

Lecture 4: Defining Functions (Ch. 3.4-3.11)

CS 1110
Introduction to Computing Using Python



[E. Andersen, A. Bracy, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White]

Things to Do Before Next Class

Readings:

- Sections 8.1, 8.2, 8.4, 8.5, first paragraph of 8.9

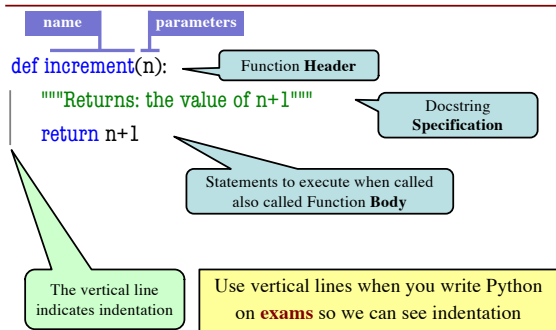
Labs:

- Go to Lab! (Lab 2 is this week)
- Get Credit for Lab 1:
 - can be checked off during Tuesday's consulting hours 4:30-9:30 in the ACCEL lab
 - cannot be checked off after 3:45pm Wednesday
 - check online if you received credit:

<http://www.cs.cornell.edu/courses/cs1110/2018sp/labs/index.php>

2

Anatomy of a Function Definition



8

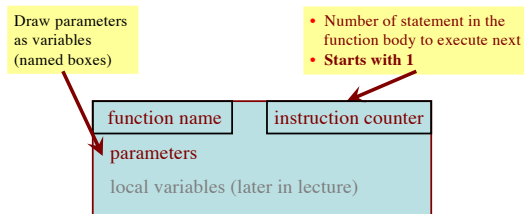
The return Statement

- Passes a value from the function to the caller
- Format:** `return <expression>`
- Any statements after **return** are ignored
- Optional (if absent, no value will be sent back)

9

Understanding How Functions Work

- We will draw pictures to show what is in memory
- Function Frame:** Representation of function call

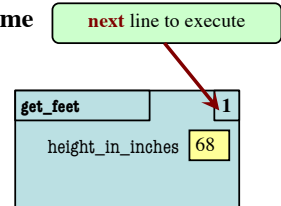


Note: slightly different than in the book (3.9) Please do it **this** way. 15

Example: `get_feet(68)`

PHASE 1: Set up call frame

- Draw a frame for the call
- Assign the argument value to the parameter (in frame)
- Indicate next line to execute



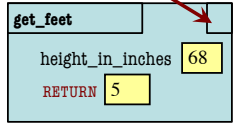
```
def get_feet(height_in_inches):  
1 | return height_in_inches // INCHES_PER FOOT
```

17

Example: get_feet(68)

PHASE 2: Execute function body

The return terminates;
no next line to execute



```
def get_feet(height_in_inches):
1 | return height_in_inches // INCHES_PER FOOT
```

19

Example: get_feet(68)

PHASE 3: Erase call frame

But don't actually
erase on an exam

ERASE WHOLE FRAME

```
def get_feet(height_in_inches):
1 | return height_in_inches // INCHES_PER FOOT
```

21

Local Variables (4)

- Call frames can make “local” variables

```
>>> import room_numbers
>>> room_numbers.lab_rooms()
```

```
def lab_rooms():
1 | red_room = 235
2 | orange_room = 236
```

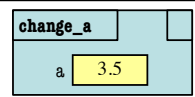
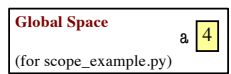
Variables are gone! This
function is useless.

ERASE WHOLE FRAME

25

Function Access to Global Space

- All function definitions are in some module
- Call can access global space for **that module**
 - math.cos: global for math
 - height.get_feet uses global for height
- But **cannot** change values
 - Assignment to a global makes a new local variable!

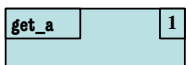
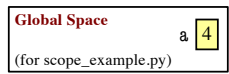


```
# scope_example.py
"""Show how globals work"""
a = 4 # global space
def change_a():
    a = 3.5 # local variable
    return a
```

35

Function Access to Global Space

- All function definitions are in some module
- Call can access global space for **that module**
 - math.cos: global for math
 - height.get_feet uses global for height
- But **cannot** change values
 - Assignment to a global makes a new local variable!



```
# scope_example.py
"""Show how globals work"""
a = 4 # global space
def get_a():
    return a # returns global
```

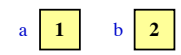
36

Call Frames and Global Variables

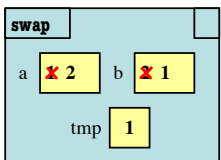
```
def swap(a,b):
    """Swap global a & b"""
1 | tmp = a
2 | a = b
3 | b = tmp
```

```
>>> a = 1
>>> b = 2
>>> swap(a,b)
```

Global Variables



Call Frame



40