They call that an alien?

Spock mind-melds three-eyed aliens!

Provably.

Lab lookback...

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

Finished with lab? OK! No need to stay longer

Four fours is ~
sometimes too many...
othertimes too few...

Email help: Start w/ Piazza...

for many questions, Piazza is a great resource:

In-person help: "grutoring"

every day there are tutoring hours in the LAC lab

this link: Useful/Helpful
- GradeScope
- CS 5 Piazza
- Grutoring!

this Q&A page

DON'T sign up JUST GO!
Grutors there to provide support...
Welcome back to CS 5!

Homework 1

due Mon. night (11:59pm)

- Problem 0: Reading + response...
- Problem 1: Four-fours program: Can be done for lab...
- Problem 2: Rock-paper-scissors program (Maybe done already!)
- Problems 3-4: Picobot! empty room (3) maze (4)

Welcome back to CS 5!

Homework 1

due Mon. night (11:59pm)

- Problem 0: Reading + response...
- Problem 1: Four-fours program: Can be done for lab...
- Problem 2: Rock-paper-scissors program (Maybe done already!)
- Problems 3-4: Picobot! empty room (3) maze (4)

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

This week’s article might not seem like CS at first…

Welcome back to CS 5!

Homework 1

due Mon. night (11:59pm)

- Problem 0: Reading + response...
- Problem 1: Four-fours program: Can be done for lab...
- Problem 2: Rock-paper-scissors program (Maybe done already!)
- Problems 3-4: Picobot! empty room (3) maze (4)

Welcome back to CS 5!

Homework 1

due Mon. night (11:59pm)

- Problem 0: Reading + response...
- Problem 1: Four-fours program: Can be done for lab...
- Problem 2: Rock-paper-scissors program (Maybe done already!)
- Problems 3-4: Picobot! empty room (3) maze (4)

Welcome back to CS 5!

Homework 1

due Mon. night (11:59pm)

- Problem 0: Reading + response...
- Problem 1: Four-fours program: Can be done for lab...
- Problem 2: Rock-paper-scissors program (Maybe done already!)
- Problems 3-4: Picobot! empty room (3) maze (4)

Welcome back to CS 5!

Homework 1

due Mon. night (11:59pm)

- Problem 0: Reading + response...
- Problem 1: Four-fours program: Can be done for lab...
- Problem 2: Rock-paper-scissors program (Maybe done already!)
- Problems 3-4: Picobot! empty room (3) maze (4)
Last time...

What is programming?

Programming as recipe-writing vs. Programming as learning a foreign language

1) Expect it to be different!
2) Don’t memorize anything!
3) Immerse == Experiment!

What about the *Python* programming language?

One possible relationship...
**Python!**

One possible relationship...

Happy co-existence...

*It can even be comfy!*

---

The **foreign language** of python...

<table>
<thead>
<tr>
<th>syntax</th>
<th>semantics</th>
<th>intent</th>
</tr>
</thead>
<tbody>
<tr>
<td>How it looks</td>
<td>What it does</td>
<td>What it should do</td>
</tr>
</tbody>
</table>

- **syntax**
  - how punctuation is used
  - the language **keywords** used
  - use of whitespace

- **semantics**
  - peculiarities of formatting
  - how behavior is affected...

- **intent**
  - How it looks
  - What it does
  - What it should do

---

Flowchart...

A graphical view of a program’s **semantics**

**machine-produced output**
The challenge of programming...

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
<th>Intent</th>
</tr>
</thead>
<tbody>
<tr>
<td>How it looks</td>
<td>What it does</td>
<td>What it should do</td>
</tr>
</tbody>
</table>

human-typed input ➔ machine-produced output ➔ human-desired output

This is somehow familiar...?!

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.' )
else:
    print( 'The result is, YOU WIN!' )
print( 'unless you’re a CS 5 grader, then YOU WIN!' )

(1) Find and correct as many errors as you can in this code:

(2) This one line does three things... what are they?

(3) Extra! Can you find 7 punctuation marks used in more than one way here?

Name(s):

Syntax challenge!

HW1PR2a: RPS...

# RPS example starting point
import random
user = input( "Choose wisely: ")
comp = random.choice([ 'rock', 'paper', 'scissors' ])
print( 'You chose', user )
print( 'I chose', comp )
if user == 'rock':
    if comp == 'paper':
        print( "paper defeats rock - I win!" )
    else:
        print( "You win!" )
print( "unless you’re a CS 5 grader, then YOU WIN!" )

(1) Find and correct as many errors as you can in this code:

(2) This one line does three things... what are they?

(3) Extra! Can you find 7 punctuation marks used in more than one way here?

Name(s):

Syntax challenge!

HW1PR2b: Your Quest!

Create a short text-adventure in Python...

We look forward to adventuring!
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!

---

**HW problems 3 and 4: Picobot!**

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

---

**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.
**Surroundings**

How many distinct surroundings are there?

- N
- W
- E
- S

---

**State**

I am in state 0. My surroundings are xxWS.

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**

- **Current state**: 0
- **Surroundings**: Nxxx
- **Direction**: S
- **New state**: 0

*Rule (A)*

- **Current state**: 0
- **Surroundings**: xxxx
- **Direction**: N
- **New state**: 0

*Rule (B)*

*Notes*

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

---

**Rules**

I am in state 0. My surroundings are xxWS.

Picobot acts through a *set of rules*

- *Each rule expresses your intent* for Picobot!

<table>
<thead>
<tr>
<th>current state</th>
<th>surroundings</th>
<th>direction</th>
<th>new state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>xxWS</td>
<td>N</td>
<td>0</td>
</tr>
</tbody>
</table>

*Syntax*

*Semantics*

If Picobot's in state 0 seeing xxWS,

Then move North, and "change" to state 0.

---

**Notes**

Self contained but not simplistic
Wildcards

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

- Current state
- Surroundings
- Direction
- New state

N must be empty

EWS may be wall or empty space

The Rule is **One step per rule**

Picobot’s world

0 N*** --> W 1
0 x*** --> N 0
1 ***x --> S 1

1. Run Picobot! Which rule A, B, or C runs first? _______
   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 upward! _______

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?

---

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>

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

---

CS ~ **Complexity Science**

Problem 3
Shortest Picobot program: **6 rules**

Problem 4
Shortest Picobot program: **8 rules**

---

Notes
Maze strategies?

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

Why might this be difficult for Picobot?

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!??!

**Picobot**

**Rules**

Is it working?

```
# twelve-rule maze-solver:
Enter rules for Picobot
Be sure to hit "Enter rules" after making
```

Messages

Lab/hw

- Onward -

You are not alone!

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

Happy Picobotting!

And, good luck with the *adventure* of Python!
(1) Find and correct as many errors as you can in this code:

```python
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.'
    print('unless you're a CS 5 grader, then YOU WIN!')
```

(2) This one line does **three** things... what are they?

```python
import random
```

(3) Extra! Can you find 7 punctuation marks used in **more than one way** here?