Welcome to CS5 Green

Three handouts today…

Lecture notes
About you…
“Worksheet”

“We don’t have strong enough words to describe this class” - U.S. News and Course Report

“We give this course two thumbs!” - Ebert and Roeper

“Of all the courses we’ve ever taken, this was - without a doubt - one of them!” - New York Review of Courses
Computing in the context of biological problems

How does salmonella cause disease?

Are sex determination systems in birds and mammals related?

How are Neanderthals related to modern humans?

http://foodtestinguae.blogspot.com/
http://humanorigins.si.edu/evidence/human-fossils/fossils/la-chapelle-aux-saints

Neandertal Museum, Mettman Germany
Introductions and Course Overview

Prof. Eliot Bush

Official turtle of CS 5 Green

Guest appearances by other HMC computational biologists…
My name: Eliot Bush

My hometown: Mentor, Ohio

Something that I really like (a food, a movie, a color, anything!)
   Camping, frisbee, foreign travel

Something about me (it can be random!)
   For my sabbatical a few years ago, my family and I lived in Istanbul, Turkey.
   I don’t believe in washing cars.

https://www.aol.co.uk/2012/04/17/russia-kills-off-the-lada-while-lada-nearly-kills-russian-video
Course Website
www.cs.hmc.edu/cs5green
Syllabus in a Nutshell

Lectures: Tuesday and Thursday, 9:35-10:50, Shan B442

Labs: Fridays 3-5 PM in Beckman B105
Recommended (and incentivized), but not required

Office hours and grutoring hours on the website!

Piazza Q&A system

Homework
Lab Problem
Several additional homework problems

Pair programming encouraged on some problems
Due, Tuesdays at 11:59 PM

Three CS 5 Greenbacks (aka “Euros”)
Syllabus in a Nutshell

Pair Programming Policy: For some questions, you are (optionally) allowed to work as a pair. In a pair you should always program together and switch every 30 minutes.

Honor Code Policy: Other than pair programming, discussions OK, sharing or searching for code not permitted.

Grading:
- Homework + Final Project: 65%
- Midterm: (Tuesday, Oct 16): 10%
- Final Exam: (Tuesday Dec 18, 9 am): 20%
- Participation/worksheets: 5% (missing up to 3 is OK)

To pass CS 5, one must have a passing grade on all components (Homework, Exams, Participation)
Textbook

Computing for Biologists
Python Programming and Principles

Ran Libeskind-Hadas
Eliot Bush
Q: Will I learn as much CS here as I would in CS 5 Gold?
A: Yes!

Q: Are there other courses combining CS and Bio at Mudd
A: Yes! Bio 52, MCB118b, Bio 188, and a whole major (Mathematical and Computational Biology)
Introductions: Picobot

Goal: whole-environment coverage with only *local sensing*…
Environment in the NEWS!

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

We can represent a particular environment with a text “code”.

Surroundings are always in NEWS order.
How many distinct surroundings are there?
Surroundings

How many distinct surroundings are there?

$2^4 = 16$ possible …
Picobot's memory is a single number, called its state. State is the *internal context* of computation.

Picobot always starts in state 0.

State and surroundings represent everything the robot knows about the world.
Picobot moves according to a set of rules:

I am in state 0. My surroundings are \( \text{xxWS} \).

Aha!
I should move \( \text{N} \).
I should enter state 0.

If I'm in state 0 seeing \( \text{xxWS} \),
then I move \( \text{N} \)orth, and change to state 0.

A capital “X” here means “Don’t Move.”
Wildcards

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

I am in state 0. My surroundings are **WS.

Aha! This matches x***

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

<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>N</td>
<td>0</td>
</tr>
</tbody>
</table>

and EWS may be wall or empty space

N must be empty
<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>-&gt; N</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>N***</td>
<td>-&gt; X</td>
<td>0</td>
</tr>
</tbody>
</table>

- Picobot checks its rules from the top each time.
- When it finds a matching rule, that rule runs.
- Only one rule is allowed per state and surroundings.

1. What will this set of rules do to Picobot?

2. Try to add some rules so that we go to the bottom now and then back up forever!

*Worksheet*
1. What will this set of rules do to Picobot?

2. Try to add some rules so that we go to the bottom now and then back up forever!
This week!

Write rules that will always cover these two rooms. *(separate sets of rules are encouraged…)*

Lab Problem 2

Problem 4

Your “program” can be slow but it should work for any starting location and for any wall-connected maze!

**our best:** 3 states, 7 rules

**our best:** 4 states, 8 rules
A word about states

Imagine state 0 means “pointing north”
state 1 means “pointing east”
state 2 …
state 3 …
Salmonella typhimurium invading human cells

Richer syntax allows greater expressiveness!

```python
def alifeSim(numGens, popSize, numToSelect, Network, inhibitorL):
    """Do an artificial life simulation for numGens generations with popSize organisms.""
    fitD={}
    # get initial pop and create popL of (fitness,org) tuples
    popL=[]
    for org in createInitialPop(popSize, Network, inhibitorL):
        fitness = org.getFitness()
        popL.append((fitness, org))
        fitD[hash(org)]=fitness
    topL = getTopOrgs(popL, numToSelect)  # get top orgs
    for i in range(numGens):
        popL=[]
        for j in range(popSize):
            toReplicate = random.choice(topL)
            neworg = toReplicate[1].replicate()  # get fitness
            if hash(neworg) in fitD:
                fitness=fitD[hash(neworg)]
            else:
                fitness=neworg.getFitness()
                fitD[hash(neworg)]=fitness
            popL.append((fitness, neworg))
    topL = getTopOrgs(popL, numToSelect)
    print("gen":",",",",topL[0])
    if i%50==0:
        fitD.clear()
    return topL[0]
```

Learning to program is a bit like learning a foreign language!

Strange syntax!
Funky grammar
Why Python?

- Relatively “nice” syntax
- Emerging as language of choice in many fields
- Packages for graphics, audio, scientific computing, ...

```python
print("Hello World!")
```

```java
class HelloWorld {
    static public void main( String args[] ) {
        System.out.println( "Hello World!" );
    }
}
```

```befunge
> v v ,,,,*"Hello"
>48*,
 v ,,,,*"World!"
>25*,@
```

After all, there are thousands of languages to choose from!
Hello World...

```cpp
#include <iostream.h>

main()
{
    cout << "Hello World!" << endl;
    return 0;
}
```
The Python interpreter

[bush@EB-Laptop:~$ python
Python 3.5.2 |Anaconda 4.1.1 (x86_64)| (default, Jul  2 2016, 17:52:12)
[GCC 4.2.1 Compatible Apple LLVM 4.2 (clang-425.0.28)] on darwin
Type "help", "copyright", "credits" or "license" for more information.

>>> [end]
Python strings

```python
>>> biologist1 = "Watson"
>>> biologist1
'Watson'

>>> biologist2 = 'Crick'
>>> biologist2
'Crick'
```
DNA is double stranded

![DNA Structure Diagram](http://upload.wikimedia.org/wikipedia/commons/thumb/e/e4/DNA_chemical_structure.svg/450px-DNA_chemical_structure.svg.png)
Representing DNA molecules on a computer

5' – AATGCCGTGCTTGTAGACGTA – 3'
3' – TTACGGCACGAACATCTGCAT – 5'

By convention, we represent as a single string going 5' to 3'.

\[
\begin{align*}
\text{AATGCCGTGCTTGTAGACGTA} \\
\text{or} \\
\text{TACGTCTTCAAGCACGGCATT}
\end{align*}
\]

● Either of these two strings could be used
● These are reverse complements of each other
Using strings: length and index

```python
>>> myDNA = "AATGCCGTGCTT"

>>> len(myDNA)
12

>>> myDNA[0]
'A'

>>> myDNA[3]
'G'

>>> myDNA[20]
IndexError: string index out of range
```
Using strings: slicing

>>> myDNA = "AATGCCGTGCTT"

>>> myDNA[0:4]
'AATG'

>>> myDNA[3:7]
'GCCG'

>>> myDNA[1:]
'ATGCCGTGCTT'

>>> myDNA[:4]
'AATG'

>>> myDNA[10:42]
'TT'