Welcome to CS 5!

WARTS FOREVER!

YOU'RE FLYING! HOW?

PYTHON!

I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST PRINT "HELLO, WORLD!"

I DUNNO...
DYNAMIC TYPING?
WHITESPACE?

COME JOIN US!
PROGRAMMING IS FUN AGAIN!
IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?

I JUST TYPED
import antigravity

THAT'S IT?

... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.

BUT I THINK THIS IS THE PYTHON.

xkcd, CS's id
Welcome to CS 5!

Wally Wart, a protrusive advocate of concrete computing

We don't have words strong enough to describe this class.
- US News and Course Report

Everyone will get out of this course – a lot!
- NYTimes Review of Courses

We give this course two thumbs...
- Metametacritic

1 handout...
slides & syllabus

Grab these lecture notes...

Introduction to CS
Welcome, not only to HMC, but to all 5Cs!
Introductions...

Zach Dodds
Olin B163 (HMC)
dodds@cs.hmc.edu

pursuer of *low-level* AI

taker of *low-quality* selfies

fan of *low-tech* games

Speaking of introductions
How I spend my summers ...?

actually, this "I" is not quite accurate...

Robots

Outreach

Chairs?

Who?!??  Dinos!
Algorithmic improvisation
Start-up ideas...
... to formal pitches
CS Staff: *Rising sophomores*, unite!

Robotics Outreach in Boston

Teacher Outreach in S.F.
CS Staff: *Rising sophomores, unite!*

*Where is this?*
CS Staff: *Rising sophomores, unite!*

Lots of opportunities surrounding computing... *(at the 5Cs and beyond)*
Take-home message...

Yay! In 2019: just Google for hmc cs5

www.cs.hmc.edu/cs5
You're here ~ what's next?

1) How CS 5 runs...

2) Python?!

the first Python HW is *choice*!

3) What *is* CS?

CS is just programming, right?

I'm not so sure...

Whatever it is, it's definitely *alien*!
CS vs. programming?
Spot the difference here?

\texttt{print('hi')} \quad \texttt{print 'hi'}

I still confuse these!
Spot the difference here?

```
print('hi')
```

We'll be using python 3 this term...
Spot the difference here?

\texttt{print('hi')} \quad \texttt{print 'hi'}

Syntax!

We'll be using python 3 this term...
A minute of cs5 programming...

Lab 0: getting everything running on your own machine

Demo

Python source code, a plain-text file
(here, edited by the VS Code text editor)

lab and hw instructions

shell or command-line or terminal
(the execution environment)
A minute of cs5 programming...

Lab 0: getting everything running on your own machine

Lab+hw

Python source code, a plain-text file (here, edited by the VS Code text editor)

shell or command-line or terminal (the execution environment)

lab and hw instructions

Edit

Run
Lab 0: **Happiness Suggestion**

Download the software BEFORE coming to lab:

https://www.cs.hmc.edu/twiki/bin/view/CS5/OwnMachines
Logically, I've got game!

rock – paper – scissors – lizard – Spock!

http://www.youtube.com/watch?v=fjqIDc2VICZ0 start at about :28

Let's play! Maybe two out of three?

hw0: rock-paper-scissors
Soundbite Syllabus

**Lectures**

**T and Th:** 8:10-9:25 am

Key topics: theory and their applications

Insight into the HW problems (what, why, how)

We'd like to see you! Let me know if you'll be sick...

**Lab**

**T or W:** 2:45 - 4:45pm or 6-8 pm

Guided progress on the week's hw

Not required, but encouraged: full credit for lab

Will SAVE you time and effort in CS 5

**Office hrs**

**F:** 3-4:00 pm - Linde Activities Center lab

or, come to any of the many tutoring hrs!

**HW**

**Monday:** Hw is due on Monday nights...

Come to Lectures!

Come to Labs!

Lots of help is available!
Syllabus, briefly

**Lectures**

**T and Th:** 8:10-9:25 am  
Key skills, topics, and their motivation  
Insight into the HW problems (what, **why**, how)  
**We’d like to see you!** Let me know if you’ll be sick...

**Lab**

**T or W:** 2:45 - 4:45pm or 6-8 pm  
Guided progress on the week's hw  
Not required, but encouraged: *full credit for lab*  
Will **SAVE** you time and effort in CS 5

**Office hrs**

**F:** 2:30-4:30 pm, *Linde Activities Center lab*  
feel free to work on HW, to just stop by,  
or, come to any of the **many** tutoring hrs!

**HW**

**Monday nights:** due by 11:59 pm
Each week's lab...

0) Find the lab!  *Sign in*...

1) Get Python running...

2) Edit, run, + submit a file...

Encouraged: *bring your laptop*
Each week's lab...

Labs are optional, but incentivized.

If you come to lab, give a good-faith effort, and sign in, you'll receive *full credit for the lab problems* even if you don't finish.

(you *do* need to submit by the usual hwk due date)

Encouraged: *bring your laptop*
Evening lab?

Olin's Southeast door is open!

Enter through Olin building through the SE door to Beckman B102, B105, B126
Edwards Macalister Pryne

coffee

cool machines - drills, lathes, etc.

other keyboard-free machines

Physicists, chemists & other parenthesis-needling individuals,

CS Hallway and Labs

B102

B105

Map to CS Labs

Laptop? Bring it!

Galileo

Beckman

Biologists, bees, spiders and other arachnophiles

Big Beckman (B126)

to Olin (Bio + CS)
Submissions: *GradeScope*
Homework

Assignments  ~ 5 problems/week

Due **Monday** evenings by 11:59 pm.

Extra credit is usually available...

You have 3 **CS 5 Euros** to use...
"Late Days"

Some problems are specified “individual-only.”
Others offer the option of working as pairs/partners:

- You don't have to work in pairs/partners (that said, it's fun!)
- If you do, you must share the work equally - typing and coaching
- Be sure to indicate who your partner was at the submission site!

**Eur-ollowed to use one Euro for up to three hwks.**

No need to let us know, even.
Pairs

one computer
tradeoff typing/debugging ~
about every 20 minutes

Partners
two computers
both partners type/debug ~
provide help as needed

Standard is the same either way:
After finishing the hw, (a) each person has contributed equally and (b) both could complete the problems on their own

Submit with a partner as full co-owners of the work.
Honor Code

• You're encouraged to discuss problems with other students – or tutors - or any instructors.

• You may not share written, electronic or verbal solutions with other students, present or past:

  Please do use the internet for Python language references.

  Pleas do use other's eyes for finding syntax errors.

  Do not use the internet (or intranet) to (try to) find solutions...

  If you work as a pair/partners, the rules apply for the duo.

Sign & submit CS's honesty policy online in this week's lab.
Grading

~ 65% Assignments

~ 30% Exams

~ 5% Participation/“quizzes”

if perc > .95:
    print('A')
elif perc > .90:
    print('A-')
elif perc > .70:
    print('Pass')

see online syllabus for the full grade list...

Midterm? This feels more like a 2/3-term!

Exams

Midterm
Th, Nov. 7, in-class

Final
Wed., Dec. 18

using a page of notes is OK on exams
the exams are written, not coded
the problems are modeled on the in-class "quizzes"
Choices, choices!

Let's set the value of `perc` to 0.91...

\[
\text{perc} = 0.91
\]

```python
if perc > 0.95:
    print 'A'
elif perc > 0.90:
    print 'A-
elif perc > 0.70:
    print 'Pass'
else:
    print 'Aargh!'
```

What will this program print, if `perc` is 0.91?

First – do you see the *syntax errors* here !?
Choices, choices!

Let's set the value of `perc` to 0.91...

```
perc = 0.91
```

If `perc` is greater than 0.95:
```
    print('A')
```

If `perc` is greater than 0.90:
```
    print('A-')
```

If `perc` is greater than 0.70:
```
    print('Pass')
```

Else:
```
    print('Aargh!')
```

What will this program print, if `perc` is 0.91?

Aargh! ;-)

Lots of Illuminating Solid Parentheses!
Choices, choices!

Let's set the value of `perc` to 0.91...

```
perc = 0.91

if perc > 0.95:
    print('A')
elif perc > 0.90:
    print('A-')
elif perc > 0.70:
    print('Pass')
else:
    print('Aargh!')
```

What will this program print, if `perc` is 0.91?
Choices, choices!

perc = 0.80

if perc > 0.95:
    print('A')
elif perc > 0.90:
    print('A-')
elif perc > 0.70:
    print('Pass')
else:
    print('Aargh!')

What does each of these programs print out, if \texttt{perc} is 0.8?

What value of \texttt{perc} gives an \texttt{'A-'} on the right?

How can you get a \textbf{better} grade on the right than the left?
Exclusive Choices

if ... elif ... else

```python
if perc > 0.95:
    print('A')

elif perc > 0.90:
    print('A-')

elif perc > 0.70:
    print('Pass')

else:
    print('Aargh!')
```

When using `if . elif ... . else` at most one block will run: the first whose test is `True`. If all fail, the `else` will run

4 mutually exclusive blocks in a single control structure

*Elif* and *else* are optional
Exclusive Choices

Every **if** starts a new control structure.

- **if** and **else** are optional
- **elif** and **else** are optional

When using **if . elif … . else**, at most one block will run: the first whose test is **True**. If all fail, the **else** will run.

4 mutually exclusive blocks

Every **elif** and **else** continues an existing control structure.
<table>
<thead>
<tr>
<th>mutually exclusive blocks</th>
<th>non exclusive blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>What if perc == .99?</td>
<td>(How would we set it?)</td>
</tr>
<tr>
<td>How many separate control structures does each side have?</td>
<td></td>
</tr>
</tbody>
</table>

```python
perc

if perc > .95:
    print('A')

elif perc > .90:
    print('A-')

elif perc > .70:
    print('Pass')

if perc > .95:
    print('A')

if perc > .90:
    print('A-')

if perc > .70:
    print('Pass')
```
What's the difference?

mutually exclusive blocks

What if \texttt{perc} == \texttt{.99} ? (How would we set it?)

How many separate \textit{control structures} does each side have?

\texttt{perc} = \texttt{.99}

\begin{itemize}
  \item \textbf{if} \texttt{perc} > \texttt{.95}:
    \begin{itemize}
      \item \texttt{print('A')}
    \end{itemize}
  \item \textbf{elif} \texttt{perc} > \texttt{.90}:
    \begin{itemize}
      \item \texttt{print('A-')}
    \end{itemize}
  \item \textbf{elif} \texttt{perc} > \texttt{.70}:
    \begin{itemize}
      \item \texttt{print('Pass')}
    \end{itemize}
\end{itemize}

nonexclusive blocks

\texttt{perc} = \texttt{.99}

\begin{itemize}
  \item \textbf{if} \texttt{perc} > \texttt{.95}:
    \begin{itemize}
      \item \texttt{print('A')}
    \end{itemize}
  \item \textbf{if} \texttt{perc} > \texttt{.90}:
    \begin{itemize}
      \item \texttt{print('A-')}
    \end{itemize}
  \item \textbf{if} \texttt{perc} > \texttt{.70}:
    \begin{itemize}
      \item \texttt{print('Pass')}
    \end{itemize}
\end{itemize}
Nesting for decision-making, we now have it all...
Nesting for decision-making, we now have it all...
Nesting for decision-making, we now have it all...

So, let's catch 'em all...
comp = 'rock'
user = 'paper'

if comp == 'paper' and user == 'paper':
    print('We tie. Try again?

elif comp == 'rock':
    if user == 'scissors':
        print('I win! *_*'
    else:
        print('You win. Aargh!')

Does this program print the correct RPS result this time? Does it always?
Pair up with someone nearby – answer these questions together...

Name ______________________

Your favorite __________ is ____________.

Your least favorite ____________ is ____________.

Name ______________________

Your favorite __________ is ____________.

Your least favorite ____________ is ____________.

What is something non-Claremont-collegey you have in common?

Then, try these Python q’s:

(0) Find the 3 tests and 4 blocks here.

(1) What does this code print?

```python
comp = 'rock'
user = 'rock'

if comp == 'rock':
    if user == 'paper':
        print('I win *_!*')
    elif user == 'scissors':
        print('You win.')
else:
    print('Tie.')
```

(2) As written, what output does this print?

```python
comp = 'rock'
user = 'rock'

if comp == 'rock':
    print('I win *_!*')
if user == 'paper':
    print('You win.')
else:
    print('Tie: Ugh')
```

(3) Change these inputs to produce a completely correct RPS output here.

(4) How many of the 9 RPS input cases are fully correctly handled here?

(5) What is the smallest number of blocks and tests you’d need for a full game of RPS?

(Extra) What if it were RPS-5, which includes Lizard and Spock? How about RPS-101?
Pair up with someone nearby – answer these questions together...

Name ______________________
Your favorite __________ is ____________.
Your least favorite ____________ is ____________.

Name ______________________
Your favorite __________ is ____________.
Your least favorite ____________ is ____________.

What is something non-Claremont-collegey you have in common?

Then, try these Python q's:

(0) Find the 3 tests and 4 blocks here.

(1) What does this code print?

```
comp = 'rock'
user = 'rock'
if comp == 'rock':
    if user == 'paper':
        print('I win *_!*!')
    elif user == 'scissors':
        print('You win.')
    else:
        print('Tie.')
else:
    print('Tie.')
```

(2) As written, what output does this print?

(3) Change these inputs to produce a completely correct RPS output here.

(4) How many of the 9 RPS input cases are fully correctly handled here?

(5) What is the smallest number of blocks and tests you'd need for a full game of RPS?

(Extra) What if it were RPS-5, which includes Lizard and Spock? How about RPS-101?
Pair up with someone nearby – answer these questions together...

Name   ______________________
Your favorite __________ is ____________.
Your least favorite __________ is ____________.

Name    ______________________
Your favorite __________ is ____________.
Your least favorite __________ is ____________.

What is something non-Claremont-collegey you have in common?

Our taste in hats!

(1) What does this code print?

```python
comp = 'rock'
user = 'rock'

if comp == 'rock':
    if user == 'paper':
        print('I win *_*!')
    if user == 'scissors':
        print('You win.')
else:
    print('Tie.')
```

(2) As written, what output does this print?

(3) Change these inputs to produce a completely correct RPS output here.

(4) How many of the 9 RPS input cases are fully correctly handled here?

(5) What is the smallest number of blocks and tests you’d need for a full game of RPS?

(Extra) What if it were RPS-5, which includes Lizard and Spock? How about RPS-101?
Pair up with someone nearby – answer these questions together...

Name ______________________  
Your favorite __________ is ____________.
Your least favorite ____________ is ____________.

Name ______________________  
Your favorite __________ is ____________.
Your least favorite ____________ is ____________.

What is something non-Claremont-collegey you have in common?

Then, try these Python q’s:

(0) Find the 3 tests and 4 blocks here.

(1) What does this code print?

```python
comp = 'rock'
user = 'rock'

if comp == 'rock':
    if user == 'paper':
        print('I win *_*!')
    elif user == 'scissors':
        print('You win.')
else:
    print('Tie.')
```

(2) As written, what output does this print?

```python
comp = 'rock'
user = 'rock'

if comp == 'rock':
    print('I win *_*!')
if user == 'paper':
    print('You win.')
else:
    print('Tie: Ugh')
```

(3) Change these inputs to produce a completely correct RPS output here.

(4) How many of the 9 RPS input cases are fully correctly handled here?

(5) What is the smallest number of blocks and tests you’d need for a full game of RPS?

(Extra) What if it were RPS-5, which includes Lizard and Spock? How about RPS-101?
Pair up with someone nearby – answer these questions together…

In a moment… pass these up the aisles (taking a picture, if you’d like)

then, turn back to the notes

... and tests you’d need for a full game of RPS?

(Extra) What if it were RPS-5, which includes Lizard and Spock? How about RPS-101?

(0) Find the 3 tests and 4 blocks here.

(1) What does this code print?

(comp = 'rock' 
user = 'rock'
if comp == 'rock':
print ('I win *_*!')
if user == 'paper':
print ('You win.')
else:
print ('Tie.'))

(2) As written, what output does this print?

(3) Change these inputs to produce a completely correct RPS output here.

(4) How many of the 9 RPS input cases are fully correctly handled here?

(5) What is the smallest number of blocks and tests you’d need for a full game of RPS?
comp = 'rock'
user = 'rock'

if comp == 'rock':
    if user == 'paper':
        print('I win *_*!')
    elif user == 'scissors':
        print('You win. ')
    else:
        print('Tie.')

... what if this else block were indented?
comp = 'rock'
user = 'rock'

if comp == 'rock':
    print('I win *_*!')

if user == 'paper':
    print('You win. ')

else:
    print('Tie: Ugh')

What does this program print?
How many possible “input cases” are there?
For how many is this program correct?

How efficient can we be?
For RPS-3? RPS-5? RPS-101?
```python
comp = 'rock'
user = 'rock'

if comp == 'rock':
    print('I win *_*!')

if user == 'paper':
    print('You win.')

else:
    print('Tie: Ugh')
```

How many possible “input cases” are there? How efficient can we be? For how many is this program correct? For RPS-3? RPS-5? RPS-101?
How many possible “input cases” are there?

For how many is this program correct?

How efficient can we be?

For RPS-3? RPS-5? RPS-101?
Pair up with someone nearby – answer these questions together.

Name ______________________
Your favorite ________ is ____________.
Your least favorite ________ is ____________.

Pair up with someone nearby – answer these questions together.

Name ______________________
Your favorite ________ is ____________.
Your least favorite ________ is ____________.

What is something non-Claremont-collegey you have in common?

comp = 'rock'
user = 'rock'
if comp == 'rock':
    if user == 'paper':
        print('I win *_*!')
    else:
        print('Tie.')
else:
    print('You win.

(0) Find the inputs that completely produce a correct output here.
(1) What output does this code print?
(2) As written, what output does this print?
(3) Change these inputs to produce a completely correct RPS output here.
(4) How many of the 9 RPS input cases are fully correctly handled here?
(5) What is the smallest number of blocks and tests you'd need for a full game of RPS?
(Extra) What if it were RPS-5, which includes Lizard and Spock? How about RPS-101?

Ok! Pass these to the aisles + "upward" (take a picture, if you'd like) ... then, turn back to the notes

CS != programming

programming : CS ::
longboards : HMC maybe 5Cs?
capital : business venture
equations : mathematics
language : ideas
web search : knowledge
Tesla : Google

programs are a vehicle, but not the destination
CS != programming

"not equal to"
CS != programming

So, what is CS?
Today in CS5

1) How CS 5 runs...

2) Python?!

3) What *is* CS?

CS is just programming, right?

I'm not so sure...

Whatever it is, it's definitely *alien*! Shouldn't there be an alien in this game?
What is CS a science of?

the study of complexity:

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

it ~ information

or, more precisely, a process transforming information from one form to another

We'll look at 3 examples – each of which you'll construct in CS 5...at least to some extent!

3 examples?
That's it for me!
What is CS?

Can you solve the problem?

How can it be done?

How well can it be done?

Can it be done at all?

Can you create a process to solve such problems?

What is the Longest Common Subsequence between 2 strings?

biology's string-matching problem, "LCS"

'HUMAN'

'CHIMPANZEE'

'CGCTGAGCTAGGCC...'

'ATCCTAGGTAACTG...'

Eye oneder if this haz othur aplications?
What is CS?

How can it be done?

How well can it be done?

Can it be done at all?

How quickly can you find a solution?

Is your solution the "best" possible?

How much work is needed to simulate $N$ stars?

chemistry's + physics's "N-body" problem

What if $N$ is a million-and-one...?
Can we build a 3d model from one 2d image?

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

Is your problem solvable?
How can you tell !?

many problems are uncomputable...
... and you'll prove this!

Andrew Ng's "Make3d"

All three eyes tell me that Make3d has just failed ~ epically!
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!

CS's 6 big questions

Only *one* is programming. *Which one?*
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?

**CS's 6 big questions**

Only one is programming. Which one?

**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!
CS’s – and CS5's – philosophy:

Whatever you are, be a good one.

- Abraham Lincoln

More and more, CS can help!
Remember ~ **Lab this week**

Tue. or Wed. ~ afternoon or evening
Bring your laptop to Beckman B126 (here)
- or use one of the CS machines in B105/B102
Get started with Python/text editor/cmdline...

**See you in lab!**
*(perhaps at 2:44:44 today...?)*

though it's more than a few bits early!

Alien defeats everything –
*even Alien*

How about a peek at the rest of the week's HW... ?

... you must mean *Pic*!