Welcome back to CS 5!

**Homework 0**
due Tues. night (11:00pm)

- **Problem 0:** Reading + response...
- **Problem 1:** Four-fours program: Can be done for lab...
- **Problem 2 (rps):** Rock-paper-scissors program
- **Problem 2 (iif):** Interactive Fiction
- **Problems 3-4:** Picobot! empty room (3) maze (4)
- **Problems 5-6:** Picobot! Extra credit challenges
Computer Science Social!

Who: YOU + friends!

What: meet CS majors, prospective CS majors, and CS profs!

When: Thursday August 31st, 4:30 PM-5:30 PM

Where: McGregor courtyard

Why: meet new people, eat trader joe’s snacks, drink boba
Computer Science Social!
Who: YOU + friends!
What: meet CS majors, prospective CS majors, and CS profs!
When: Thursday August 31st, 4:30 PM-5:30 PM
Where: McGregor courtyard
Why: meet new people, eat trader joe’s snacks, drink boba

Right after Thursday lab!
Reminder of your Challenge: Learn names!

Practice – introduce yourself and the person next to you!

● 30 sec: talk to the person next to you, make sure you know each other!
● Around the room: “This is X and I am Y.”
Mike, Shadab, Anirudh, and Tanya ask…

What is the expected amount of homework per week?
Welcome back to CS 5!

Homework 0
due Tues. night (11:00pm)

Problem 0: Reading + response...
Problem 1: Four-fours program: Can be done for lab...
Problem 2rps: Rock-paper-scissors program
Problem 2if: Interactive Fiction
Problems 3-4: Picobot! empty room (3) maze (4)
Problems 5-6: Picobot! Extra credit challenges
Problem 0?

Typically, an article on CS or an application...

Submit a one-paragraph response

A few sentences that raise or address questions, using the article as a guide.

Small part (5 pts)

5 – insightful, careful
4 – thoughtful
3 – complete, on topic
0-2 – less than complete

Does Your Language Shape How You Think?

This week's article might not seem like CS at first...
Seventy years ago, in 1940, a popular science magazine published a short article that set in motion one of the trendiest intellectual fads of the 20th century. At first glance, there seemed little about the article to augur its subsequent celebrity. Neither the title, “Science and Linguistics,” nor the magazine, M.I.T.’s Technology Review, was most people’s idea of glamour. And the author, a chemical engineer who worked for an insurance company and moonlighted as an anthropology lecturer at Yale University, was an unlikely candidate for international superstardom. And yet Benjamin Lee Whorf let loose an alluring idea about language’s power over the mind, and his stirring prose seduced a whole generation into believing that our mother tongue restricts what we are able to think.
But then a remote Australian aboriginal tongue, Guugu Yimidhirr, from north Queensland, turned up, and with it came the astounding realization that not all languages conform to what we have always taken as simply “natural.” In fact, Guugu Yimidhirr doesn’t make any use of egocentric coordinates at all. The anthropologist John Haviland and later the linguist Stephen Levinson have shown that Guugu Yimidhirr does not use words like “left” or “right,” “in front of” or “behind,” to describe the position of objects. Whenever we would use the egocentric system, the Guugu Yimidhirr rely on cardinal directions. If they want you to move over on the car seat to make room, they’ll say “move a bit to the east.” To tell you where exactly they left something in your house, they’ll say, “I left it on the southern edge of the western table.” Or they would warn you to “look out for that big ant just north of your foot.” Even when shown a film on television, they gave descriptions of it based on the orientation of the screen. If the television was facing north, and a man on the screen was approaching, they said that he was “coming northward.”
Seventy years ago, in 1940, a popular science magazine published a short article that set in motion one of the trendiest intellectual fads of the 20th century. At first glance, there seemed little about the article to augur its subsequent celebrity. Neither the title, "Science and Linguistics," nor the magazine, M.I.T.'s Technology Review, was most people’s idea of glamour. And the author, a chemical engineer who worked for an insurance company and moonlighted as an anthropology lecturer at Yale University, was an unlikely candidate for international superstardom. And yet Benjamin Lee Whorf let loose an alluring idea about language’s power over the mind, and his stirring prose seduced a whole generation into believing that our mother tongue restricts what we are able to think.

Q'n of the Day: Which direction are you facing right now?
Welcome back to CS 5!

Homework 0
due Tues. night (11:00pm)

Problem 0: Reading + response...
Problem 1: Four-fours program: Can be done for lab...
Problem 2rps: Rock-paper-scissors program
Problem 2if: Interactive Fiction
Problems 3-4: Picobot! empty room (3) maze (4)
Problems 5-6: Picobot! Extra credit challenges
Lab checkin...

*Lab's goal:* Get things working
Complete 25-50% of the hw

*Finished with lab?* Check in! No *need* to stay longer

Four fours is ∼
*sometimes too many... sometimes too few...*

never enough! -- Prof. Su
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>44/44</td>
<td>34</td>
<td>4! + ( \frac{4!}{4} + 4 )</td>
</tr>
<tr>
<td>2</td>
<td>4 + ( \frac{4}{4} )</td>
<td>35</td>
<td>4! + ( \frac{4!}{4} )</td>
</tr>
<tr>
<td>3</td>
<td>4 + ( \frac{4}{4} )</td>
<td>36</td>
<td>4! + 4(4) - 4</td>
</tr>
<tr>
<td>4</td>
<td>4! - 4(4) - 4</td>
<td>37</td>
<td>4! + ( \frac{4!}{4} )</td>
</tr>
<tr>
<td>5</td>
<td>4 + ( \frac{4}{4} )</td>
<td>38</td>
<td>4! - ( \frac{4!}{4} )</td>
</tr>
<tr>
<td>6</td>
<td>4 + ( \frac{4}{4} )</td>
<td>39</td>
<td>4! - ( \frac{4!}{4} )</td>
</tr>
<tr>
<td>7</td>
<td>4 + ( \frac{4}{4} )</td>
<td>40</td>
<td>4! + 4! - 4</td>
</tr>
<tr>
<td>8</td>
<td>4! - 4(4) + 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>4 + ( \frac{4}{4} )</td>
<td>41</td>
<td>4! - ( \frac{4!}{4} )</td>
</tr>
<tr>
<td>10</td>
<td>4! - 4(4) + 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>4! - (4 + 4 + 4)</td>
<td>44</td>
<td>4! + ( \frac{4!}{4} )</td>
</tr>
<tr>
<td>12</td>
<td>4! - ( \frac{4!}{4} )</td>
<td>45</td>
<td>4! + ( \frac{4!}{4} )</td>
</tr>
<tr>
<td>13</td>
<td>( \frac{4}{4} ) + 4 + 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>( \frac{4}{4} ) + 4 + 4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

F. Su
15  \[ \frac{4(4)}{3} \left( \frac{4}{3} \right) \]
16  \[ \frac{4(4)}{3} \]
17  \[ \frac{4(4)}{3} \]
18  \[ (\frac{4}{3}) \frac{4}{3} + 4 \]
19  \[ 4 - \frac{4}{3} \]
20  \[ \frac{4}{3} + \frac{4}{3} \]
21  \[ \frac{4}{3} + \frac{4}{3} \]
22  \[ \frac{4}{3} + \frac{4}{3} \]
23  \[ \frac{4}{3} + \frac{4}{3} \]
24  \[ \frac{4}{3} + \frac{4}{3} \]
25  \[ \frac{4}{3} + \frac{4}{3} \]
26  \[ \frac{4}{3} + \frac{4}{3} \]
27  \[ \frac{4}{3} + \frac{4}{3} \]
28  \[ \frac{4}{3} + \frac{4}{3} \]
29  \[ \frac{4}{3} + \frac{4}{3} \]
30  \[ \frac{4}{3} + \frac{4}{3} + \frac{4}{3} \]
31  \[ \frac{4}{3} + \frac{4}{3} + \frac{4}{3} \]
32  \[ \frac{4}{3} + \frac{4}{3} + \frac{4}{3} \]
33  \[ \frac{4}{3} + \frac{4}{3} + \frac{4}{3} \]
Last time...

CS != Programming

What is CS?

CS is the study of complexity

How can it be done?
How well can it be done?
Can it be done at all?

Can you solve this problem?
Can you create a process to solve such problems?

How quickly can you find solutions?
Do you have the “best” solution?

Is every problem solvable?
Is there a way to tell? There isn’t always!

But only one is programming. Do you see which?
# RPS example starting point

```python
import random

print("Welcome to rock/paper/scissors, human!\n")

comp = random.choice(['rock', 'paper', 'scissors'])
user = input(" +++ Choose wisely: ")

print(" You chose", user)
print(" I chose", comp)

if user == 'rock':
    if comp == 'paper':
        print(" paper defeats rock - I win!")
```
import random

user = input( "Choose your weapon! " )

comp = random.choice( [ 'rock', 'paper', 'scissors' ] )

print('user (you) chose:', user)

print('comp (me!) chose:', comp)

if user == 'rock' and comp == 'paper':
    print('The result is, YOU LOSE.

    you're a CS5 grader, then YOU WIN : )')
Please pass these **Forward & Right ... !**

*(take a photo, if you'd like!)*
Create a short text-adventure in Python...

Use at least five control structures with decisions: (if/elif/else)

Use lists, strings, and dictionaries as you like ... not required ...

We look forward to adventuring!
Welcome back to CS 5!

Homework 0

due Tues. night (11:00pm)

Problem 0: Reading + response...

Problem 1: Four-fours program: Can be done for lab...

Problem 2rps: Rock-paper-scissors program

Problem 2if: Interactive Fiction

Problems 3-4: Picobot! empty room (3) maze (4)

Problems 5-6: Picobot! Extra credit challenges
Welcome back to CS 5!

Homework 0

due Tues. night (11:00pm)

Problem 0: Reading + response...

Problem 1: Four-fours program: Can be done for lab...

Problem 2rps: Rock-paper-scissors program

Problem 2if: Interactive Fiction

Problems 3-4: Picobot! empty room (3) maze (4)

Problems 5-6: Picobot! Extra credit challenges
Another language!

Let's not only add another language...

... but also make it half the hw!

Even with three eyes, I must be misreading this!
Another language already?

**Python**

*General-purpose language*

you might see 50% by the end of the term

even then, <1% of its libraries!

**Picobot**

*Special-purpose language*

you'll see 100% in the next 10 minutes

The Picobot simulator

[www.cs.hmc.edu/picobot](http://www.cs.hmc.edu/picobot)
HW problems 3 and 4: Picobot!

**Goal:** full-room coverage with only *local sensing*...

Inspiration?
HW problems 3 and 4: Picobot!

**Goal:** full-room coverage with only *local sensing*...

The Roomba! can't tell "vacuumed" from "unvacuumed" area

Let's see it!
Surroundings

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

For example, here its surroundings are

Surroundings are always in NEWS order.
What are these surroundings?

Surroundings are always in NEWS order.

Wow - this one is disgusting!
Surroundings

How many distinct surroundings are there?

\[ 2^4 = 16 \text{ possible} \]

Aargh!
Picobot's memory is a single number, called its state.

State is the *internal context* of a computation, i.e., its subtask.

Picobot always starts in state 0.

*State* and *surroundings* represent everything Picobot knows about the world.
## Picobot programming ~ rules

<table>
<thead>
<tr>
<th>current state</th>
<th>surroundings</th>
<th>step</th>
<th>new state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Nxxx</td>
<td>S</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>xxxx</td>
<td>N</td>
<td>0</td>
</tr>
</tbody>
</table>

**Notes**

Picobot checks its rules from the top each time. *When it finds a matching rule*, that rule runs.

---

*These two rules are a complete Picobot program*
Picobot programming ~ *rules*

<table>
<thead>
<tr>
<th>current state</th>
<th>surroundings</th>
<th>step</th>
<th>new state</th>
</tr>
</thead>
<tbody>
<tr>
<td>rule A</td>
<td>0</td>
<td>Nxxx</td>
<td>S</td>
</tr>
<tr>
<td>rule B</td>
<td>0</td>
<td>xxxx</td>
<td>N</td>
</tr>
</tbody>
</table>

These two rules are a complete Picobot program.

**Notes**

Picobot checks its rules from the top each time. *When it finds a matching rule*, that rule runs.

---

```
Start

Step 1

Step 2

Step 3

Step 4

...?
```
Picobot programming ~ *rules*

<table>
<thead>
<tr>
<th>current state</th>
<th>surroundings</th>
<th>step</th>
<th>new state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Nxxx</td>
<td>S</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>xxxx</td>
<td>N</td>
<td>0</td>
</tr>
</tbody>
</table>

These two rules are a complete Picobot program.

**Notes**

Picobot checks its rules from the top each time. *When it finds a matching rule*, that rule runs.

---

Start

![Start](image)

Step 1

![Step 1](image)

Step 2

![Step 2](image)

Step 3

![Step 3](image)

Step 4

![Step 4](image)

...
## Picobot programming ~ *rules*

<table>
<thead>
<tr>
<th>current state</th>
<th>surroundings</th>
<th>step</th>
<th>new state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Nxxx</td>
<td>S</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>xxxx</td>
<td>N</td>
<td>0</td>
</tr>
</tbody>
</table>

these two rules are a complete Picobot program

---

### Notes

Picobot checks its rules from the top each time. *When it finds a matching rule*, that rule runs.

---

![Picobot steps](image)
Picobot programming ~ *rules*

<table>
<thead>
<tr>
<th>rule A</th>
<th>current state</th>
<th>surroundings</th>
<th>step</th>
<th>new state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Nxxx</td>
<td>S</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>rule B</td>
<td>0</td>
<td>xxxx</td>
<td>N</td>
<td>0</td>
</tr>
</tbody>
</table>

These two rules are a complete Picobot program.

**Notes**

Picobot checks its rules from the top each time.  
*When it finds a matching rule*, that rule runs.

...
Picobot programming ~ *rules*

**Notes**

Picobot checks its rules from the top each time. *When it finds a matching rule*, that rule runs.

These two rules are a complete Picobot program.

---

**Start**

- Current state: 0
- Surroundings: Nxxx
- Step: S
- New state: 0

**Step 1**

- Current state: 0
- Surroundings: xxxx
- Step: N
- New state: 0

**Step 2**

**Step 3**

**Step 4**

These cycle back and forth...
Rules

I am in state 0. My surroundings are xxWS.

Picobot acts through a set of rules

Each rule expresses your intent for Picobot!

If Picobot's in state 0 seeing xxWS,

Then move North, and "change" to state 0.
Asterisks * are wild cards. They match walls or empty space:

I only care about NORTH being EMPTY

N must be empty

EWS may be wall or empty space

8 surroundings in one rule

that's it!
1. Run Picobot! Which rule A, B, or C runs first? **B**
   1a. How many times does rule (A) run? _______
   1b. How many times does rule (B) run? _______
   1c. How many times does rule (C) run? _______

2. Picobot stops when no rule matches. *Where does it stop?*

3. Add a rule so that Picobot continues *back upwards!*
   *Hint: A step of *x* stays in place...*

Extra #1 Rule A has a bug! What is it?
Extra #2 Add rules to finish exploring the empty room from any starting point...
Extra #3 How to do this in only 6 rules total?!
1. Run Picobot! Which rule A, B, or C runs first? B
   1a. How many times does rule (A) run? 1
   1b. How many times does rule (B) run? 3
   1c. How many times does rule (C) run? 4

2. Picobot stops when no rule matches. Where does it stop?

3. Add a rule so that Picobot continues back upward!

Extra #1: Rule A has a bug! What is it? should be N**x**
Extra #2: Add rules to finish exploring the empty room from any starting point... hw0pr3
Extra #3: How to do this in only 6 rules total?!
**Warning!  What's wrong here?**

<table>
<thead>
<tr>
<th>state</th>
<th>surroundings</th>
<th>direction</th>
<th>new state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x***</td>
<td>S</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>***x</td>
<td>N</td>
<td>0</td>
</tr>
</tbody>
</table>

**Notes**

Picobot checks its rules from the top each time. *When it finds a matching rule*, that rule runs.
Warning! What's wrong here?

<table>
<thead>
<tr>
<th>state</th>
<th>surroundings</th>
<th>direction</th>
<th>new state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>***x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

These two situations COULD BE the same!

Picobot checks its rules from the top each time. When it finds a matching rule, that rule runs.

Notes

There can only be **ONE** rule per situation!

and a "situation" is state and surroundings
CS ~ *Complexity Science*

- **problem 3**
  - Shortest Picobot program: *6 rules*

- **problem 4**
  - Shortest Picobot program: *8 rules*

- **pr. 5 (extra!)**

- **pr. 6 (extra!)**
Maze strategies?
Maze solving algorithm

There are a number of different maze solving algorithms, that is, automated methods for the solving of mazes. The random mouse, wall follower, Pledge, and Trémaux algorithms are designed to be used inside the maze by a traveler with no knowledge of the maze, whereas the dead-end filling and shortest path algorithms are designed to be used by a person or computer program that can see the whole maze at once.

Mazes containing no loops are known as "standard", or "perfect" mazes, and are equivalent to a tree in graph theory. Thus many maze solving algorithms are closely related to graph theory. Intuitively, if one pulled and stretched out the paths in the maze in the proper way, the result could be made to resemble a tree.[1]

Contents [hide]
1 Random mouse algorithm

Right Hand Rule
Maze strategies?  

Right Hand Rule

Keep your "right hand" on the wall, Picobot!

Why might this be difficult for Picobot?
Maze strategies?

Right Hand Rule

Keep your "right hand" on the wall, Picobot!

We'll need to use state to represent the direction Picobot is facing.
Suppose Picobot wants to traverse a maze *with its right hand always on the wall*...

(A) **CORRIDOR** rule

*If you're facing N with a wall at right and space ahead* then *go forward*”

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

(B) **INTERSECTION** rule

“If you're facing North and lose the wall, then get over to the wall now!”

```
0  ->
```

(C) **DEAD END** rule

Write 1 rule to tell Picobot to do the right thing if it hits a dead end.

Repetition of this IDEA for all four states, representing all four *facing directions.*
Suppose Picobot wants to traverse a maze *with its right hand always on the wall*...

(A) CORRIDOR rule

*If you're facing N with a wall at right and space ahead then go forward*”

(B) INTERSECTION rule

“If you're facing North and lose the wall, then get over to the wall now!”

(C) DEAD END rule

Write 1 or 2 rules to tell Picobot to do the right thing if it hits a dead end.

Repeat this IDEA for all four states, representing all four *facing directions.*
Hooray!?!?

Is it working?

# twelve-rule maze-solver:

Enter rules for Picobot

Be sure to hit "Enter rules" after making

Messages
Come to tutoring hours!
Post questions to piazza...

Happy Picobotting!

You are not alone!

And, good luck with the **adventure** of Python!