61A Lecture 6

Friday, September 12
Announcements

• Homework 2 due Monday 9/15 @ 11:59pm

• Project 1 due Wednesday 9/17 @ 11:59pm

• Optional Guerrilla section Saturday 9/13 @ 12:30pm in 306 Soda about higher-order functions
  - Organized by Andrew Huang and the readers
  - Work in a group on a problem until everyone in the group understands the solution

• Project party on Monday 9/15, 3pm–4pm in Wozniak Lounge and 6pm–8pm in 2050 VLSB

• Midterm 1 on Monday 9/22 from 7pm to 9pm
  - Details and review materials will be posted next week
  - There will be a web form for students who cannot attend due to a conflict

• There's a pinned Piazza thread to find partners
Lambda Expressions

(Demo)
Lambda Expressions

```python
>>> x = 10

An expression: this one evaluates to a number

>>> square = x * x

Also an expression: evaluates to a function

>>> square = lambda x: x * x

A function with formal parameter x that returns the value of "x * x"

>>> square(4)
16

Important: No "return" keyword!

Must be a single expression

Lambda expressions are not common in Python, but important in general

Lambda expressions in Python cannot contain statements at all!
Lambda Expressions Versus Def Statements

\[ \text{square} = \text{lambda } x: x \times x \quad \text{VS} \quad \text{def square}(x): \text{return } x \times x \]

- Both create a function with the same domain, range, and behavior.
- Both functions have as their parent the frame in which they were defined.
- Both bind that function to the name square.
- Only the def statement gives the function an intrinsic name.
Currying
Function Currying

```python
def make_adder(n):
    return lambda k: n + k

>>> make_adder(2)(3)
5
>>> add(2, 3)
5
```

There's a general relationship between these functions

Curry: Transform a multi-argument function into a single-argument, higher-order function.

Currying was discovered by Moses Schönfinkel and re-discovered by Haskell Curry.
Newton's Method
Newton's Method Background

Quickly finds accurate approximations to zeroes of differentiable functions!

Application: a method for computing square roots, cube roots, etc.

The positive zero of $f(x) = x^2 - a$ is $\sqrt{a}$. (We're solving the equation $x^2 = a$.)
Newton's Method

Given a function $f$ and initial guess $x$,

Repeatedly improve $x$:

1. Compute the value of $f$ at the guess: $f(x)$
2. Compute the derivative of $f$ at the guess: $f'(x)$
3. Update guess $x$ to be:
   $$x \leftarrow x - \frac{f(x)}{f'(x)}$$

Finish when $f(x) = 0$ (or close enough)

Using Newton's Method

How to find the square root of 2?

```python
>>> f = lambda x: x*x - 2
>>> df = lambda x: 2*x
>>> find_zero(f, df)
1.4142135623730951
```

\( f(x) = x^2 - 2 \)
\( f'(x) = 2x \)

How to find the cube root of 729?

```python
>>> g = lambda x: x**3 - 729
>>> dg = lambda x: 3*x**2
>>> find_zero(g, dg)
9.0
```

\( g(x) = x^3 - 729 \)
\( g'(x) = 3x^2 \)
Iterative Improvement
Special Case: Square Roots

How to compute square_root(a)

**Idea:** Iteratively refine a guess x about the square root of a

**Update:** \[ x = \frac{x + \frac{a}{x}}{2} \]

**Babylonian Method**

**Implementation questions:**

- What guess should start the computation?
- How do we know when we are finished?
Special Case: Cube Roots

How to compute cube_root(a)

**Idea:** Iteratively refine a guess $x$ about the cube root of $a$

\[
\text{Update: } \quad x = \frac{2 \cdot x + \frac{a}{x^2}}{3}
\]

**Implementation questions:**

What guess should start the computation?

How do we know when we are finished?
Implementing Newton's Method

(Demo)