



Computational Structures in Data Science



UC Berkeley EECS
Lecturer
Michael Ball

Lecture #6: Higher Order Functions



Computing in the News

- Lots of discussion about facial recognition
- "The End of Privacy as We Know It?"
 - <https://www.nytimes.com/2020/02/10/podcasts/the-daily/facial-recognition-surveillance.html>
 - " A secretive start-up promising the next generation of facial recognition software has compiled a database of images far bigger than anything ever constructed by the United States government: over three billion, it says. Is this technology a breakthrough for law enforcement — or the end of privacy as we know it?"
- "**Facial Recognition Moves Into a New Front: Schools**"
 - <https://www.nytimes.com/2020/02/06/business/facial-recognition-schools.html>
 - " Lockport's Aegis software studies images of faces captured by 300 newly installed cameras and calculates whether those faces match a "persons of interest" database compiled by school administrators; if the system finds a match, it alerts security staff who vet the image for confirmation."



iClicker Check-In

- How are you feeling about CS88 so far?
- A) It's going very well! 😊
- B) It's going good...
- C) So, so...
- D) Not so great...
- E) Terribly. ☹️



Announcements!

- **Bonus Questions in Labs**
 - **Extra practice, not just coding.**
 - **Goal is to focus on the concepts.**
 - **Different styles of questions promote different ways of thinking and synthesizing information.**
 - **Half point extra for each lab you complete, ~5 points throughout the semester. (Labs are 40 points total.)**
 - **<https://codestyle.herokuapp.com/cs88-lab02>**



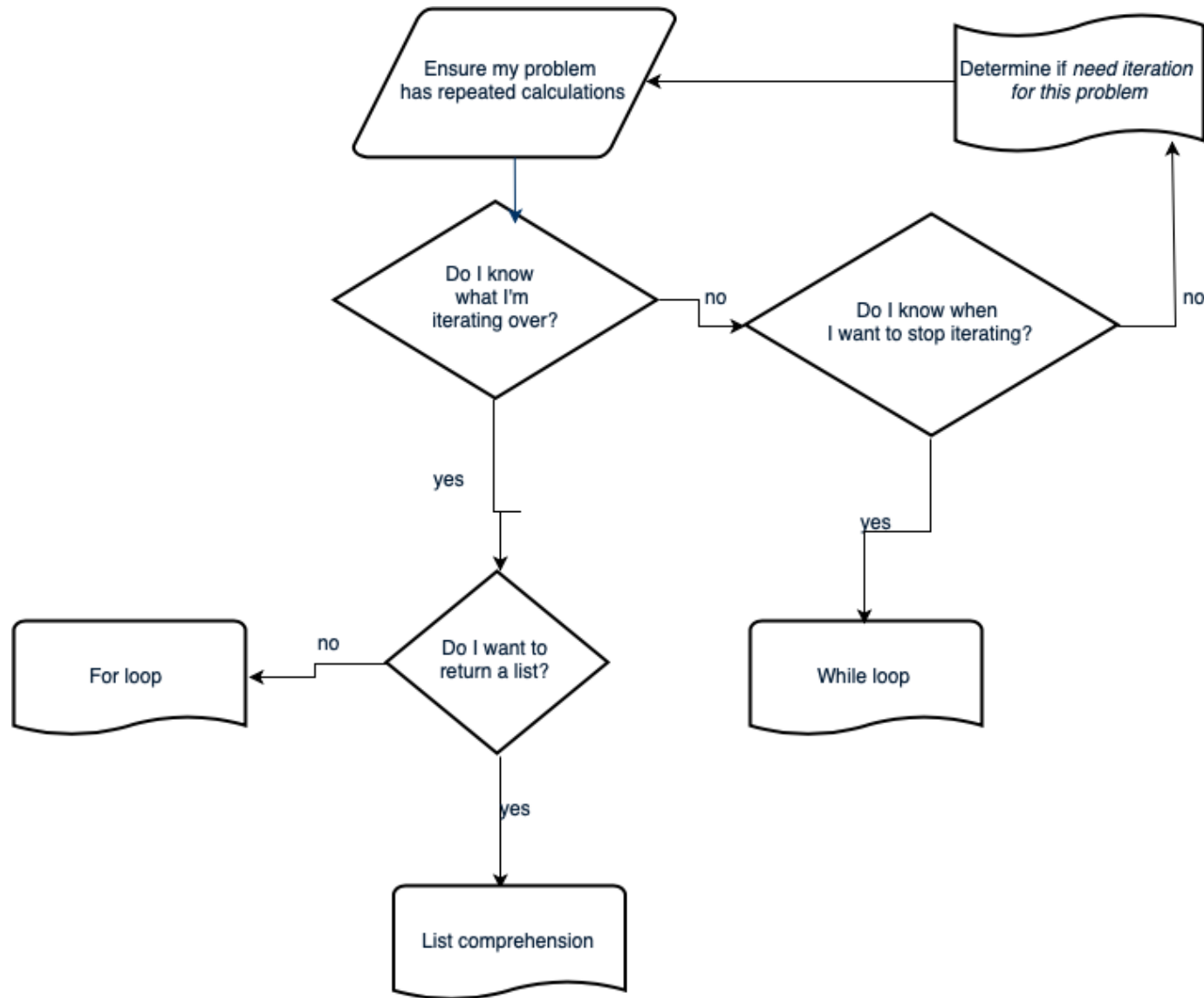
Computational Concepts Toolbox

- **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





Iteration flow chart





Computational Concepts today

- **Higher Order Functions**
 - **Functions as Values**
 - **Functions with functions as argument**
 - **Functions that *return* a function**
- In Python, we use () to *call* a function.
- We don't need to do this!



Functions: A New Kind of Data!

Lists, Numbers, Strings: All kinds of data

Code *is its own* kind of data, too!

Why?

More expressive programs, a new kind of abstraction.

This will be one of the trickier concepts in CS88.

Big Idea: Software Design Patterns



iClicker Question

Question: What's the result of the following?

```
def greet(name):  
    print('Hello, ' + name)
```

```
hello = greet
```

```
def greet(name):  
    print('Hi, ' + name)
```

```
hello('CS88')
```

- A) Error**
- B) prints "Hello, CS88"**
- C) prints "Hi, CS88"**
- D) "I'm lost...."**



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 ]
```



What does this do?

```
list(map(capitalize,  
        ['michael', 'Alex', 'Srinath', 'julia']  
))
```

Assume `capitalize('michael') == 'Michael'`

- A) ['michael', 'Alex', 'Srinath', 'julia']
- B) ['Michael', 'Alex', 'Srinath', 'Julia']
- C) []
- D) Error
- E) I'm lost.



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) ]
```



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
```




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]
```



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