Norwegian University of Life Sciences

Modelling the Ecosystem of Rossumøya



Project in Advanced Porgramming at REALTEK/ NMBU, January 2019

By: Ivan Cherednikov

Date: 2020/01/22

Director: Yngve Mardal Moe - Enviromental protection agency of Pylandia

**Abstract**

Ecosystems on Earth have defined the fauna and the flow of life for millions of years, balancing the flow of life on our planet. Only with the arrival of homo sapiens, the wise men, has this balance been disturbed and altered. However, this project is about a classic ecosystem on the island Rossumøya, where the grass-eating herbivore population is balanced out by the meat-eating carnivores. The purpose of this project was to simulate the population dynamics on Rossumøya, observe the changes in the population and see if this a stable population of both species could be achieved.

Rossumøya is not an actual island, but rather just a theoretical one. The geography of the island is a variable, chosen by those running the simulation and many properties of both of the species are also changeable. The simulation project was written in python 3.7 programming language with the help of PyCharm build 2019.2.3.

**Introduction**

The enviromental agency of Pylandia has given out a task to develop a program for the simulation of population dynamics on the island Rossumøya. An expert is expected to develop a population dynamics simulation by the end of January 2020.

Rossumøya is populated by two species, herbivores and carnivores. These are different in many ways, with one being a plant eater and the other one a meat eater, as the names suggest. The island has several different landscape types with different kinds of characteristics, with the animals living in these. As expected with an island, it is surrounded by water, which animals can not live in or migrate to.

The main concerns going into the project were about the structure of the simulation. Since it is the group's first time doing a project of this scale, a well working plan needs to be established before writing any code. This is to have well structured, working and bug free code that could be understood even by those uninitiated in the project. Therefore, well chosen function and variable names are crucial, as well as writing code following PEP8 python coding guidelines.

The main questions that are to be answered in the project:

**• Could a working island population dynamics program be created for Rossumøya?**

**• Could a stable population of both species on the island be achieved after an amount of years?**

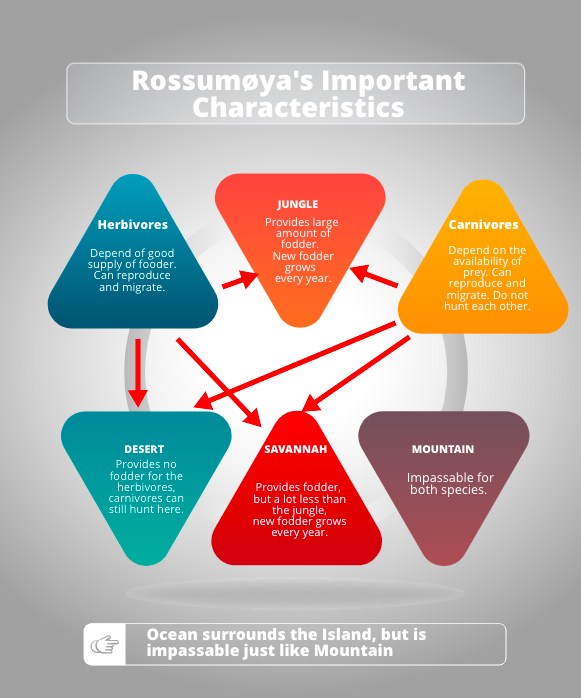
**Theory/Background**

PEP8 provides guidelines for writing Python code, dating back to 2001 where it was written by Guido van Rossum, Barry Warsaw and Nick Coghlan. The main goal is to improve readability of code and make it easier for others to understand.[[1]](#footnote-1) Since PEP8 checking is a built in feature, writing code following it's guidelines is much easier. The maximal line length chosen for the project is 80. Anything above that limit is to be transferred onto the next line using a line break or other methods.

A lot of information comes straight from the project assignment paper. Therefore, it will be referenced once and included in the source list, with all other information from it used in this paper being referenced to the instance of it in the source list and to the keynote. [[2]](#footnote-2)

Rossumøya The island is inhabited by two species, herbivores and carnivores. Herbivores require fodder in order to survive and reproduce, while the carnivores depend on the availability of herbivores. The landscape of the island is divided into 4 different kinds, excluding the Ocean, which is impassable for both species and surrounds the island. These are: Savannah, Desert, Mountain and Jungle.

Here is a visual representation of the most import characteristics on the island:



Picture 1: The arrows represent the landscape types to which each of the species can migrate.

Ocean: Surrounds the island, impassable for the animals

Mountais: Impassable for the animals

Desert: Animals can stay in and migrate to the desert, but there is no fodder here. Carnivores can still hunt.

Savannah: Offers fodder, but in limited quantity. Fodder grows every year according to "grow\_food" function for the Savannah subclass.

Jungle: Lots of fodder, maximal fodder growth every year.

Each animal is born with the age of 0 and a weight decided by normal distribution. The overall condition of an animal is described by fitness, calculated by a formula. Fitness is recounted after an animal gives birth, loses weight and eats.

Animals migrate from one cell to another depending on their fitness and fodder availability in neighbouring, non-diagonal cells. First, the migration probability is calculated, where the chance is higher if the fitness of the animal is higher. Afterwards the propensity and probability of the animal fron cell i to j is calculated using formulas given in the task paper. The probability to move to an Ocean or Mountain cell is always 0.

Animals mate if there are at least two same specied animals in a cell, without gender playing any role. The chance to give offspring is calculated by a formula, include the python function "min()". The birth giving mother loses weight and the animal's fitness is recalculated.

An animal dies if it's fitness is 0, or with a probability formula using the fitness value of the animal.

Rossumøya has an annual cycle, with following components:

0. Food growth in the Jungle and Savannah cells.

1. Feeding, animals eat. First herbivores, then carnivores.

2. Procreation, animals give birth.

3. Migrations, animals migrate.

4. Aging, each animal becomes one year older.

5. Loss of weight, all animals lose weight

6. Death.

**Method**

**Materials used in the project:**

Macbook with macOS High Sierra version 10.13.6 , PyCharm Professional 2019.2.3 (Professional Edition), licensed to Ivan Cherednikov, GitKraken version 6.4.1, GitHub.

**Execution of the project:**

It was decided that there is need of use of superclasses and subclasses. The animals file lays the foundation for the project, with other functions depending on the ones from animals.py. Therefore, it was written first. A superclass "Animal" was created, containing all the functions that could be applied to both herbivores and carnivores. The species bound parameters were stored in the "Herbivore" and "Carnivore" class. The herbivore subclass only had the species bound parameters and a "super() "function. Usage of inheritance and superclasses greatly reduces the amount of code needed to be written in the project. The carnivore subclass included the species parameters, a "super()" function and a "kill\_carnivore" function. Since herbivores do not hunt, the function did not need to be in the Animal superclass. It is also worth noting that the default parameters from the assignment paper were used.

Many of the functions in the animals.py file are simple, just plugging in the formulas and using right parameter names, or even just adding a 1, like in the age function. However, as already noted, together it build the much needed foundation for the rest of the code.

The calculation of fitness was split up into two, a class method and a static method. The static method behaves like a function and no object istance is passed as the first argument. In the class method the class of the istance is then passed as the first argument, with the class method finishing the fitness calculation. Fitness is recounted after yearly weight loss, weight loss at reproduction and after eating.

The calculation of the weight at birth of an animal was a simple, yet interesting class method. Since the project worker has not worked with numpy random destribution, it was a hassle figuring it out at first. But it was, in fact as simple as using numpy's "random.normal" normal distribution, putting in the values for the mean and standard deviation.

Moving on the the landscape file, a superclass Square was created. The class represents a single cell on the island. All types of cells are subclasses to it, including impassable ones, Ocean and Mountain. The constructor includes two empty dictionaries, one that is later filled with animals of both species in a cell and the other one with migrants, used in the migration functions.

The landscape file was built with the help of functions written in the animals.py file. Abundance is calculated in a similar way to formula\_fitness in animals.py, using a static method with a simple return. However, propensity differs from the fitness calculation as it is a harder process requiring the use of numpy and multiple local variables.

The migration process was by far the biggest challenge of the project, with constant problems arising in the process. It seems, however, that using multiple "for" loops combined with "if" tests and a "while" loop, together with numpy works wonders. The migration process was split up into two functions, one for addding them to "migrants" or keeping them in the cell if they do not move and the other one for adding them to the cell.

The subclasses contain constructors with super() functions, fodder growth functions for Jungle and Savannah, and propensity calculation function that return a 0 for the impassable landscape types, Ocean and Mountain.

Simple, yet crucial for the simulation code was further written in the landscape.py file. Yearly weight loss for animals in a cell and aging of animals in a cell. Furthermore, code for feeding both the carnivores and the herbivores was written. For herbivores, it was a simpler combination of a "for" loop and "if" and "elif" testing. If the herbivore ate all the food available in the cell, the cell's fodder supply was set to 0. For carnivores, however, it was a harder process. This part was assissted by the TA's, helping the student with the function. The herbivores that are not eaten in the carnivore hunting process are added to the "kill\_failed" list, they are the survivors. After the carnivore tries to hunt, the surviving animals are placed back in the cell. If a herbivore gets killed and it's weight is more that carnivore's hunger (how much the carnivore has eaten), the rest of the herbivore's food goes to waste, which means no food is stored.

Thereafter, a function was made that simply calls on food growth, carnivores and herbivores feeding. It was done to make the code more compact and to not have to call on all three functions in the yearly cycle function, later on in the project.

The island file combined all the previous code to get the island running. The island map construction was introduced to build the cells of the island. Since an animal that is in a cell can only move the one of four neighboring cells which are non diagonal, a function for getting the adjacent cells was written. Further, a function for adding initial animal population was done with the help of calling functions and code written previously.

In the end, simple functions calling the ones from previous files were done, to initialize the yearly cycle. The yearly cycle, as stated before, is feeding, procreation, migrations, aging, loss of weight and death. It is, however, important to note that the growth of food in savannahs and jungles occured before feeding, as a step 0. That is why the growth function is included in "feed\_everyone()".

Moving on to the simulation file, which has yet not been fully developed. The three functions for animal distribution on the island were done, "animal\_distibution", "num\_animals\_per\_species" and "num\_animals" with each returning the amount of animals, whether it is per species or a total amount, A pandas dataframe is used in the first function and is then passed on to the other two.

The "simulate" function checks if the simulation should still be running, by checking if the year has not passed the maximum amount of years and if any animals are still alive. If that is not the case, the simulation stops. Else, the yearly cycle is started again and a year passed in the simulation.

It should be noted a very small amounts of tests has been done, as they were left for last. That, I apologize for, since I know that it is a great percentage of the grade.

**Results**

Although the code if not fully finished and the simulation can not be called a finished products, results show that a simulation is working, and an initial population is added when altering the compatibility check for Biosim simulations written by Hans Ekkehard Plesser. The island is sucessfully simulated and both herbivores and carnivores walk the island.

**Discussion/Analysis**

The project was a big challenge, but a fun one. Although a full visualisation of the simulation has not been finished and a final product has not been achieved, the code written could be improved on at a later time, as a home project.

The task taught the student new skills within programming and has helped answer previously unaswered questions. The fact that the structure of the code was not pre-determined and had to be worked on by the student has shown a different side of programming and allowed for personal development. Finally has the math from previous math courses become useful, which is great. As Yngve has stated, there is a difference between those taking the INF200 course and those just taking programming courses without doing advanced math prior to it. With more effort and at a later time, more programming projects with advanced mathematics can be developed with the things learned in the course.

GitKraken working together with PyCharm works great as well, with PyCharm checking for possible mistakes and PEP8 guidelines and with GitKraken allowing you to see the changes you have done to your project, making it possible to roll back changes and go to a previous version if needed. The assisstance of these two programs has made structuring the project and fixing bugs a much easier task.

**Sources**

Moe, Yngve M (Director, enviromental protection agency of Pylandia). "Modelling the Ecosystem of Rossumøya", <https://github.com/yngvem/INF200-2019/blob/master/INF200_H19_BioSim.pdf> . Information gathered 2020.01.21. The official Rossumøya population dynamics project assignment paper. The front page image is taken from the paper as well, as I have nothing to do with the enviromental protection agency of Pylandia. January 2020.

<https://realpython.com/python-pep8/> . Information gathered 2020.01.21. The credibility of the source has not been checked because of lack of time. It is however assumed that the source is credible.

1. <https://realpython.com/python-pep8/> . The credibility of the source has not been checked because of lack of time. It is however assumed that the source is credible. [↑](#footnote-ref-1)
2. <https://github.com/yngvem/INF200-2019/blob/master/INF200_H19_BioSim.pdf> . The official population dynamics project assignment paper. [↑](#footnote-ref-2)