Introduction to Computer Programming

Lecture 5
Simple Functions and Simple IF Statements
Important!!!

First Midterm Exam (20%) Tuesday, Feb. 25
– Chapters 1, 2, 4 (beginning of 3) in Gaddis

Assignment #2 must be submitted by 11:55PM tonight!!!

Next Homework is posted (assign #3):

http://cs.nyu.edu/courses/spring14/CSCI-UA.0002-008/Assignments
Today

• NEW: IF statement, Defining your own functions
• We will be writing programs:
  – Salary with bonus
  – Overtime Pay
  – Is the temperature too hot, too cold, just right?
  – Vending machine
  – Is your lifestyle healthy?
  – How to bake a cake

• Demo: Game of Stones
CLOSE YOUR LAPTOPS!
a = 10000/6
b = format (a, ‘.2f’) # format as a 2 digit float
c = format (a, ‘.5f’) # format as a 5 digit float
d = format (a, ‘,.5f’) # format as a 5 digit float + # comma separators
e = format (a, ‘20,.2f’) # format as a 2 digit # float + commas + 20 # character min field width
Review: IF Statement

if grade >= 90 :
    letter_grade = "A"
    print("Good job!")
Review: Boolean Expressions

Evaluate to True or False

a > b  # is a greater than b?
a < b  # is a less than b?
a == b  # is a equal to b?
a >= b  # is a greater than OR equal to b?
a <= b  # is a less than OR equal to b?
a != b  # is a not equal to b?
Practice

What else is true?

a > b

a == b

a <= b

a != b
Practice

What else is true?

a > b  # a != b  a >= b  b < a  b <= a

a == b

a <= b

a != b
Practice

What else is true?

a > b

a == b  # a <= b  a >= b  b <= a  b >= a

a <= b

a != b
Practice

What else is true?

a > b

a == b

a <= b  # b >= a

a != b
Practice

What else is true?

a > b

a == b

a <= b

a != b  # b != a
How to define your own functions
Kinds of Functions

• Python Built-in Functions
• Capture multi-statement operation used frequently
  – Sort items, draw a particular graphic
• Modularize your program into parts
Figure 3-1  Using functions to divide and conquer a large task

This program is one long, complex sequence of statements.

```
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
```

In this program the task has been divided into smaller tasks, each of which is performed by a separate function.

```
def function1():
    statement
    statement
    statement

def function2():
    statement
    statement
    statement

def function3():
    statement
    statement
    statement

def function4():
    statement
    statement
    statement
```
Designing a Program with Functions

• Top-Down approach
  – First, break program into functions
  – Then define each function

• Top-Down Pseudo-code

• Top-Down Flow chart

• Hierarchy chart (call graph)
  – Shows who calls who
Top-level Pseudo-code for Baking a Cake

1. Measure all of your ingredients
2. Combine your dry ingredients
3. Combine your wet ingredients
4. Incorporate your wet ingredients into your dry ingredients
5. Bake at 350 degrees for 30 minutes
Hierarchy Chart for Baking a Cake

main()

.measure()  mix_dry()  mix_wet()  incorporate()  bake()
Anatomy of a Function Definition

def main() :
    print("Hello there.")
    print("So long.")
Anatomy of a Function Definition

def main() :
    print("Hello there.")
    print("So long.")
def main() :
  print(“Hello there.”)
  print(“So long.”)
Anatomy of a Function Definition

def main() :
    print(“Hello there.”)
    print(“So long.”)
Anatomy of a Program with a Function

def main() :
    print("Hello there.")
    print("So long.")

main()
Anatomy of a Program with a Function

```python
def main() :
    print(“Hello there.”)
    print(“So long.”)
```

```python
main()
```

Function Definition
Functions are defined before they are used
Anatomy of a Program with a Function

def function_name() :
    statement
    statement

    function_name()
def function_name() :
    statement1
    statement2

function_name()
print(“Done”)
def function_name() :
    statement1
    statement2

function_name()
print("Done")
Simple Function Call Execution Trace

def function_name() :
    statement1
    statement2

function_name()
print("Done")
def function_name() :
    statement1
    statement2

function_name()

print("Done")
def function1() :
    statement1
    statement2

def function2() :
    statement1
    statement2

function1()
function2()
print("Done")
def function1() :
    statement1
    statement2

def function2() :
    statement1
    statement2

function1()
function2()
print(“Done”)
Execution Trace with Two Functions

```python
def function1() :
    statement1
    statement2

def function2() :
    statement1
    statement2

function1()
function2()
print("Done")
```
Execution Trace with Two Functions

```python
def function1() :
    statement1
    statement2

def function2() :
    statement1
    statement2

function1()
function2()
print("Done")
```
Execution Trace with Two Functions

```python
def function1() :
    statement1
    statement2

def function2() :
    statement1
    statement2

function1()
function2()
print("Done")
```
Execution Trace with Two Functions

```python
def function1() :
    statement1
    statement2

def function2() :
    statement1
    statement2

function1()
function2()
print(“Done”)```
Execution Trace with Two Functions

```python
def function1() :
    statement1
    statement2

def function2() :
    statement1
    statement2

def function1()
function2()
print("Done")
```
def function1() :
  statement1
  function2()

def function2() :
  statement1
  statement2

function1()
print("Done")
def function1() :
    statement1
    function2()

def function2() :
    statement1
    statement2

function1()
print(“Done”)
Execution Trace with Nested Function Call

def function1() :
    statement1
    function2()

def function2() :
    statement1
    statement2

function1()
print("Done")
Execution Trace with Nested Function Call

```python
def function1() :
    statement1
    function2()

def function2() :
    statement1
    statement2

function1()
print(“Done”)```
Execution Trace with Nested Function Call

```python
def function1() :
    statement1
    function2()

def function2() :
    statement1
    statement2

def function1() :
    print(“Done”)
```
def function1() :
    statement1
    function2()

def function2() :
    statement1
    statement2

function1()
print(“Done”)
Things to Remember

• Don’t mix up “==” and “=”
• Avoid “==” with floating point numbers
• Implement all cases that require an action
• Indent consistently
• Don’t forget to call your function!
  – It won’t execute by itself
• Test all paths through the program
Write a program that inputs the scores for the Bronco’s and the Seahawks, and prints out the winner or “tied” if the score is tied.