

Lecture 15:

FUNCTIONS PT. 2

CSC111: Introduction to CS through Programming

R. Jordan Crouser

Assistant Professor of Computer Science

Smith College

Announcements 1/2

Terry-Ann Craigie

Friday, October 12 • 5:30 p.m.
Seelye Hall, Room 106



Intersections of Race, Data Science and Public Policy

Terry-Ann Craigie, Ph.D., associate professor of economics at Connecticut College, will introduce key public policy issues, describe the limitations of current research and illustrate how big data can help facilitate innovative solutions for improving racial equity.

Her research explores economics of the family, crime and labor. She is currently focusing on equity issues facing the U.S. correctional population, the majority of whom are young racial-ethnic minority males.

Sponsored by the Statistical and Data Sciences Program and the Smith College Lecture Committee.

Free, open to the public and wheelchair accessible. For disability access information or accommodations requests, please call 413-585-2407. To request a sign language interpreter, call 413-585-2071 (voice or TTY) or send email to csd@smith.edu, at least 10 days before the event.



Announcements 2/2

- A3 feed back is out, A4 will come out over the weekend
- Overall notes:
 - You code is getting **much** cleaner and more organized
 - Very few syntax errors (but more frequent **logical errors**)
 - **Important:** test your code against several different inputs (especially “edge cases”)

Lab 5 debrief

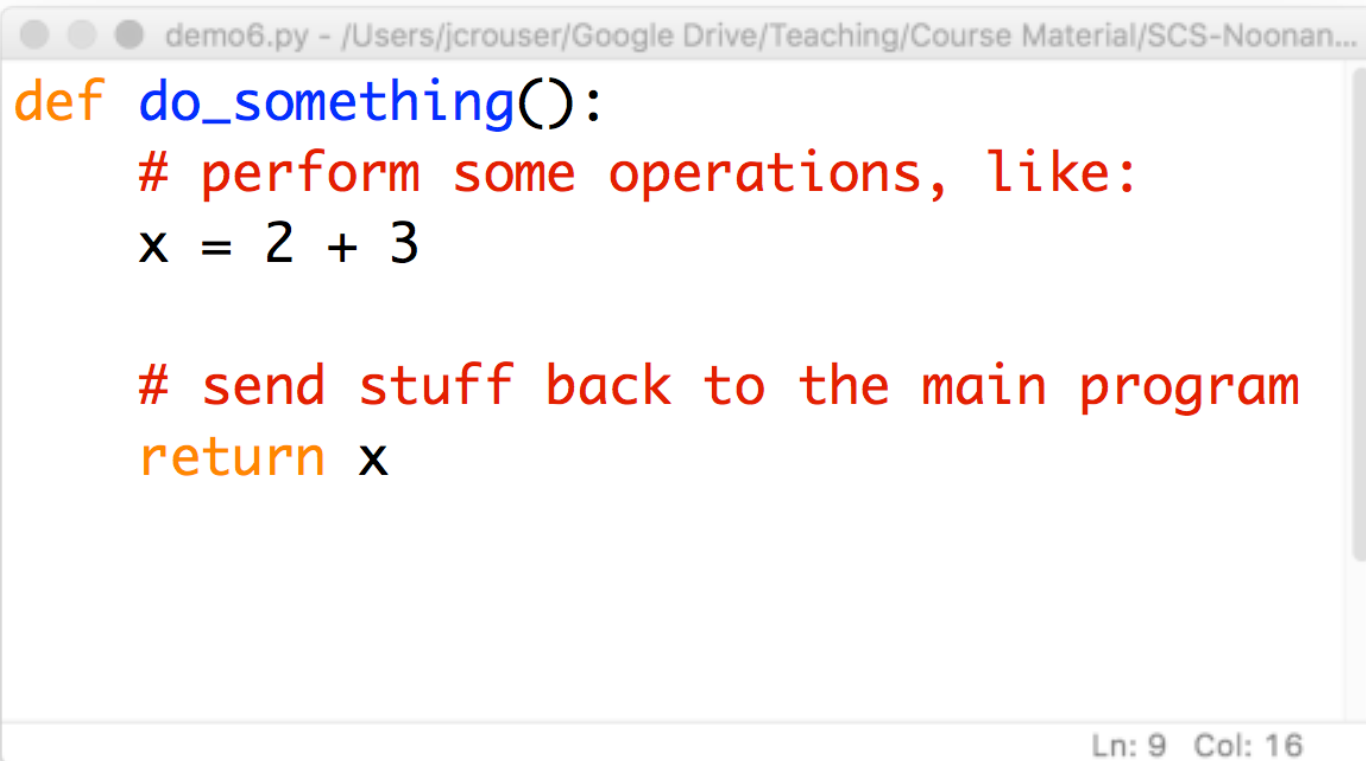
How did it go?



Outline

- ✓ Monday: FALL BREAK
- ✓ Wednesday: Functions
 - ✓ basic components
 - ✓ definition vs. call
 - ✓ an analogy
 - ✓ parameters
 - ✓ returning values
- ✓ LAB: MadLibs (debrief)
- **Friday: ~~Catching Exceptions~~ More on Functions**

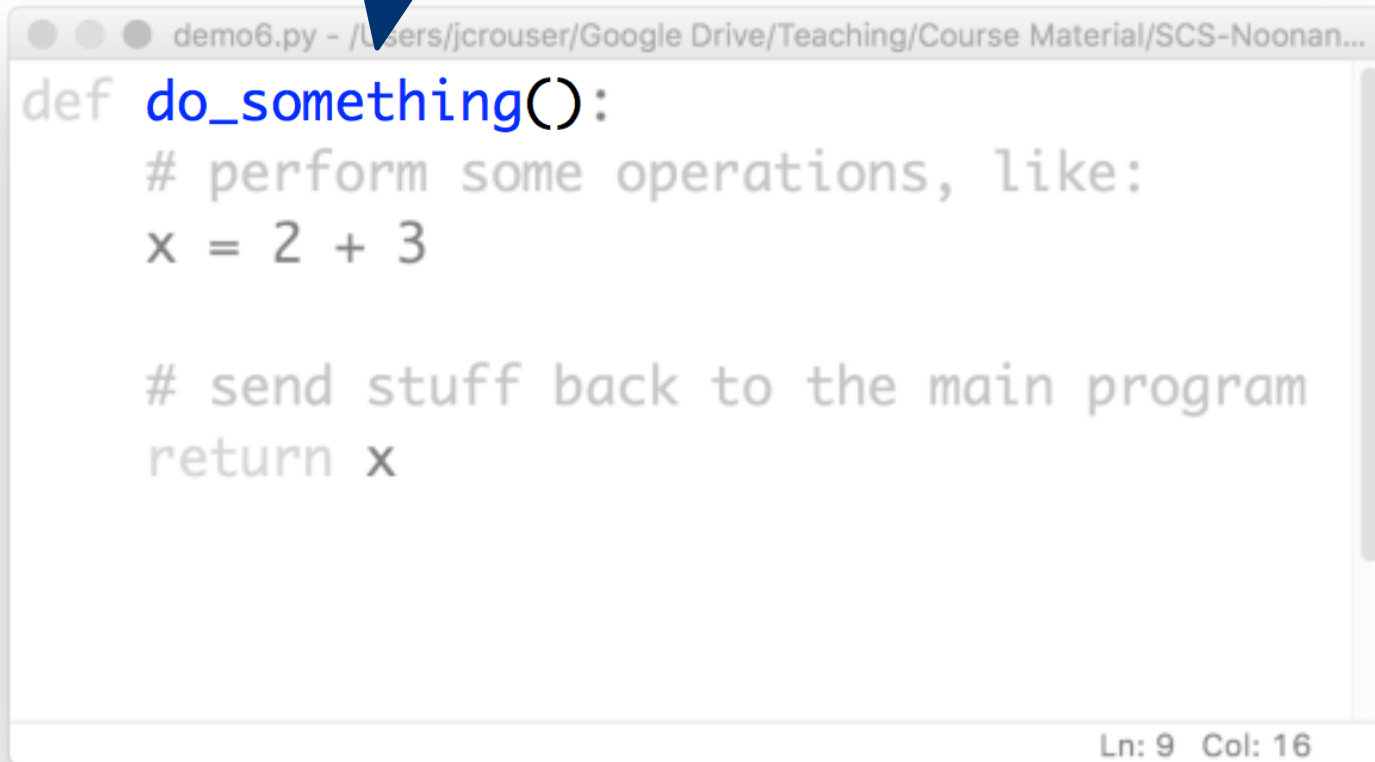
To recap: a “function definition”

A screenshot of a code editor window. The title bar at the top reads "demo6.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan...". The editor contains Python code for a function definition. The code is color-coded: "def" is orange, "do_something()" is blue, and the rest is black or red. The code defines a function named "do_something" that takes no arguments, performs an addition, and returns the result. The status bar at the bottom right shows "Ln: 9 Col: 16".

```
def do_something():  
    # perform some operations, like:  
    x = 2 + 3  
  
    # send stuff back to the main program  
    return x
```

To recap: a “function definition”

a name

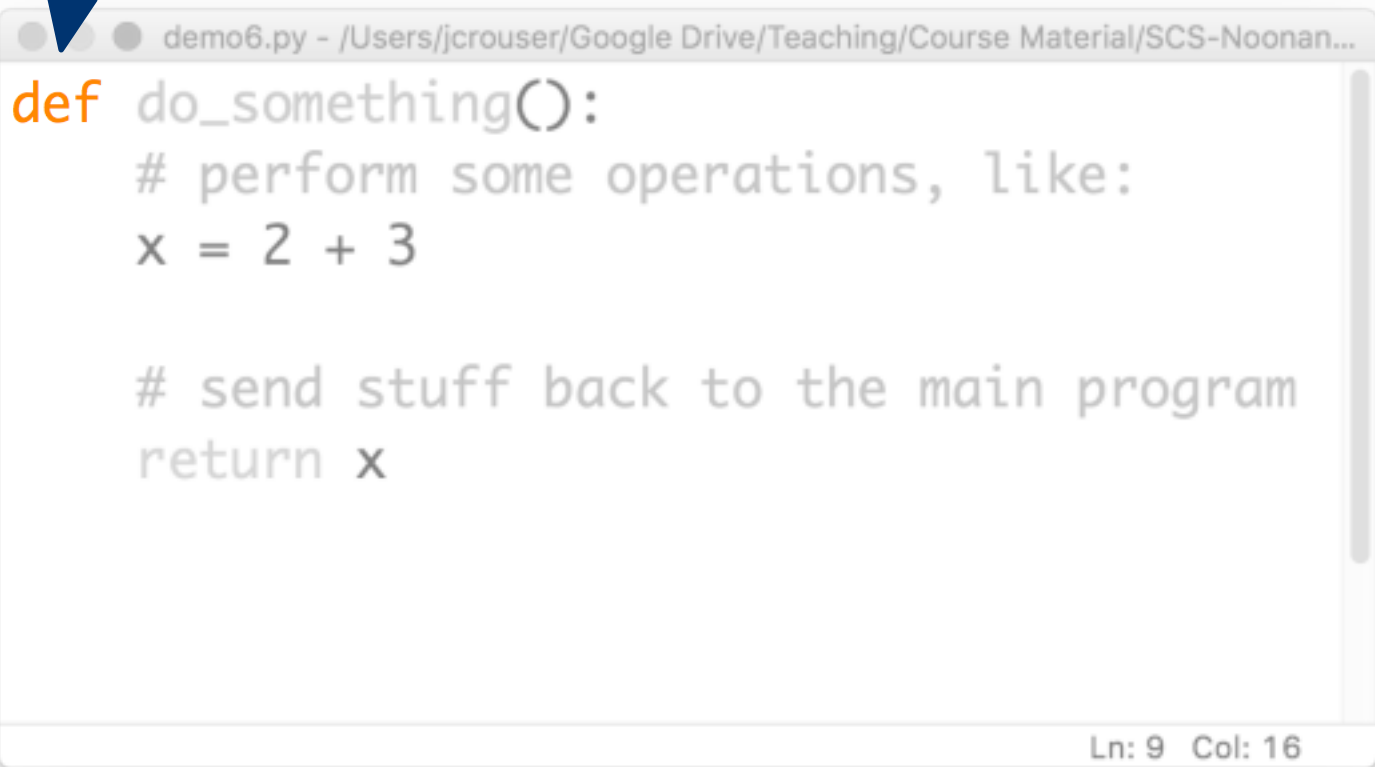


```
demo6.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan...  
def do_something():  
    # perform some operations, like:  
    x = 2 + 3  
  
    # send stuff back to the main program  
    return x  
  
Ln: 9 Col: 16
```

Convention: use `_underscores_` or `camelCase`

To recap: a “function definition”

defined using
the **def** keyword

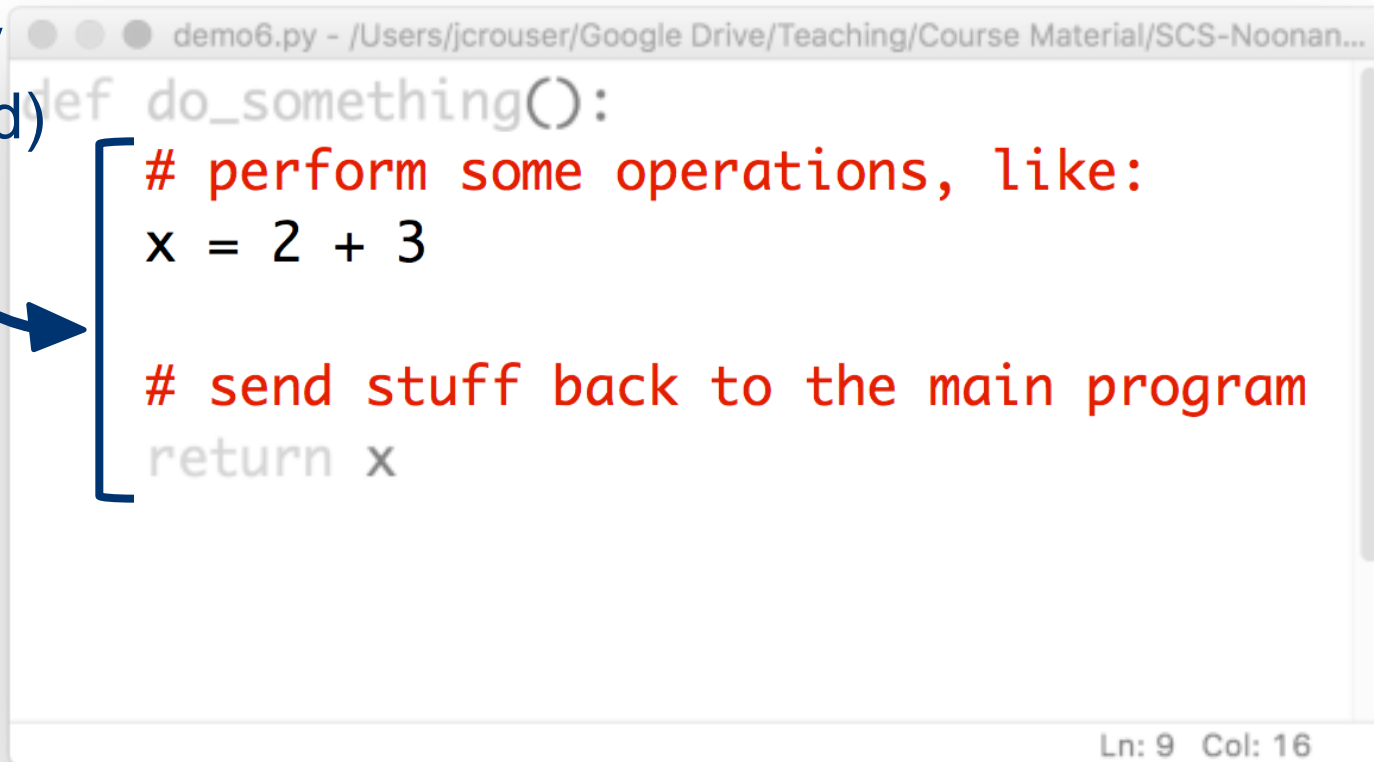


```
def do_something():  
    # perform some operations, like:  
    x = 2 + 3  
  
    # send stuff back to the main program  
    return x
```

Ln: 9 Col: 16

To recap: a “function definition”

a **body**
(indented)



The image shows a screenshot of a code editor window titled "demo6.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan...". The code defines a function named "do_something()". The function's body is indented and highlighted with a blue bracket. A blue arrow points from the text "a body (indented)" to the bracketed body of the function.

```
def do_something():  
    # perform some operations, like:  
    x = 2 + 3  
  
    # send stuff back to the main program  
    return x
```

Ln: 9 Col: 16

To recap: a “function definition”

```
demo6.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan...  
def do_something():  
    # perform some operations, like:  
    x = 2 + 3  
  
    # send stuff back to the main program  
    return x
```

← a **return** (optional)

Ln: 9 Col: 16

To recap: function calls



```
demo6.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan...  
def do_something():  
    # perform some operations, like:  
    x = 2 + 3  
  
    # send stuff back to the main program  
    return x  
  
y = do_something()
```

Ln: 9 Col: 16

a function **call**

To recap: function calls



```
demo6.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan...
def do_something():
    # perform some operations, like:
    x = 2 + 3

    # send stuff back to the main program
    return x

```

y =

Ln: 9 Col: 16

Definitions vs. calls

Function Definition

- Step-by-step **instructions** for how to perform a given set of operations
- Analogy: a **recipe**
- Think of the function's name as **shorthand**
(i.e. “okay, when I say `do_something()`, here's what I want you to do”)

Function Call

- An **actual request** to perform the operations
- **Control** is turned over to the “minion” (temporarily)
- Once complete, we go back to the **exact place** in the program where the call was issued

Discussion

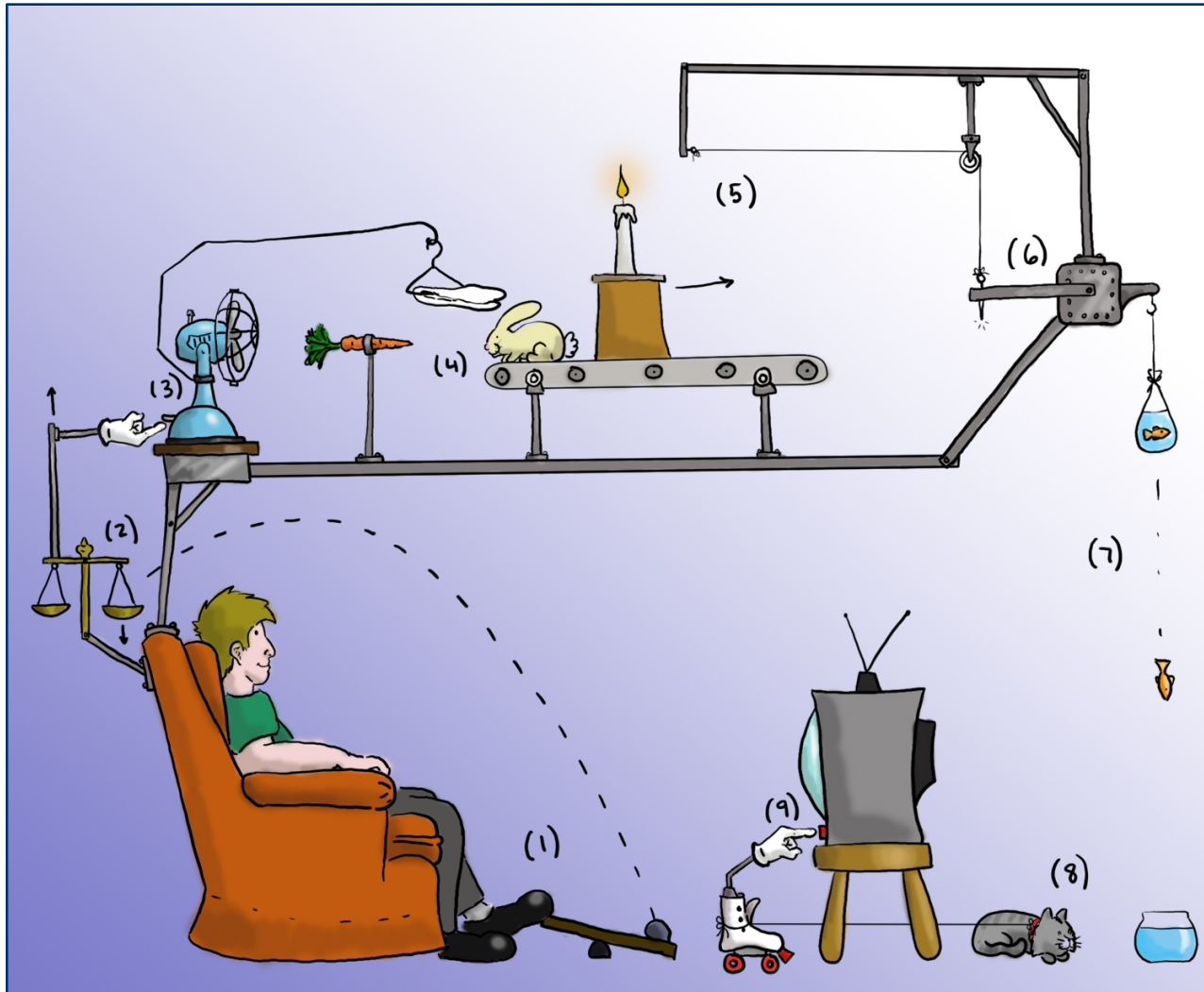
What does it mean for a function
to **return** a value?

And what's the difference
between **return** and **print(...)**?



DEMO
TIME

Demo: human Rube Goldberg machine




```
def addOne ( x ) :
```

- Take the value of x, and add 1 to it
- **return** the modified value

```
def doubleIt(x):
```

- Take the value of x, and double it (i.e. multiply by 2)
- **return** the modified value

```
def print(x):
```

- Look at the value of x
- Write it on the board in big letters
- **return** nothing

```
def woohoo ( ) :
```

- Say out loud a single, loud “WOO HOO!”
- **return** nothing

Discussion

Lingering questions?



Remainder of class



Coming up next

- *A5: Encrypt/Decrypt* is due **Sunday 11:55pm**
- **Mon 10/15:** Lists and Dictionaries
- **Weds 10/17:** Working with Files
- **Lab:** TBD
- **Fri 10/19:** Life Skill #4 - Code Reuse