BioSim

By -

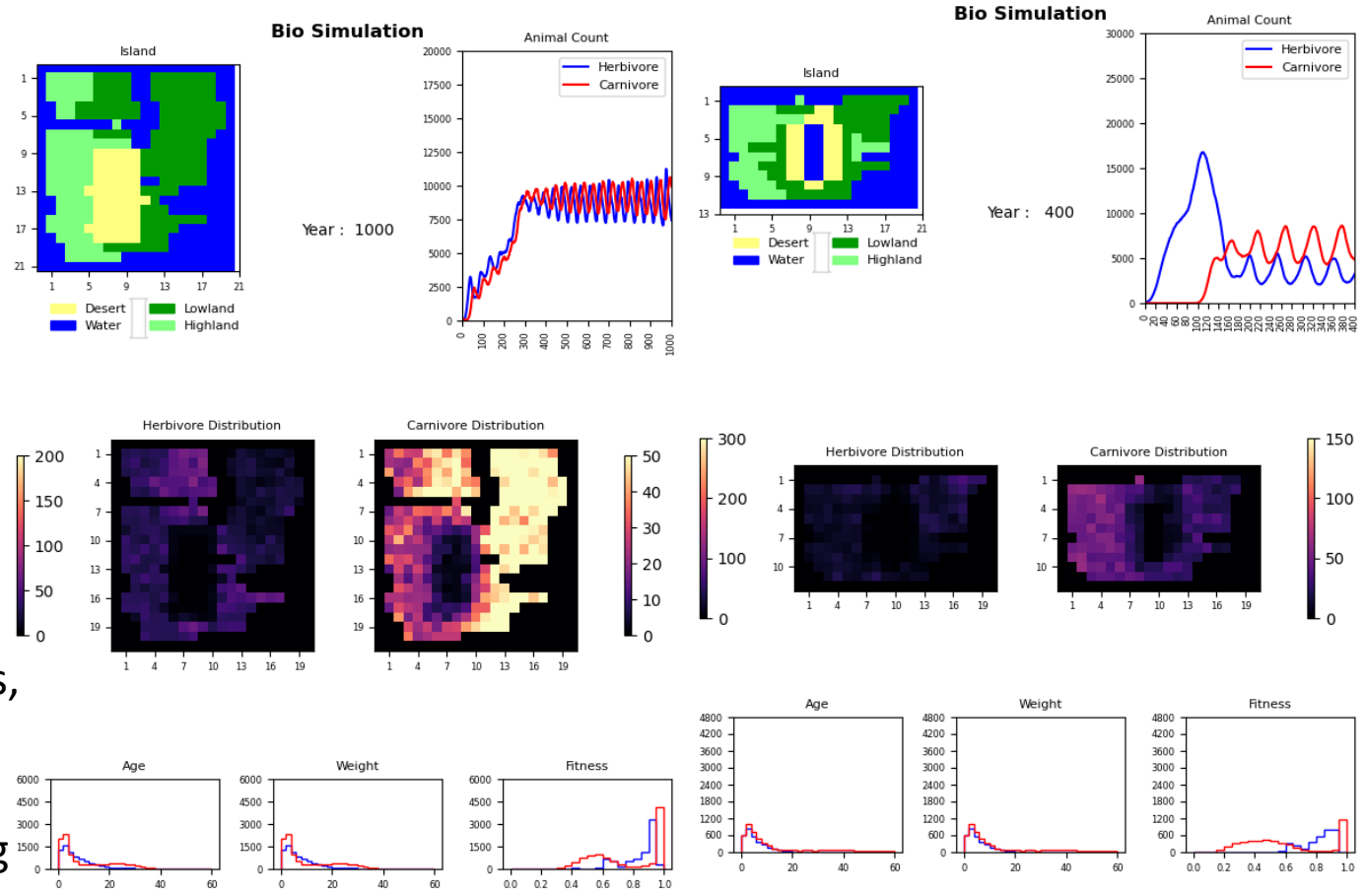
Aditya Dey

Olutomi Samuel Okubadejo

Introduction

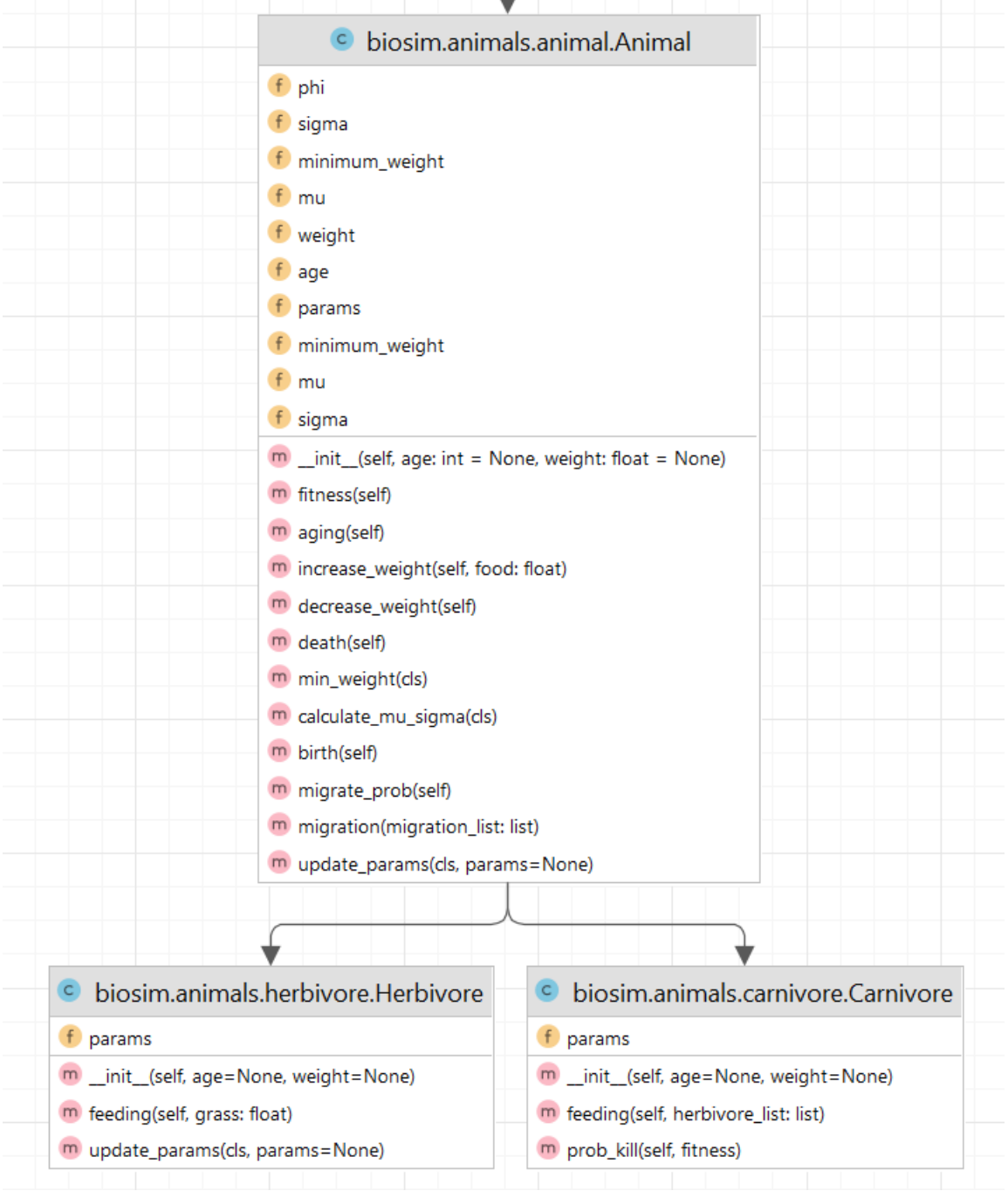
BioSim package provides:

- view of Island Map.
- Animal count of Herbivores and Carnivores.
- Heatmap distribution of both animals.
- Age, weight and fitness histogram of both animals.
- Can perform simulation for 200 years in 46 secs, with visuals disabled.
- Animal Count y_axis can be adjusted by passing parameters and based on number of years of simulation, the x_axis adjusts automatically.

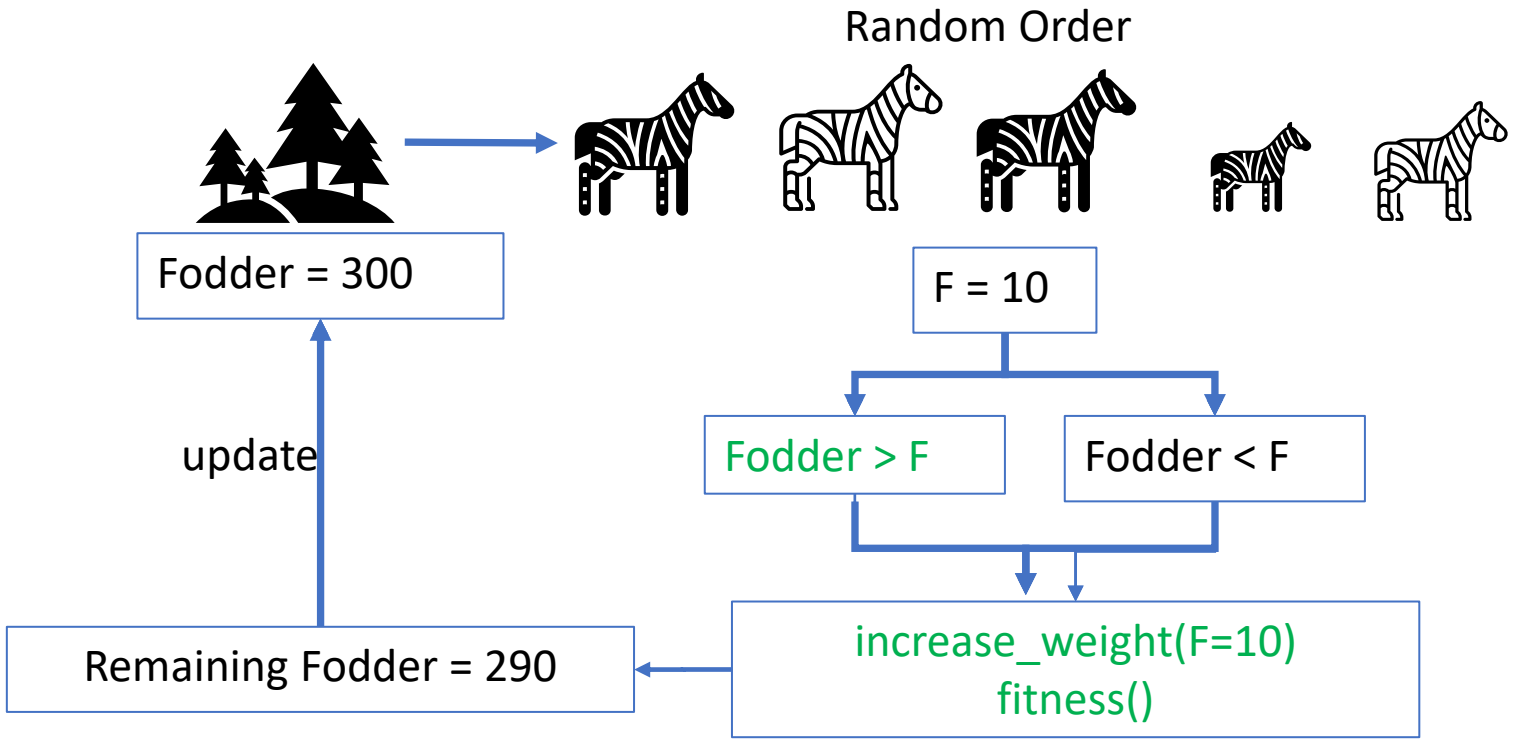


Animal Class Structure

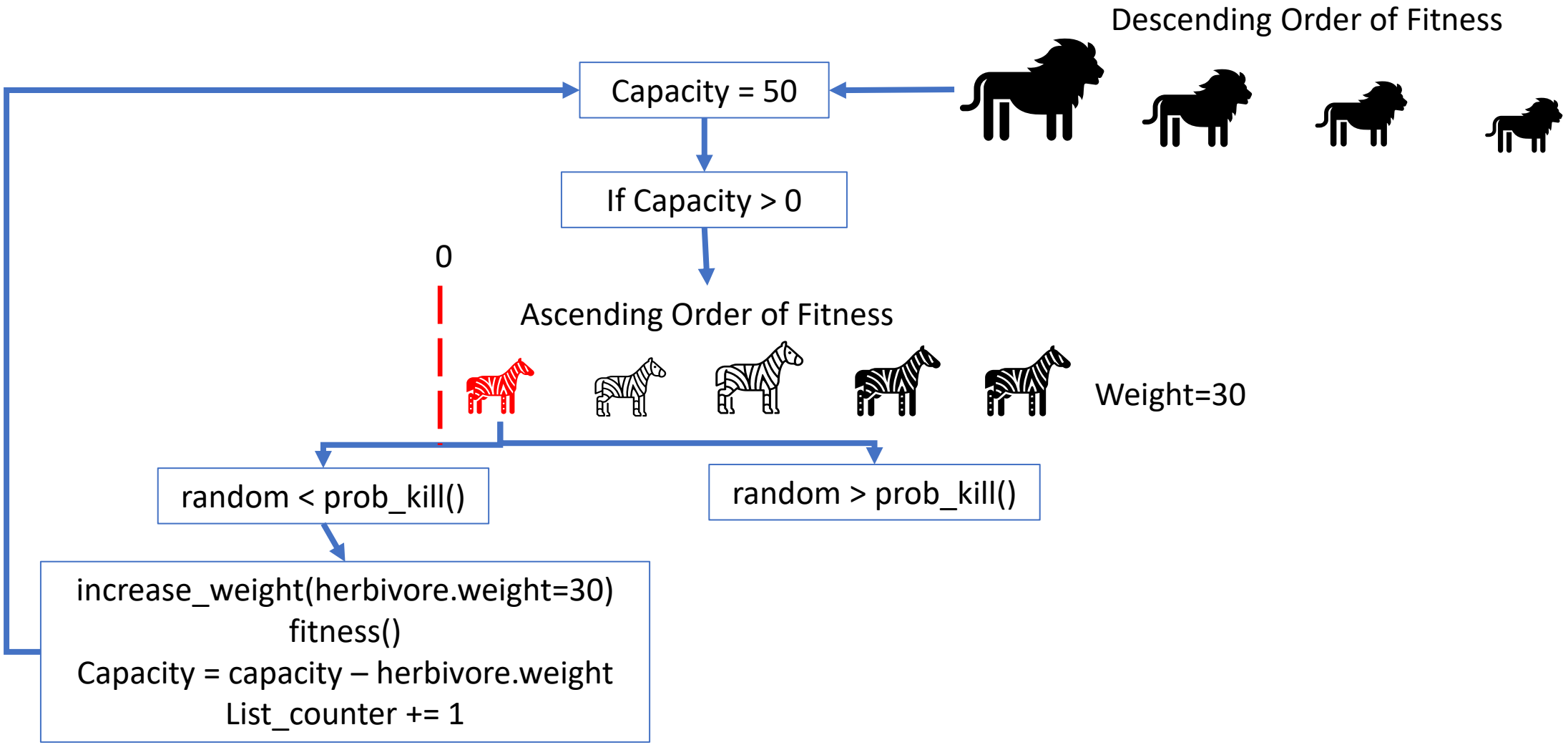
- Animal is the parent Class from which Herbivore and Carnivore Inherit functions and variables.
- A single instance of a class stores age, weight and phi(fitness) variables for each instance and share class variables like parameters, and (mu, sigma, and minimum weight) information for child birth.
- Herbivore and Carnivore have same function name feeding but operate differently.
  - Herbivore feeding input is float(fodder). It returns float(fodder remaining).
  - Carnivore feeding input is list(herbivore population). It returns list(survived herbivore population).
- Herbivore's update_params overrides Animal Class to further block user to update their DeltaPhiMax to anything other than None.



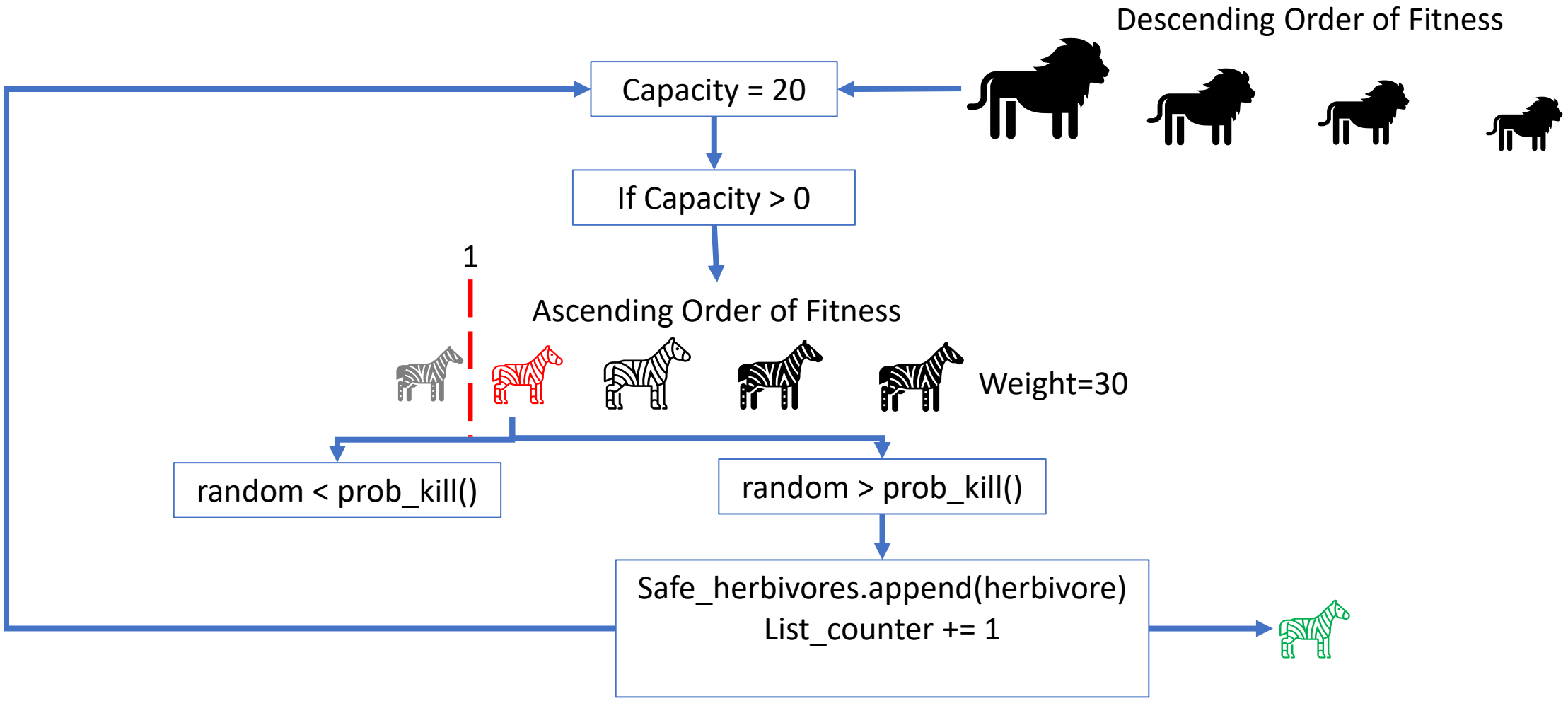
Herbivore Feeding:



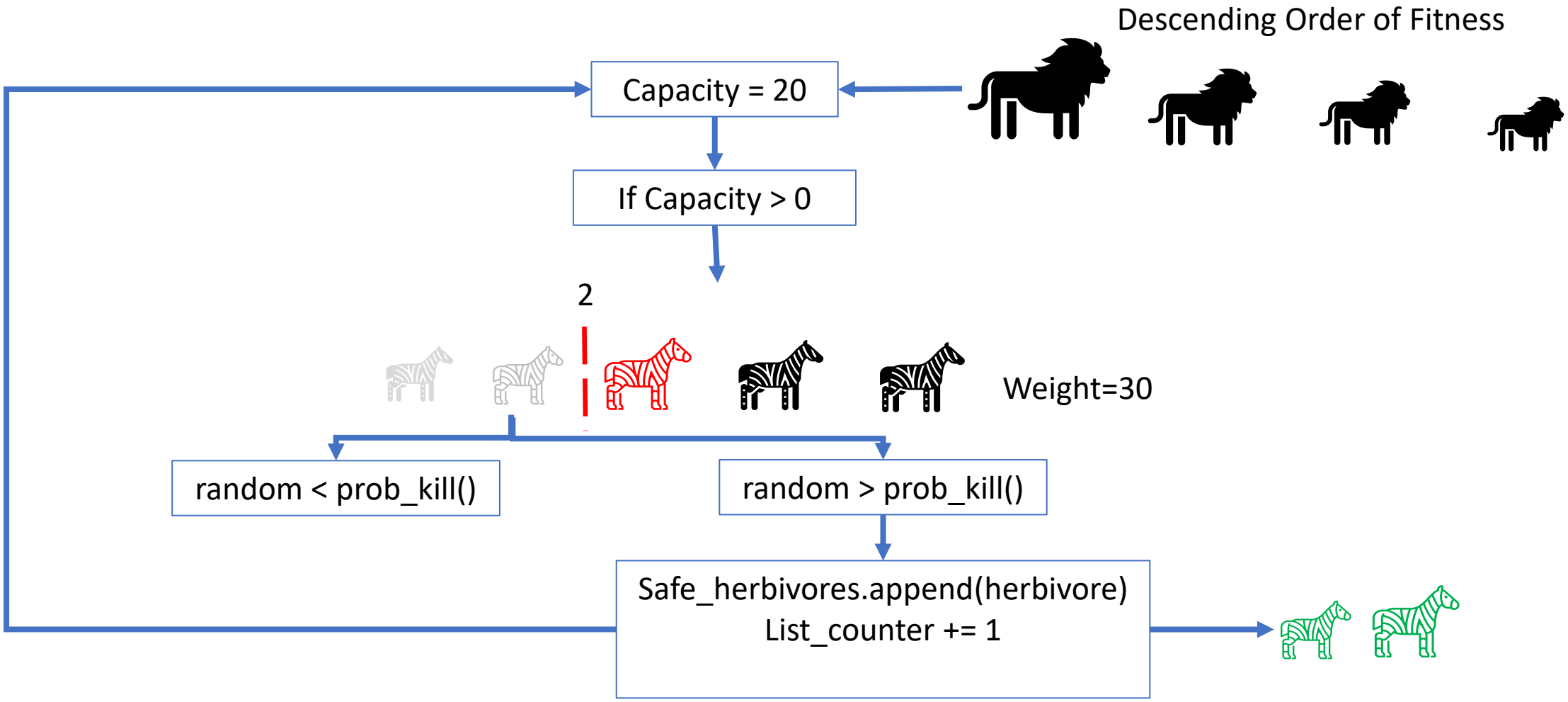
Carnivore Feeding:



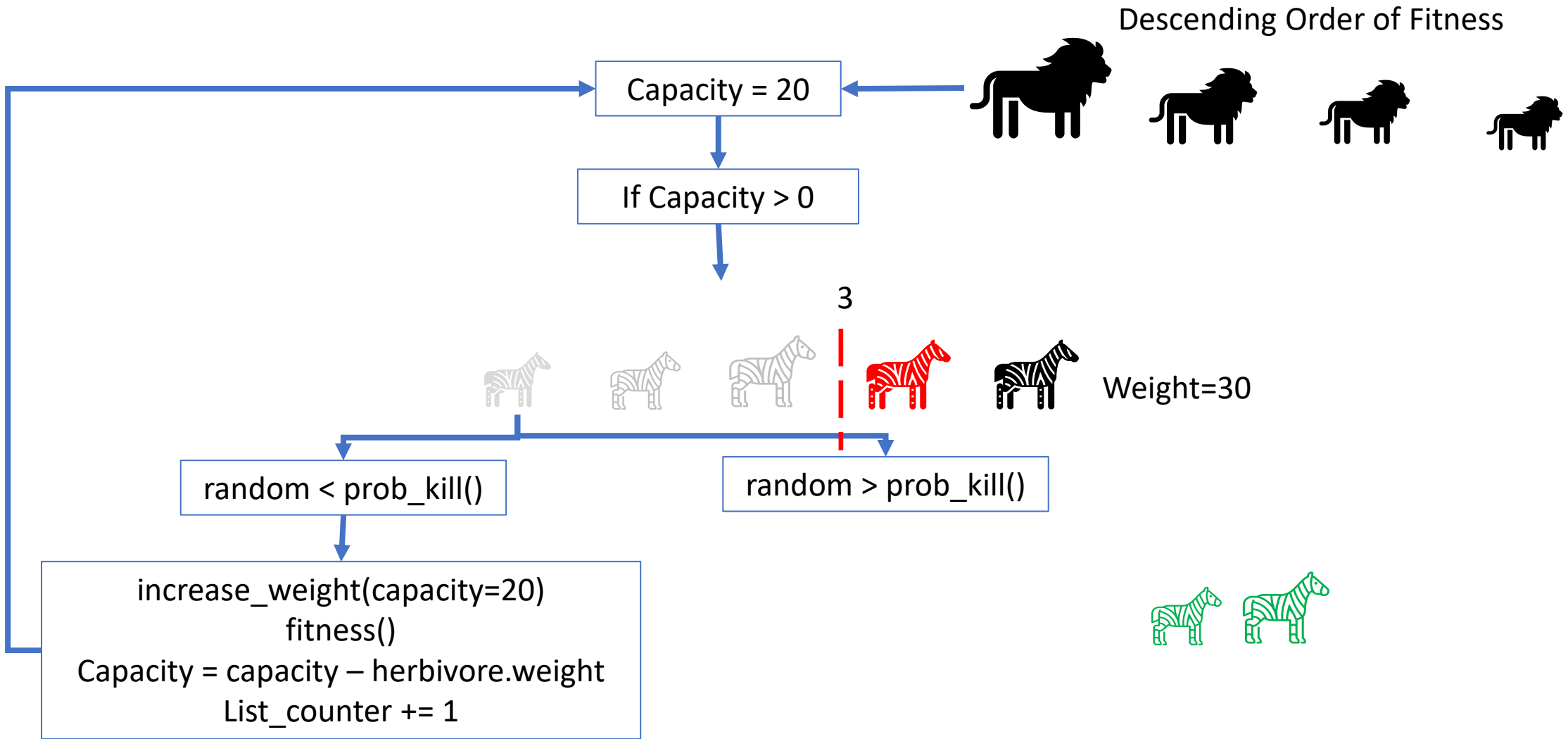
Carnivore Feeding(contd.):



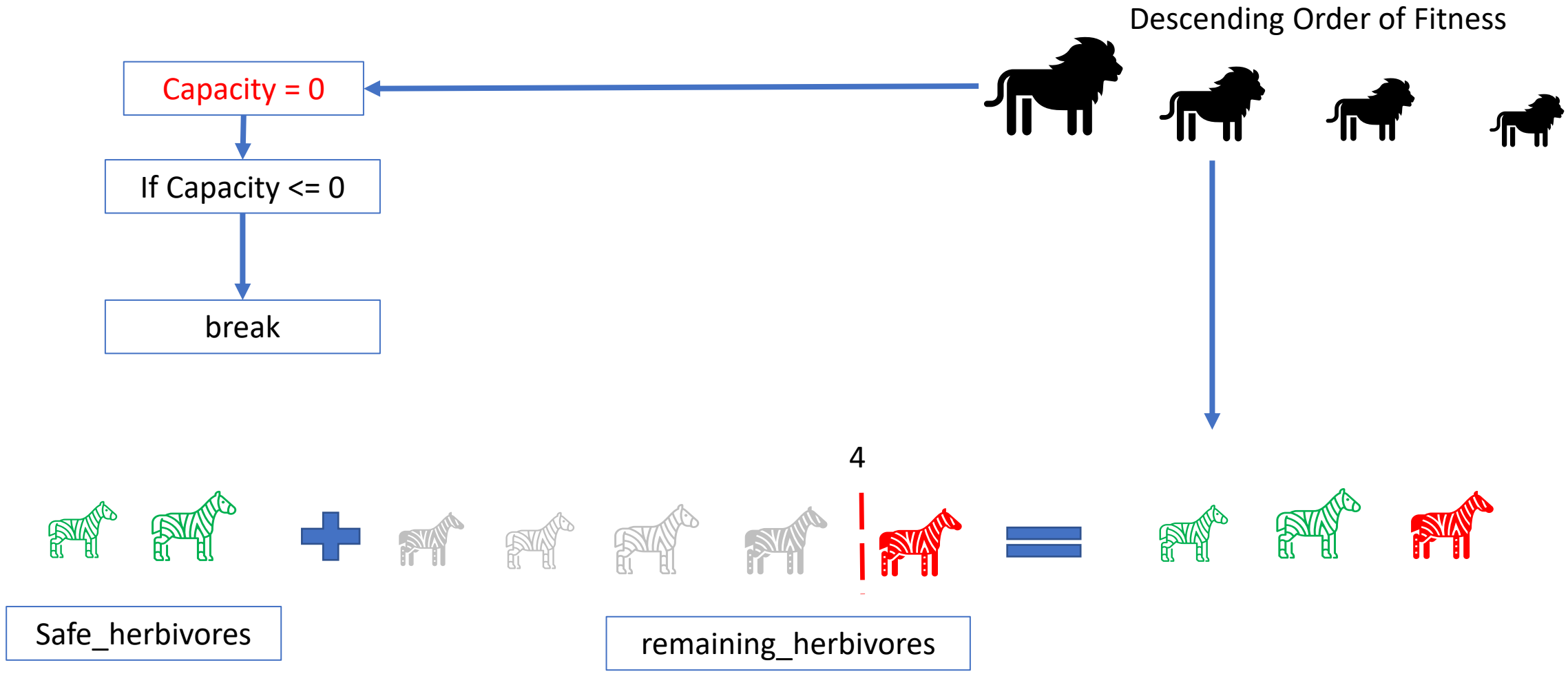
Carnivore Feeding(contd.):



Carnivore Feeding:



Carnivore Feeding(contd.):



Carnivore Feeding(contd.)

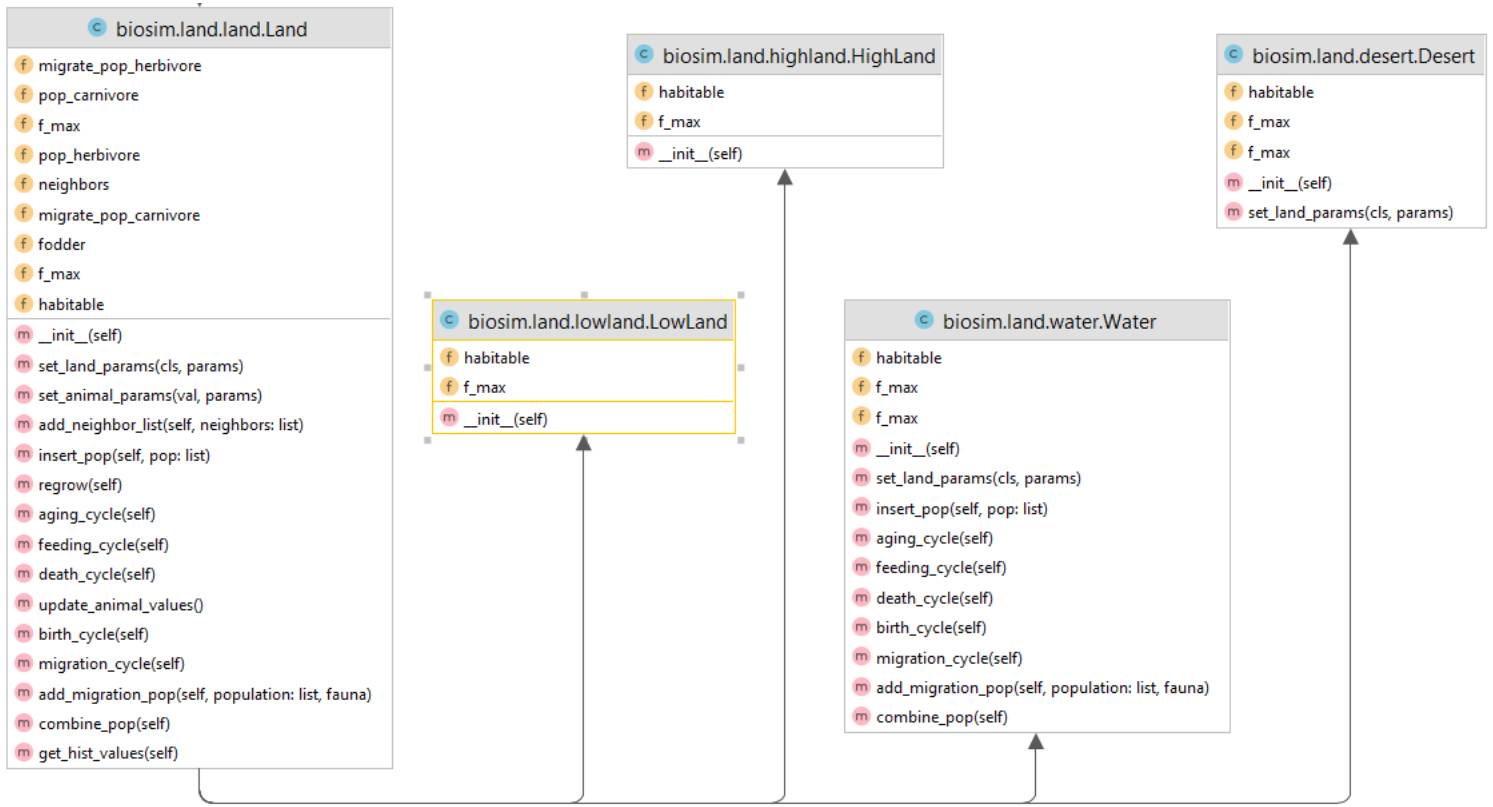
- Pop_herbivore = safe_herbivore + pop_herbivore[counter:]
  This ensures the next carnivore gets the list in descending order of fitness.

- Increase_weight() updates weight and recalculates fitness for next hunt.

- Iteration over pop_herbivore occurs until capacity becomes 0 or list is finished.

Land Class Structure

- Stores variables:
  - Pop_herbivore & pop_carnivore → list
  - Migrate_pop_herbivore & migrate_pop_carnivore → list.
  - “f_max” is class variable for maximum fodder.
  - “Fodder” is instance variable for tracking current fodder.
- “habitable” variable allows to determine if land is liveable.
- Desert Class overrides set_land_params() from Land Class to stop setting f_max anything other than zero.
- Similarly Water Class overrides
  - set_land_params() to stop setting f_max to anything other than None.
  - insert_pop() to raise ValueError.
  - All cycles are overridden to perform nothing.

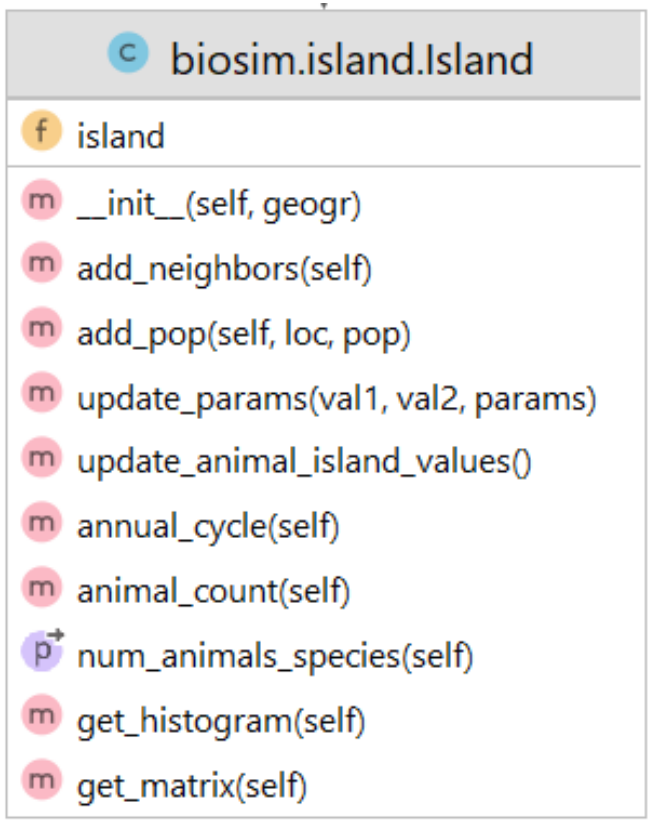


Island Class Structure

- Stores information as:
  - {(1,1):Lowland, (1,2): Highland,…….,(m,n): Land}

- add_neighbors() runs once when BioSim is initialized to update Land of its neighbours. Performed to reduce migration list passed to animals.

- Update_animal_island_values() runs once when simulate() is called, to calculate mu, sigma for lognormvariate and minimum weight for animal class.



Annual Cycle

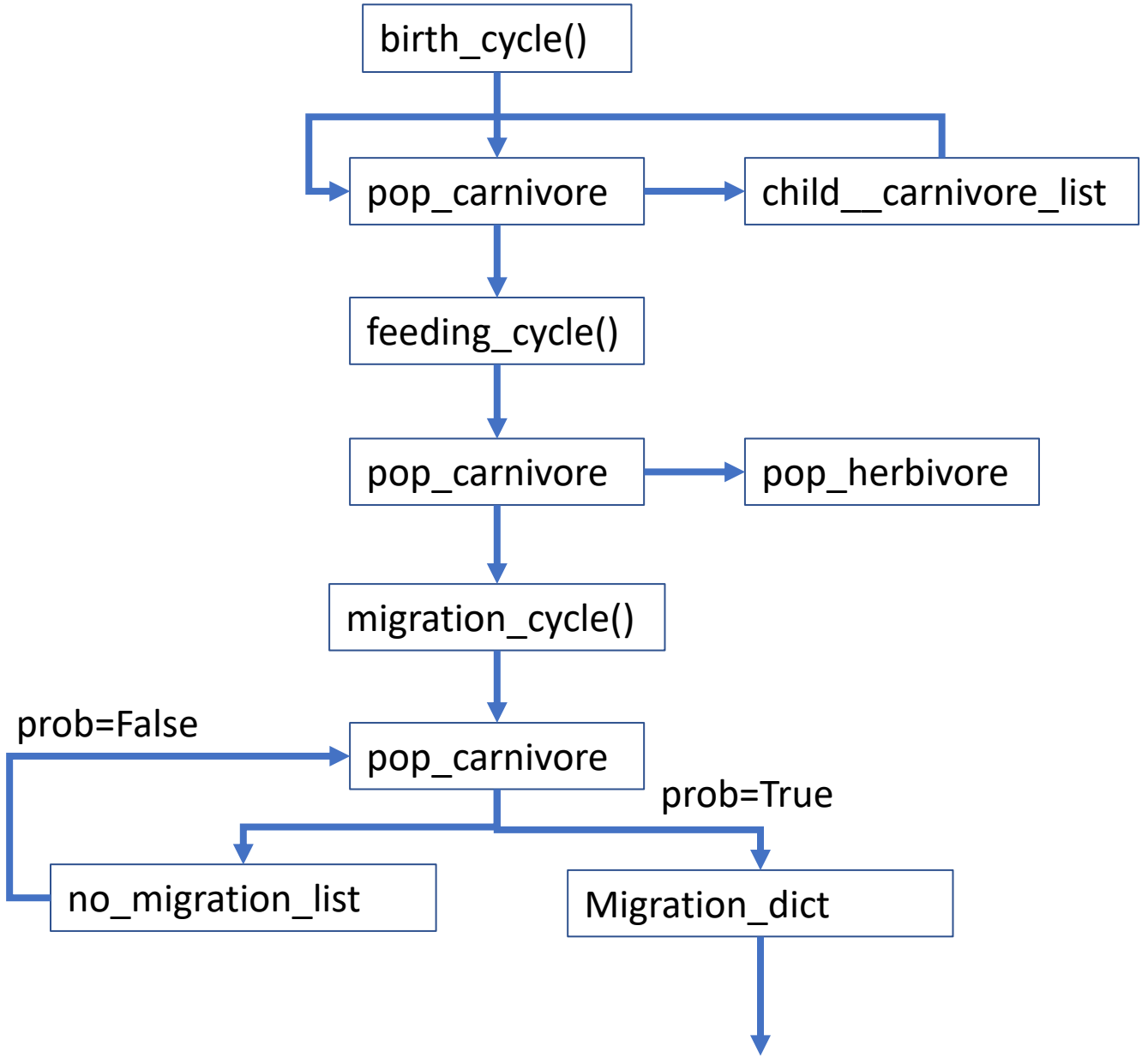
The Annual Cycle for a single Land instance in an Island →

Loop 1:

1. Birth_cycle() create a child_list and appends it back to original population list.

2. Feeding cycle as explained earlier.

3. Migration_cycle() for each animal class returns two values:
   - no_migration_list is updated as the current population.
   - Migration_dict is passed off to island class to perform migration, since island knows the locations.



Annual Cycle(contd.)

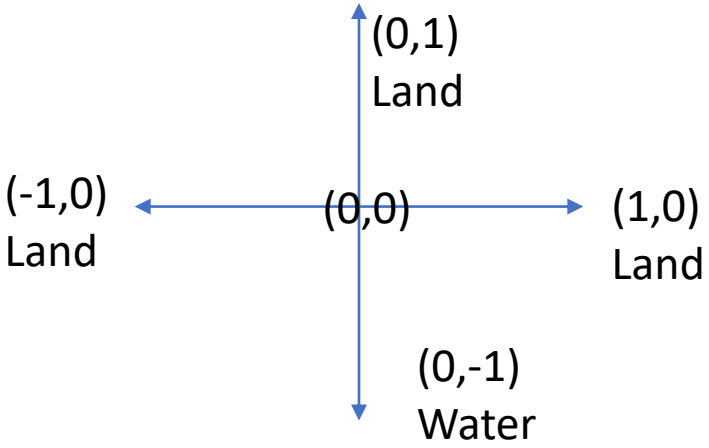
For location, population in migration_dict.items():

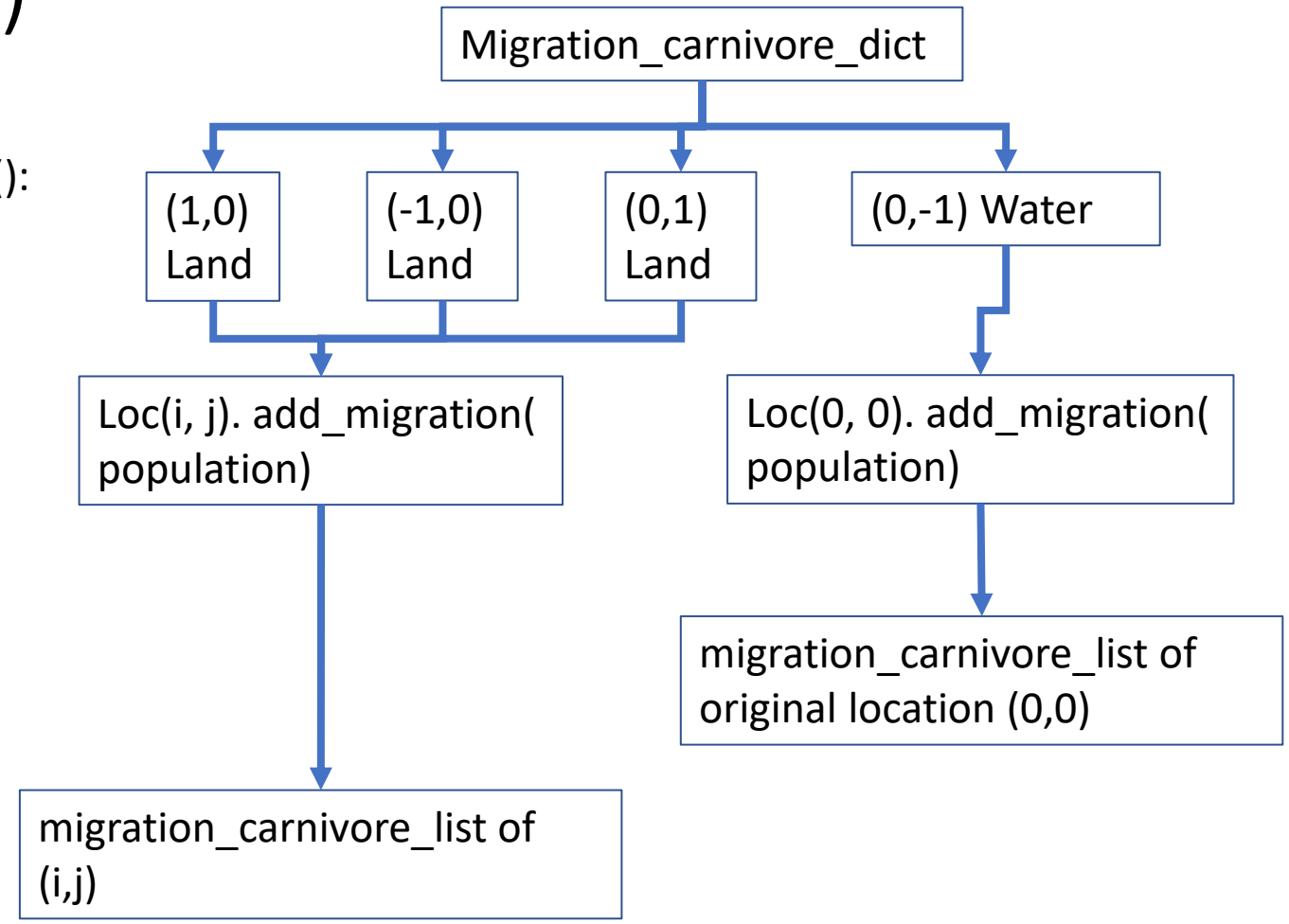
if loc.habitable is True:

island[location]. add_migration_pop
(population)

else:

island[origin]. add_migration_pop
(population)





Annual Cycle(contd.)

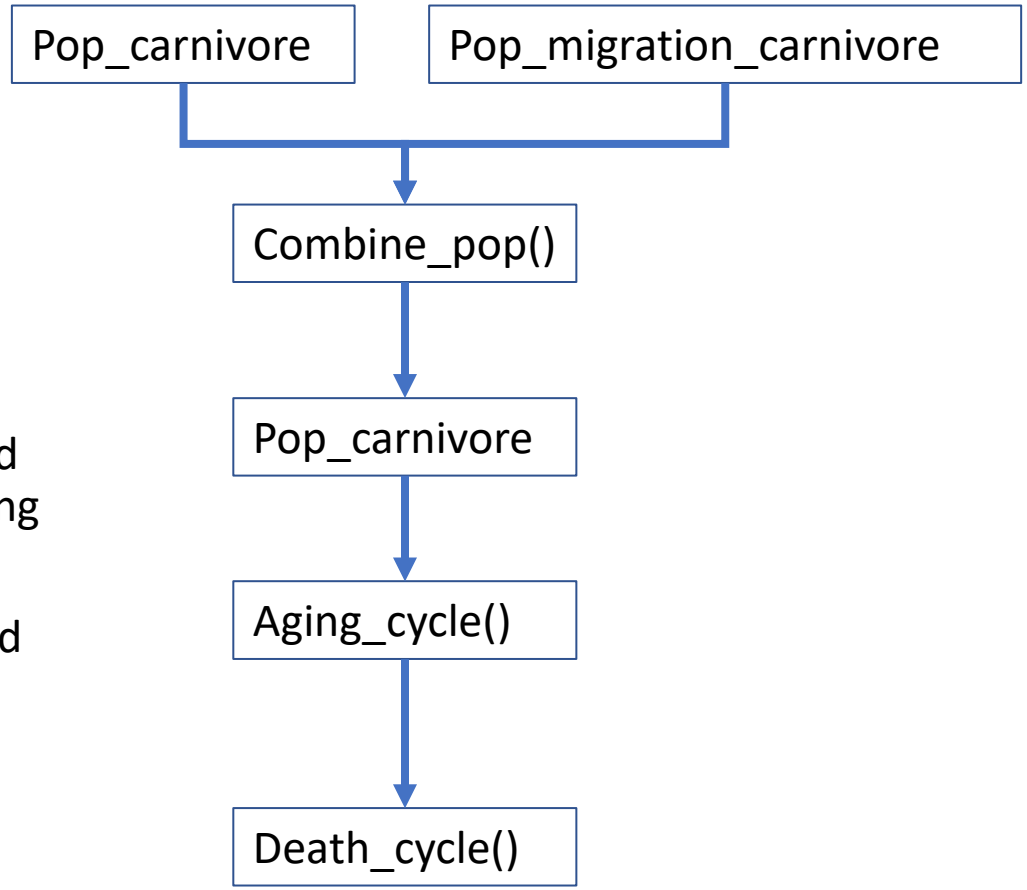
- **Loop_1:**
  - Birth
  - Feeding
  - Migration.

Loop_1 ensures migration is completed for all cells in the island and migrated animals are moved to migrate_pop to avoid double feeding or birth situations.

Aging avoided in Loop_1 to avoid no aging for someone who moved in backward cell.

- **Loop_2:**
  - combine pop →
    - Migration pop + original pop = original pop
  - Aging
  - Weight loss
  - Death

Loop_2 ensures both populations are combined and then rest of the cycles are done.



Optimization

- Without visual, when first run was done on Check_sim for 200 cycles, annual_cycle took 145 seconds.
    - Birth_cycle(51s / 31,400 times) → birthing(51s /62,800 times) → birth (42s / 2,287, 837 times) → lognormvariate(11s / 2, 287, 837 times)
    - Fitness (27s/8,786,672times)

- Reduced birth cycle time by making “minimum_weight” as a class method that will be called once during simulate to update the values and in birth_cycle that minimum_weight will be first criteria to check within loop to avoid excess time consumption inside loop. Also multiple times calculating “mu” and “sigma” also avoided by making it a class method in animal.
    - Birth_cycle(5-6s/31,400 times) → birthing(5-6s/ 62,800 times) → birth(~3s/ 360,313 times)

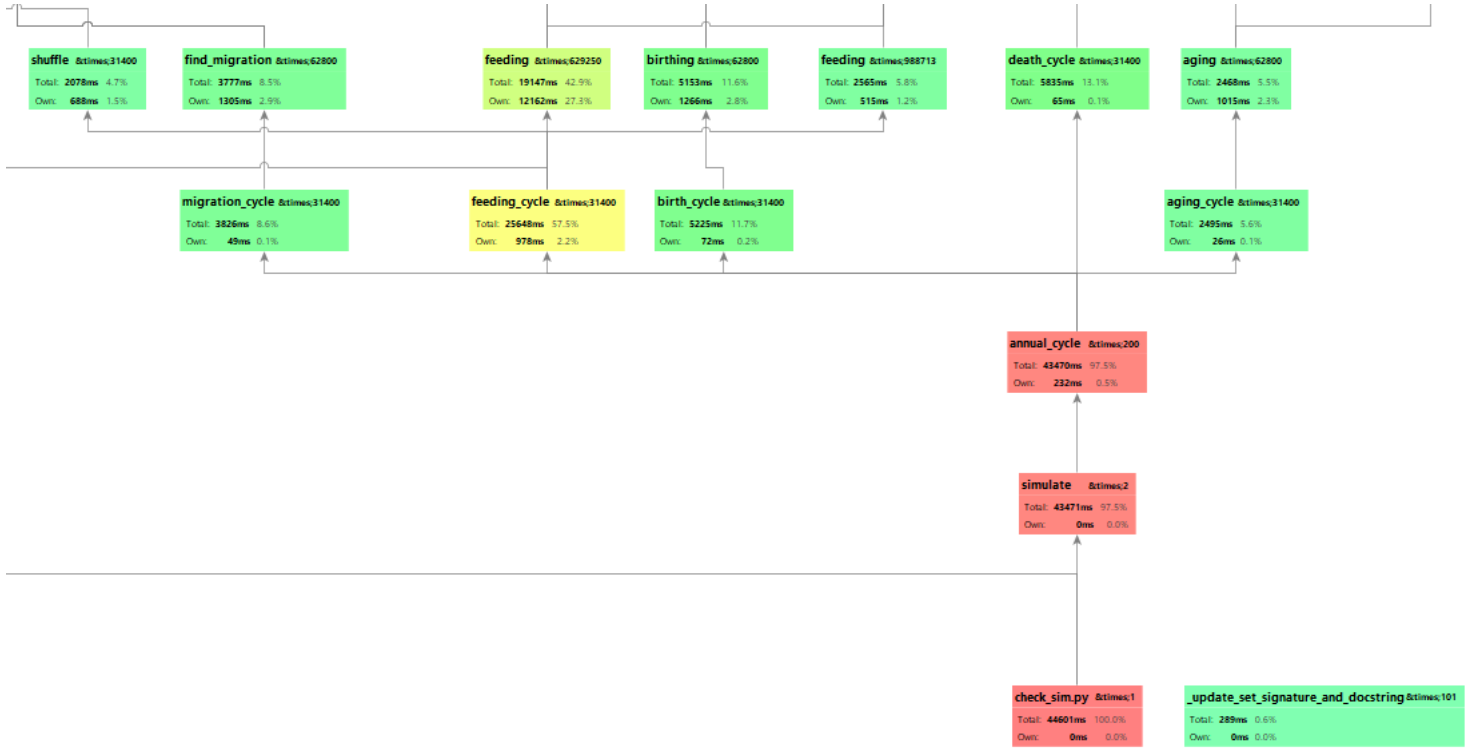
Optimization

- Fitness was all calculated earlier at init, birth, feeding, after weight loss, migration and before death. Identified that fitness can be called in the below sequence:
  - As soon as animal is inserted. (required for first birth cycle) – 1st time
  - During feeding's increase weight.(required for sorting and hunting. Cannot be avoided.) – 2nd time
  - Migration can use the same updated fitness calculated during feeding. (can be avoided).
  - Perform aging and then after weight loss, calculate fitness. – 3rd time
  - Same fitness value can be used to validate death. (can be avoided)
  - Since death does not deal with any changes in age or weight, same fitness can be used for birth in next cycle. (can be avoided)
- Earlier : Fitness (27s/8,786,672times)
  Now : Fitness(6s/4,115,275 times)

Optimization

- Before:
  annual_cycle → 145s / 200 cycles

- Now:
  annual_cycle → 43.4s / 200 cycles . (ranges between 43-45s)



END