

Intro to Coding with Python— More Functions

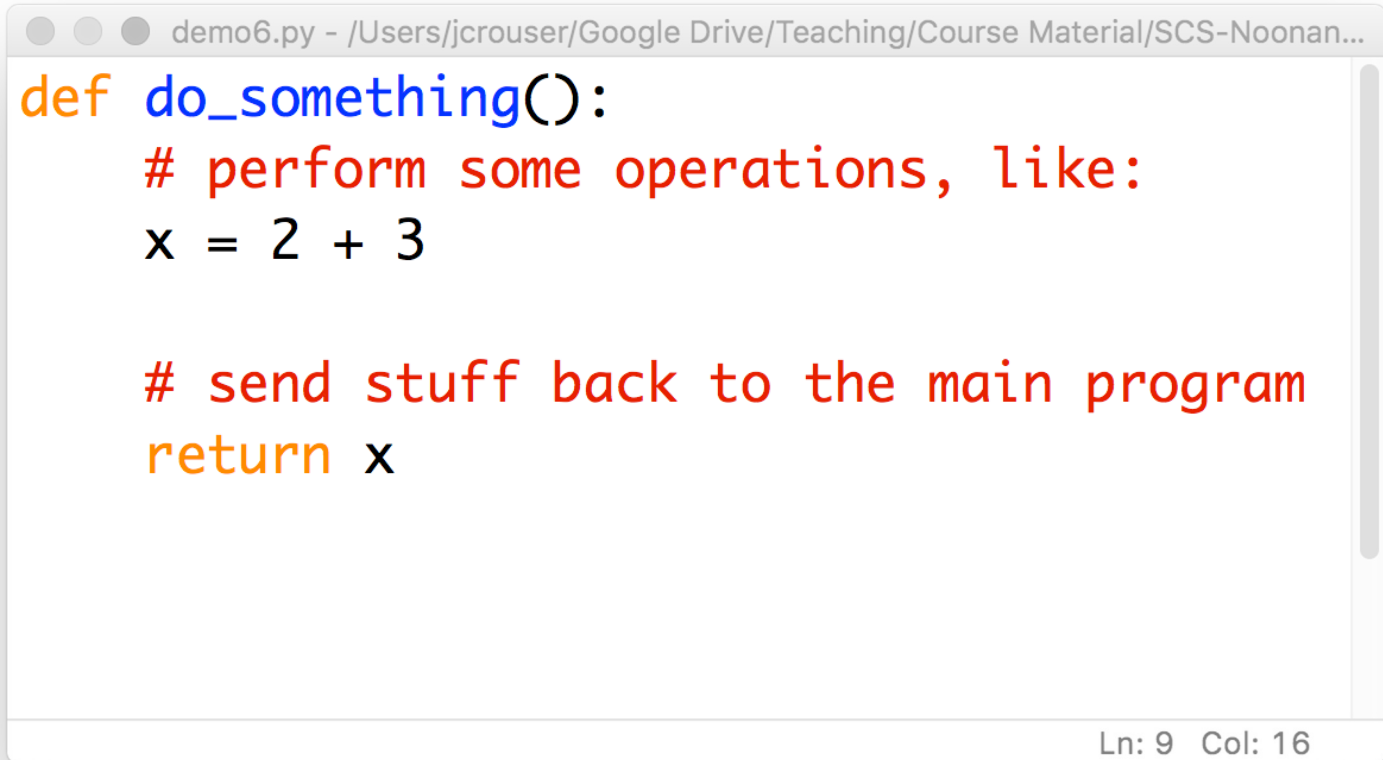
Dr. Ab Mosca (they/them)

Slides based off slides courtesy of Jordan Crouser (<https://jcrouser.github.io/>)

Plan for Today

- Recap defining and calling functions
- String functions together

To recap: a
“function
definition”




```
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 name



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

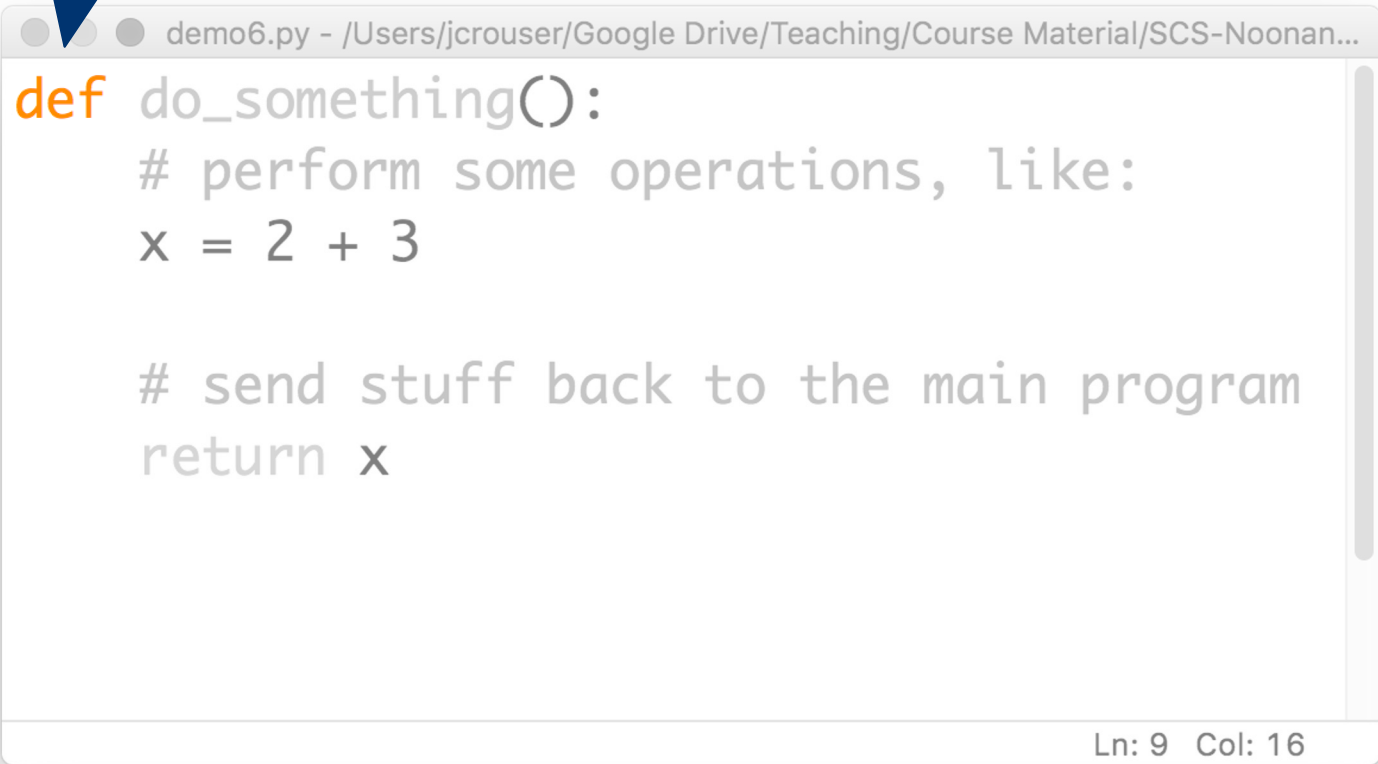
    # send stuff back to the main program
    return x
```

Ln: 9 Col: 16

Convention: use `_underscores_` or `camelCase`

To recap: a
“function
definition”

defined using
the **def** keyword

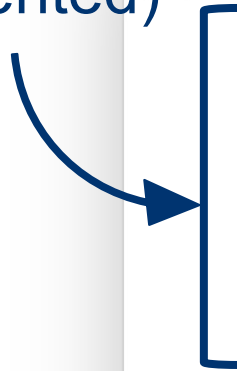


```
demo6.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan...  
def do_something():  
    # perform some operations, like:  
    x = 2 + 3  
  
    # send stuff back to the main program  
    return x
```

Ln: 9 Col: 16

To recap: a
“function
definition”

a **body**
(indented)



```
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
```

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() ← a function call
```

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 =
```

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

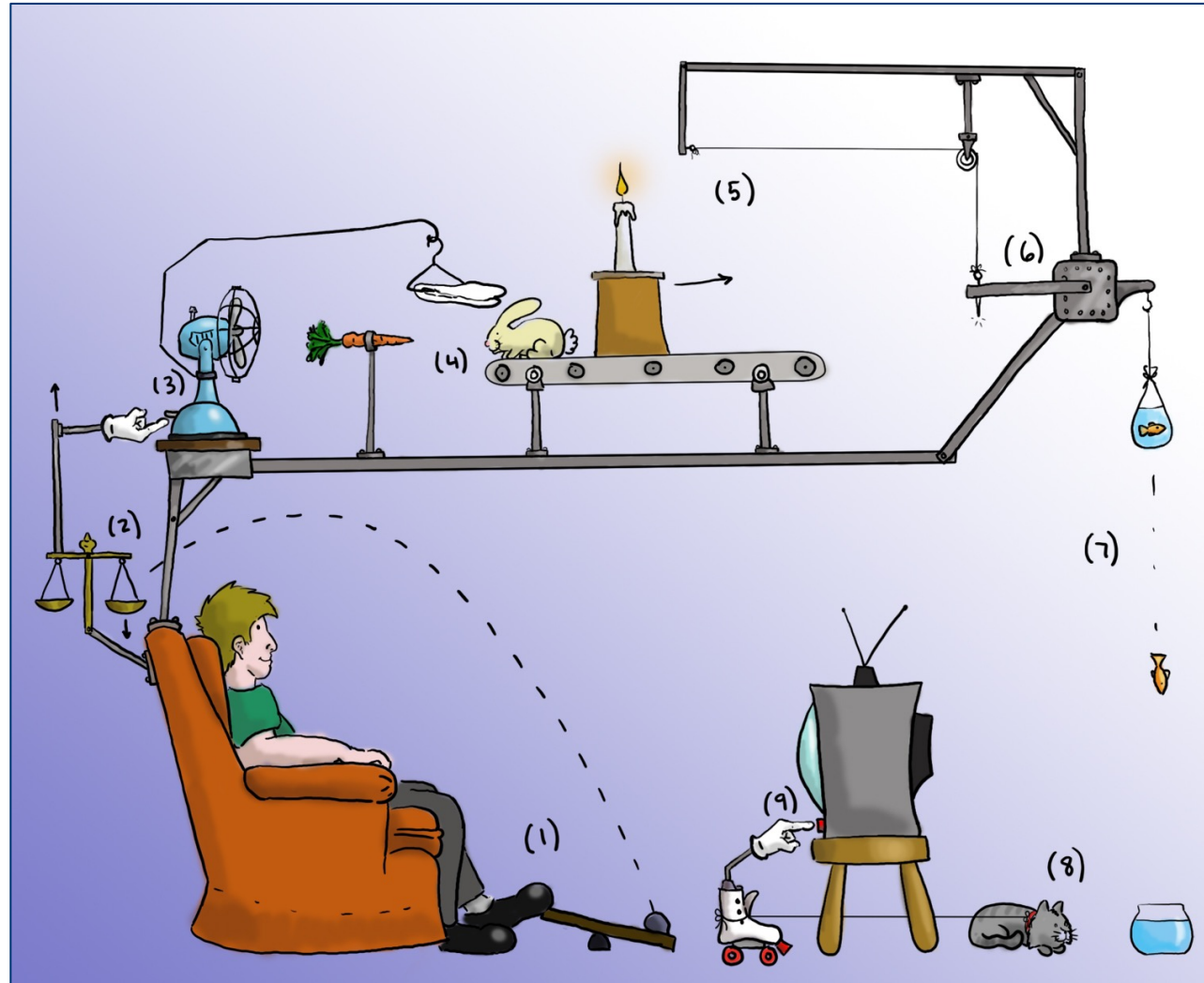
Discussion

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

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

DEMO
TIME

Demo: 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  
printWith  
Stars(x) :
```

- Look at the value of x
- Print it out as *'s (i.e. if x = 3, print ***)
- **return** nothing


```
def  
woohoo () :
```

- Print "WOO HOO!"
- Call
 `printWithStars (addOne (doubleIt (2)))`
- **return** nothing

Discussion

Lingering questions?



**KEEP
CALM
AND
MODULARIZE
YOUR CODE**