



UC Berkeley EECS  
Lecturer  
Michael Ball

# Computational Structures in Data Science

---



## HOFs & Environment Diagrams



## Announcements

---

- The weeks-long saga is over!
- **Everyone getting off the waitlist today!**
- **Please see Ed about 61A assignment transfers.**
- if I don't respond to your email about enrollment, I'm sorry.



UC Berkeley EECS  
Lecturer  
Michael Ball

# Computational Structures in Data Science

---



## HOFs and Sequences



## Functional List Operations

- Goal: Transform a list, and return a new result
- We'll use 3 functions that are hallmarks of functional programming
- Each of these takes in a function and a sequence

Function Name	Action	Input arguments	Input Fn. Returns	Output List
<b>map</b>	Transform every item	1 argument (each item)	"Anything", a new item	List of the same length, but possibly new values
<b>filter</b>	Return a list with fewer items	1 argument (each item)	A Boolean	List with possibly fewer items, but values are the same
<b>reduce</b>	"Combine" items together	2 arguments (current item, and the previous result)	Type should match the type each item	Usually a "single" item



## Filter and Non-Boolean Functions

---

```
>>> list(filter(add_2, range(10)))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> if 0:
...     print("0 is a true value")
... else:
...     print("0 is a false value")
...
0 is a false value
```

**Why is 0 in the output of our filter?**



## Filter and Non-Boolean Functions

---

```
>>> [ x for x in range(10) if add_2(x) ]
```

**Why is 0 in the output of our filter?**

Filter calls our function, but always returns the original value!



## Today's Task: Acronym

---

Input: "The University of California at Berkeley"

Output: "UCB"

```
def acronym(sentence):  
    """ (Some doctests)  
    """  
    words = sentence.split()  
    return reduce(add, map(first_letter, filter(long_word,  
words)))
```

P.S. Pedantry alert: This is really an *initialism* but that's rather annoying to say and type. ☺ (However, the code we write is the same, the difference is in how you pronounce the result.) The more you know!



## Acronym With HOFs

---

What is we want to control the filtering method?

```
def keep_words(word):  
    specials = ['Los']  
    return word in specials or long_word(word)
```

```
def acronym_hof(sentence, filter_fn):  
    words = sentence.split()  
    return reduce(add, map(first_letter,  
filter(filter_fn, words)))
```





## Functional List Operations

- Goal: Transform a list, and return a new result
- We'll use 3 functions that are hallmarks of functional programming
- Each of these takes in a function and a sequence

Function Name	Action	Input arguments	Input Fn. Returns	Output List
<b>map</b>	Transform every item	1 argument (each item)	"Anything", a new item	List of the same length, but possibly new values
<b>filter</b>	Return a list with fewer items	1 argument (each item)	A Boolean	List with possibly fewer items, but values are the same
<b>reduce</b>	"Combine" items together	2 arguments (current item, and the previous result)	Type should match the type each item	Usually a "single" item



UC Berkeley EECS  
Lecturer  
Michael Ball

# Computational Structures in Data Science

---



## Functions That Return Functions



## Learning Objectives

---

- Learn how to use and create higher order functions:
- Functions can be used as data
- Functions can accept a function as an argument
- **Functions can return a new function**



## Review: What is a Higher Order Function?

---

- A function that takes in another function as an argument

OR

- A function that returns a function as a result.



# Higher Order Functions

---

- **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
```

```
>>> [x for x in range(7) if leq_maker(3)(x)]  
[0, 1, 2, 3]
```



UC Berkeley EECS  
Lecturer  
Michael Ball

# Computational Structures in Data Science

---



## Environment Diagrams



## Environment Diagrams

---

- Organizational tools that help you understand code
- **Terminology:**
  - **Frame:** keeps track of variable-to-value bindings, each function call has a frame
  - **Global Frame:** global for short, the starting frame of all python programs, doesn't correspond to a specific function
  - **Parent Frame:** The frame of where a function is defined (default parent frame is global)
  - **Frame number:** What we use to keep track of frames,  $f_1$ ,  $f_2$ ,  $f_3$ , etc
  - **Variable vs Value:**  $x = 1$ .  $x$  is the **variable**, 1 is the **value**



## Environment Diagrams Reminders

---

1. Always draw the global frame first
2. When evaluating assignments (lines with single equal), always evaluate right side first
3. When you CALL a function MAKE A NEW FRAME!
4. When assigning a primitive expression (number, boolean, string) write the value in the box
5. When assigning anything else (lists, functions, etc.), draw an arrow to the value
6. When calling a function, name the frame with the intrinsic name – the name of the function that variable points to
7. The parent frame of a function is the frame in which it was defined in (default parent frame is global)
8. If the value for a variable doesn't exist in the current frame, search in the parent frame





## Example 1

---

```
a = "chipotle"
```

```
b = 5 > 3
```

```
c = 8
```

```
def foo(c):  
    return c - 5
```

```
def bar():  
    if b:  
        a = "taco bell"
```

```
result1 = foo(10)
```

```
result2 = bar()
```



## Example 2

---

```
def make_adder(n):  
    def adder(k):  
        return k + n  
    return adder
```

```
n = 10  
add_2 = make_adder(2)  
x = add_2(5)
```



## Python Tutor Example #3

---

```
add_2 = make_adder(2)
add_3 = make_adder(3)
```

```
x = add_2(2)
def compose(f, g):
    def h(x):
        return f(g(x))
    return h
```

```
add_5 = compose(add_2, add_3)
z = add_5(x)
```



## Demo

---

Example 1:

- [Primitives and Functions: Environment Diagram Python Tutor:](#)

Example 2:

- [make\\_adder Higher Order Function: Environment Diagram Python Tutor Link](#)

Example 3:

- [Compose Python Tutor Link](#)



## Environment Diagram Tips / Links

---

- NEVER draw an arrow from one variable to another.
- Useful Resources:
  - [http://markmiyashita.com/cs61a/environment\\_diagrams/rules\\_of\\_environment\\_diagrams/](http://markmiyashita.com/cs61a/environment_diagrams/rules_of_environment_diagrams/)
  - <http://albertwu.org/cs61a/notes/environments.html>