Student-Computer Interaction Design

for Introductory Computer Science

John DeNero

How A Computer Science Course Begins

Photos by Brian Ly, Daily Cal Staff

How Students Actually Learn to Program

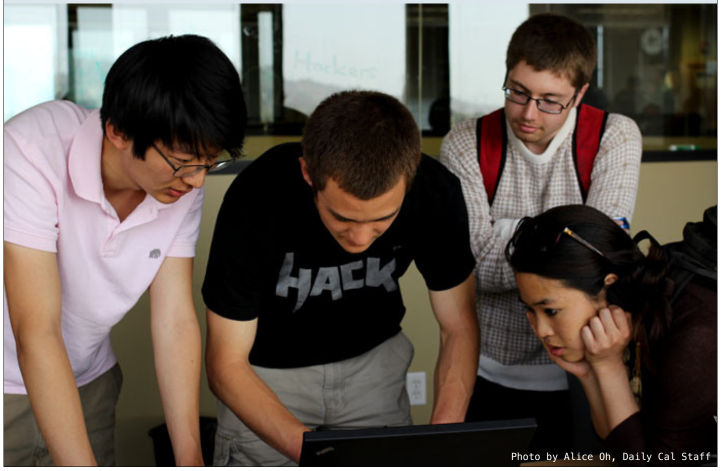
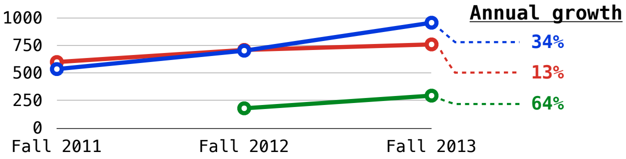


Photo by Alice Oh, Daily Cal Staff

Student-Computer Interaction Design

Design Principle: Create interactions that are consistently productive and challenging.

- No prolonged periods of frustration or confusion
- Every distinct activity involves a new idea
- Students solve problems that they didn't think they could solve (especially ones worth solving)



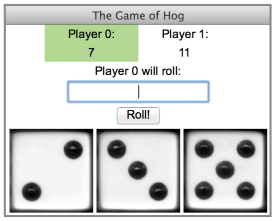
Programming Projects

Scaffolded Programming Projects

Fill-in-the-blank starter code and a full test suite

Advantages of scaffolding:

- Modular design taught by example
- Test-driven development from the first assignment
- Automated feedback localizes problems



The Game of Hog

Player 0: 7

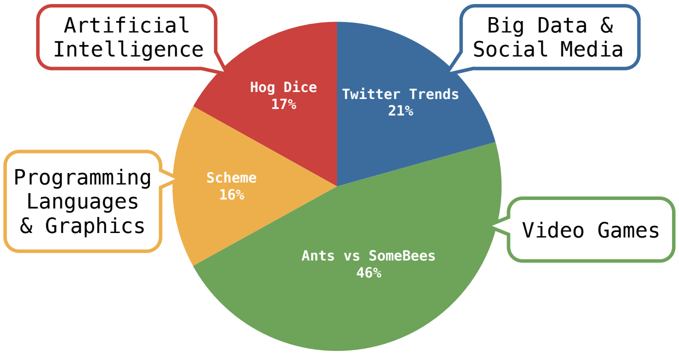
Player 1: 11

Player 0 will roll:

```
def make_averaged(fn, num_samples=1000):
    """Return a function that returns the average
    return value of FN called NUM_SAMPLES times.

    >>> dice = make_test_dice(3, 1, 5, 6)
    >>> averaged_dice = make_averaged(dice, 1000)
    >>> averaged_dice()
    3.75
    """
    "*** YOUR CODE HERE ***"
```

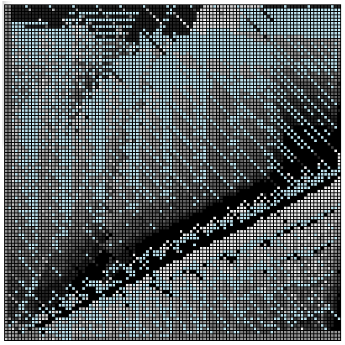
Rewarding Project Outcomes

Which project did you enjoy the most (Fall 2013)?



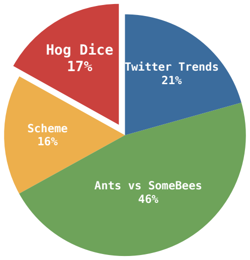
Rewarding Project Outcomes: Hog Dice

The Hog strategy contest encourages exploration



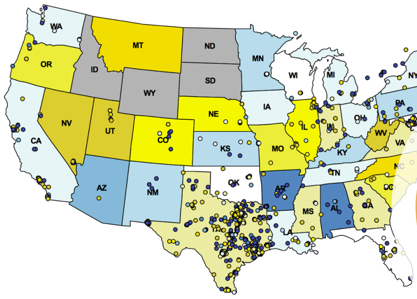
Strategy computed by Chenyang Yuan (class of Fall '12)
Visualization by Kevin Chen (class of Fall '13)

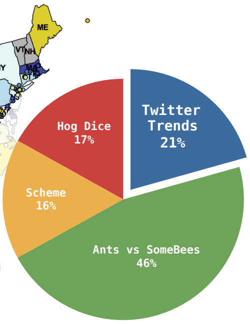
Visualization created for a student blog post



Rewarding Project Outcomes: Twitter Trends

The Twitter Trends project plots the average sentiment of tweets, aggregated by US state.



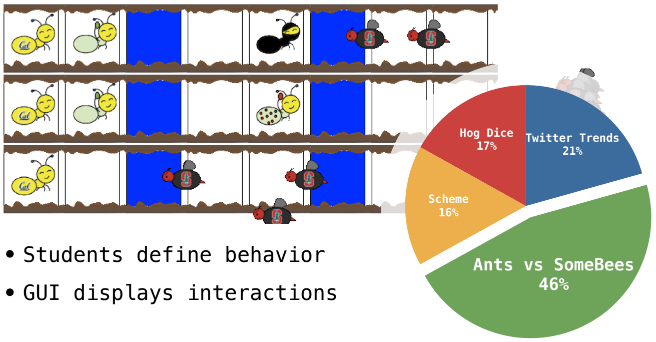


Tweets containing "Texas"

Nifty Assignments Track, SIGCSE 2013

Rewarding Project Outcomes: Ants vs SomeBees

Ants vs **SomeBees** is a clone of a popular game, **Plants** vs **Zombies**



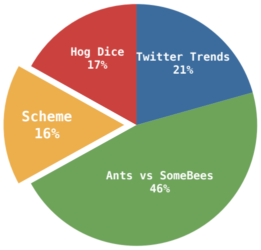
- Students define behavior
- GUI displays interactions

Nifty Assignments Track, SIGCSE 2014

Rewarding Project Outcomes: Scheme

In the Scheme Recursive Art Contest, students draw using Turtle commands interpreted by their own code

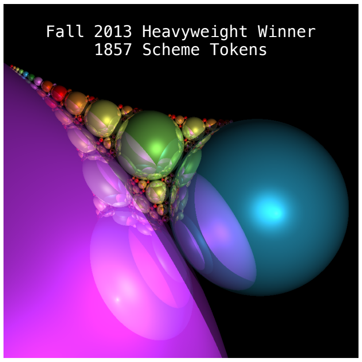
Fall 2012 Featherweight Winner
176 Scheme Tokens

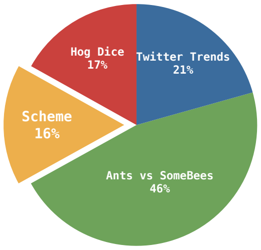


Rewarding Project Outcomes: Scheme

In the Scheme Recursive Art Contest, students draw using Turtle commands interpreted by their own code

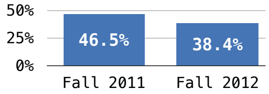
Fall 2013 Heavyweight Winner
1857 Scheme Tokens





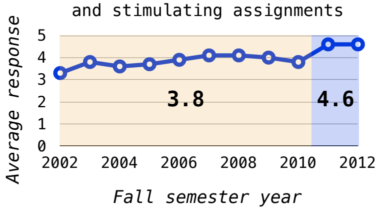
Student Responses to Projects

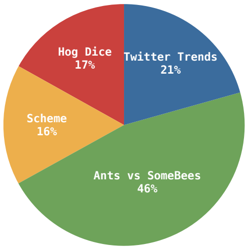
When asked, "what worked best for you in CS 61A," did students mention the projects?



26% mentioned working with partners & group study

This instructor gives interesting and stimulating assignments





Community

Promoting Collaboration

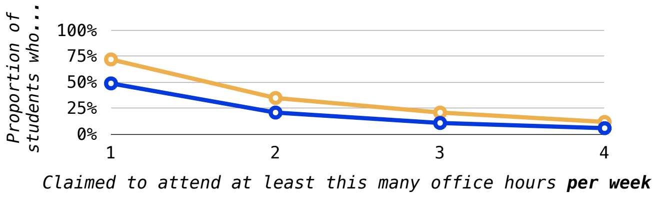
Computing is a collaborative, social discipline

I strongly encourage students to:

- Work with a partner on projects
- Discuss problems with classmates
- Attend office hours with questions

61% had a partner for all projects

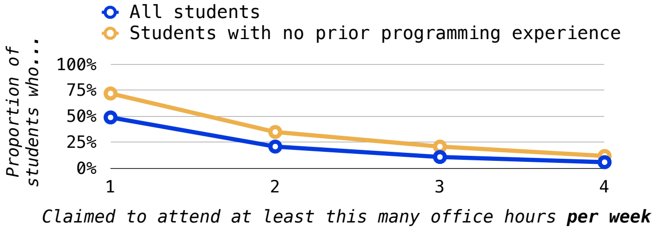
True for **67%** of students with no prior programming experience



Promoting Collaboration

Computing is a collaborative, social discipline

"I attended office hours religiously to get help with homework, projects, and key concepts of the class from all the TA's. Moreover, I worked with other students in office hours to further my understanding of the material by explaining concepts I already understood to them."



Connecting Students to External Communities

Python is maintained and used by a large community of open-source developers.

Benefits to students:

- Targeted explanations of language behavior (40,000+ questions answered on Stack Overflow)
- Online worked examples for many problem domains
- Strong library support for extracurricular projects

Materials & Tools

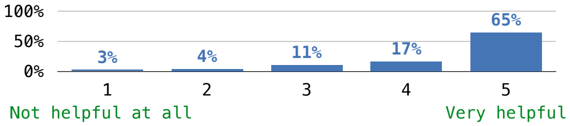
Composing Programs: An Interactive Textbook

Composing Programs is a free online introduction to programming and computer science.

A product of public domain and open source content:

- Derived from Structure and Interpretation of Computer Programs, the former CS 61A text
- Examples diagrammed by the Online Python Tutor

How helpful did you find the online tool for drawing environment diagrams in understanding course material?

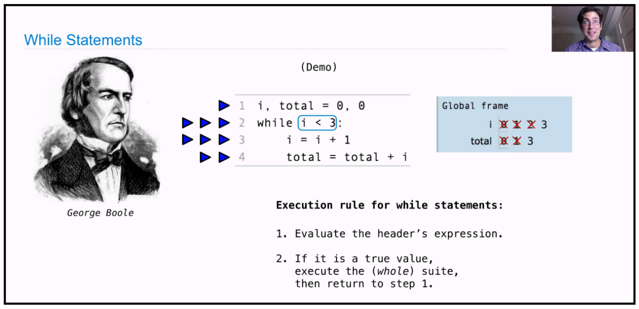


Demo: http://composingprograms.com/pages/16-higher-order-functions.html#functions-as-arguments

Demo: http://composingprograms.com/pages/23-sequences.html#recursive-lists

Lecturing Outside the Lecture Hall

Video lectures allow students to pause & experiment.



While Statements

(Demo)

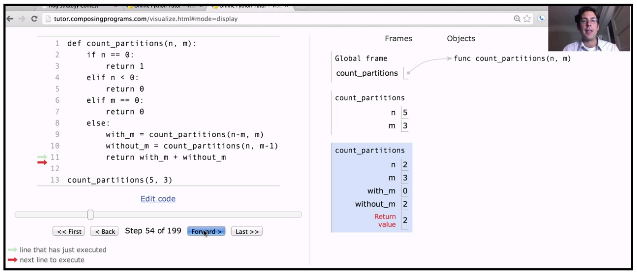
```
i, total = 0, 0
while i < 3:
    i = i + 1
    total = total + i
```

Execution rule for while statements:

1. Evaluate the header's expression.
2. If it is a true value, execute the (*whole*) suite, then return to step 1.

Lecturing Outside the Lecture Hall

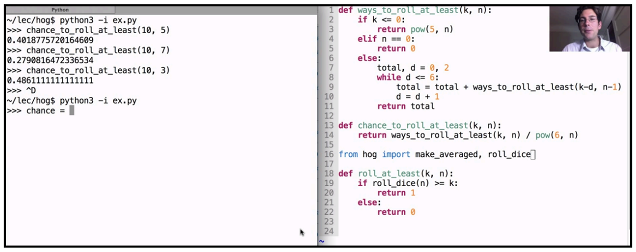
Video lectures allow students to pause & experiment.



```
def count_partitions(n, m):
    if n == 0:
        return 1
    elif n < 0:
        return 0
    elif m == 0:
        return 0
    else:
        with_m = count_partitions(n-m, m)
        without_m = count_partitions(n, m-1)
        return with_m + without_m

count_partitions(5, 3)
```

Lecturing Outside the Lecture Hall

Video lectures allow students to pause & experiment.



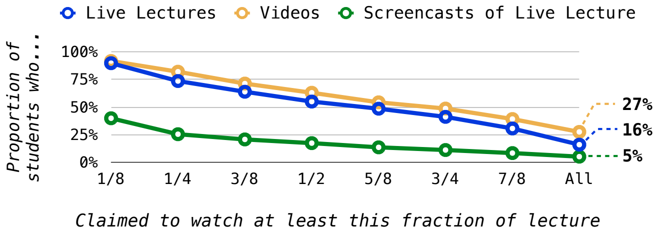
```
~/lec/hog$ python3 -i ex.py
>>> chance_to_roll_at_least(10, 5)
0.4018775720164609
>>> chance_to_roll_at_least(10, 7)
0.2790816472336534
>>> chance_to_roll_at_least(10, 3)
0.4861111111111111
>>> ^D
~/lec/hog$ python3 -i ex.py
>>> chance =
```

```
def ways_to_roll_at_least(k, n):
    if k <= 0:
        return pow(5, n)
    elif n == 0:
        return 0
    else:
        total, d = 0, 2
        while d <= 6:
            total = total + ways_to_roll_at_least(k-d, n-1)
            d = d + 1
        return total

def chance_to_roll_at_least(k, n):
    return ways_to_roll_at_least(k, n) / pow(6, n)

from hog import make_averaged, roll_dice

def roll_at_least(k, n):
    if roll_dice(n) >= k:
        return 1
    else:
        return 0
```

Lecturing Outside the Lecture Hall

Video lectures allow students to pause & experiment.

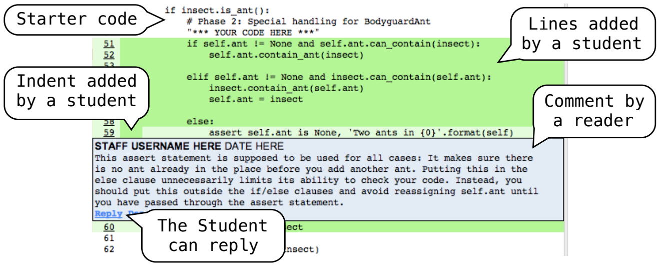
"I watched most of the videos at home where I was able to pause when I didn't understand a concept. I thought that being able to do so really made it so that I could learn at my own pace and thoroughly understand something before moving on."



Online Code Review for Education

CS 61A uses a custom version of Google's (former) code review tool to give feedback about composition.

"Programs must be written for people to read, and only incidentally for machines to execute."
(Structure and Interpretation of Computer Programs)



```
if insect.is_ant():
    # Phase 2: Special handling for BodyguardAnt
    "*** YOUR CODE HERE ***"
    if self.ant != None and self.ant.can_contain(insect):
        self.ant.contain_ant(insect)

    elif self.ant != None and insect.can_contain(self.ant):
        insect.contain_ant(self.ant)
        self.ant = insect

    else:
        assert self.ant is None, 'Two ants in {0}'.format(self)
```

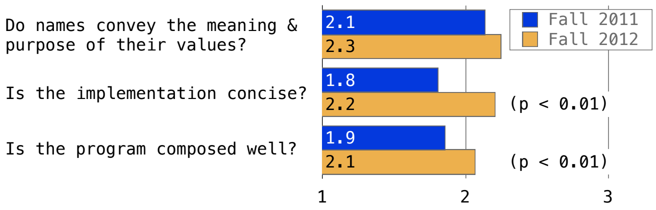
STAFF USERNAME HERE DATE HERE
This assert statement is supposed to be used for all cases: It makes sure there is no ant already in the place before you add another ant. Putting this in the else clause unnecessarily limits its ability to check your code. Instead, you should put this outside the if/else clauses and avoid reassigning self.ant until you have passed through the assert statement.

Online Code Review for Education

CS 61A uses a custom version of Google's (former) code review tool to give feedback about composition.

"Programs must be written for people to read, and only incidentally for machines to execute."
(Structure and Interpretation of Computer Programs)

Blind evaluation of a sample of submissions



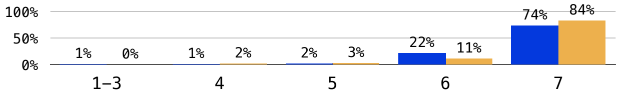
Teaching Composition Quality at Scale, DeNero and Martinis, SIGCSE 2014

Course Trends

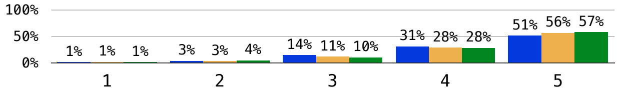
Overall Student Experience

🟦 Fall 2011 🟧 Fall 2012

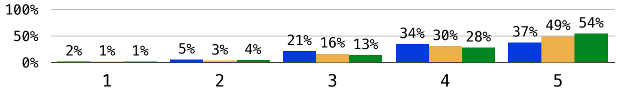
How worthwhile was this course compared with others at U.C.?



How much do you feel you learned (new ideas and skills)?

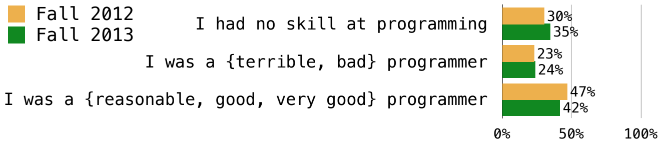


Relative to your expectations, how much do you feel you learned?

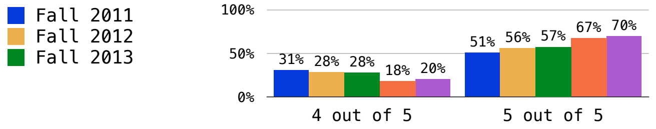


No Programming Experience Required

Before taking this course this semester, how good a programmer did you consider yourself to be?



How much do you feel you learned (new ideas and skills)?



(new): "I had no prior skill at programming"

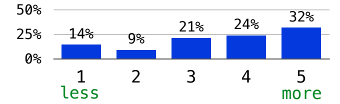
Women in Computer Science 61A

The total number of women in CS 61A increased by **59%** from Fall 2012 to Fall 2013.

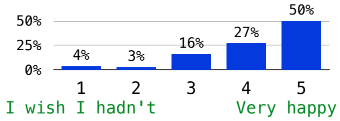
46% of women in Fall 2013 did not have prior programming experience.

33% of students without prior experience were women.

Relative to before you took CS 61A, how interested are you in pursuing computer science as a major?



How happy are you about your decision to take CS 61A?



Thanks