## ABM

1. The provided notebook contains a basic implementation of a predator-prey model. The model consists of sheep, wolves, and grass. The sheep and wolves move about on a SingleGrid, while grass is implemented as a NumPy array.

Both sheep and wolves use one unit of energy per tick. If their energy level drops below zero, they die. If they are still alive, they move about, possibly eat to gain energy, and possibly reproduce. Sheep move to an empty random neighbouring location and eat all the grass at this new location. One unit of grass equals one unit of energy for a sheep. Wolves prefer to move to a random neighboring location containing a sheep. If such a location is available, they first eat the sheep at this location, gain all the energy of the sheep, and next move to this location. If no neighboring location with a sheep is available, a wolf will move to a random empty location instead. Wolves and sheep reproduce in the same way. There is a small chance that an animal reproduces (asexually). Moreover, the animal must have sufficient energy to reproduce, and there must be at least one empty neighboring location on which to position the offspring. If an offspring is created, it is placed in a random empty neighboring location, and it gets half the energy of its parent. Grass also regrows. If there is some grass on a given location, it gains 1/3 unit of energy up to a maximum of 5. If there is no grass in a given location, grass will only regrow with a small probability. If it regrows, it also adds 1/3 unit of energy.

A key implementation detail: a MESA SingleGrid is indexed from the bottom left corner using x (column) and y (row), while a NumPy array is indexed from the top left using i (row) and j (column). The Sheep class comes with properties for doing this conversion.

The provided notebook contains the implementation of the complete model except for the step method of the Sheep.

1. (20/100) Sketch a flowchart or provide in pseudo-code a conceptual description of the step method of the sheep.
2. (30/100) Implement the step method. For validation purposes, the figure below gives you a sense of the dynamics the model produces. The necessary code to create this visual is included in the notebook.

Chart, histogram

Description automatically generated

**Figure 1 Indicative results**

1. (20/100) Desertification is a major problem in many parts of the world. For a long while, it was thought that grazing by e.g., sheep contributed to desertification. So, policies focussed on reducing grazing. However, in specific places this resulted in an increase of desertification. It turned out that if grass was eaten, it was turned into manure that replenished the nutrient stock in the soil. If grass was not eaten, it would dry out and not return to the nutrient stock in the soil. Over time, this resulted in exhausting the soil and thus resulted in a speed up of the desertification. How would you extend the model to bring in this additional dynamic? In your answer, think about potential modifications to the agents in the model, the space you would use, possible modification in scheduling and/or step methods.
2. The model as provided is sensitive to initial conditions. Explore the behaviour of the model by varying the grass regrowth probability between 0.05 and 0.15 with a step size of 0.025.
   1. (10/100) Implement the code necessary for performing the exploration.
   2. (10/100) Provide a clear visualization that reveals how the dynamics of the model over time changes across the different parametrisations for the number of sheep, wolves, and amount of grass.
   3. (10/100) Explain the observed changes in behaviour as shown in response to question B.