UC Berkeley EECS
Lecturer
Michael Ball

Computational Structures in Data Science

Lecture #4: Lists and Functions

February 3, 2020

https://cs88.org

Tech in the News

- “Scientists are unraveling the Chinese coronavirus with unprecedented speed and openness”
  - https://www.washingtonpost.com/science/2020/01/24/scientists-are-unraveling-chinese-coronavirus-with-unprecedented-speed-openness/
- “An AI Epidemiologist Sent the First Warnings of the Wuhan Virus”
  - https://www.wired.com/story/ai-epidemiologist-wuhan-public-health-warnings/

Announcements!

- Register iClickers at any point
- CS Mentors Drop-In Sections
    - https://piazza.com/class/k5kga9pwx0l754?cid=47
    - Amazing student group that provides tutoring
- Midterm: Weds 3/4, 7-9pm.
- Final Exam:
    - Trying to only have 1 exam. Section 2, look out for a message soon.
- If you have DSP accommodations, please let us know! We're here to help. ☺

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

Control Structures Review

- **The result of** *list(range(0,10))* **is…**
-
- **A)** *[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]*
- **B)** *[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]*
- **C)** *[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]*
- **D)** *[1, 2, 3, 4, 5, 6, 7, 8, 9]*
- **E) an error**

- **http://bit.ly/88Lec3Q1**

Solution:
A) *list(range(m,n))* **creates a list with elements from m to n-1.**

Types of Things We’ve Seen So Far

- `ints` / Integers
  - `1, -1, 0`, …
- `floats` (“decimal numbers”)
  - `1.0, 3.14159, 20.0`
- `strings`
  - `“Hello”, “CS88”`
- `list` / Arrays
  - `['CS88', 'DATA8', 'POLSCI2', 'PHILR1B’]`
- `functions`
  - `max(), min()`

Additional Types

- `ranges`
  - A function, but is also its own type
  - range(0, 10)
  - A “sequence”.
- `tuple` / A list you cannot change.
  - `('CS88', 'DATA8', 'POLSCI2', 'PHILR1B’)`
- More sequence types:
  - map
  - filter

`for` statement – iteration control

- Repeat a block of statements for a structured sequence of variable bindings

```
<initialization statements>
for <variables> in <sequence expression>:
  <body statements>

<rest of the program>
```

```
def cum_OR(lst):
   """Return cumulative OR of entries in lst.
   >>> cum_OR([True, False])
   True
   >>> cum_OR([False, False])
   False
   """
   co = False
   for item in lst:
         co = co or item
   return co
```

`while` statement – iteration control

- Repeat a block of statements until a predicate expression is satisfied

```
<initialization statements>
while <predicate expression>:
    <body statements>

<rest of the program>
```

```
def first_primes(k):
    """ Return the first k primes.
    """
    primes = []
    num = 2
    while len(primes) < k :
        if prime(num):
            primes = primes + [num]
        num = num + 1
    return primes
```

Data-driven iteration

- describe an expression to perform on each item in a sequence
- let the data dictate the control
- *"List Comprehensions"*

```
[ <expr with loop var> for <loop var> in <sequence expr > ]
```

```
def dividers(n):
    """Return list of whether numbers greater than 1 that divide n.

    >>> dividers(6)
    [True, True]
    >>> dividers(9)
    [False, True, False]
    """
    return [divides(n,i) for i in range(2,(n//2)+1) ]
```

Control Structures Review

- **The result of** *[i for i in range(3,9) if i % 2 == 1]* **is…**
-
- **A)** *[3, 4, 5, 6, 7, 8, 9]*
- **B)** *[3, 4, 5, 6, 7, 8]*
- **C)** *[1, 3, 5, 7, 9]*
- **D)** *[3, 5, 7, 9]*
- **E)** *[3, 5, 7]*
-
- ***http://bit.ly/88Lec3Q2***

Solution:
E) *[3, 5, 7]*

Types And Actions

- Every *object* has a bunch of functions or actions that you can use with that object.
- len()
- + , - , *, /, **
- min(), max()
- Strings:
  - <string>.split(<sep>) → List
  - <string>.join(<list>) → String

```
thing = [ print('I like '+ course) for course in courses
]
```

iClicker Question

A) Nothing
B) [“I like CS88”, “I like DATA8”, …]
C) []
D) [None, None, None, None]
E) Error

Control Structures Review

The result of *len([i for i in range(1,10) if i % 2 == 0)])* **is…**

A) 5
B) 4
C) 3
D) 2
E) 1

http://bit.ly/88Lec3Q3

Solution:
B) *len([2, 4, 6, 8]) == 4*

Example: “Acronym”

“The University of California at Berkeley” → “UCB"

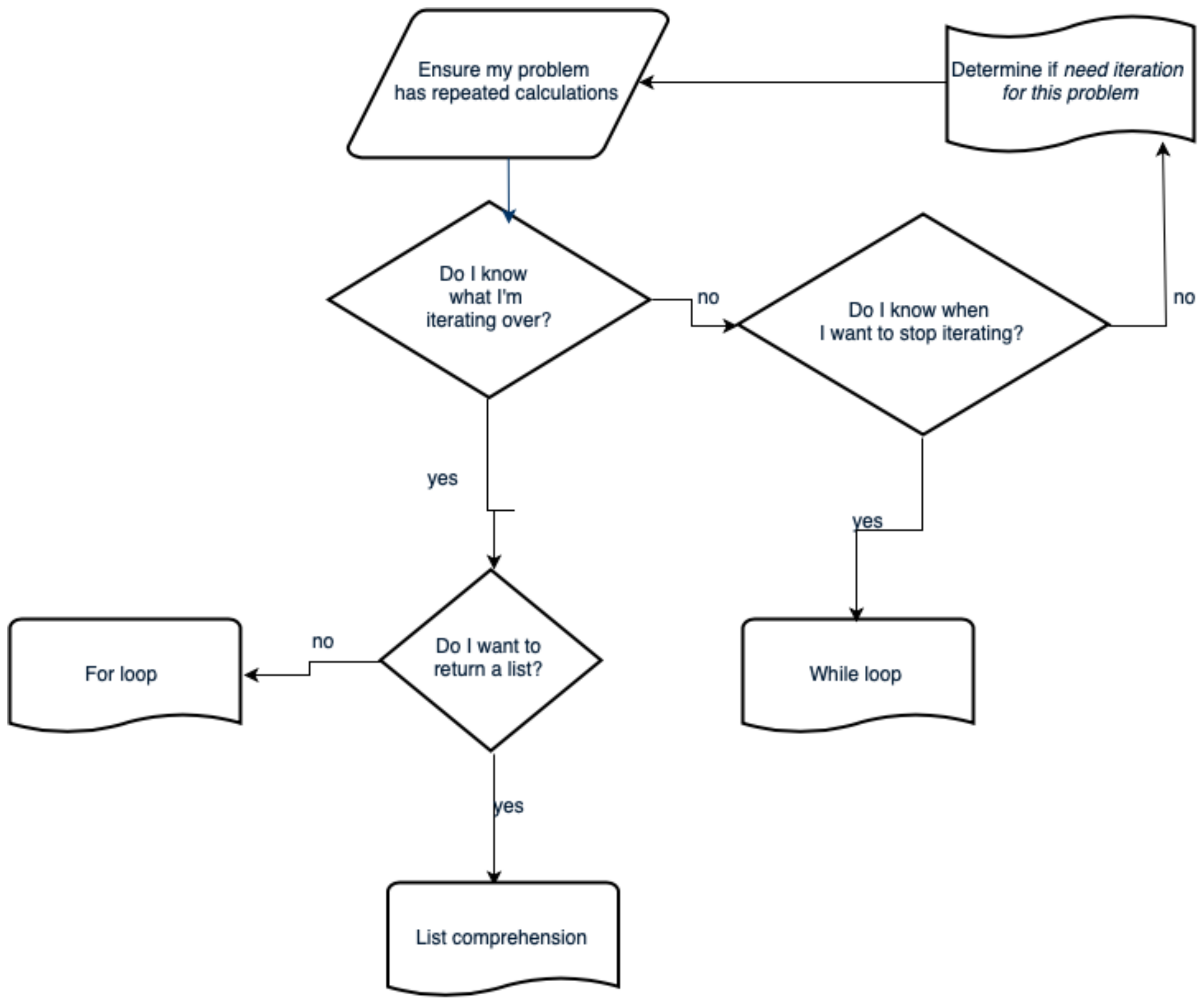
iClicker Question

```
>>> uni = ‘The University of California at Berkeley’
>>> words = uni.split(‘ ‘)
>>> thing = [ w[0] for w in words ]
```

A) []
B) [‘The’, ’University’, ‘of’, ‘California’, ‘at’, ‘Berkeley’]
C) ‘TUoCaB’
D) [‘T’, ‘U’, ‘o’, ‘C’, ‘a’, ‘B’]
E) Error

Solution:
D)

Iteration flow chart


Ensure my problem has repeated calculations
Determine if *need iteration for this problem*
Do I know what I'm iterating over?
no
Do I know when I want to stop iterating?
no
yes
yes
Do I want to return a list?
no
For loop
While loop
yes
List comprehension

An Interesting Example

$$\sum_{k=1}^{5} \boxed{k} = 1 + 2 + 3 + 4 + 5 \qquad = 15$$

$$\sum_{k=1}^{5} \boxed{k^3} = 1^3 + 2^3 + 3^3 + 4^3 + 5^3 \qquad = 225$$

$$\sum_{k=1}^{5} \boxed{\frac{8}{(4k-3)\cdot(4k-1)}} = \frac{8}{3} + \frac{8}{35} + \frac{8}{99} + \frac{8}{195} + \frac{8}{323} \qquad = 3.04$$

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%20squar
e%28x%29%3A%0A%20%20%20%20return%20x%20*%20x%0A%20%20%20%20%0A
s%20%3D%20square%0Ax%20%3D%20s%283%29%0A%0Adef%20make_adder%28
n%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%2
0%20%20%20%0Aadd_2%20%3D%20make_adder%282%29%0Aadd_3%20%3D%20m
ake_adder%283%29%0Ax%20%3D%20add_2%28x%29%0A%0Adef%20compose%2
8f,%20g%29%3A%0A%20%20%20%20def%20h%28x%29%3A%0A%20%20%20%20%2
0%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%2
9%283%29&cumulative=true&mode=edit&origin=composingprograms.js
&py=3&rawInputLstJSON=%5B%5D
```

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

Why is this not 'odd' ?

```
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 (Wait for lab)

* 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)
```

Reduces the list to a result, given the function

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