Foxes and Rabbits

Milestone 2

Exam 2021 - Group Project

DM536	Introduction to Programming
DM562	Scientific Programming
DM857	Introduction to Programming
DS830	Introduction to Programming

Functionality

In this phase of the project you will refine and extend the program from Milestone 1 by implementing the middle layer which consists primarily of the simulation stage of the program. During this stage, the programs performs the following activities.

Initialisation

The first activity is the initialisation of opportune data structures to represent the status of the simulated model and its entities (patches of grass, foxes, rabbits, etc.) according to the parameters provided as input (and obtained from the configuration stage, see Milestone 1). In particular, the program creates the 2D surface of patches (the specific data structure is up to you) and populate it with individuals of each species. Individuals created in this phase must be assigned random positions on the grid and random ages (within the respective ranges of validity) following a uniform distribution (see below). Moreover, alive individuals of the same species cannot share the same position whereas individuals of different species can.

During this activity, the program also starts visualisers for the progress and status of the simulation.

Update

The program then simulates the model in discrete time steps until the maximum number of steps is reached or there are no animals alive in the simulation. During each step, the status of the simulation is update as follows.

1. The "internal clock" of every entity of the simulation "ticks": patches grow grass, animals age and consume energy (dying by old age or starvation).
2. Then, the surviving animals feed using the resources in the patch they currently occupy.
3. The survivors attempt to reproduce provided that there is a patch free of predators and individuals of the same species near the parent (if there is more than one, a random one is uniformly selected) as discussed in the Overview.
4. Animals that failed to reproduce, try to move provided there is a nearby patch free of individuals of the same species (if there is more than one, a random one is uniformly selected) as discussed in the Overview.

Collection

The program has to track data and statistics required by the next reporting stage (which will be the object of Phased 3) and listed below.

- The total size of each population over the entire execution of the simulation.
- The number of alive individuals of each population and each time step.
- The age at the time of death of each individual, separated by population.
- The average energy level (relative to the maximum energy level) of the combined and separate populations at each time step.
- The total of deaths by each cause (old age, predation, starvation) and population
- The number of deaths by predation on each patch of the grid.

Randomness

To pick (pseudo)random values, use module `random`[1] from the standard library. Most of its functions (e.g., `random`, `randrange`, `randint`, `choice`, `sample`, `uniform`) can be considered based on uniform variables, for the purpose of this project. Observe that the result of `choices` allows repetitions, if you need to avoid repetitions use `sample`.

Structure

The program is organised in a number of modules as shown in Figure 1.

- Modules `fox_and_rabbits`, `simulation`, `reporting`, and `parameters` follow the same contract discussed in Phase 1. In particular, your implementation of `fox_and_rabbits` from Phase 1 should work also in this settings.
- Module `results` contains classes for storing data and statistics of a simulation run.
- Module `entities` contains classes for storing and manipulating the status of grass patches (class `Patch`) and animals (`Animal`, `Fox`, and `Rabbit`). The class `Animal` contains definitions shared by the classes `Fox` and `Rabbit` (which can be though of as special cases).
- Module `visualisers` defines some classes for visualising the progress and status of the simulation (this module requires some third-party packages, see below).

Of the modules listed above, you are asked to implement only module `simulation` while its dependencies are provided (obfuscated or as mock implementations) together with their documentation as exam material.

The implementations of `visualisers` and `reporting` used in this phase require two third-party packages: `numpy`[2] and `matplotlib`[3]. Although these packages are widely used, they are not part of the standard Python distribution and might not be on your machine. You can obtain these packages using the utility PIP which is part of the standard Python installation—see “How to: Installing Packages with PIP” from the Course Material or `https://pip.pypa.io/en/stable/`.

[1] `https://docs.python.org/3/library/random.html`

[2] `https://numpy.org/`

[3] `https://matplotlib.org/`

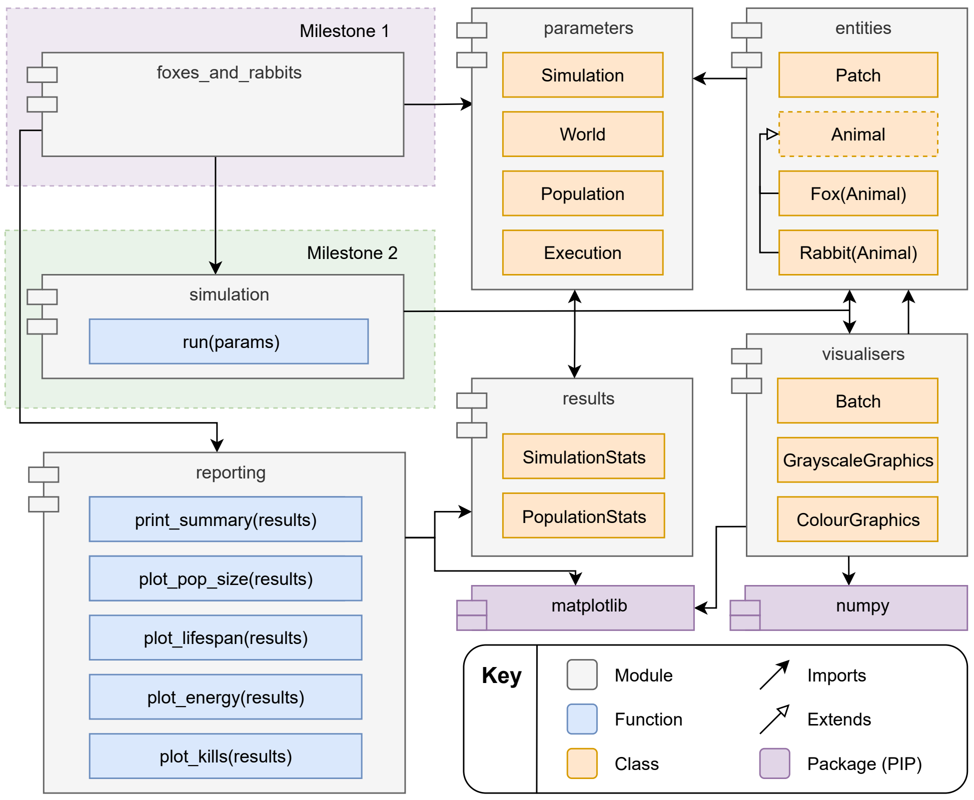

Figure 1: Structure - Phase 2.

Milestone 2

Module `simulation`.

Hand-in

You must hand in a zip file containing:

- A PDF document named `report.pdf` containing your report.
- A file `simulation.py` containing your implementation of module `simulation` as described here.

The name of the zip file must be the name of your group e.g., `Group C18.zip` (capitalisation is immaterial).

The report must be written in English and delivered as a single PDF file printable in black and white and long at most 15 pages excluding front and back matter e.g., an appendix (examiners are not required to consider appendices or anything above the page limit in their evaluation). The report must include the name of the group and its members (full name and SDU email where applicable).

Your code must follow the structure detailed in this document, be clearly documented, and adhere to the common coding conventions and rules of Python and this course. For this phase your code can use any module in the standard library of Python (third-party modules and packages are not allowed, code provided as exam material like `visualiser.py` is exempt from this rule).