Lambda Expressions

61A Lecture 6
Friday, September 7

Lambda Expressions

>>> ten = 10
>>> square = x * x
>>> square = lambda x: x * x
>>> square(4)
16

An expression: this one evaluates to a number
Also an expression: evaluates to a function
A function with formal parameter x and body "return x * x"
Notice: no "return"
Must be a single expression

Lambda expressions are rare in Python, but important in general

Lambda Expressions Versus Def Statements

square = lambda x: x * x VS def square(x):
return x * x

• Both create a function with the same arguments & behavior
• Both of those functions are associated with the environment in which they are defined
• Both bind that function to the name "square"
• Only the def statement gives the function an intrinsic name

Function Currying

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

Currying: Transforming a multi-argument function into a single-argument, higher-order function.

Fun Fact: Currying was discovered by Moses Schönfinkel and later re-discovered by Haskell Curry.

Newton's Method Background

Finds approximations to zeroes of differentiable functions

Application: a method for (approximately) computing square roots, using only basic arithmetic.
The positive zero of \( f(x) = x^2 - a \) is \( \sqrt{a} \)

Newton's Method

Begin with a function \( f \) and an initial guess \( x \)

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 to be: \( x - \frac{f(x)}{f'(x)} \)
Visualization of Newton's Method

Iterative Improvement

(Demo)

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?

Using Newton's Method

How to find the square root of 2?

\[ f(x) = x^2 - 2 \]

How to find the log base 2 of 1024?

\[ g(x) = 2^x - 1024 \]

What number is one less than its square?

\[ h(x) = x^2 - (x+1) \]

Special Case: Cube Roots

How to compute cube_root(a)

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

Update:

\[ x = \frac{2 \cdot x + \frac{a}{x \cdot x \cdot x}}{3} \]

Implementation questions:

What guess should start the computation?

How do we know when we are finished?

Iterative Improvement

(Demo)

Derivatives of Single-Argument Functions

\[ f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \]

(Demo)
Approximating Derivatives

Implementing Newton's Method

```python
def newton_update(f):
    """Return an update function for f using Newton's method."
    return x - f(x) / approx_derivative(f, x)

def approx_derivative(f, x, delta=1e-5):
    """Return an approximation to the derivative of f at x."
    return df/delta

def find_root(f, guess):
    """Return a guess of a zero of the function f, near guess."
    return iter_improve(newton_update(f), lambda x: f(x) == 0, guess)
```

Could be replaced with the exact derivative

Limit approximated by a small value

Definition of a function zero