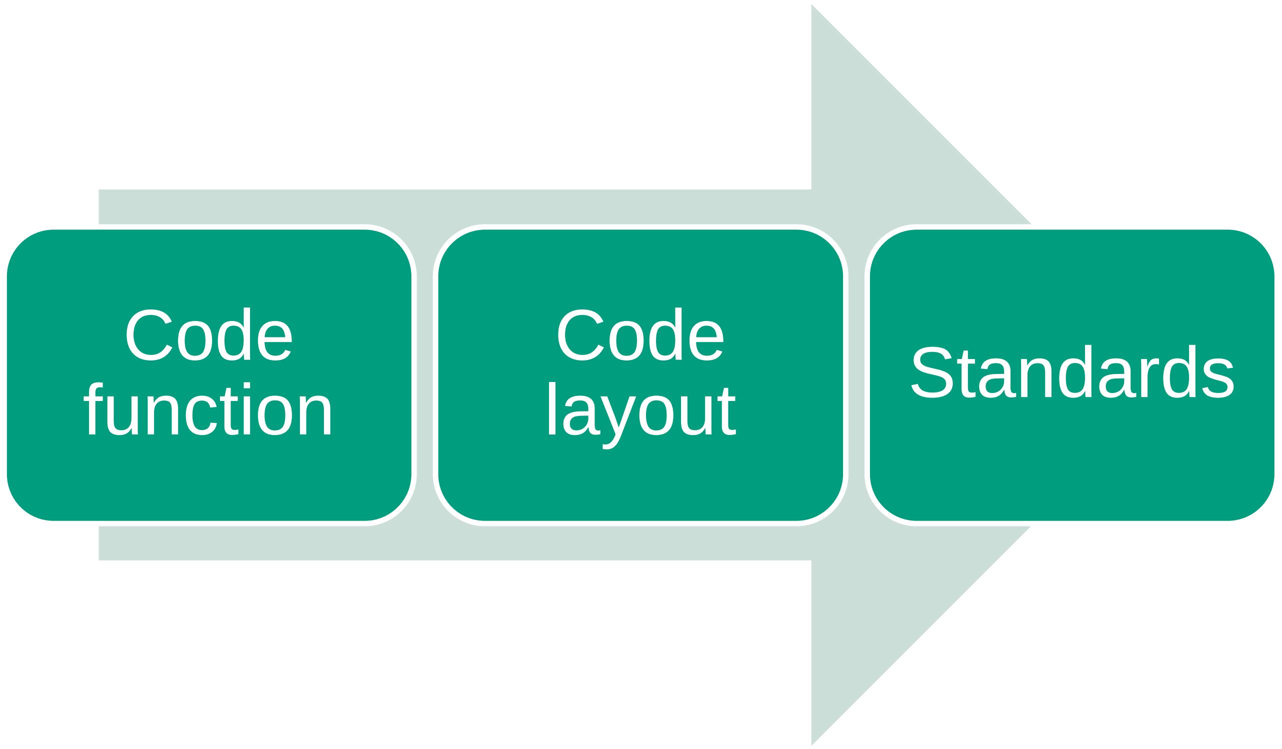
- Mockers



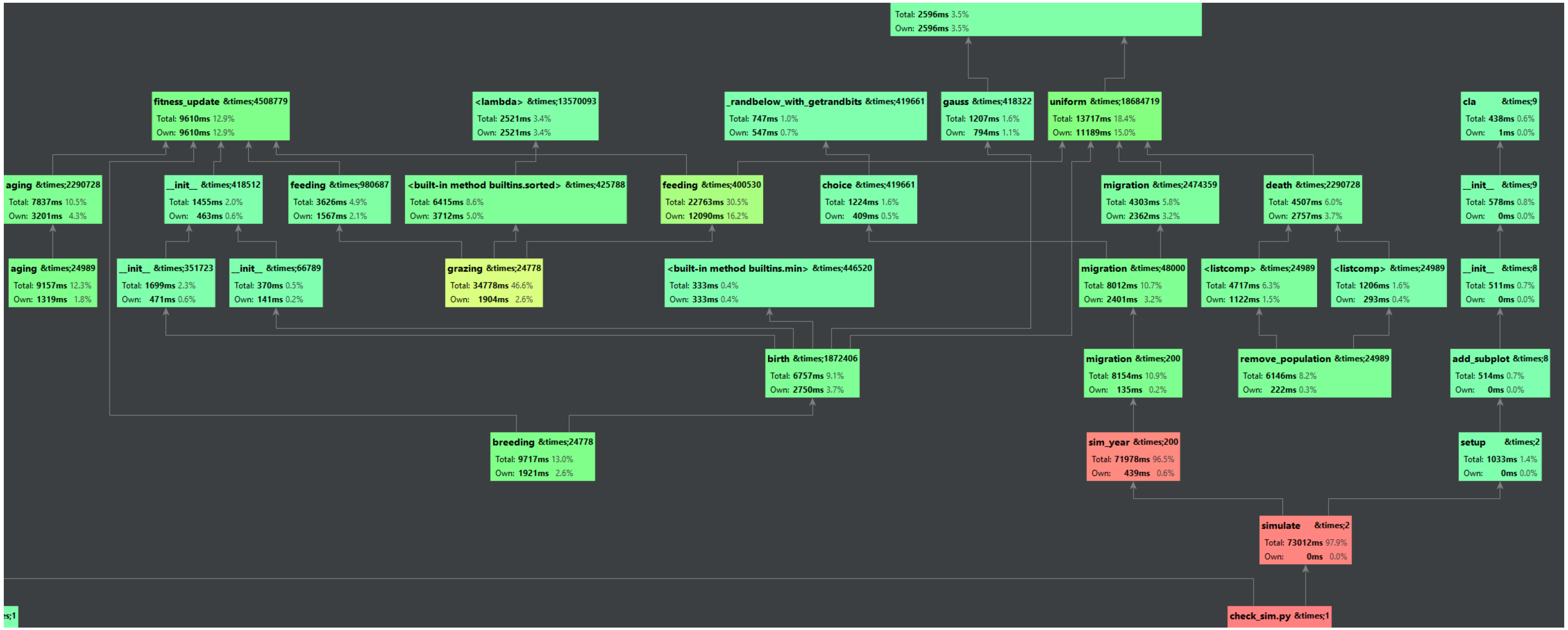
```
def test_cell_procreation(mocker):
    # test no babies are born
    a = rd.randint(1, 50)
    b = rd.randint(1, 50)
    A = lowland((a, b))
    pop_size = 20
    pop = [{'species': 'Carnivore',
            'age': 5,
            'weight': 20}
           for _ in range(pop_size)]
    A.add_population(pop)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

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

Gitlab automated testing



Profiling and optimization

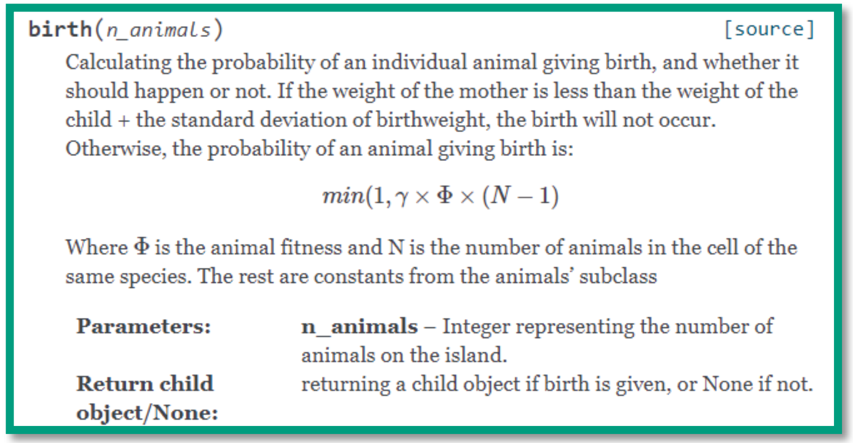

Total: 2596ms 3.5% Own: 2596ms 3.5%
fitness_update &times;4508779 Total: 9610ms 12.9% Own: 9610ms 12.9%
<lambda> &times;13570093 Total: 2521ms 3.4% Own: 2521ms 3.4%
_randbelow_with_getrandbits &times;419661 Total: 747ms 1.0% Own: 547ms 0.7%
gauss &times;418322 Total: 1207ms 1.6% Own: 794ms 1.1%
uniform &times;18684719 Total: 13717ms 18.4% Own: 11189ms 15.0%
cla &times;9 Total: 438ms 0.6% Own: 1ms 0.0%
aging &times;2290728 Total: 7837ms 10.5% Own: 3201ms 4.3%
__init__ &times;418512 Total: 1455ms 2.0% Own: 463ms 0.6%
feeding &times;980687 Total: 3626ms 4.9% Own: 1567ms 2.1%
<built-in method builtins.sorted> &times;425788 Total: 6415ms 8.6% Own: 3712ms 5.0%
feeding &times;400530 Total: 22763ms 30.5% Own: 12090ms 16.2%
choice &times;419661 Total: 1224ms 1.6% Own: 409ms 0.5%
migration &times;2474359 Total: 4303ms 5.8% Own: 2362ms 3.2%
death &times;2290728 Total: 4507ms 6.0% Own: 2757ms 3.7%
__init__ &times;9 Total: 578ms 0.8% Own: 0ms 0.0%
aging &times;24989 Total: 9157ms 12.3% Own: 1319ms 1.8%
__init__ &times;351723 Total: 1699ms 2.3% Own: 471ms 0.6%
__init__ &times;66789 Total: 370ms 0.5% Own: 141ms 0.2%
grazing &times;24778 Total: 34778ms 46.6% Own: 1904ms 2.6%
<built-in method builtins.min> &times;446520 Total: 333ms 0.4% Own: 333ms 0.4%
migration &times;48000 Total: 8012ms 10.7% Own: 2401ms 3.2%
<listcomp> &times;24989 Total: 4717ms 6.3% Own: 1122ms 1.5%
<listcomp> &times;24989 Total: 1206ms 1.6% Own: 293ms 0.4%
__init__ &times;8 Total: 511ms 0.7% Own: 0ms 0.0%
birth &times;1872406 Total: 6757ms 9.1% Own: 2750ms 3.7%
migration &times;200 Total: 8154ms 10.9% Own: 135ms 0.2%
remove_population &times;24989 Total: 6146ms 8.2% Own: 222ms 0.3%
add_subplot &times;8 Total: 514ms 0.7% Own: 0ms 0.0%
breeding &times;24778 Total: 9717ms 13.0% Own: 1921ms 2.6%
sim_year &times;200 Total: 71978ms 96.5% Own: 439ms 0.6%
setup &times;2 Total: 1033ms 1.4% Own: 0ms 0.0%
simulate &times;2 Total: 73012ms 97.9% Own: 0ms 0.0%
s;1
check_sim.py &times;1

Documentation

Short and to the point!

Documentation

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



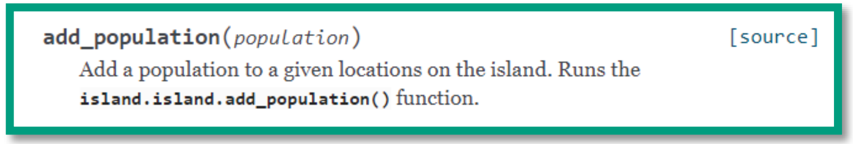
birth(*n_animals*) [source]

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:

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

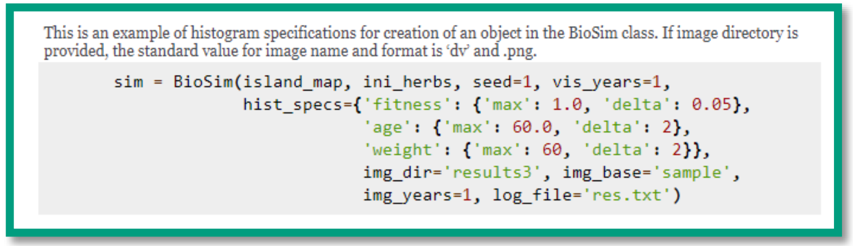
Where Φ is the animal fitness and N is the number of animals in the cell of the same species. The rest are constants from the animals' subclass

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?