Welcome to CS 5!

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...
- Metameticritic

Read sections 1.1, 2.1-2.6

Introduction to CS

Geoff Kuenning
geoff@cs.hmc.edu
Put “CS 5” in your subject line!

I think hard drives are cool...

Not entirely sensible!
Chat grutors: Natalie Couch, Ammar Fakih, Amy Tam

Zoominess

Keep cameras on (if possible)
Slides pre-posted on class Web site
Lectures recorded and posted here
Search for “HMC CS5”!
Zoom chat is for questions, not conversation
- Posting in chat is like raising your voice in class

4:15 Zoom this Thursday!
See class e-mail for details

Summer research

Lots of opportunities surrounding computing... (at the 5Cs and beyond)
Getting Help

No Piazza this semester

This is the e-mail for all help!
cs5help@cs.hmc.edu

Note the "@cs" part!

DO NOT include screenshots!
- Please copy and paste your entire program
- …or even better, attach the file to your email

Spot the differences here?

print('hi')  print 'hi'

Syntax!

Today in CS5...

2) How CS 5 runs...

3) Python?!

1) What is CS?

Whatever it is, it’s definitely alien!

I’m not so sure...

What is CS?

CS is the study of complexity

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

How can it be done?
How quickly can you find solutions?

How well can it be done?
Do you have the “best” solution?

Can it be done at all?
Is every problem solvable?

But only one is programming.
Do you see which?

There isn’t always!
Take-home message...

www.cs.hmc.edu/cs5

Syllabus, briefly

**Lectures**

*Tu and Th:* 12:45-2:00pm
Key skills, topics, and their motivation
Insight into the HW problems (what, why, how)
We'd like to see you! Let us know if you'll be sick...

**Lab**

*Tu:* 2:30-4:30pm and 6:00-8:00pm
Guided progress on the week's homework
Not required, but encouraged: full credit for lab
Will SAVE you time and effort in CS 5

**Office hrs**

See https://www.cs.hmc.edu/~geoff/geooff-schedule.html
or, come to any of the many tutoring hrs!

**HW**

*Monday nights: due by 11:59 pm*

This week: Lab 0

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

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

Get everything running on your own machine
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"

Collaborate!

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!
• Put your name(s) in the code, as well!

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, past, present or future:
  Please do use the Internet for Python language references.
  Please 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.

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 homework, (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.

Grading

~ 65% Assignments
~ 30% Exams
~ 5% Participation/"quizzes"

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

many take Cs P/NC
see online syllabus for the full grade list...

Exams

Midterm
Thu, Mar 25, in-class
Wed, May 12 (2pm)

Final

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 \( \text{pct} \) to 0.91...

\[
\text{pct} = 0.91
\]

What will this program print, if \( \text{pct} \) is 0.91?

\[
\begin{align*}
\text{if } \text{pct} &> 0.95: \\
&\text{print('A')} \\
\text{elif } \text{pct} &> 0.90: \\
&\text{print('A-')} \\
\text{elif } \text{pct} &> 0.70: \\
&\text{print('Pass')} \\
\text{else:} \\
&\text{print('Aargh!')} \\
\end{align*}
\]

What's here?

- \# of BLOCKS here:
- \# of TESTS here:
- \# of \text{TESTS} here:

Exclusive Choices

\[
\begin{align*}
\text{pct} &= 0 \\
\text{if } \text{pct} &> 0.95: \\
&\text{print('A')} \\
\text{elif } \text{pct} &> 0.90: \\
&\text{print('A-')} \\
\text{elif } \text{pct} &> 0.70: \\
&\text{print('Pass')} \\
\text{else:} \\
&\text{print('Aargh!')} \\
\end{align*}
\]

4 mutually exclusive blocks in a single control structure

When using \text{if} \ldots \text{elif} \ldots \text{else} \ldots \text{at most one block will run: the first whose test is True. If all fail, the else will run}

What's the difference?

\[
\begin{align*}
\text{mutually exclusive blocks} \\
\text{pct} \\
\text{if } \text{pct} &> 0.95: \\
&\text{print('A')} \\
\text{elif } \text{pct} &> 0.90: \\
&\text{print('A-')} \\
\text{elif } \text{pct} &> 0.70: \\
&\text{print('Pass')} \\
\text{else:} \\
&\text{print('Aargh!')} \\
\end{align*}
\]

\[
\begin{align*}
\text{non-exclusive blocks} \\
\text{pct} \\
\text{if } \text{pct} &> 0.95: \\
&\text{print('A')} \\
\text{elif } \text{pct} &> 0.90: \\
&\text{print('A-')} \\
\text{elif } \text{pct} &> 0.70: \\
&\text{print('Pass')} \\
\end{align*}
\]

What if \( \text{pct} \) = 0.99? (How would we set it?)

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

```python
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:

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

(2) 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.')
```

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

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

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

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

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

(6) 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?