Training

Types of agents in NetLogo:

- **Observer**  
  Gives instructions from an external points of view

- **Patches**  
  Cannot move, have integer coordinates

- **Turtles**  
  Can move in the “world”

- **Links**  
  Connect two turtles
Training

ask patch 0 0 [ set pcolor green ]
Training

ask patch 0 0 [ set pcolor green ]

Who is the subject of this command?
The observer
Training

ask patch 0 0 [ set pcolor green ]

Who is the subject of this command?
Patch 0 0

In the [ ], the point of view changes
Training

```
ask patch 0 0 [ set pcolor green ]
```

```
ask agentset [ do something ]
```
ask patch 0 0 [ set pcolor green ]

pcolor is a built-in variable of patches, there are many other built-in variables, for example pxcor and pycor

“green” is a value that pcolor can have
We can now define another “agentset”
How can we turn all the patches into blue?
Training

ask agentset [ do something ]

We can now define another “agentset”
How can we turn all the patches into blue?

ask patches [ set pcolor blue ]
Training

ask agentset [ do something ]

We can now define another “agentset”
How can we turn all the patches into blue?

ask patches [ set pcolor blue ]

Patches is an agentset that the observer can access, it represents all the patches in the “world”
Training

Now type

clear-all

And repeat the same command with reduced speed. What do you notice?
Training

Now type

clear-all

And repeat the same command with reduced speed. What do you notice?

When a command is assigned to an agentset, it is not executed simultaneously by all the components of the agentset. Components of the agentset execute the command one after another in random order.
ask \textit{agentset} \ [ \textit{do something} \ ]

How to ask the patches in the upper half of the world to turn their color into yellow?
Training

ask `agentset` [ `do something` ]

How to ask the patches in the upper half of the world to turn their color into yellow?

ask patches with [ `pycor > 0` ]
[ `set pcolor yellow` ]
Training

ask *agentset* [ *do something* ]

How to ask the patches in the upper half of the world to turn their color into yellow?

ask patches with [ *pycor* > 0]
[ *set pcolor* *yellow* ]

*agentset* with [ *condition* ]
Training

ask \textit{agentset} [ do something ]

How to ask the patches in the upper half of the world to turn their color into yellow?

ask patches with [ pycor > 0]
[ set pcolor yellow ]

\textit{agentset} with [ condition ]

Here the point of view belongs to the agent within the agentset. Each agent knows its own pycor.
ask agentset [ do something ]

How to turn the neighbors of patch 0 0 into pink?
Training

ask agentset [ do something ]

How to turn the neighbors of patch 0 0 into pink?

Each patch has a built-in variable (an agentset) called “neighbors”
ask agentset [ do something ]

How to turn the neighbors of patch 0 0 into pink?

Each patch has a built-in variable (an agentset) called “neighbors”

ask patch 0 0
[ ask neighbors [ set pcolor pink ] ]
ask agentset [ do something ]

How to turn the neighbors of patch 0 0 into pink?

Each patch has a built-in variable (an agentset) called “neighbors”

ask patch 0 0
    [ ask neighbors [ set pcolor pink ] ]

The observer can delegate the command ask to an agenset
Training

Ask to 10 random patches to set the color equal to BLUE

ask agentset [ do something ]

ask n-of 10 patches [ set pcolor blue]
Training

Ask to 10 random patches to set the color equal to BLUE

ask `agentset` [ `do something` ]

ask `n-of 10 patches` [ `set pccolor blue` ]
Training

Ask to 10 random patches to set the color equal to BLUE

\[
\text{ask} \quad \text{agentset} \quad [ \quad \text{do something} \quad ]
\]

\[
\text{ask} \quad \text{n-of 10 patches} \quad [ \quad \text{set pcolor blue} \quad ]
\]

\text{n-of number agentset}
Now,

Set the neighbors of the blue patches RED
Now,

Set the neighbors of the blue patches RED

Then

Set the neighbors of the RED patches equal to RED IF THEY ARE NOT BLUE

Repeat this latter command over and over
Basic simulation of pest growth and spread in a landscape

Each patch is characterized by:
• Cultivated fractions $p_1$, $p_2$, $p_3$ ...
• A normalized density of pests (from 0 to 1)

We need to define a rule for
- Pest growth within each patch
- Pest spread from one patch to another
Basic simulation of pest growth and spread in a landscape

Assign a cultivated fraction to each patch:

Let's assume that we have only two types of cultivated fractions, $p_1$ and $p_2 = 1 - p_1$.

(we only have to assign $p_1$)

$p_1$ is not a built-in variable of patches in NetLogo, so we have to define it.
Basic simulation of pest growth and spread in a landscape

patches-own [ p1 ]
Basic simulation of pest growth and spread in a landscape

patches-own [ p1 ]

to setup
clear-all
ask patches
[ set p1 random-float 1 ]
end
Assigning cultivated fractions to patches

```
patches-own [ p1 ]

to-setup
  clear-all
  ask patches
    [ set p1 random-float 1 ]
end
```

We are telling to NetLogo to assign to each patch a variable called `p1`
Assigning cultivated fractions to patches

patches-own [ p1 ]

to setup
  clear-all
  ask patches
  [ set p1 random-float 1 ]
end

This is a way to define a procedure in Netlogo. It can be “called” in the command line or it can be associated with a button.
Assigning cultivated fractions to patches

patches-own [ p1 ]

to setup
   clear-all
ask patches
   [ set p1 random-float 1 ]
end

Usually it is better to write this as a first command in a setup procedure
Assigning cultivated fractions to patches

patches-own [ p1 ]

to setup
clear-all
ask patches
[ set p1 random-float 1 ]
end

random-float n
generates a real number between 0 and n.
Assigning cultivated fractions to patches

By calling the procedure setup nothing happens, why?
Assigning cultivated fractions to patches

By calling the procedure setup nothing happens, why?

Each patch has a value for p1, it is just not visible.

For example, type:

```
ask patch 0 0 [ show p1 ]
```
Assigning cultivated fractions to patches

By calling the procedure setup nothing happens, why?

Each patch has a value for p1, it is just not visible.

How to assign the color green to all the patches with p1 > 0.5?
Assigning cultivated fractions to patches

By calling the procedure setup nothing happens, why?

Each patch has a value for p1, it is just not visible.

How to assign the color green to all the patches with p1 > 0.5?

\begin{verbatim}
ask patches with [ p1 > 0.5 ]
[ set pcolor green ]
\end{verbatim}
Assigning cultivated fractions to patches

patches-own [ p1 ]

to setup
clear-all
ask patches
[ set p1 random-float 1 ]
ask patches
[ set pcolor scale-color green p1 0 1 ]
end

This assigns a shade of green proportional to p1.
Now, let’s introduce pests
Pest population growth in each patch

patches-own [ p1 PESTS ]  We need to assign another variable to the patches
to setup
...
end
Pest population growth in each patch

patches-own [ p1 PESTS ]

to setup
...
ask patch 0 0
[ set PESTS 0.1 ]
end

As a default, all variables are initialized to 0. For initializing, we need to specify in the setup function
Pest population growth in each patch

patches-own [ p1 PESTS ]

to setup
...
ask patch 0 0
[ set PESTS 0.1 ]
color-patches-pests
end

to color-patches-pests
ask patches with [ PESTS > 0 ]
[ set pcolor scale-color-color brown PESTS 0 1 ]
end

We can also color patches in order to show the density of pests.
Pest population growth in each patch

patches-own [ p1 PESTS ]
globals [ S R ]

to setup
...
set S 0.8
set R 0.2
end

There are some parameters which are the same for all the patches, so they are globals (the parameters S – survival and R – reproduction)
Their value is assigned in the setup function directly by the observer
Pest population growth in each patch

patches-own [ p1 PESTS ]
globals [ S, R ]

to setup
...
end

... 

to go
grow-pests
color-patches-pests
end

to grow-pests
ask patches
[ set PESTS ( S * PESTS + R * PESTS * ( p1 – PESTS ) ) ]
end

The procedure go executes a procedure for growing pests in each patch and for coloring patches

It can be setup as a forever procedure
Pest population spreading from patch to neighbors

patches-own [ pl PESTS Contagious ]
globals [ S, R ]

...  
To spread-pests
ask patches with [ PESTS > 0.2 ]
[ set Contagious 1 ]
end

First, we assume that PESTS can spread to neighbors if their density of PESTS is greater than 0.2
Pest population spreading from patch to neighbors

patches-own [ p1 PESTS Contagious ]
globals [ S, R ]

... to spread-pests
ask patches with [ PESTS > 0.2 ]
[ set Contagious 1 ]
ask patches with [ Contagious = 1 ]
[ ask neighbors4 with [ PESTS = 0] ]
[ set PESTS 0.1 ] ]
end

First, we assume that PESTS can spread to neighbors if their density of PESTS is greater than 0.2.