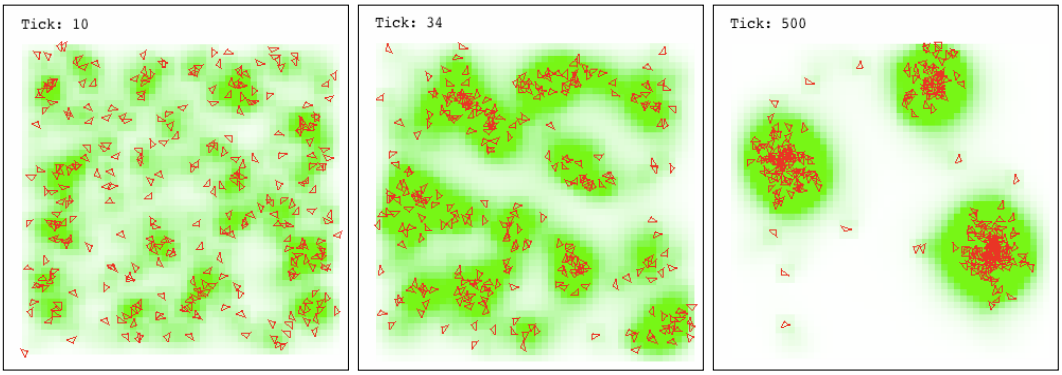
Project 12.3 Slime mold aggregation

In this project, you will write an ***agent-based simulation*** that graphically depicts the emergent "intelligence" of a fascinating organism known as slime mold (*Dictyostelium discoideum*). When food is plentiful, the slime mold exists in a unicellular amoeboid form. But when food becomes scarce, it emits a chemical known as cyclic AMP (or cAMP) that attracts other amoeboids to it. The congregated cells form a ***pseudoplasmodium*** which then scavenges for food as a single multicellular organism. We will investigate how the pseudoplasmodium forms. A movie linked from the book website shows this phenomenon in action.

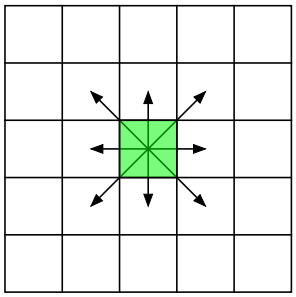
The following sequence of images illustrates what your simulation may look like. The red triangles represent slime mold amoeboids and the varying shades of green represent varying levels of cAMP on the surface. (Darker green represents higher levels.)



Slime world

In our simulation, the slime mold's world will consist of a grid of square patches, each of which contains some non-negative level of the chemical cAMP. The cAMP will be deposited by the slime mold (explained next). In each time step, the chemical in each patch should:

1. Diffuse to the eight neighboring patches. In other words, after the chemical in a patch diffuses, 1/8 of it will be added to the chemical in each neighboring patch. (Note: this needs to be done carefully; the resulting levels should be as if all patches diffused simultaneously.)



2. Partially evaporate. (Reduce the level in each patch to a constant fraction, say 0.9, of its previous level.)

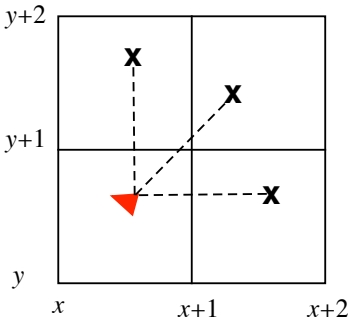
Slime world will be modeled as an instance of a class (that you will create) called `World`. Each patch in slime world will be modeled as an instance of a class called `Patch` (that you will also create). The `World` class should contain a grid of `Patch` objects. You will need to design the variables and methods needed in these new classes.

There is code on the book website to visualize the level of chemical in each patch. Higher levels are represented with darker shades of green on the turtle's canvas. Although it is possible to recolor each patch with a Turtle during each time step, it is far too slow. The supplied code modifies the underlying canvas used in the implementation of the `turtle` module.

Amoeboid behavior

At the outset of the simulation, the world will be populated with some number of slime mold amoeboids at random locations on the grid. At each time step in the simulation, a slime mold amoeboid will:

1. "Sniff" the level of the chemical cAMP at its current position. If that level is above some threshold, it will next sniff for chemical `SNIFF_DISTANCE` units ahead and `SNIFF_DISTANCE` units out at `SNIFF_ANGLE` degrees to the left and right of its current position. `SNIFF_ANGLE` and `SNIFF_DISTANCE` are parameters that can be set in the simulation. In the graphic below, the slime mold is represented by a red triangle pointing at its current heading and `SNIFF_ANGLE` is 45 degrees. The `X`'s represent the positions to sniff.



Notice that neither the current coordinates of the slime mold cell nor the

coordinates to sniff may be integers. You will want to write a function that will round coordinates to find the patch in which they reside. Once it ascertains the levels in each of these three patches, it will turn toward the highest level.

2. Randomly wiggle its heading to the left or right a maximum of `WIGGLE_ANGLE` degrees.
3. Move forward one unit on the current heading.
4. Drop `CHEMICAL_ADD` units of cAMP at its current position.

A slime mold amoeboid should, of course, also be modeled as a class. At the very least, the class should contain a `Turtle` object that will graphically represent the cell. Set the speed of the `Turtle` object to 0 and minimize the delay between updates by calling `screen.tracer(200, 0)`. The remaining design of this class is up to you.

The simulation

The main loop of the simulation will involve iterating over some number of time steps. In each time step, every slime mold amoeboid and every patch must execute the steps outlined above.

Download a bare-bones skeleton of the classes needed in the project from the book website. These files contain *only the minimum amount of code necessary* to accomplish the drawing of cAMP levels in the background (as discussed earlier).

Before you write any Python code, think carefully about how you want to design your project. Draw pictures and map out what each class should contain. Also map out the main event loop of your simulation. (This will be an additional file.)