

UC Berkeley EECS
Lecturer
Michael Ball

Computational Structures in Data Science

Lecture #3: Loops and Functions

January 31, 2020 <https://cs88.org>

1

Administrivia


- More spots opened for lab sections
- Please try to attend labs you signed up for. (See Piazza)
- Reminder: iClickers next week.
 - Can register them at any time during the semester.
- We're going to be doing live coding, so review videos, not just slides.

1/31/2020 UCB CS88 Sp20 L3

2

Computational Concepts Today

- Conditional Statement
- Functions
- Iteration



1/31/2020 UCB CS88 Sp20 L3

3

Things you can do now:

- Write a program that makes a decision.
- Write your own functions
- Use loops so you can process lots of data.

1/31/2020 UCB CS88 Sp20 L3

4

A Brief Review: Files, Terminals

- This is mostly lab 0 review.
- It will take time to get used to everything!
- Things we'll do:
 - Use the command line to run files
 - Review the difference between notebooks and files

8/26/16 UCB CS88 Fa16 L1

5

Let's talk About Python

- Expression `3.1 * 2.6`
- Call expression `max(0, x)`
- Variables
- Assignment Statement `x = <expression>`
- Define Function: `def <function name> (<parameter list>):`
- Control Statements:
 - if ...
 - for ...
 - while ...
 - list comprehension

1/31/2020 UCB CS88 Sp20 L3

6

Conditional statement

- Do some statements, conditional on a *predicate* expression

```
if <predicate>:
    <true statements>
else:
    <false statements>
```

- Example:

```
if (temperature>98.6):
    print("fever!")
else:
    print("no fever")
```

1/31/2020

UCB CS88 Sp20 L3

7

7

Defining Functions

```
def <function name> (<argument list>) :
```

return  expression

- Abstracts an expression or set of statements to apply to lots of instances of the problem
- A function should *do one thing well*

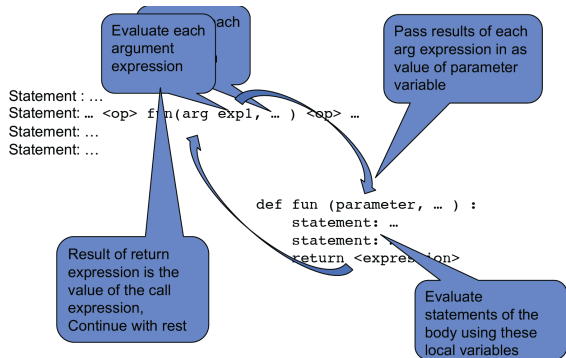
1/31/2020

UCB CS88 Sp20 L3

8

8

Functions: Calling and Returning Results



1/31/2020

UCB CS88 Sp20 L3

9

9

Functions and Arguments

```
>>> x = 3
>>> y = 4 + max(17, x + 4) * 0.5
>>> z = x + y
>>> print(z)
15.5
```

```
def max(x, y):
    return x if x > y else y

def min(x, y):
    if x > y:
        return x
    else:
        return y
```

8/26/16

UCB CS88 Fall 11

10

10

How to write a good Function

- Give a descriptive name
 - Function names should be lowercase. If necessary, separate words by underscores to improve readability. Names are extremely suggestive!
- Chose meaningful parameter names
 - Again, names are extremely suggestive.
- Write the docstring to explain *what* it does
 - What does the function return? What are corner cases for parameters?
- Write doctest to show what it should do
 - Before you write the implementation.

Python Style Guide: <https://www.python.org/dev/peps/pep-0008/>

1/31/2020

UCB CS88 Sp20 L3

11

11

Example: Prime Numbers

```
1 def prime(n):
2     """Return whether n is a prime number.
3
4     >>> prime(2)
5     True
6     >>> prime(3)
7     True
8     >>> prime(4)
9     False
10
11
12     return "figure this out"
```

Prime number

From Wikipedia, the free encyclopedia

"Prime" redirects here. For other uses, see Prime (disambiguation).
A prime number (or a prime) is a natural number greater than 1 that cannot be formed by multiplying two smaller natural numbers. A natural number greater than 1 that is not prime is called a composite number. For example, 5 is prime because the only ways of writing it as a product, 1 × 5 or 5 × 1, involve 5 itself. However, 6 is composite because it is the product of two numbers (2 × 3) that are both greater than 1. Primes are central in number theory because of the fundamental theorem of arithmetic: every natural number greater than 1 is either a prime itself or can be factored as a product of primes that is unique up to their order.

Why do we have prime numbers?

<https://www.youtube.com/watch?v=e4kevnd2vPI&t=72s&index=6&list=PL17CIGMLR0Xz3vNK31TG7mJlzmF78vsFO>

1/31/2020

UCB CS88 Sp20 L3

12

12

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
```

1/31/2020

UCB CS88 Sp20 L3

13