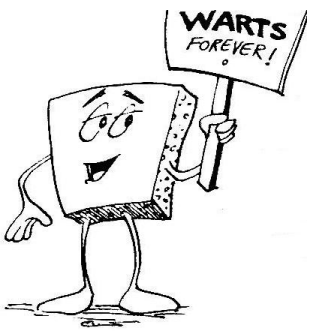


Wally Wart, a protrusive advocate of ***concrete*** computing

Welcome to CS 5 !

Read sections 1.1, 2.1-2.6

Introduction to CS

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…
- *Meta*metacritic

Introductions...

Geoff Kuenning

geoff@cs.hmc.edu

Put "CS 5" in your subject line!

Not entirely sensible!

I think hard drives are cool...

Chat grutors: Natalie Couch, Ammar Fakih, Amy Tam



Speaking of introductions

Summer research

Zoominess

Keep cameras on (if possible)

Slides pre-posted on class Web site

Lectures recorded and posted here

Search for "HMC CS5"!

https://www.cs.hmc.edu/~geoff/cs5videos
(login and PW in email you got)

Zoom chat is for *questions*, not conversation
- Posting in chat is like raising your voice in class

Getting Help

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

Today in CS5…

2) How CS 5 runs…

3) Python?!

1) What ***is*** CS?


CS is just programming, right?


Whatever it is, it's definitely *alien*!

I'm not so sure…

Spot the differences here?

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

```
print 'hi'
```

Syntax!

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 ***six*** big questions are here.

But only <u>one</u> is ***programming***. Do you see which?

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?

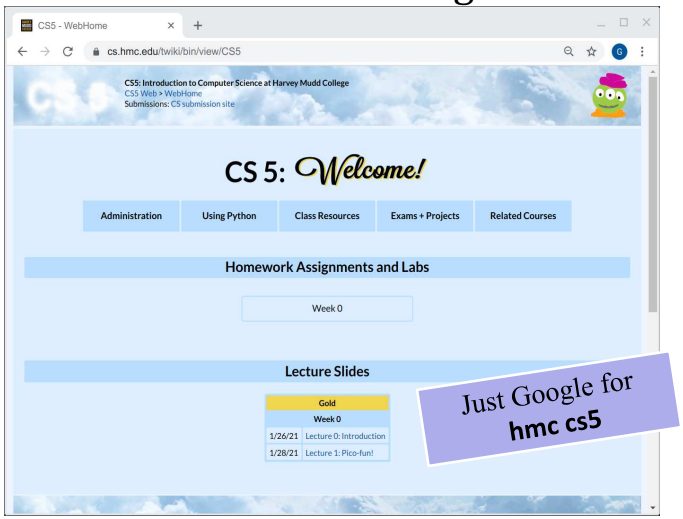
Is every problem solvable?

Is there a way to tell?

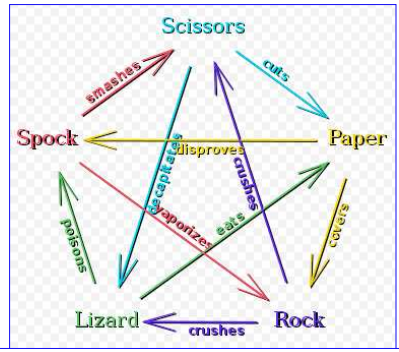
There isn't always!

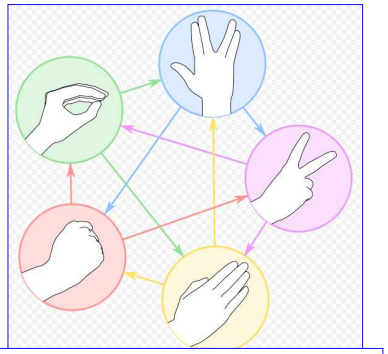
Take-home message…

depending on where "home" is, perhaps…



www.cs.hmc.edu/cs5





rock – paper – scissors – lizard – Spock!

Logically, I've got game!

http://www.youtube.com/watch?v=fqIDc2VICZ0 start at about :22

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

recommended by 4 out of 5 CS5 alums!

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

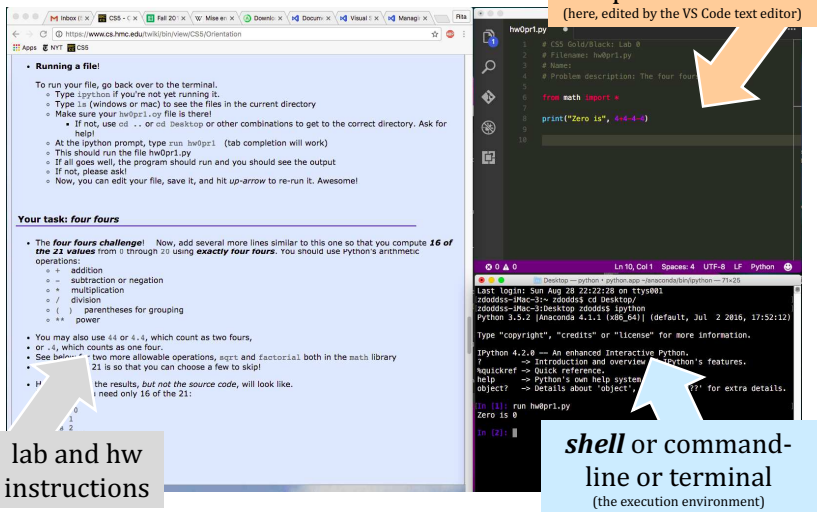
Office hrs

See https://www.cs.hmc.edu/~geoff/geoff-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)

lab and hw instructions

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"

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

No need to let us know, even.

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!

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.

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.

Pleas ***do*** use other's eyes for finding syntax erorrs.

Even with three eyes, I need to borrow others' to find the syntax errors here!

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

many take cs5 P/NC

see online syllabus for the full grade list…

Exams

Midterm	Thu, Mar. 25, in-class	Midterm? This feels more like a *3/4-term*!
Final	Wed, May 12 (2pm)	

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 `pct` to 0.91...

```
pct = 0.91

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

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

What's here?

of **BLOCKS** here:

of **TESTS** here:

# of CONTROL STRUCTURES here:

Choices, choices!

```
pct = 0.80

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

```
pct = 0.80

if pct > 0.00:
    print('Aargh!')
elif pct > 0.70:
    print('Pass')
elif pct > 0.90:
    print('A-')
else:
    print('A')
```

What does each of these programs print out, if `pct` is 0.80?

What value of `pct` gives an `'A-'` on the right?

How can you get a ***better*** grade on the right than the left?

Exclusive Choices

if ... elif ... else

pct = 0

```
if pct > 0.95:
    print('A')

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

elif pct > 0.70:
    print('Pass')

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

***4 mutually exclusive** blocks*

in a single control structure

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

`elif` and `else` are optional

What's the difference?

mutually exclusive blocks

```
pct

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

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

elif pct > .70:
    print('Pass')
```

***non*exclusive blocks**

```
pct

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

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

if pct > .70:
    print('Pass')
```

What if **pct == .99** ? (How would we set it?)

How many separate ***control structures*** does each side have?

Nesting

Does this program print the correct RPS result *this time*? *Does it always?*

```
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!')
```

BLOCKS ?

TESTS ?

Control Structures ?

Pair up with someone nearby – answer these questions together...

"Quiz"

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 *_*!')
if 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?

user \ comp	'rock'	'paper'	'scissors'
'rock'			
'paper'			
'scissors'			

(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?