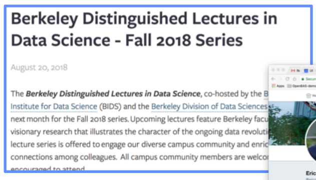
# Intro to Higher Order Functions

David E. Culler
CS8 – Computational Structures in Data Science
http://inst.eecs.berkeley.edu/~cs88

Lecture 4
Sept 17, 2018

## Data Science in the News



Berkeley Distinguished Lectures in Data Science - Fall 2018 Series

August 20, 2018

The ***Berkeley Distinguished Lectures in Data Science***, co-hosted by the B
Institute for Data Science (BIDS) and the Berkeley Division of Data Sciences
next month for the Fall 2018 series. Upcoming lectures feature Berkeley facu
visionary research that illustrates the character of the ongoing data revoluti
lecture series is offered to engage our diverse campus community and enric
connections among colleagues. All campus community members are welco
encouraged to attend

Eric Schmidt @ericschmidt · Sep 11
Delighted to support UC Berkeley's Data Collaboratives – harnessing students' ingenuity & the power of data science to solve real-world problems @BerkeleyDataSci bit.ly/2MZNGRE

Eric Schmidt @ericschmidt · Jun 28
@conservationx uses tech & scientific innovation to protect wildlife & nature. Contribute to new tech projects on their open collaboration platform & submit your bold idea to solve the extinction crisis to the #ConXTechPrize to win $20K & tech support: bit.ly/ConXTechPrize

California Water Data Hackathon

Division of Data Sciences
Enabling research, innovation, and learning across UC Berkeley

Data Collaboratives
Cultivating student-powered solutions to a range of pressing challenges. Find out more here!

## Administrative issues

- Tutoring
  - To help you prepare for exams, we will be hosting small group tutoring will start today -- to sign up, go tiny.cc/cs88tutoring ; we will also be having guerrilla sections starting this Friday from 7-9 pm, it will be in Soda 310"
- Midterm Wed 10/3 evening (6-8 working on room)
- Project 1 Follows midterm

## Computational Concepts Toolbox

- Data type: values, literals, operations,
  - e.g., int, float, string
- Expressions, Call expression
- Variables
- Assignment Statement
- Sequences: tuple, list
- Data structures
- Tuple assignment
- Call Expressions
- Function Definition Statement
- Conditional Statement
- Iteration:
  - data-driven (list comprehension)
  - control-driven (for statement)
  - while statement

## Computational Concepts today

- Higher Order Functions
- Functions as Values
- Functions with functions as argument
- Assignment of function values
- Higher order function patterns
  - Map, Filter, Reduce
- Function factories – create and return functions

Big Idea: Software Design Patterns

## Today's Notebook

- http://bit.ly/cs88-fa18-L04
- http://datahub.berkeley.edu/user-redirect/interact?account=data-8&repo=cs-connector&branch=gh-pages&path=L04-hof.ipynb

## Iteration Review

- When should we use a for loop, rather than list comprehension?

## Higher Order Functions

- Functions that operate on functions
- A function

```
def odd(x):
    return (x%2==1)

>>> odd(3)
True
```

Why is this not 'odd' ?

- A function that takes a function arg

```
def filter(fun, s):
    return [x for x in s if fun(x)]

>>> filter(odd, [0,1,2,3,4,5,6,7])
[1, 3, 5, 7]
```

## Higher Order Functions (cont)

- A function that returns (makes) a function

```
def leq_maker(c):
    def leq(val):
        return val <= c
    return leq
```

```
>>> leq_maker(3)
<function leq_maker.<locals>.leq at 0x1019d8c80>

>>> leq_maker(3)(4)
False

>>> filter(leq_maker(3), [0,1,2,3,4,5,6,7])
[0, 1, 2, 3]
>>>
```

## Three super important HOFS

`map(function_to_apply, list_of_inputs)`

Applies function to each element of the list

`filter(condition, list_of_inputs)`

Returns a list of elements for which the condition is true

`reduce(function, list_of_inputs)`

Reduces the list to a result, given the function

## One more example

- What does this function do?

```
def split_fun(p, s):
    """ Returns <you fill this in>."""
    return [i for i in s if p(i)], [i for i in s if not p(i)]
```

```
>>> split_fun(leq_maker(3), [0,1,2,3,4,5,6])
([0, 1, 2, 3], [4, 5, 6])
```

## Function Factories

```
def linemaker(m, b):
    def linefun(x):
# Create a function that embeds the parameters of the line
        return m*x + b
# Return that dynamically created function
return linefun
```

```
def make_decoder(code_map):
    """Make a decoder function specified by a map"""
    def decode(code):
        for (code_num, desc) in code_map:
            if code == code_num:
                return desc
        return "unknown"
    return decode
```

## Computational Concepts today

- Higher Order Functions
- Functions as Values
- Functions with functions as argument
- Assignment of function values
- Higher order function patterns
  - Map, Filter, Reduce
- Function factories – create and return functions

Big Idea: Software Design Patterns

## Recap: Data or Code?