Lecture 3: Functions and Lops
Announcements

• In person next week!
• Please check the CS88 Google Calendar for locations
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>):
  
  if ...

  while ...
Python: Definitions and Control
Learning Objectives

• Create your own functions.
• Write a loop to run the same code multiple times
• Use conditionals to control when a loop stops
Conditional Statement

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

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

• Example:

```python
if (temperature>37.2):
    print("fever!")
else:
    print("no fever")
```
Defining Functions

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

\[
\begin{align*}
    x &= 3 \\
    y &= 4 + \max(17, x+6) \times 0.1 \\
    z &= x / y
\end{align*}
\]

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

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

• Write doctest to show what it should do
  – Before you write the implementation.
Functions: Calling and Returning Results

Python Tutor

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

x = 3
y = 4 + max(17, x + 6) * 0.1
z = x / y
**Doctests**

- 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.
  - `python3 -m doctest [-v] file.py`
Returns and Values

• All functions always return SOME value.
• If you don’t specify return, the value is None.
Computational Structures in Data Science

Iteration with *while* Loops
Learning Objectives

• Use a while loop to repeat some task.
• Write an expression to control when a while loop stops executing
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>
Sum The Numbers

- This is a task we'll see many times!

```python
total = 0
n = 1
while n <= 10:
    total += n
    n += 1
print(total)
```
Computational Structures in Data Science

Environments & Higher Order Functions
Learning Objectives

• Use environment diagrams to model Python
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, f1, f2, f3, etc
  – **Variable vs Value**: x = 1. x is the variable (name), 1 is the value
Environment Diagrams 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) write 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 variable isn’t in the current frame, search in the parent frame
Environment Diagram Tips / Links

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

• Useful Resources:
  – http://albertwu.org/cs61a/notes/environments.html
Iteration With for Loops
Learning Objectives

• Compare a for loop and a while loop.
• Learn to use `range()`
• Use a string as a sequence of letters
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>
Sequences are a type of data that can be broken down into smaller parts.

Common sequences:
- range() – gimme all the numbers
- strings
- lists (next week!)

We'll start with two basic facts:
- range(10) is the numbers 0 to 9, or range(0, 10)
- [] means "indexing" an item in a sequence.
- "Hello"[0] == "H"
Data-Driven Iteration

• describe an expression to perform on each item in a sequence
• let the data dictate the control

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