CSC108H Lecture 5

Dan Zingaro

September 19, 2012

Functions: Motivation

- Let's say you have some code that calculates a student's course mark given the marks on labs, assignments, etc.
- What if you want to apply it to three students?
- We could copy-and-paste the same code three times
- What are the disadvantages of doing this?
  - Code length (three times as much code)
  - Bugs: if there's a bug or error, we'll replicate it in multiple places

Functions: Motivation...

- Suppose you are writing a cookbook
- It has 12 cake recipes, and three of them involve a buttercream filling
- Do we have to explain how to make the filling every time?
- No — we can "define" it once, and then "call" it from other recipes (e.g. "see page 18 for the buttercream filling recipe, then come back here to finish making the cake")
- In programs, we similarly use functions to avoid repetition

Function Parameters

```
def f(a, b, c):
  ...
```

- When we `define` a function, we specify that it takes zero or more **parameters**
- e.g. the function above takes three parameters
- When we call a function, we provide a value (an argument) for each parameter; our values are then assigned to the function's parameters
- If there are no parameters, we call the function with empty parentheses

Function Parameters...

```
def f(a, b, c):
  ...

x = 5
y = 8
z = 10
f(x, y, z)
```

- Parameter passing is just like an assignment statement
- Here, when `f` starts running, `a` gets the value of `x`, `b` gets the value of `y`, and `c` gets the value of `z`
- Changes to `a`, `b`, or `c` do not change `x`, `y`, or `z`!

ConcepTest

```
def first(a):
  a = 8
  return a

a = 20
first(a)
print(a)
```

What is the output of this code?

- A. `8`
- B. 20
- C. Error, because a cannot be assigned in two places
- D. `None`
- E. `0`

ConcepTest

```
def first(a):
  a = 8
  return a

a = 20
a = first(a)
print(a)
```

What is the output of this code?

- A. 8
- B. 20

Return vs. Print

- We have said that `return` terminates the function and returns a value to its caller
- If there is no `return`, the function returns the value `None`
- We have also introduced the `print` function, which outputs its arguments to the screen
  - `print` takes one or more arguments, and prints them out with a space separating the values
  - `print` returns `None`

ConcepTest

```
def f1():
  return 5

def f2():
  print(5)

def f3():
  return print(5)
```

Which of the following assigns 5 to x?

- A. `x = f1()`
- B. `x = f2()`
- C. `x = f3()`
- D. Two of the above
- E. All of the above

Choosing Parameters

- How do you decide on the parameters of a function?
- Ask the question "what does the function need to know to do its job"
- Style rule: Everything it has to "know" should be provided as a parameter
- Remember the LGB rule for variable lookup; functions can access
  - Local variables (including parameters)
  - Global variables (often bad style)
  - Built-in variables (e.g. `abs`)

ConcepTest

```
def a(num):
  num = 4
  return 2

def b(val):
  num = 8
  print(a(1))

b(2)
```

What is the output of this code?

- A. 1
- B. 2
- C. 4
- D. 8
- E. Error because of an undefined variable

ConcepTest

```
def a(num):
  num = 4
  return 2

def b(val):
  val = 8
  print (a(1))

b(2)
```

What is the output of this code?

- A. 1
- B. 2
- C. 4
- D. 8
- E. Error because of an undefined variable