Computational Structures in Data Science

UC Berkeley EECS
Lecturer
Michael Ball

Lecture #7: Higher Order Functions & Environments

Feb. 14, 2020

cs88.org

Announcements!

- Late Adds:
  - If you filled out the form on Piazza you'll hear from us soon.
  - If you're coming from 61A, you can copy over Labs and HW 0-2
    - The roster is delayed ☹, so please send us an email so we can add you
    - If you want E.C. for lab practice questions you'll need to turn in lab 2 – you'll get an extension to turn in lab since you cannot try the practice until we add you.
- No Class Monday, please attend any lab Tues!

Computational Concepts Toolbox

- Data type: values, literals, operations,
  - e.g., int, float, string
- Expressions, Call expression
- Variables
- Assignment Statement
- Sequences: list
- Data structures
- 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
  - **Functions that *return* a function**
- **"Environments"**
  - **These are a tools to help us understand what variables or parameters are accessible in which functions.**

Three super important HOFS

* For the builtin filter/map, you need to then call list on it to get a list.
If we define our own, we do not need to call list

```
list(map(function_to_apply, list_of_inputs))
```

Applies function to each element of the list

```
list(filter(condition, list_of_inputs))
```

Returns a list of elements for which the condition is true

```
reduce(function, list_of_inputs)
```

Applies the function, combining items of the list into a "single" value.

Today's Task: Acronym

```
Input: "The University of California at Berkeley"

Output: "UCB"

def acronym(sentence):
     """YOUR CODE HERE"""
```

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!

MAP

```
list(map(function_to_apply, list_of_inputs))
```

Transform each of items by a function.

e.g. square()

Inputs (Domain):

- Function
- Sequence

Output (Range):

- A sequence

```
def map(function, sequence):
    return [ function(item) for item in sequence ]
```

FILTER

```
list(filter(function, list_of_inputs))
```

Keeps each of item where the function is true.

Inputs (Domain):

- Function
- Sequence

Output (Range):

- A sequence

```
def filter(function, sequence):
    return [ item for item in sequence
                  if function(item) == True ]
```

Higher Order Functions

- **Functions that operate on functions**
- **A function**

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

odd(3)
True
```

- **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]
```

Why is this not ‘odd’ ?

What does this do?

```
list(filter(return_false,
       range(100)
))

Assume return_false(42) == False
```

A) range(0, 100) # A standard range object

B) [0, 1, 2, … 96, 97, 98, 99]

C) [ ]

D) Error

E) I'm lost.

REDUCE

```
reduce(function, list_of_inputs)
```

Successively **combine** items of our sequence

- function: add(), takes 2 inputs gives us 1 value.

Inputs (Domain):

- Function, with 2 inputs
- Sequence

Output (Range):

- An item, specifically, the output of our function.

```
def reduce(function, sequence):
    result = function(sequence[0], sequence[1])
    for index in range(2, len(sequence)):
        result = function(result, sequence[index])
    return result
```

Note: This reduce is slightly different than the homework one….

Question: Inputs to our reducer?

```
reduce(sub, range(5))
reduce(add, range(5))
reduce(REDUCER, range(5))

How many inputs should our reducer accept?
```

A) `0`
B) `1`
C) 2
D) Unlimited
E) I'm lost.

Question: What's the output?

```
reduce(add, range(5))
```

What is the value of this expression?

A) 0
B) 9
C) 10
D) 15
E) Error

Question: What's the output?

```
reduce(sub, range(5))
```

What is the value of this expression?

A) 0
B) - 5
C) -10
D) -15
E) Error

Map, Filter, Reduce

Each takes in a function and a sequence

- Function – what this does depends on your goal!
    - Map: Returns a new value
    - Filter: Returns a boolean value
    - Reduce: Takes in 2 values, "combines" them
- Sequence

Always consider your output!

- Am I returning a new list of different items?
- Am I excluding items from my list?
- Do I need a list as my result?

Returning a New Function

- **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]
```

Environment Diagrams aka what python tutor makes

Environment Diagrams are 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, f1, f2, f3, etc
- **Variable** vs **Value**: x = 1. x is the **variable**, 1 is the **value**

Steps:

1 Draw the global frame
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) right the value in the box
5 When assigning anything else, **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 isn't in the current frame, search in the parent frame

NEVER EVER EVER draw an arrow from one variable to another.

Source:
http://markmiyashita.com/cs61a/environment_diagrams/rules_of_environment_diagrams/

Another example

- **Higher Order Functions**

```
http://pythontutor.com/composingprograms.html#code=def%20square%28x%29%3A%0A%20%20%20%20return%20x%20*%20x%0A%20%20%20%20%0As%20%3D%20square%0Ax%20%3D%20s%283%29%0A%0Adef%20make_adder%28n%29%3A%0A%20%20%20%20def%20adder%28k%29%3A%0A%20%20%20%20%20%20%20%20return%20k%20%2B%20n%0A%20%20%20%20return%20adder%0A%20%20%20%20%0Aadd_2%20%3D%20make_adder%282%29%0Aadd_3%20%3D%20make_adder%283%29%0Ax%20%3D%20add_2%28x%29%0A%0Adef%20compose%28f,%20g%29%3A%0A%20%20%20%20def%20h%28x%29%3A%0A%20%20%20%20%20%20%20%20return%20f%28g%28x%29%29%0A%20%20%20%20return%20h%0A%0Aadd_5%20%3D%20compose%28add_2,%20add_3%29%0Ay%20%3D%20add_5%28x%29%0A%0Az%20%3D%20compose%28square,%20make_adder%282%29%29%283%29&cumulative=true&mode=edit&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D
```

Computational Concepts today

- Higher Order Functions
- Functions as Values
- Functions with functions as argument
- Functions with functions as return values
- Environment Diagrams

Big Idea: Software Design Patterns