Lambda Expressions

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

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

Lambda Expressions Versus Def Statements

Both create a function with the same domain, range, and behavior.

Both bind that function to the name square.

Only the def statement gives the function an intrinsic name.

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 that returns the value of \(x \times x\)

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

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

VS

Both create a function with the same domain, range, and behavior.

Both functions have as their parent the environment in which they were defined.

Only the def statement gives the function an intrinsic name.
Function Currying

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

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

There’s a general relationship between these functions.

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

Currying was discovered by Moses Schönfinkel and re-discovered by Haskell Curry.

Schönfinkeling?

Newton’s Method

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

```plaintext
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:
   \[ x = x - \frac{f(x)}{f'(x)} \]

Finish when \( f(x) = 0 \) (or close enough)
```

Using Newton’s Method

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

How to find the square root of 2?

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

How to find the cube root of 729?

```
```

Newton’s Method Background

```
```

Iterative Improvement
Special Case: Square Roots

How to compute $\sqrt{a}$

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

$\text{Update: } x = \frac{x + \frac{a}{x}}{2}$

Implementation questions:
- What guess should start the computation?
- How do we know when we are finished?

Babylonian Method

Special Case: Cube Roots

How to compute $\sqrt[3]{a}$

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

$\text{Update: } x = \frac{2x + \frac{a}{x^2}}{3}$

Implementation questions:
- What guess should start the computation?
- How do we know when we are finished?