Introduction to Scientific Computing and Problem Solving

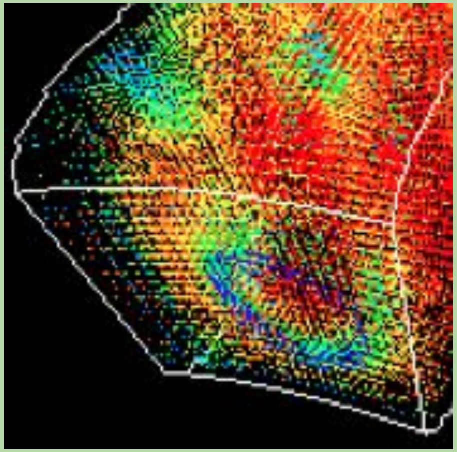
Jason Gaudette

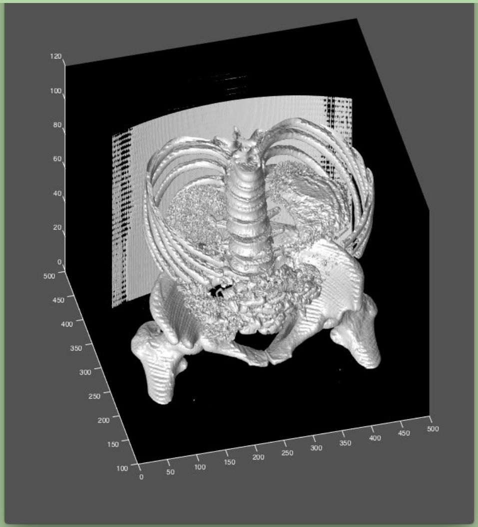
CSCI0040 - Spring 2019

CS4

An introductory scientific computing course

- Designed for non-CS concentration majors
- STEM oriented audience
- No prior programming experience is assumed
- No calculus or linear algebra prerequisites





Course Content

Computer science is not so much the science of computers as it is the science of solving problems using computers.

- Eric Roberts

- This course covers:
  - the process of developing **algorithms** to solve problems
  - the process of developing computer **programs** to express those algorithms
  - topics from **computer science** and **scientific computing**

Course Goals

- Two main goals (and parts to the course)
  - Introduction to Computer Science
  - Topics in Scientific Computing
- Students should leave the course with
  - Excellent Python and MATLAB programming skills
  - As well as the ability to *implement* mathematical models/concepts in their programs
- Assignments will include 2 quizzes, 11 homeworks, and 3 projects
  - Please review the syllabus for grading breakdown!

Computer Science -vs- Programming

- There are many different fields within CS, including:
  - software systems
  - computer architecture
  - networking
  - programming languages, compilers, etc.
  - theory
  - artificial intelligence
- Experts in many of these fields don't do much programming!
- However, learning to program will help you to develop ways of thinking and solving problems used in all fields of CS.

A Breadth-Based Introduction

Four major units:

- weeks 0-3: computational problem solving and imperative programming
- Weeks 4-5: functional programming
- Weeks 6-7: object-oriented programming
- Weeks 9-12: MATLAB, linear algebra, image processing, and special topics

These units are designed to

- help develop your computational problem-solving skills
  - including, but not limited to, coding skills
- give you a sense of the richness of computer science and scientific computing

A Comprehensive Introduction

- Intended for:
  - Engineering, math, and physical science concentrators
  - others who want a comprehensive/applied introduction
  - Beginners! No programming background required
- Allow for about **10 hours of work per week**
  - start work early!
  - utilize TA Hours, piazza, and other supporting resources

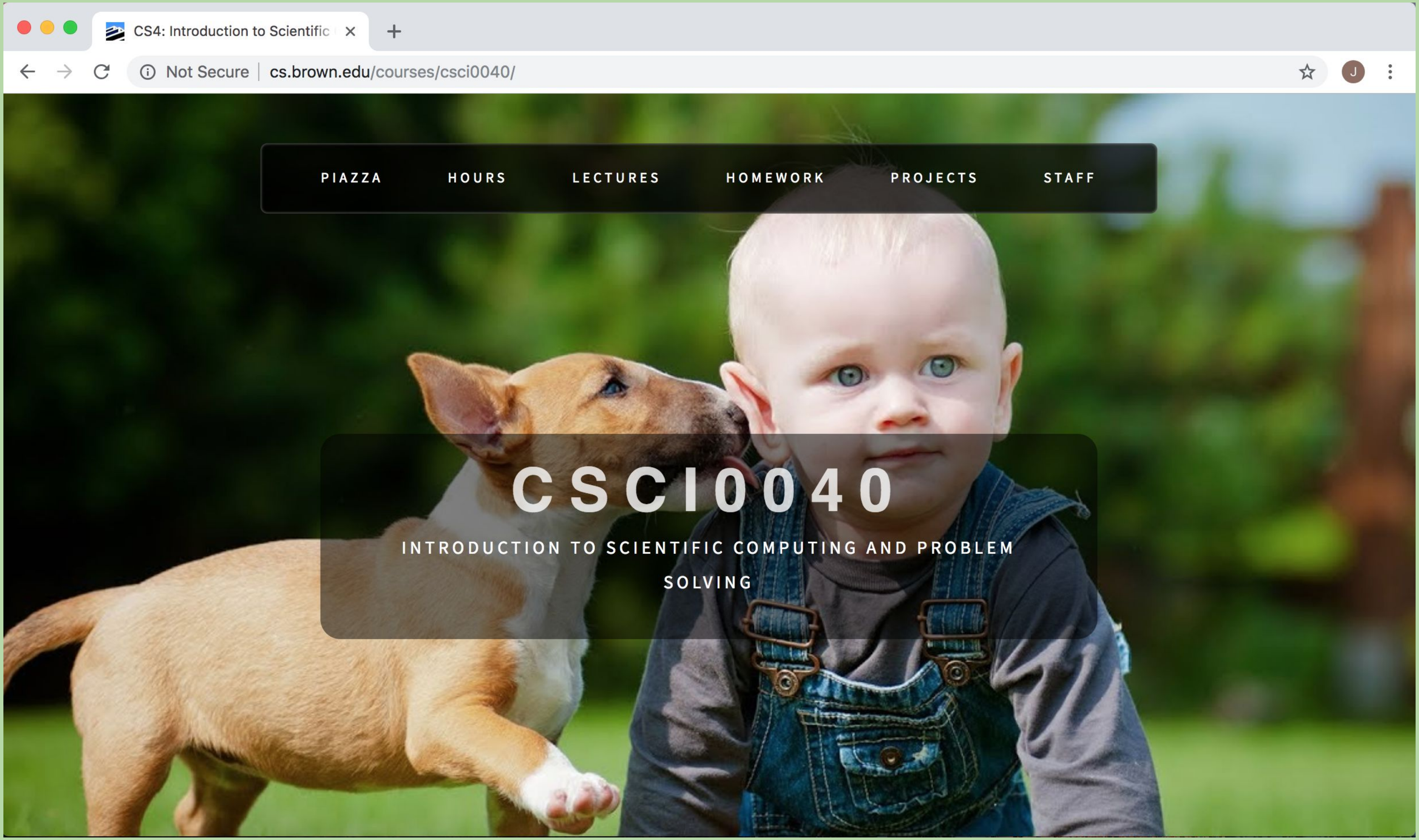
Preparing for Lecture

- We recommend doing the HMC reading(s) and reviewing the slides before each lecture

- Preparing for lecture is essential!
  - gets you ready for the lecture questions and discussions
  - we may not cover everything in the lecture material

Course Website

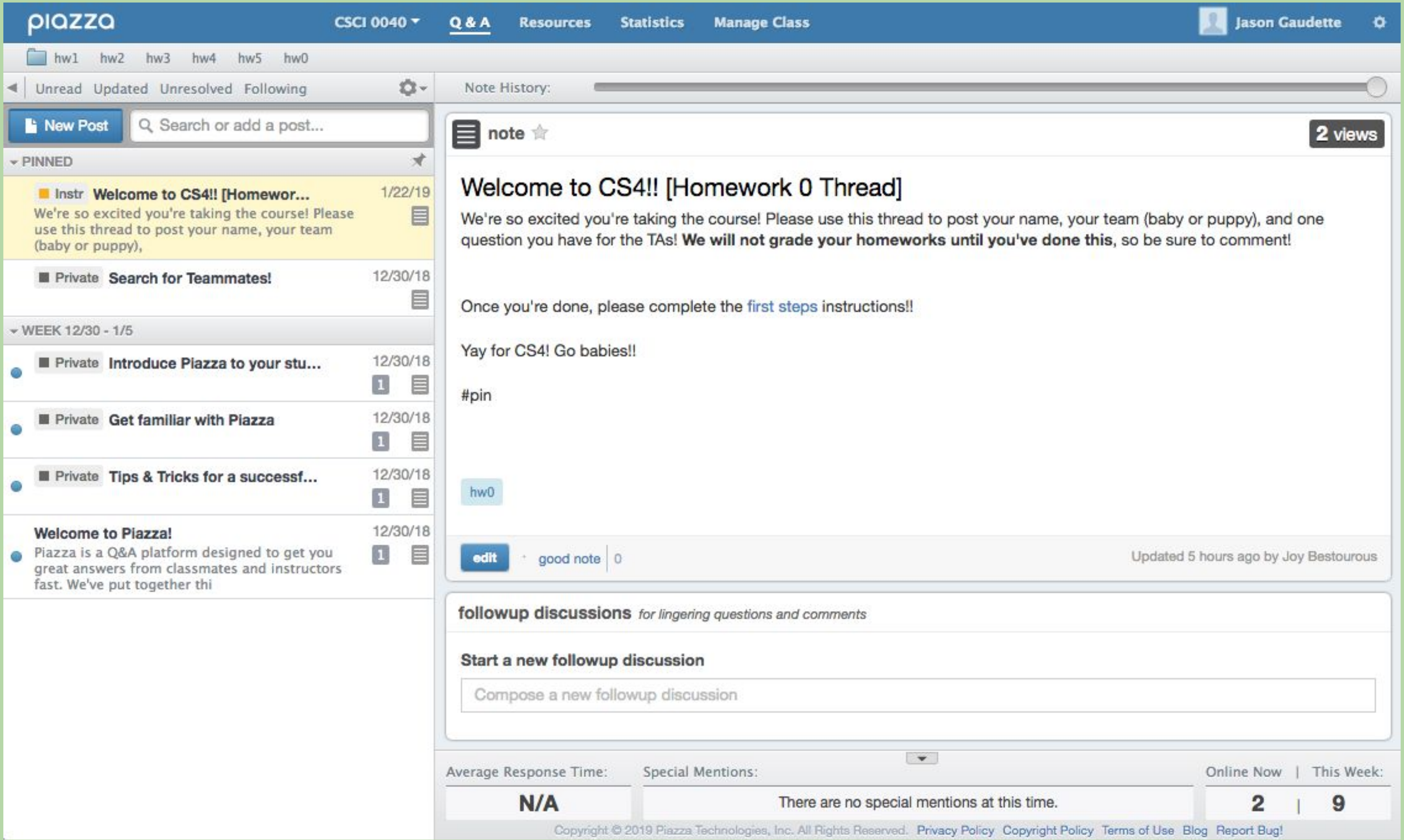
http://cs.brown.edu/courses/cs004/



Check this site frequently for updates to syllabus, lecture material, homework, and projects!

Course Discussion Forum

https://piazza.com/brown/spring2019/cs4



Start a discussion, ask questions, or help your classmates on Piazza

Teaching Staff

- Three head TAs
  - Griffin Kao, Joy Bestourous, Hersh Gupta

- Eleven UTAs
  - Annie He, Alex Liu, Aryan Srivastava, Ellen Ling, Irene Rhee, Joseph Chen, Jarrett Huddleston, Milla Shin, Pedro de Freitas, Solomon Rueschemeyer-Bailey, Tiffany Ding

TA Sections

- You will sign up for a TA Section on the website by Sunday at 11:59 PM. Sections will be held Thursdays and Fridays (starting Thursday, January 31st). **Attendance is required every week** and will account for 5% of your grade!

- These sections were created to help you get started on the assignments and offer lots of direct TA access

- Helps prevents huge lines during TA hours the night before a homework or project is due

- **Attend a Setup Section Tomorrow (11 AM - 7 PM) or Friday (11 AM - 4 PM) in the Sunlab**
  - setup a CS account and remote access, install python (bring your personal laptop if you want to set up remote access)
  - verify you have everything you need to hand-in assignments
  - sections start on the hour, so please show up on time!
  - let us know if you can't make any of those times

Assignments

- Weekly problem sets
  - Homeworks will be released on Wednesdays after lecture and will be due the following Wednesday at 4 PM
  - Can submit up to 24 hours late with a 20% penalty
  - No submissions accepted after 24 hours

- Projects
  - Projects will be released and due on Thursday (at midnight)
  - Can submit up to 72 hours late with a 20% penalty for each day it is late
  - No submissions accepted after 72 hours

- You have a combined 6 late days to use on either homeworks or projects. Please see the syllabus on the course website for more detail.

Collaboration

- Homeworks and Projects
  - Must complete on your own, but you may interact with other others at a high level - you must obey the collaboration policy!

- For both types of assignments:
  - may discuss assignment requirements and main ideas with others
  - may ***not*** **view** another student's work
  - may ***not*** **show** your work to another student
  - don't consult solutions from past semesters
  - don't consult solutions in books or online

Grading

1. 11 Weekly problem sets (40%), 3 projects (40%)
   - your lowest weekly HW score will be dropped

2. Quizzes
   - Quiz I (7.5%) - Python
   - Quiz II (7.5%) - MATLAB

3. Section attendance (5%) - split over the total sections
   - Includes TA session attendance, iClicker responses, etc.

Algorithms

- In order to solve a problem using a computer, you need to come up with one or more *algorithms*.

- An algorithm is a step-by-step description of how to accomplish a task.

- An algorithm must be:
    - *precise:* specified in a clear and unambiguous way
    - *effective:* capable of being carried out

Programming

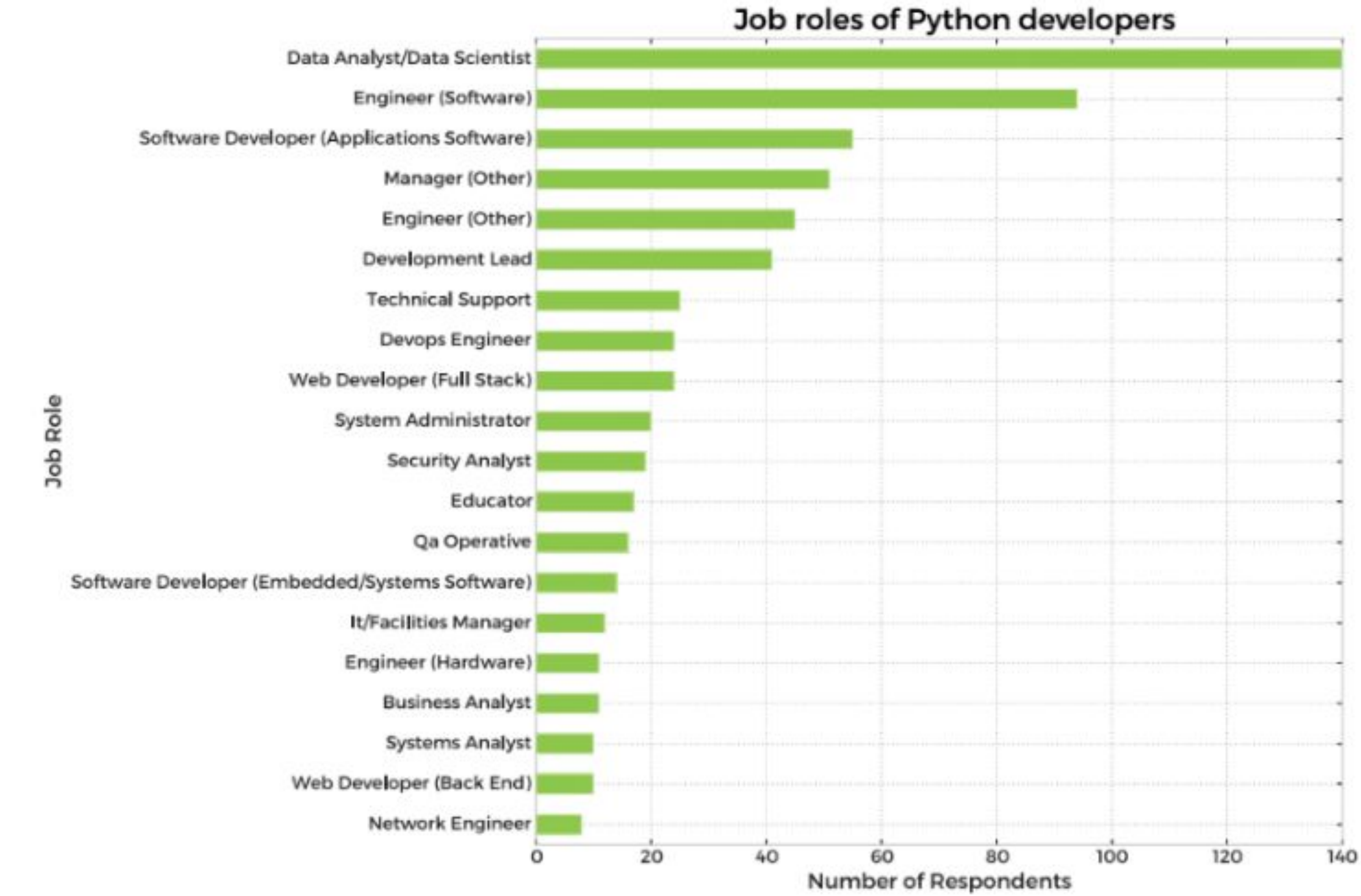
- Programming involves expressing an algorithm in a form that a computer can interpret.

- We will primarily use the Python programming language.
  - one of many possible languages
  - widely used
  - relatively simple to learn

- The key concepts of the course transcend this language.

- You can use any version of Python **3**
  - ***not*** Python 2
  - see First Steps and visit the Setup Section for details

Why Learn Programming?



Source: Skill Up 2016: Developer Skills Report

Why Learn Programming in Python?

Average Python Developer Salary Compared to Other Programming Languages

Skill	Average salaries	Monthly jobs advertised
Python	US$116,379	6,550
Ruby	US$115,005	1,080
Java	US$112,592	10,443
Perl	US$111,928	1,398
C++	US$108,123	3,567
JavaScript	US$103,503	8,764
C#	US$101,715	4,101
PHP	US$94,690	1,664
ASP.NET	US$95,551	1,289
C	US$95,166	5,639

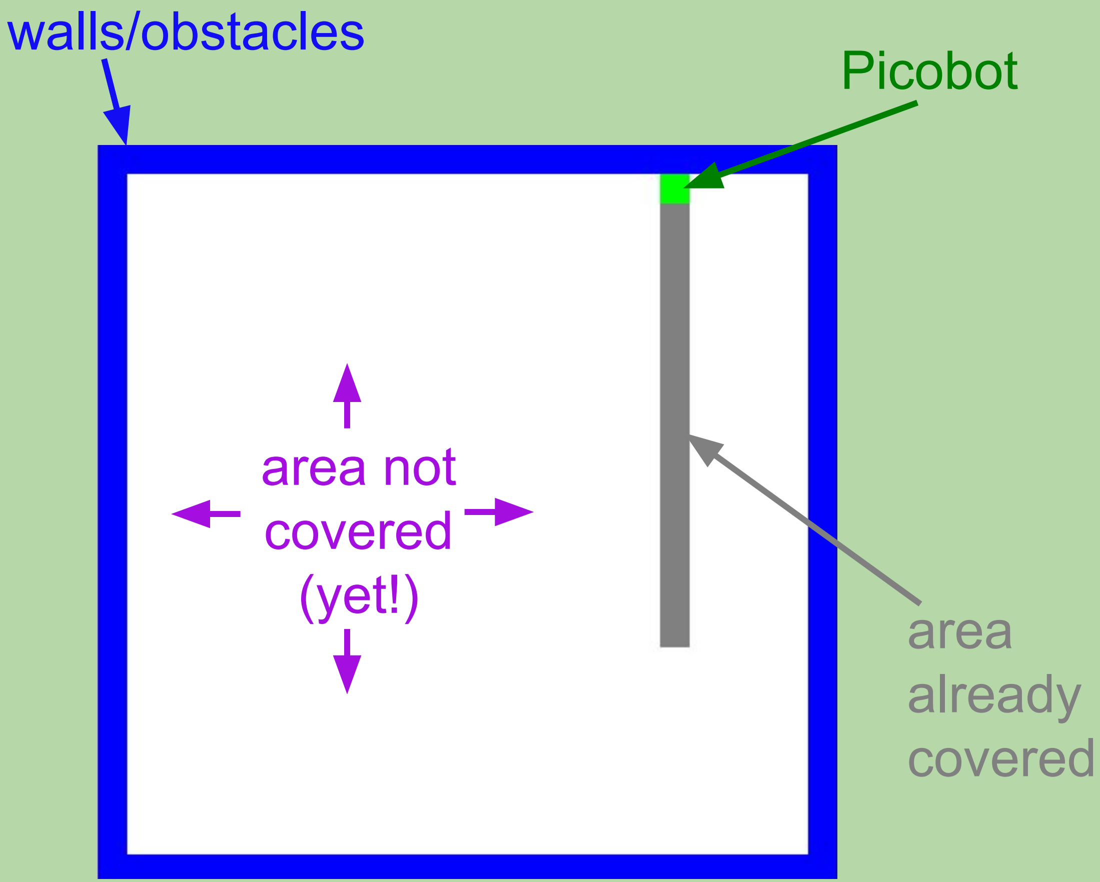
But First… Let's Learn Picobot!

- Python is a relatively simple language, but it will take several weeks to learn

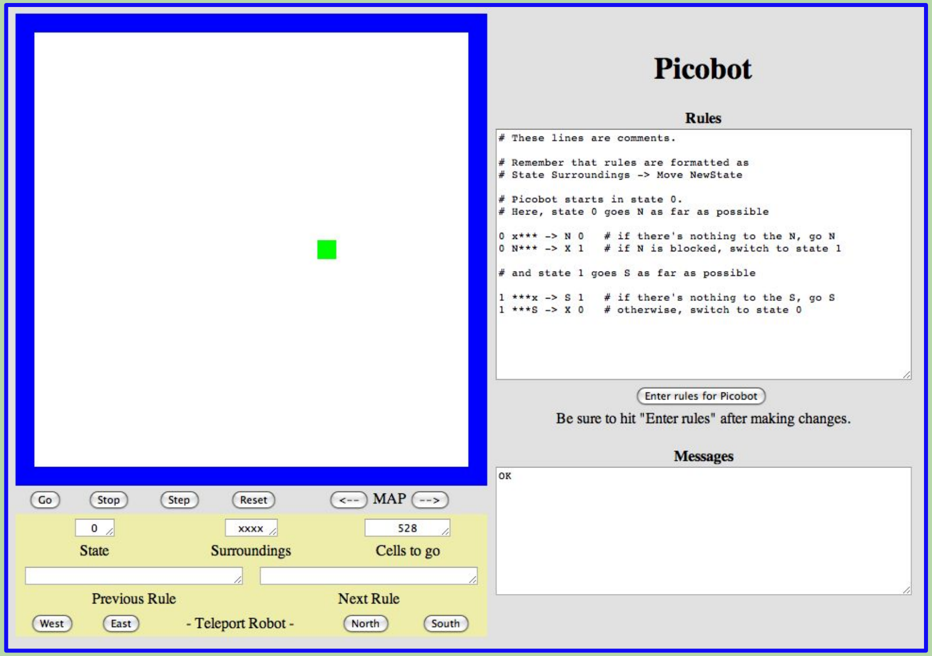
- To allow for interesting problems right away,
  we're going to start with something even simpler!

- Picobot!
  - a special-purpose language
  - controls a robot based on the Roomba vacuum cleaner robot

The Picobot Environment



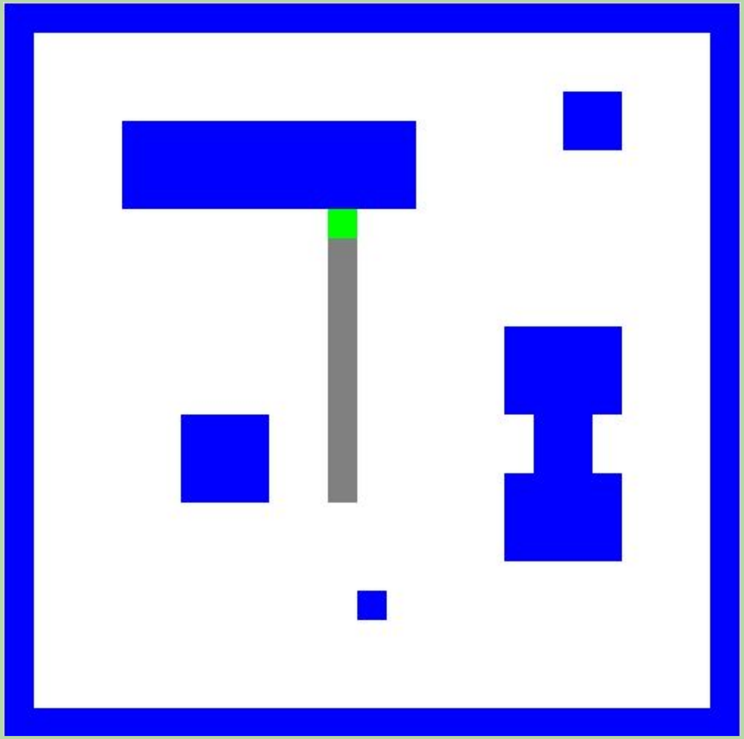
Picobot (cont.)



- Goal: to have the robot "vacuum" a small room.
  - there may be obstacles!
  - it can't remember where it's been
  - it can only sense its immediate surroundings

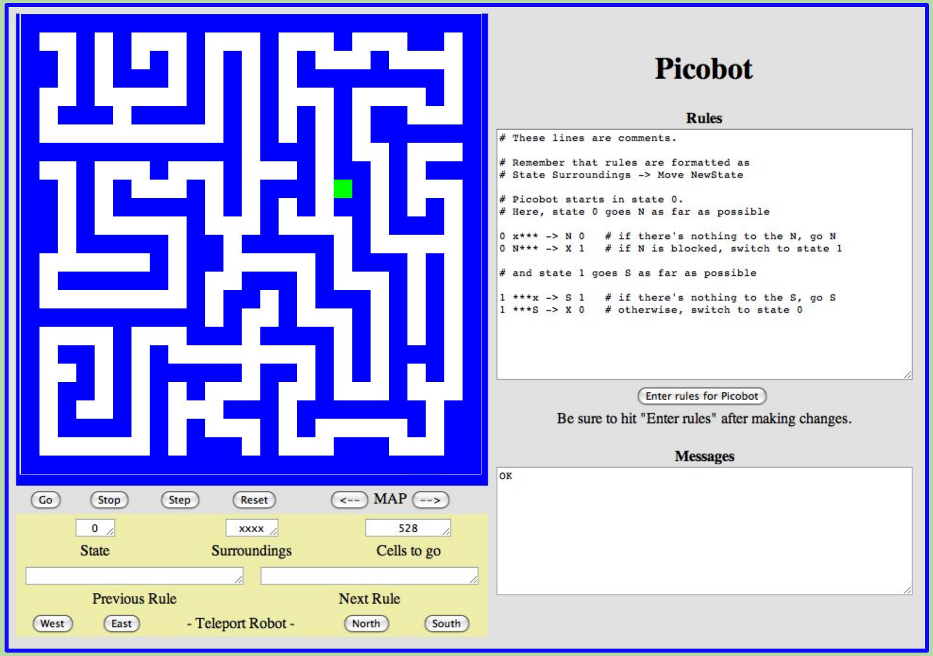
https://www.cs.hmc.edu/picobot/

The Picobot Environment (cont.)



- Rooms can have walls/obstacles "inside" the box, too!

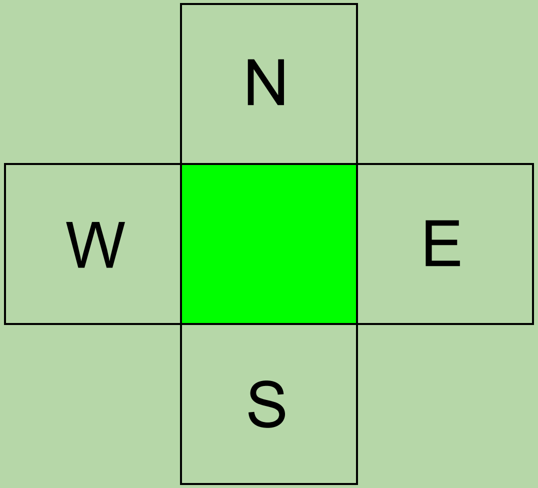
Picobot (cont.)



- Goal: to have the robot ”traverse" a maze.
  - Lots of twists and turns (obstacles)!
  - it can't remember where it's been
  - it can only sense its immediate surroundings

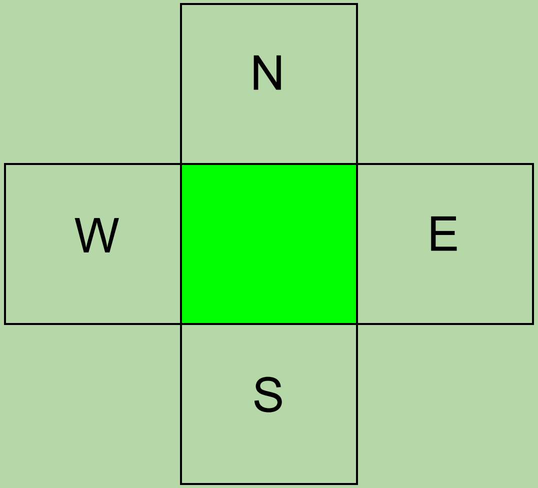
Picobot's Surroundings

- Picobot is only aware of its immediate surroundings.



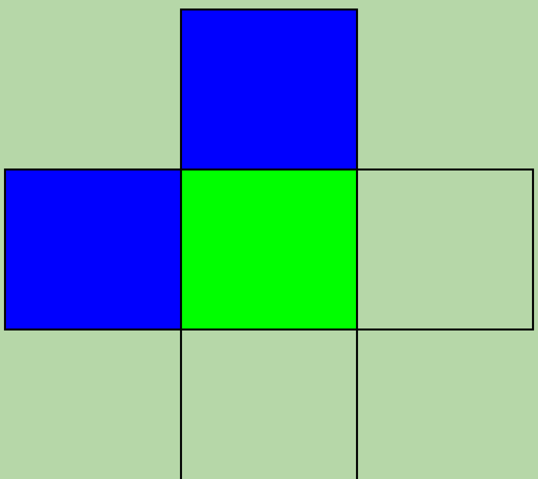
- We express the surroundings using a sequence of four characters...

Surroundings



Picobot can only sense things directly to the N, E, W, and S

For example, here the surroundings are (obstacles to the north and west)



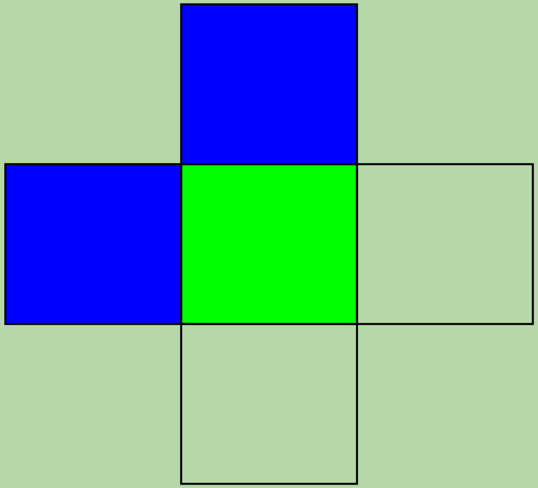
NxWx

N E W S

Surroundings are always in NEWS order.

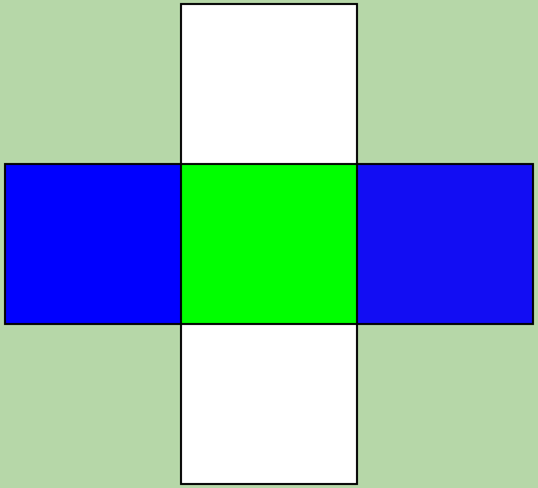
What are these surroundings?

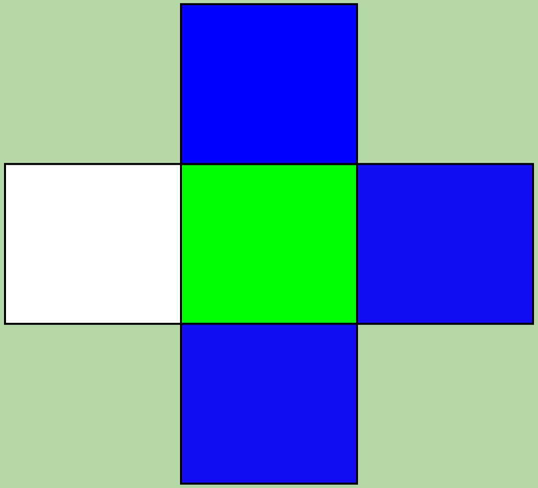
Surroundings are always in NEWS order.



N E W S

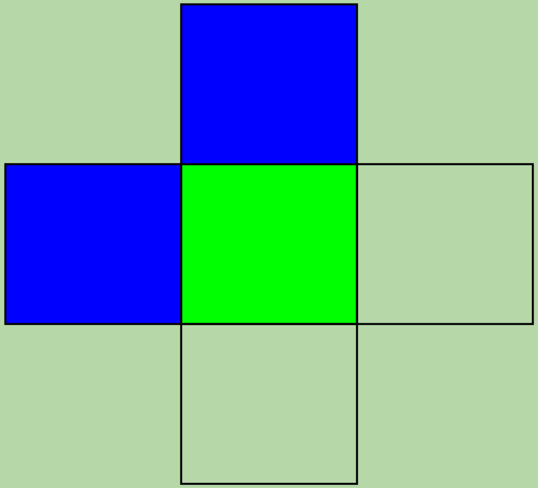
NxWx





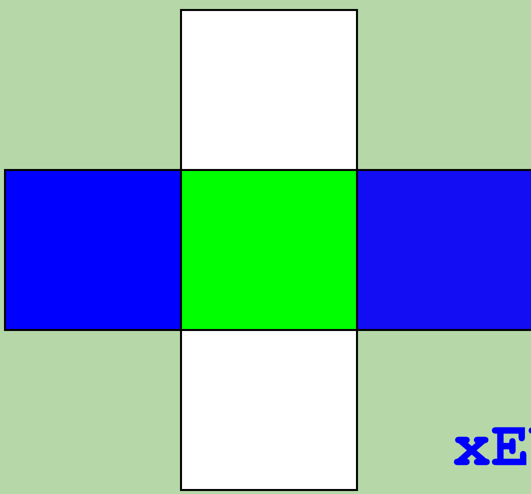
What are these surroundings?

Surroundings are always in NEWS order.

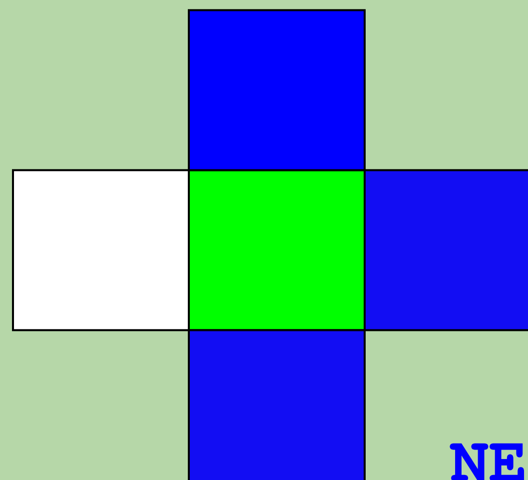


N E W S

NxWx



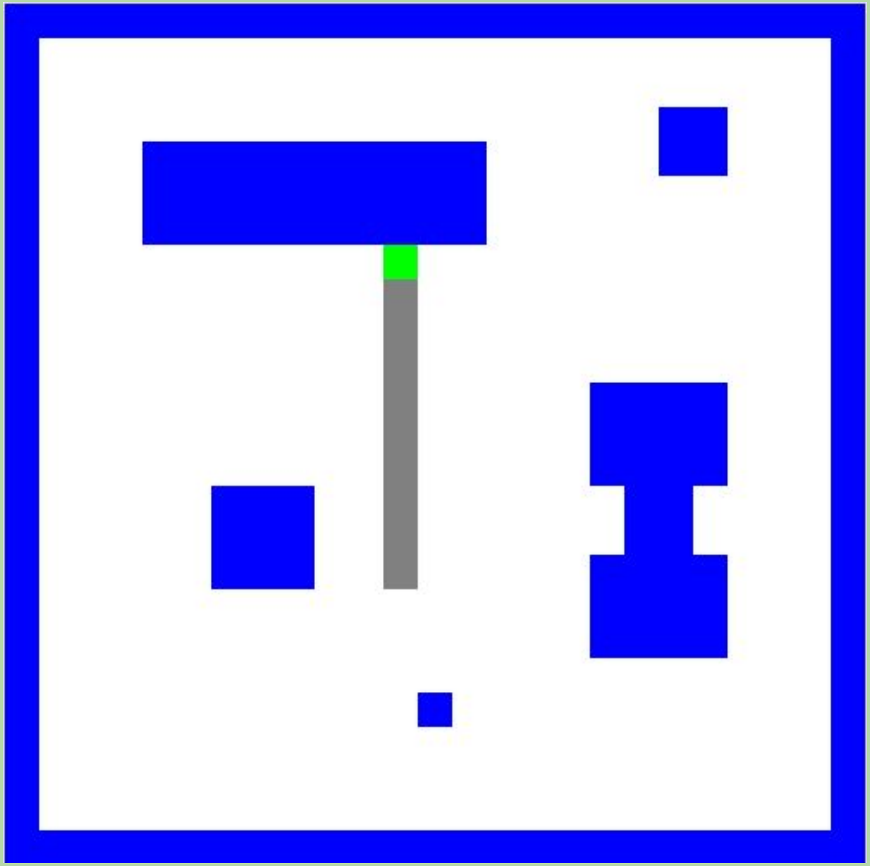
xEWx



NExS

Which of the following describes Picobot's surroundings in the figure below? (gray is not an obstacle)

A. **`eNSw`**
B. **`xNSx`**
C. **`xNxx`**
D. **`Nxxx`**
E. **`NxxS`**



Which of the following describes Picobot's surroundings in the figure below?

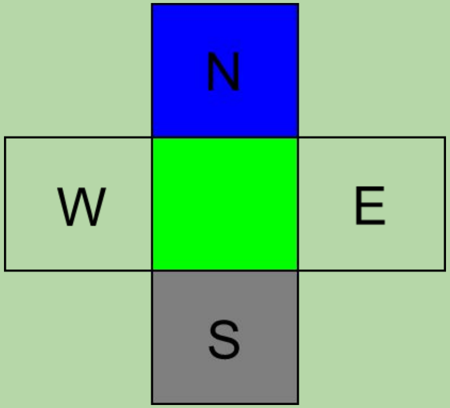
A. **`eNSw`**

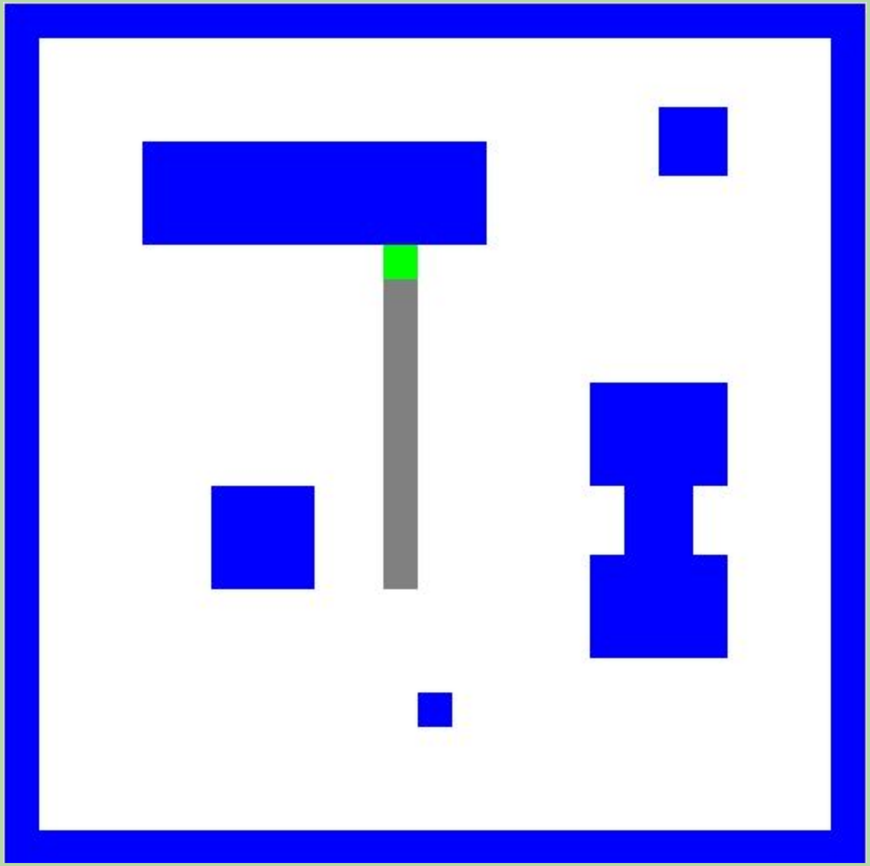
B. **`xNSx`**

C. **`xNxx`**

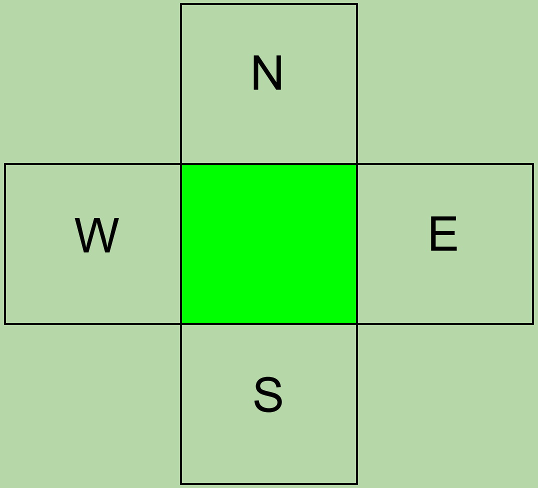
D. **`Nxxx`**

E. **`NxxS`**



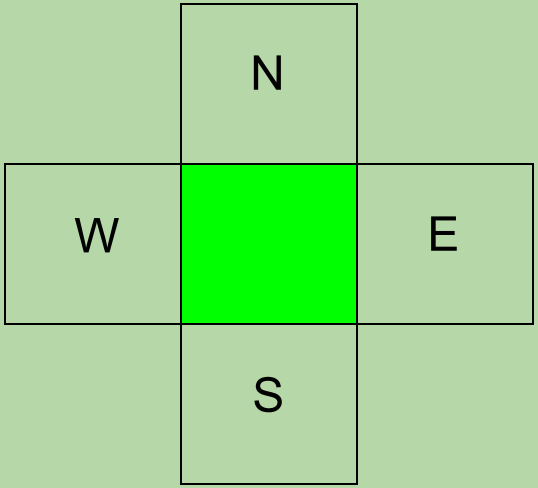


Surroundings



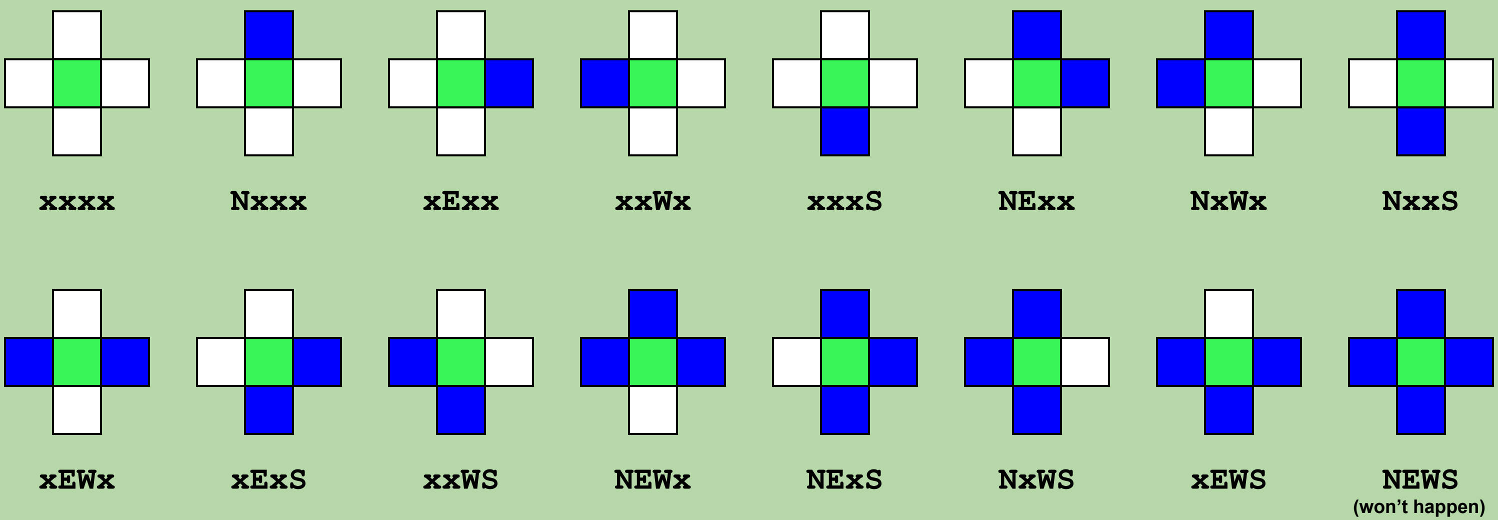
How many distinct surroundings are there?

Surroundings

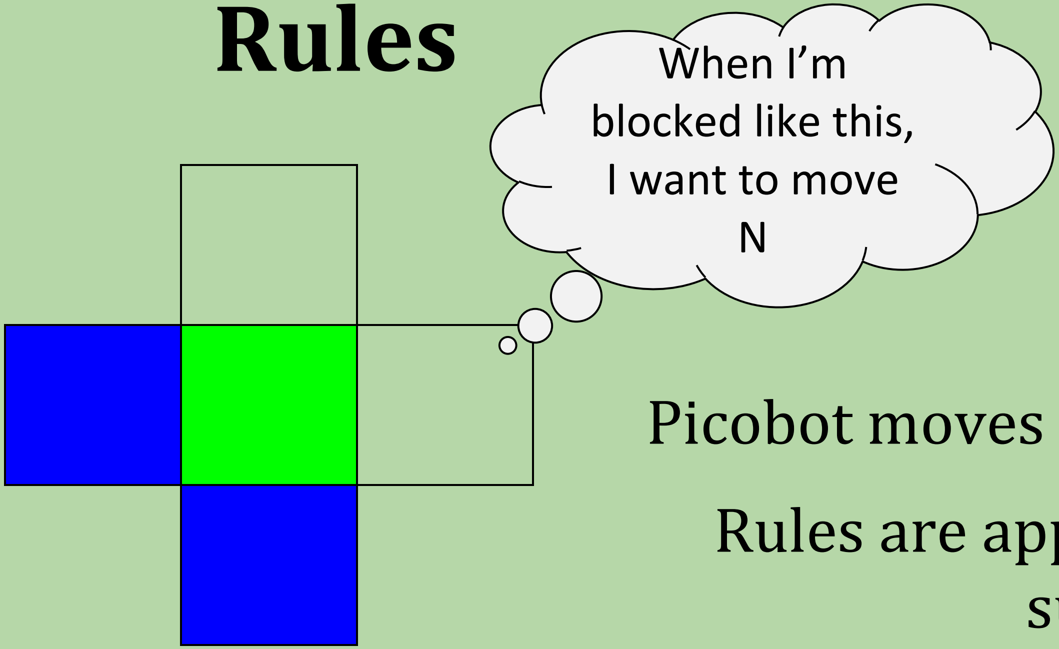


How many distinct surroundings are there?

2^4 == 16 possible …



Rules



I should move N.

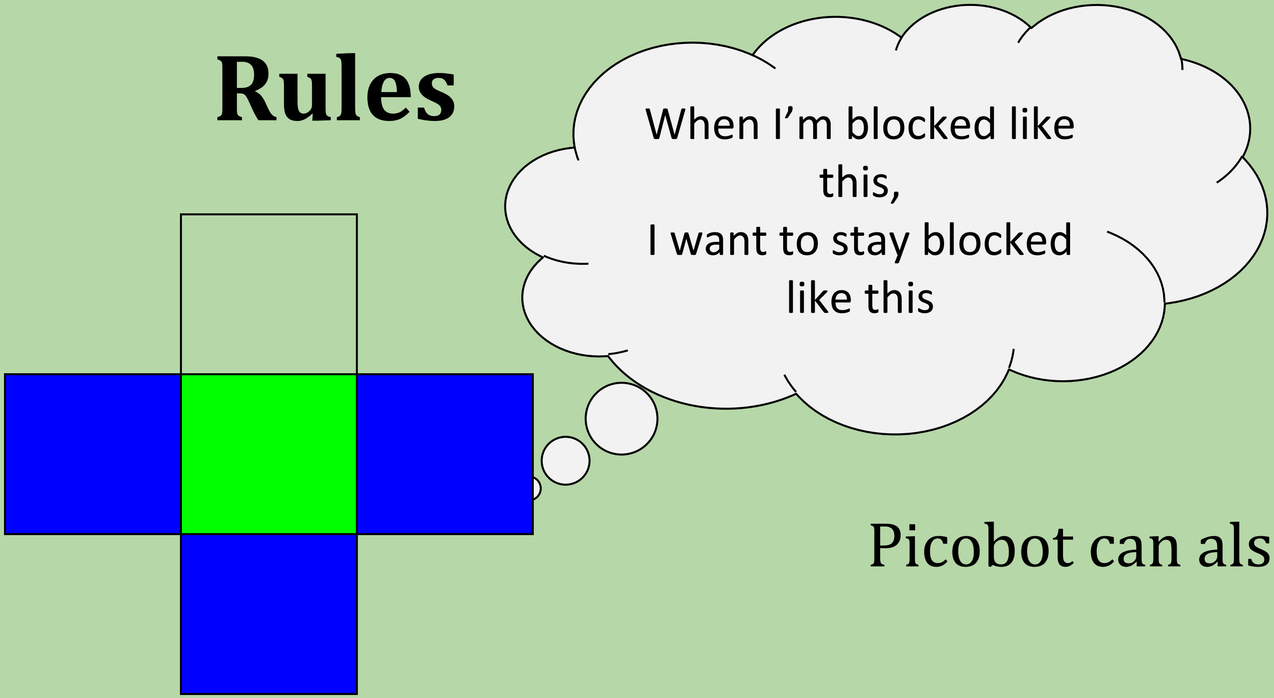
Picobot moves according to a set of rules:

Rules are applied based on picobot's surroundings

surroundings		direction
xx**WS**	→	N

If I see xx**WS**, *Then I move* **N***orth*

Rules

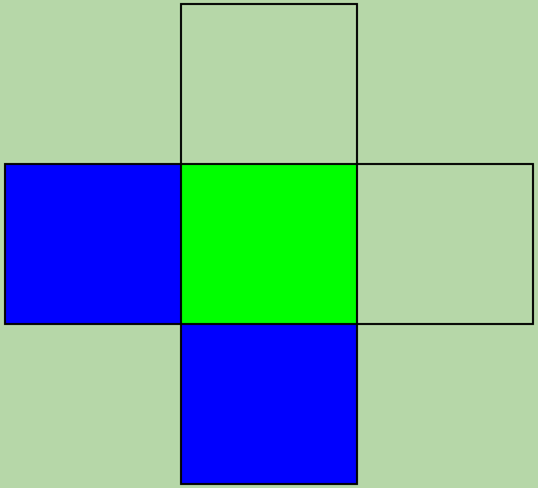


Picobot can also hold still

surroundings		direction
x**EWS**	→	X

If I see x**EWS**, *Then I hold still*

Wildcards



Asterisks * are wild cards. They match walls ***or*** empty space:

surroundings direction

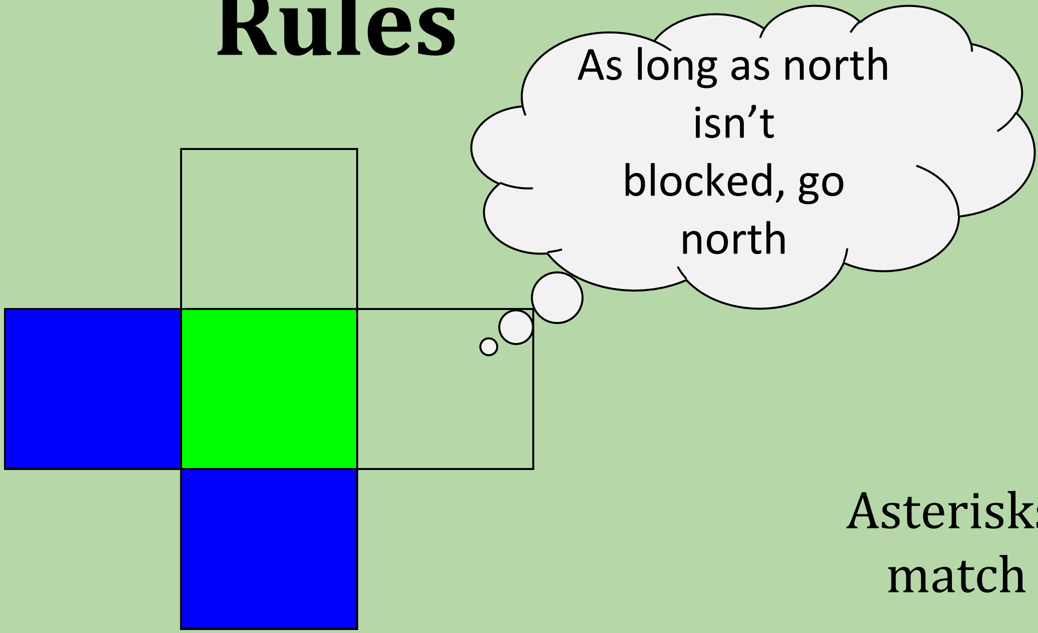
x*** → N

N must be empty

EWS may be wall ***or*** empty space

Wild stars? You should visit Alpha Centauri!

Rules



I should move **N**.

Asterisks * are wild cards. They match walls ***or*** empty space:

surroundings		direction
x***	→	N

If I see North is free (no matter what other walls there are)

Then I move **N***orth*

Picobot Programs

Computational Model

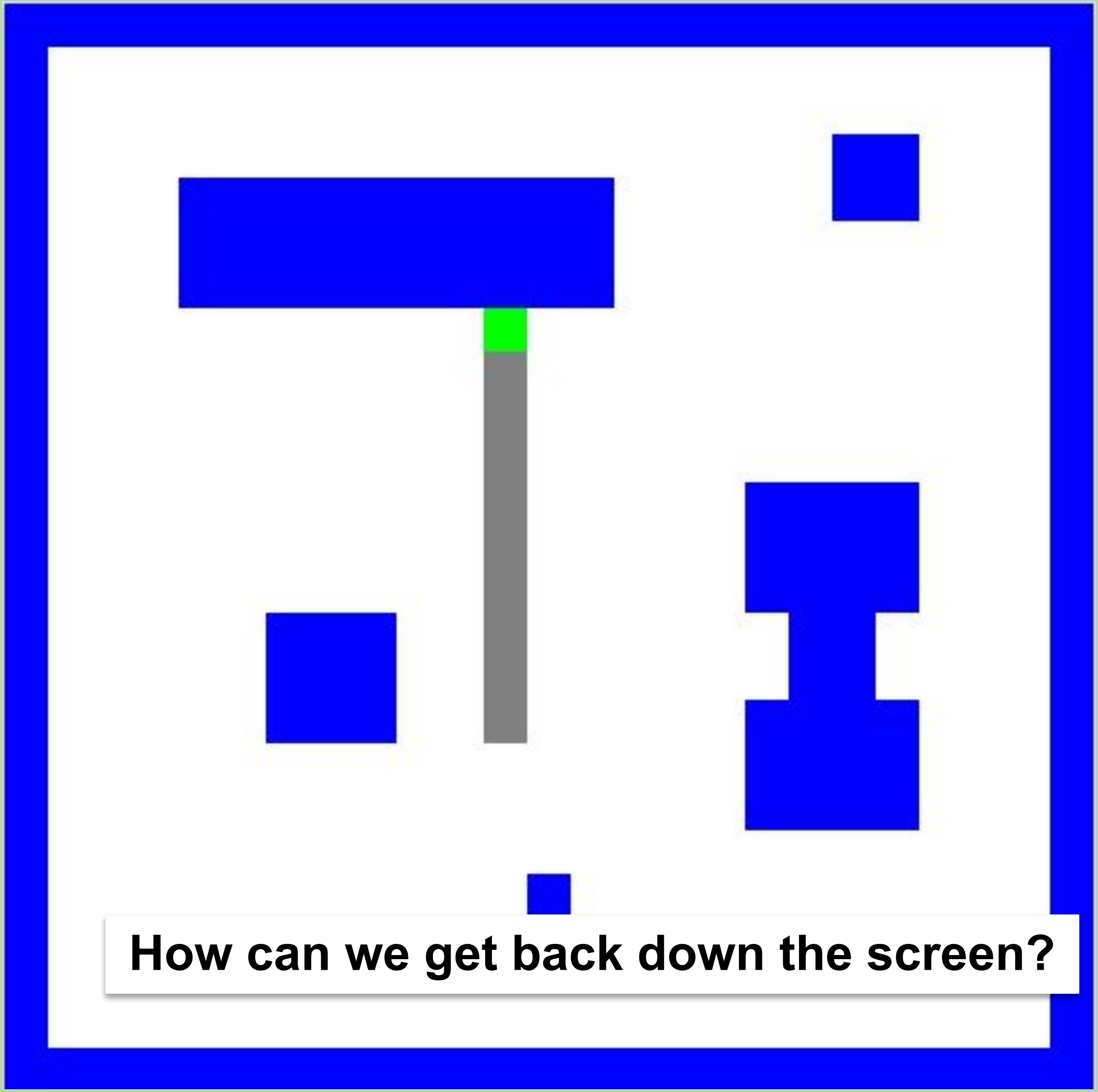
Picobot checks all of its rules.

If it finds a matching rule, that rule runs.

Only one rule is allowed per state and surroundings.

What will this set of rules (program!) do to Picobot?

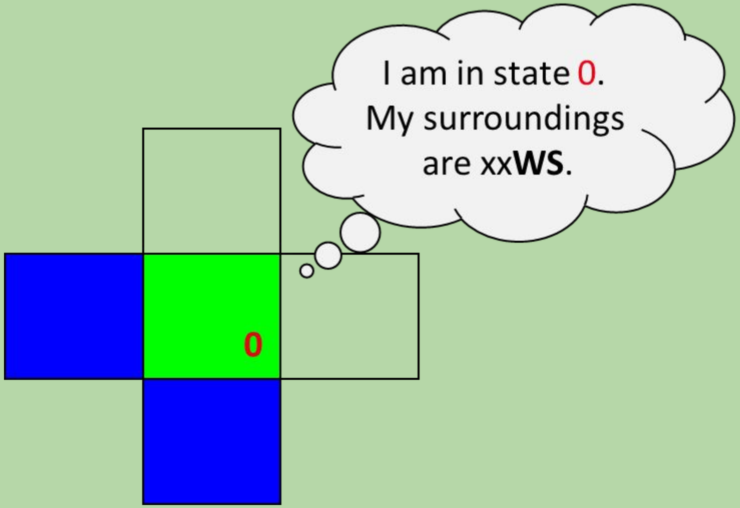
surroundings		direction
x***	->	N
N***	->	X



How can we get back down the screen?

Picobot's State

- Picobot's *state* is a single integer (from 0-99).
- It always starts in state 0.
- The state can be used to capture the current *context* or *subtask*.
  - e.g., "moving east until I get to an obstacle"
  - it's up to us to decide what each state means
- Surroundings + state = all Picobot knows about the world!



Picobot's Rules

- A Picobot *program* is a collection of *rules.*
  - allow us to tell Picobot what to do

- Here's one rule:

state	surroundings		direction to move	new state
0	**xxWS**	**->**	**N**	**0**

if you are in state 0
and
<u>only</u> have obstacles
on your West and South

then move one cell North
and
stay in state 0

- An **X** for the direction means "stay put":

0 xxWS -> X 1

Wildcards

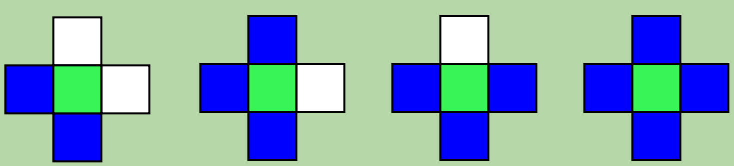
- An asterisk (*) is a wildcard.
  - matches *either* an obstacle or an empty cell.

- Here's a modified version of our earlier rule:

state	surroundings		direction to move	new state
`0`	`**WS`	`->`	`N`	`0`

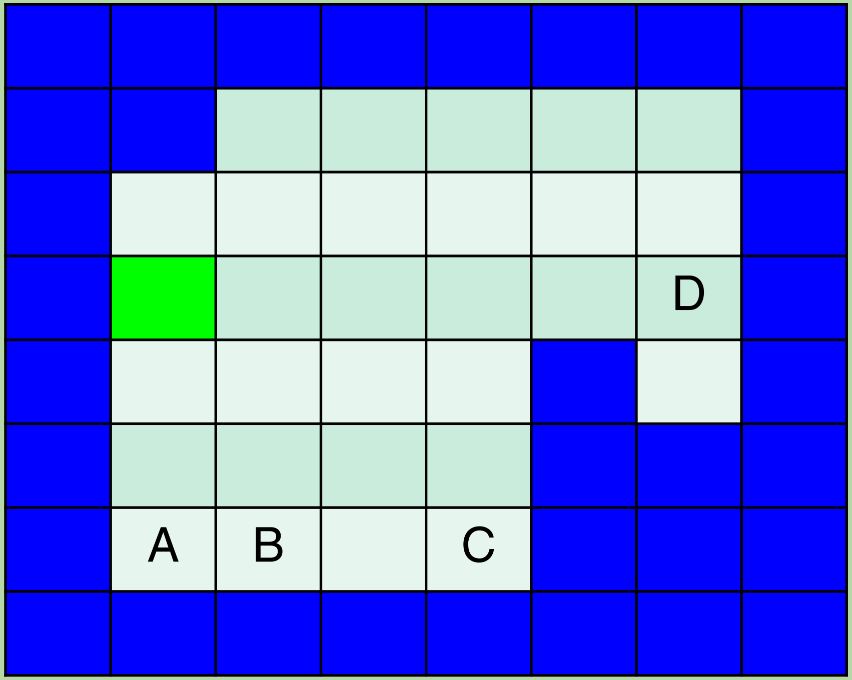
if you are in state 0
and
~~only~~ have obstacles
on your West and South
(regardless of your North or East)

then move one cell North
and
stay in state 0



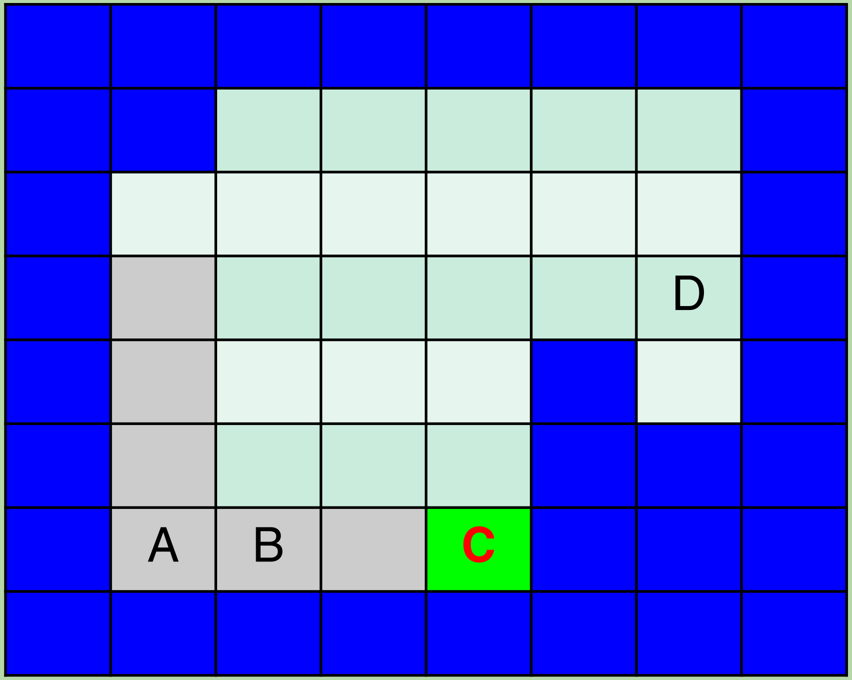
xxWS NxWS xEWS NEWS (won't happen)

Where will Picobot come to a stop?




```
0 ***x -> S 0
0 *x*S -> E 0
0 *E*S -> X 1
```

Where will Picobot come to a stop?



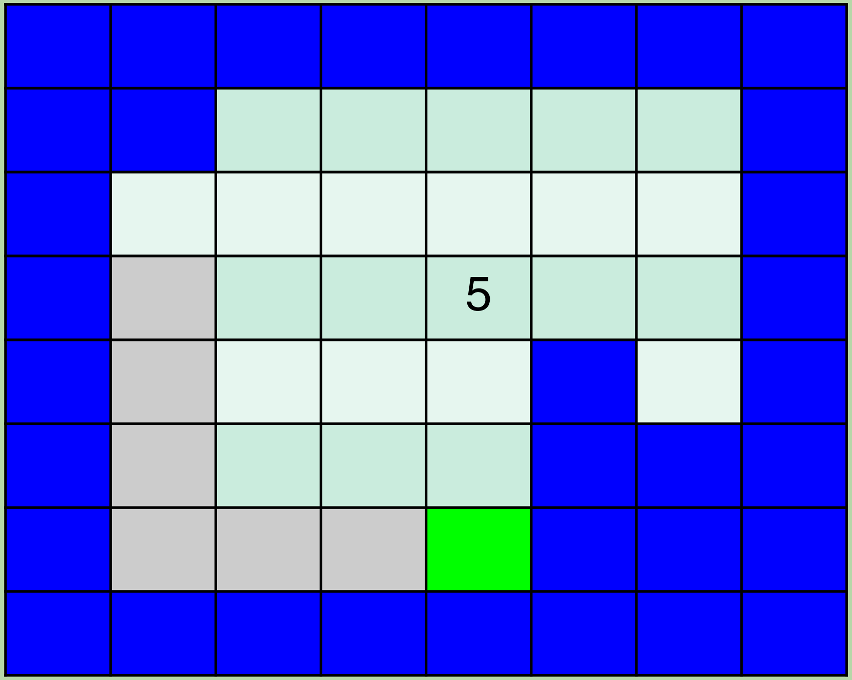

```
0 ***x -> S  0
0 *x*S -> E  0
0 *E*S -> X  1
```

The rules are applied as follows:

- first rule
- first rule
- first rule
- second rule
- second rule
- second rule
- third rule (enters state 1)

No rules for state 1, so we're done.

What rule can we add to the original ones so Picobot will continue until it stops at cell 5?




```
0 ***x -> S 0
0 *x*S -> E 0
0 *E*S -> X 1
```

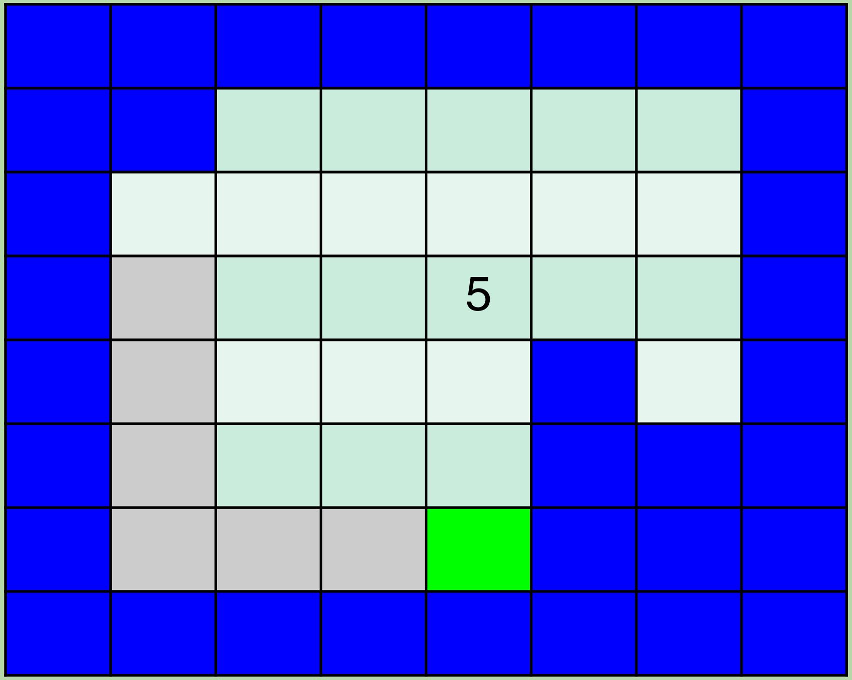
A. `1 *E*S -> N 1`

B. `1 *E** -> N 1`

C. `1 **** -> N 1`

D. more than one of the above will work

What rule can we add to the original ones so Picobot will continue until it stops at cell 5?




```
0 ***x -> S 0
0 *x*S -> E 0
0 *E*S -> X 1
```

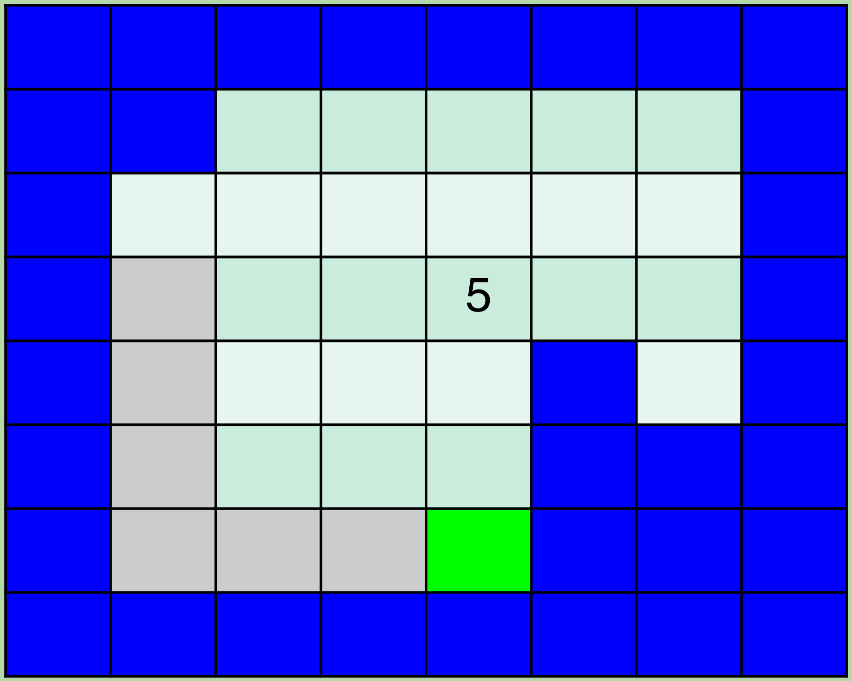
A. `1 *E*S -> N 1`

B. `1 *E** -> N 1`

C. `1 **** -> N 1`

D. more than one of the above will work

Is this set of rules an acceptable alternative?

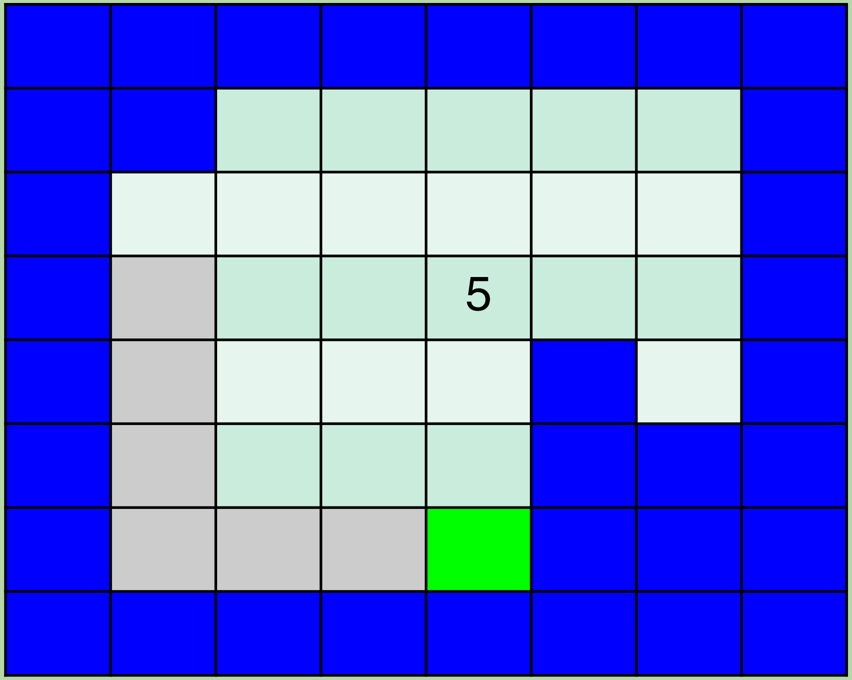



```
0 ***x -> S 0
0 *x*S -> E 0
0 *E*S -> X 1
0 *E** -> N 0
```

A. Yes! (Why?)

B. No! (Why not?)

Is this set of rules an acceptable alternative?



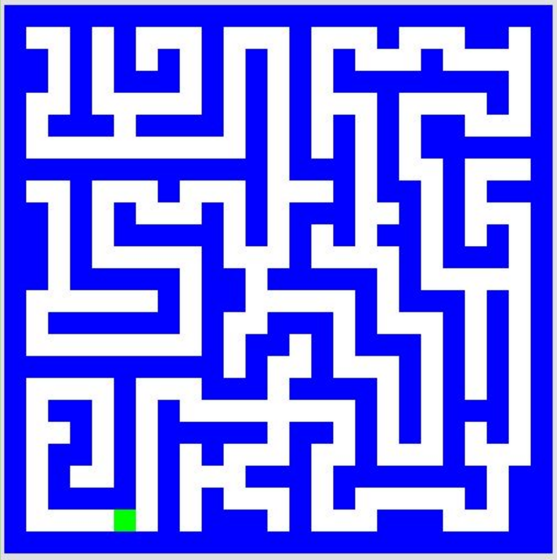

```
0 ***x -> S 0
0 *x*S -> E 0
0 *E*S -> X 1
0 *E** -> N 0
```

A. Yes! (Why?)

B. **No!** **We have *repeat rules* – rules triggered by the same state+surroundings, which causes Picobot to complain.**

(In this case, the new rule is triggered whenever the first rule or third rule is.)

Dealing With a Maze



- What strategy do humans use?
  Keep your right hand on a wall.
- Picobot can use this approach, too!
- To know where its right side is,
  you need four states:
  - facing north (right side is to the east)
  - facing south (right side is to the west)
  - facing east (right side is to the south)
  - facing west (right side is to the north)
- It doesn't matter what number you assign to which state,
  as long as one of them is state 0.

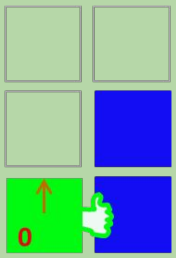
Dealing With a Maze (cont.)

- Let state 0 be facing North.

- Here's one rule for that state:

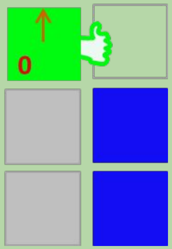
  If you're facing North with the wall on your right and nothing in front of you, go forward.

  `0 xE** -> N 0`



- Let's write a rule for the following:

  If you're facing North but you lose the wall on your right, get over to the wall now!



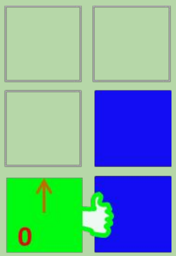
Dealing With a Maze (cont.)

- Let state 0 be facing North.

- Here's one rule for that state:

  If you're facing North with the wall on your right and nothing in front of you, go forward.

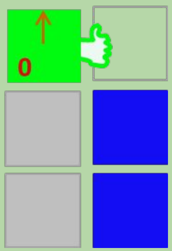
  ```
  0 xE** -> N 0
  ```



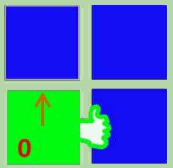
- Let's write a rule for the following:

  If you're facing North but you lose the wall on your right, get over to the wall now!

  ```
  0 *x** -> E 1
  ```



- For the homework, you'll also need:
  - one or two rules for hitting a dead end when facing North
  - similar sets of rules for the other three facing directions



Additional Tips for Picobot problems

- Thinking about the CS questions before diving into the programming will help!
  - Imagine you're blindfolded in the room. How would you solve it?
  - Solve it FIRST in English, then try to figure out the algorithm (don't worry about code!).
  - For each sentence in English, that might be a different state.
  - If you find that rules conflict with each other, you might need a different state.

CS ~ *complexity* science

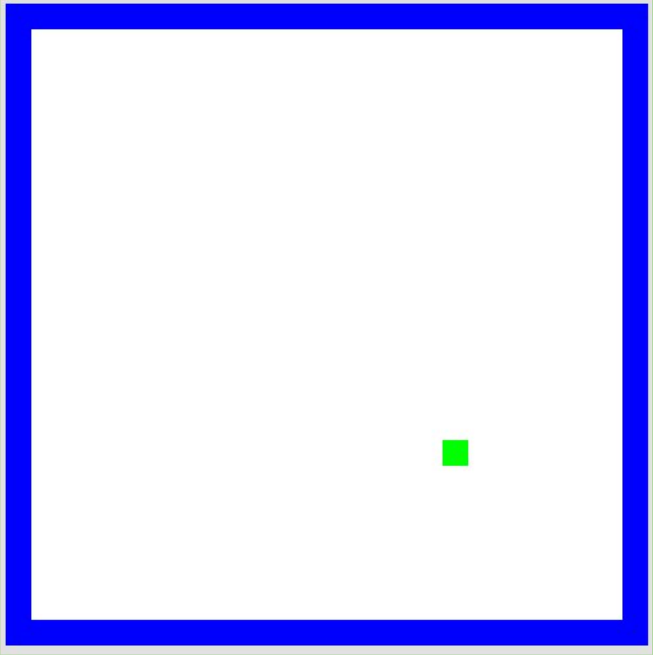
Information is intrinsic to every system…

How can we *benefit from* this information?

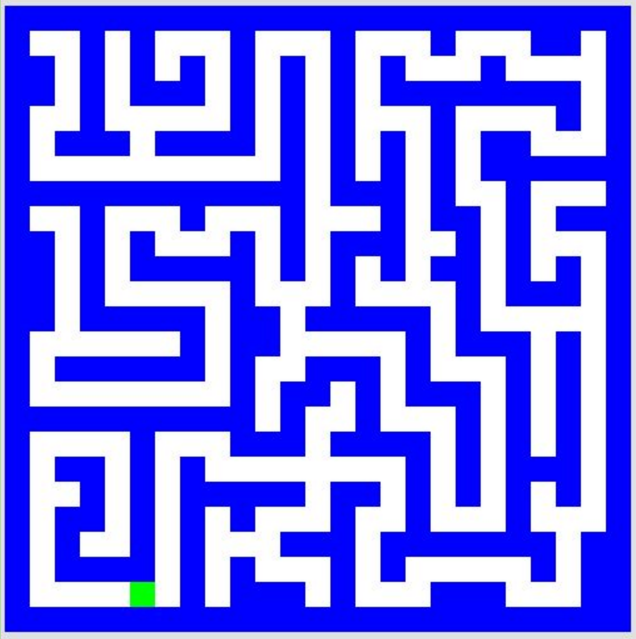
"construct with"

Representing ***it*** … Transforming ***it*** … Measuring ***it***

efficiently? effectively? possibly?



How might we ***measure*** these rooms' **complexity**?



CS ~ *complexity* science

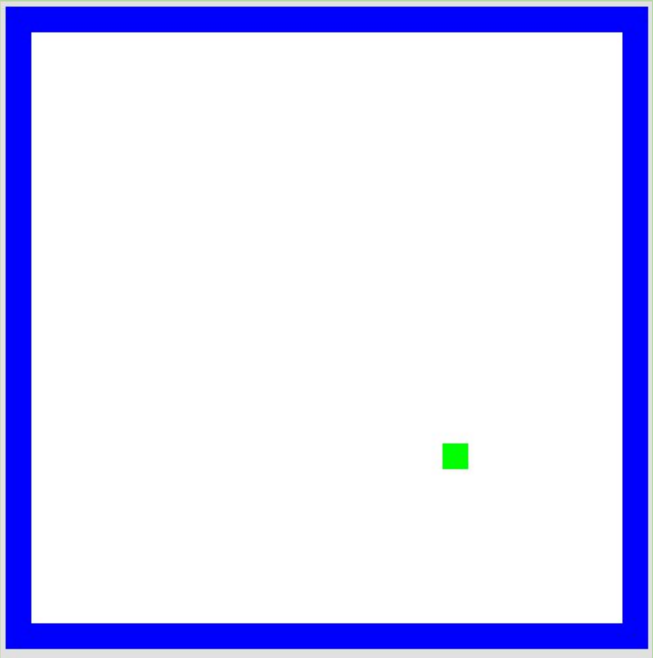
Information is intrinsic to every system…

How can we *benefit from* this information?

"construct with"

Representing ***it*** … Transforming ***it*** … Measuring ***it***

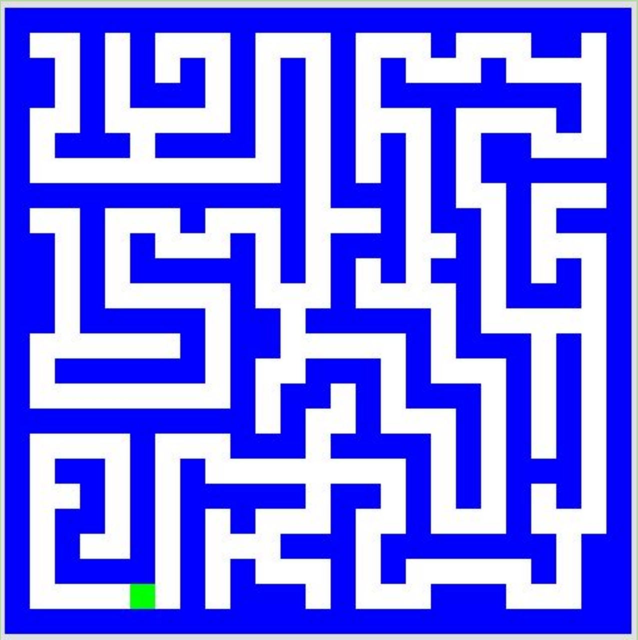
efficiently? effectively? possibly?



How might we ***measure*** these rooms' complexity?

How many states and rules are necessary ?

How much information does each room contain ?



our best: 3 states, 6 rules

our best: 4 states, 8 rules

CS ~ *complexity* science

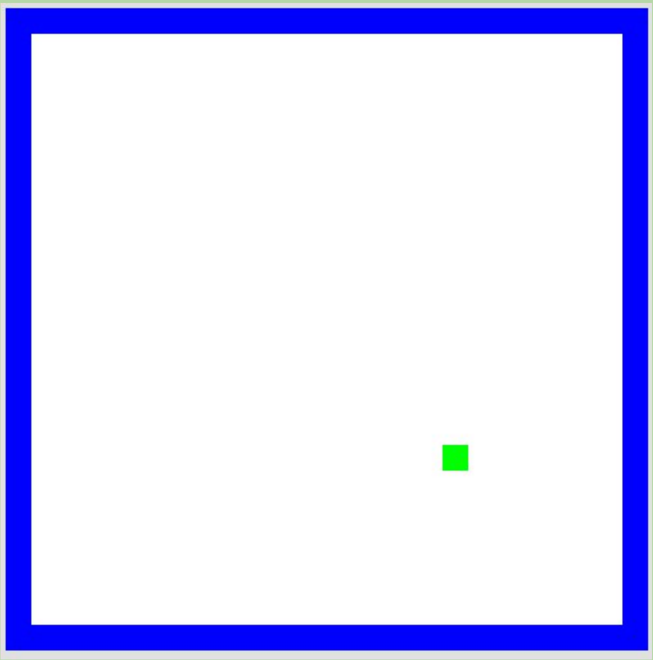
Information is intrinsic to every system…

How can we *benefit from* this information?

"construct with"

Representing ***it*** … Transforming ***it*** … Measuring ***it***

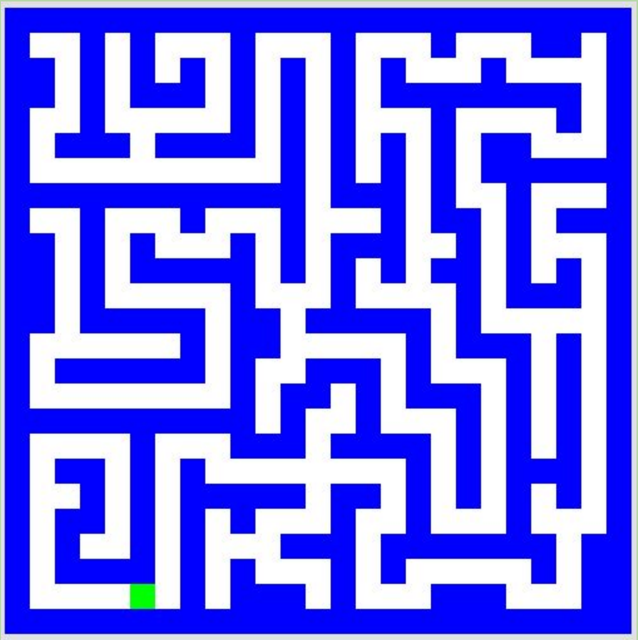
efficiently? effectively? possibly?



How might we ***measure*** these rooms' complexity?

How many states and rules are necessary ?

How much ***information*** does each room contain ?



As a file: ~5000 bytes

As a file: ~20,000 bytes!

What's Next

- Sign-up from Piazza, follow the "First Steps" to take care of other course details https://piazza.com/class/jqbfx4epmck4yd

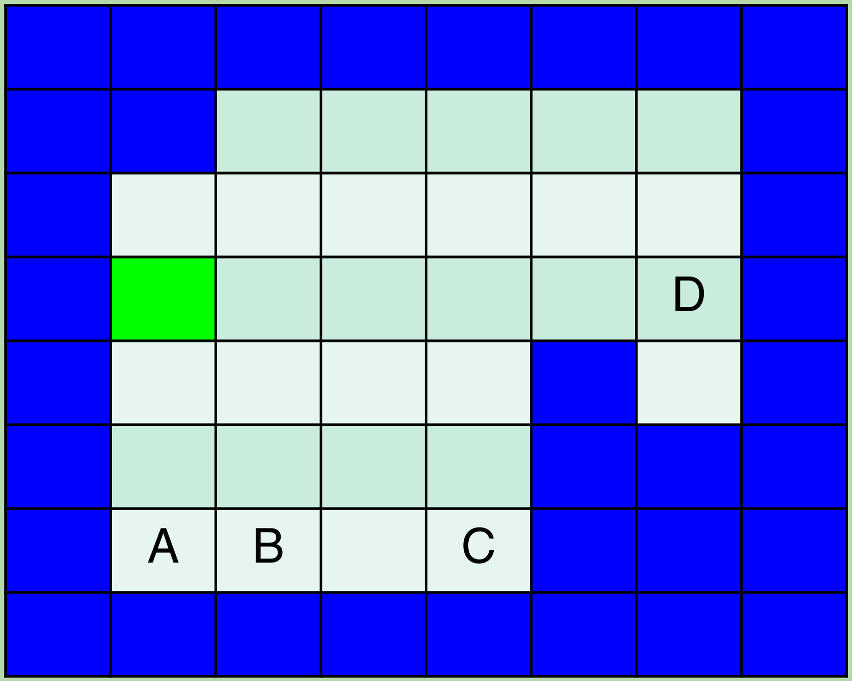
- Sections begin next week. Be sure to attend the Setup Section on Thursday or Friday in order to get a CS login and setup to turn in assignments

- Complete the reading and review these slides before the next lecture
  - Lectures slides will be posted on the website

- **Homework 0**
  - Posted on the website, due next Wednesday

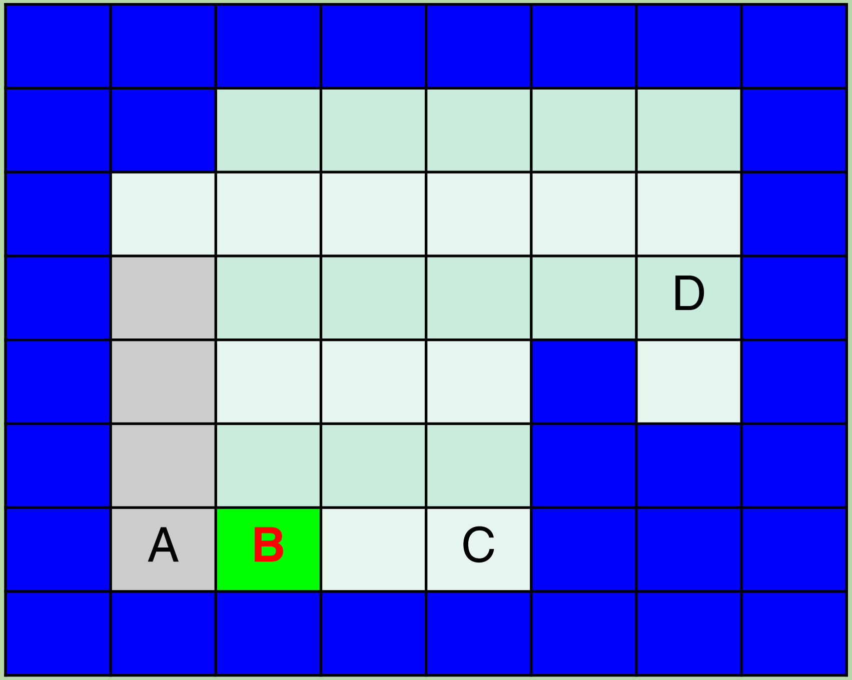
- ***many*** *opportunities for help!*
  - Piazza, Peers (while respecting the Collaboration Policy)
  - Setup Section, Assigned Section, Open Hours

Extra Practice: Where will it stop now?




```
0 ***x -> S  0
0 *x*S -> E  1
0 *E*S -> X  1
```

Extra Practice: Where will it stop now?



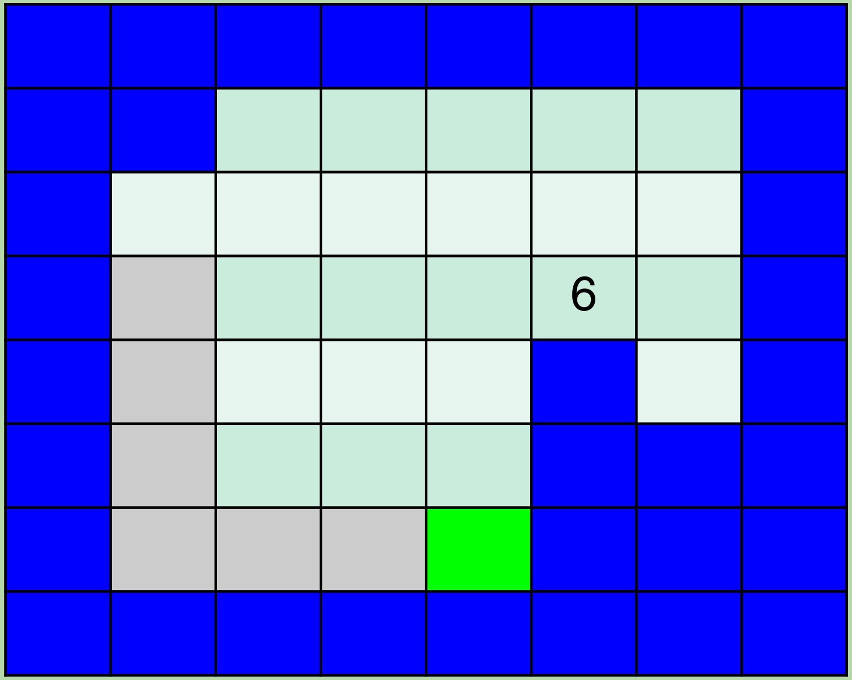

```
0 ***x -> S  0
0 *x*S -> E  1
0 *E*S -> X  1
```

The rules are applied as follows:

- first rule
- first rule
- first rule
- second rule *(enters 1)*

No rules for state 1, so we're done.

Take-Home Challenge!




```
0 ***x ->  S   0
0 *x*S ->  E   0
0 *E*S ->  X   1
```

- What **2 rules** could you add so that Picobot will still travel the same path shown above, but then continue to cell 6 and stop?