Lecture 14:

FUNCTIONS

CSC111: Introduction to CS through Programming

R. Jordan Crouser

Assistant Professor of Computer Science

Smith College

Announcements

Fall Tech Trek in Boston
Friday, October 12, 8:00am-6:00pm

Now open to all class years.
Space is limited. **Sign up in Handshake by Tuesday 10/9 at 5:00pm**

The day is an opportunity for students to explore careers in technology and meet with professionals at their workplace. Hear from the representatives about their career paths and advice on getting started in the industry.

Bus transportation to and from the program will be provided. Lunch will be served.

HubSpot - A developer and marketer of software products for inbound marketing and sales

Audible - A seller and producer of spoken audio entertainment, information, and educational programming on the Internet.

DraftKings - A Boston daily fantasy sports contest provider. The company allows users to enter daily and weekly fantasy sports-related contests and win money based on individual player and team.

Outline

- ✓Monday: FALL BREAK
- Wednesday: Functions
  - basic components
  - definition vs. call
  - an analogy
  - parameters
  - returning values
- LAB: MadLibs
- Friday: Catching Exceptions

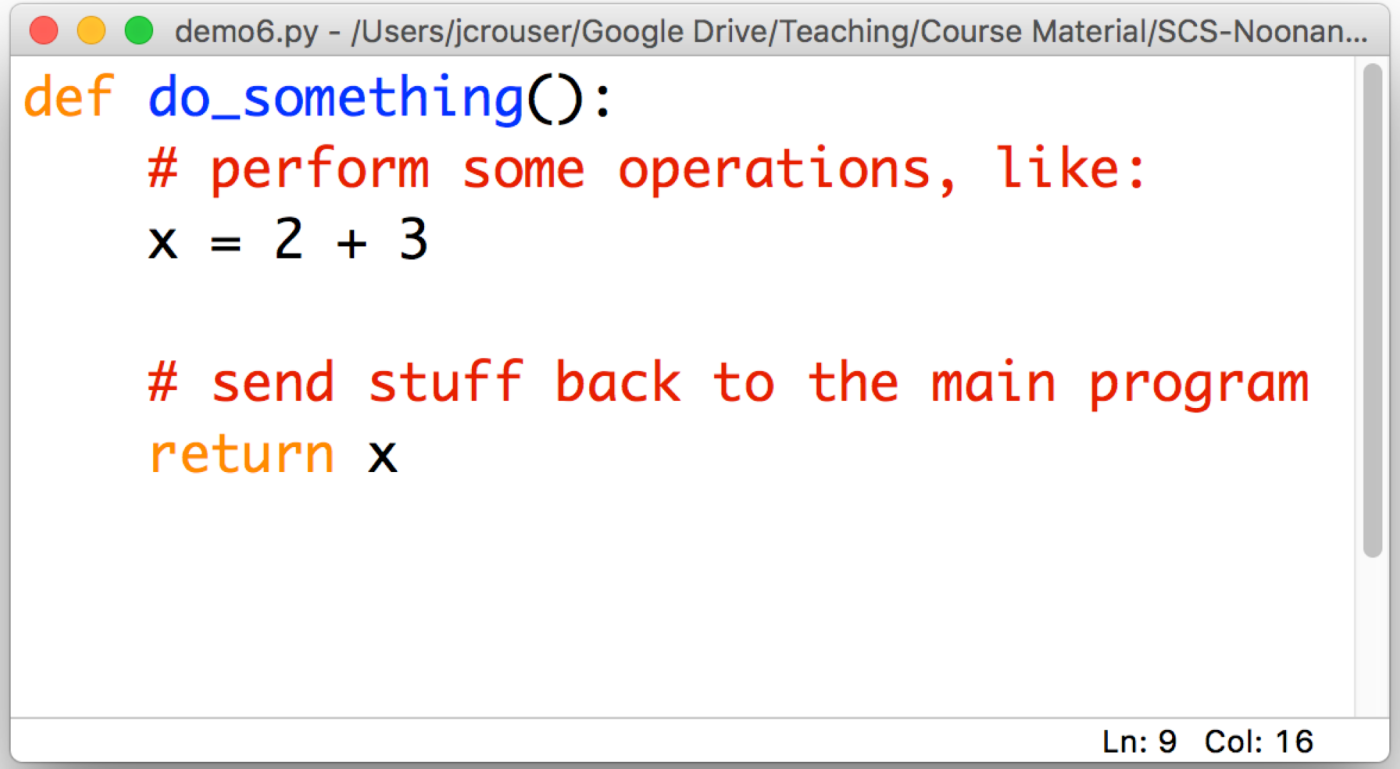
Functions

- **Recall**: a **function** is a procedure / routine that takes in some input and does something with it (just like in math)

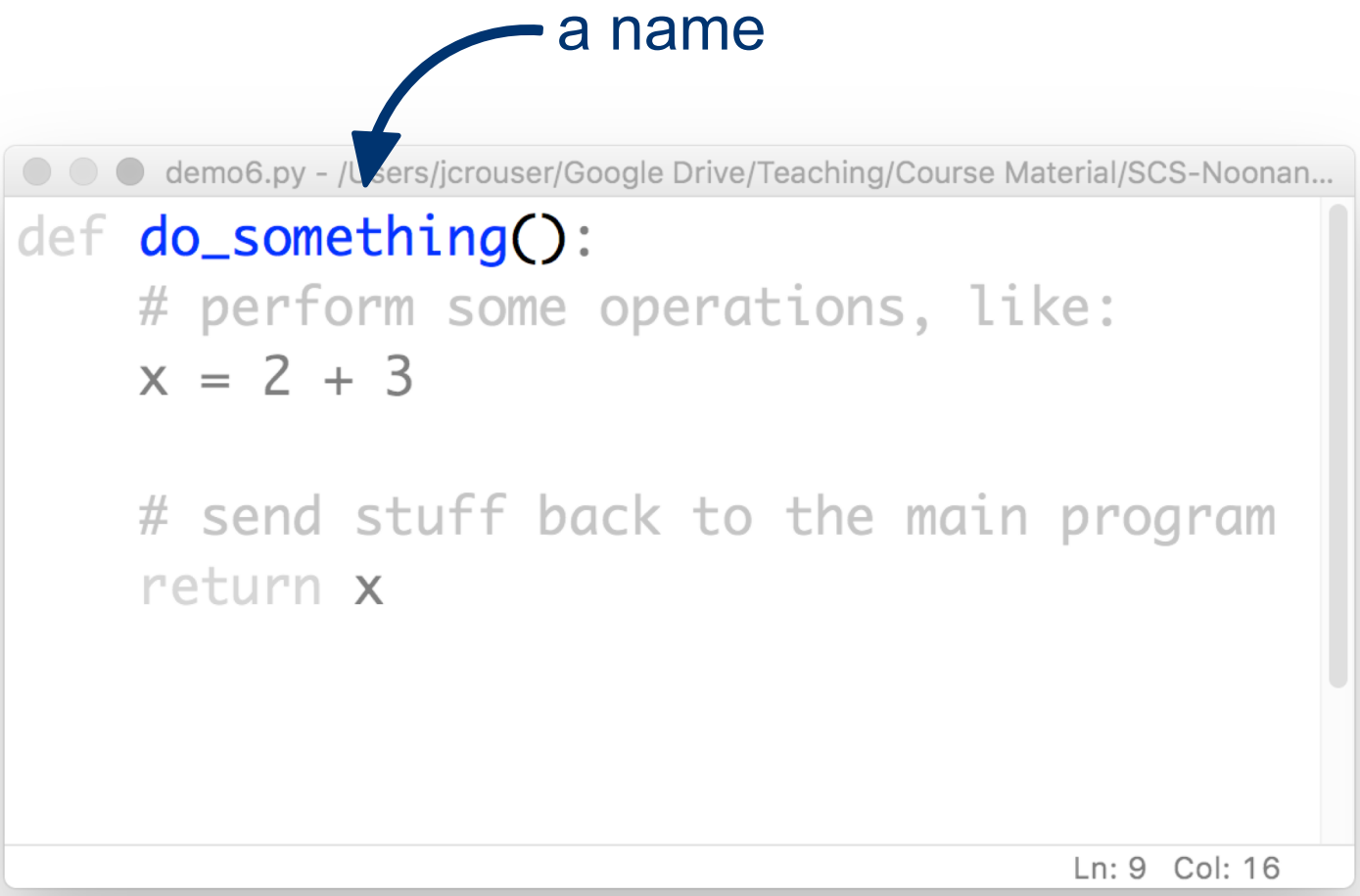
- We've seen lots of built-in functions:
  - `print(…)`
  - `input(…)`
  - `eval(…)`
  - `round(…)`

- Perhaps unsurprisingly, Python lets us write custom functions as well (like yesterday's `main()` function)

Basic components of a function



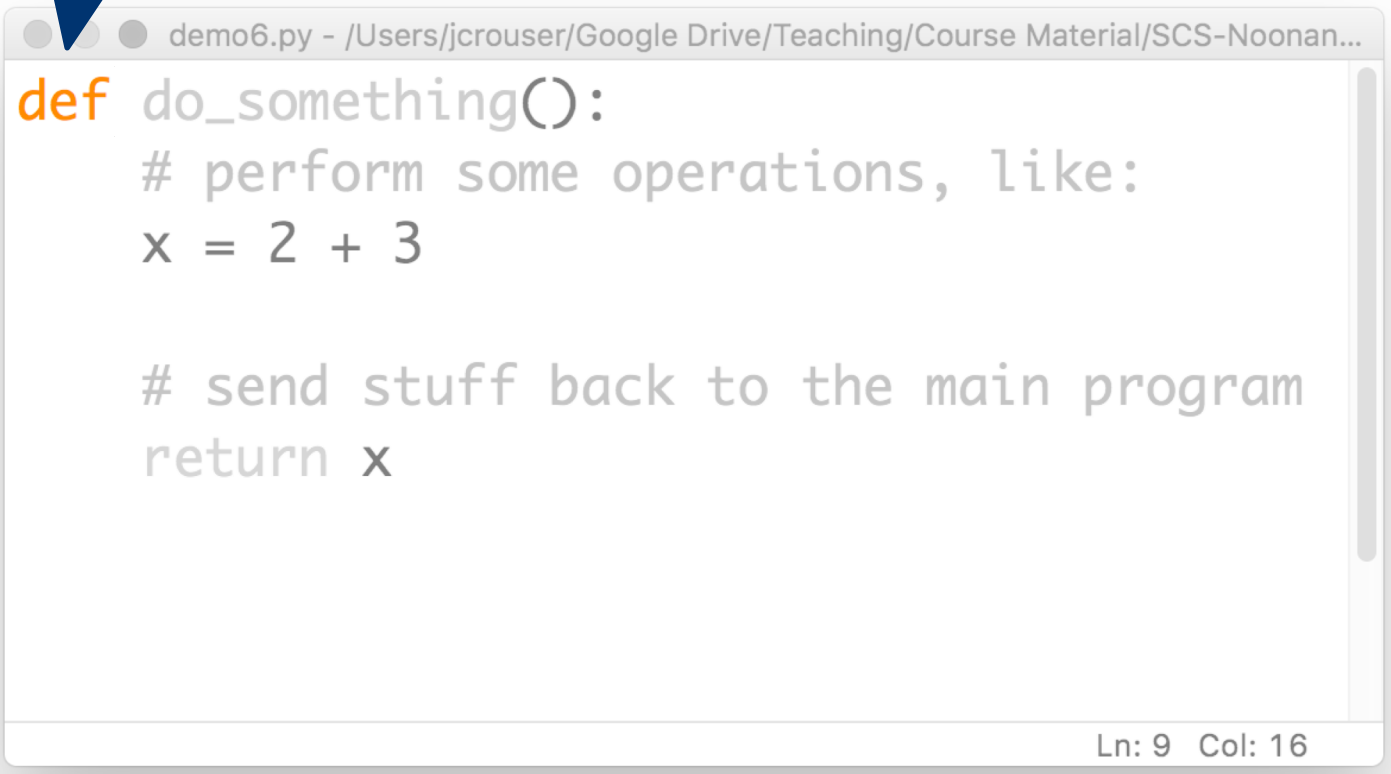
Basic components of a function



Convention: use **`_underscores_`** or **`camelCase`**

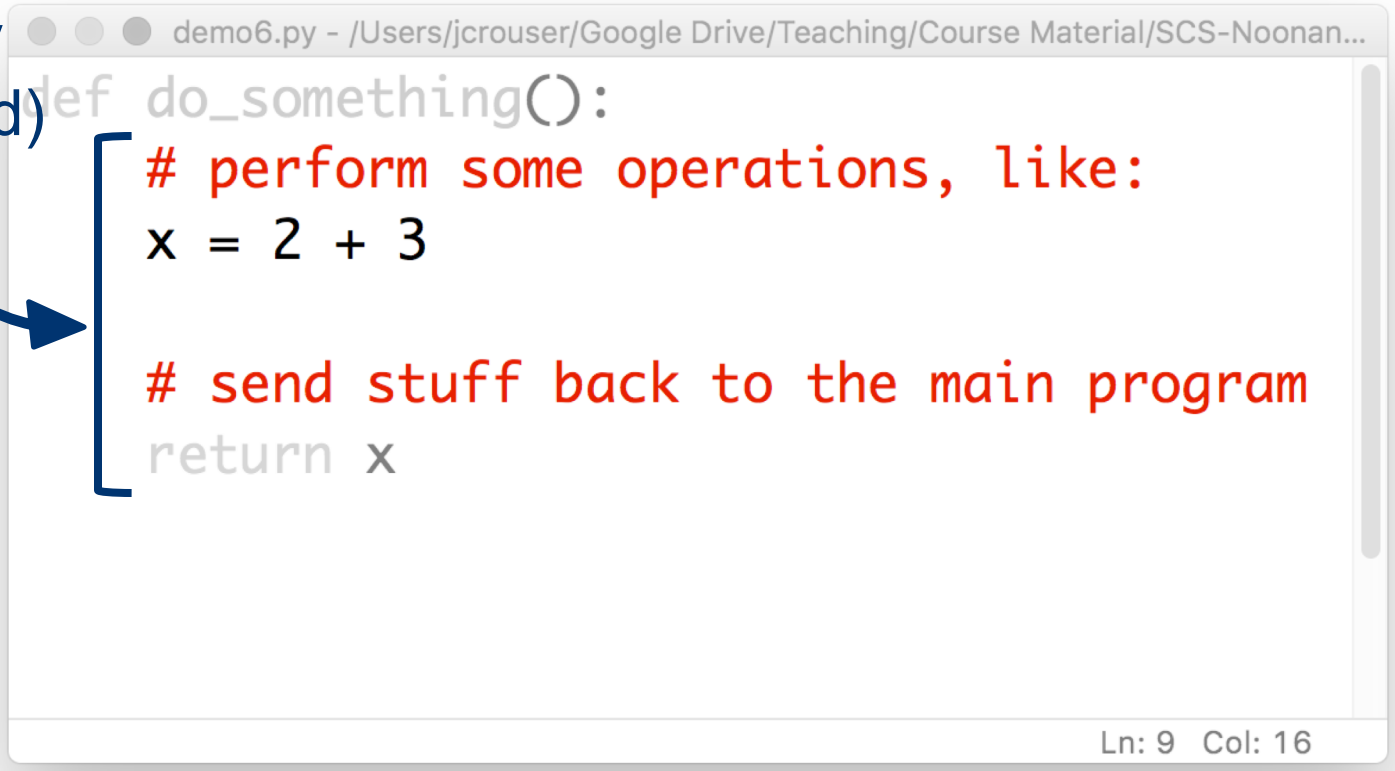
Basic components of a function

which is defined
using the **`def`** keyword

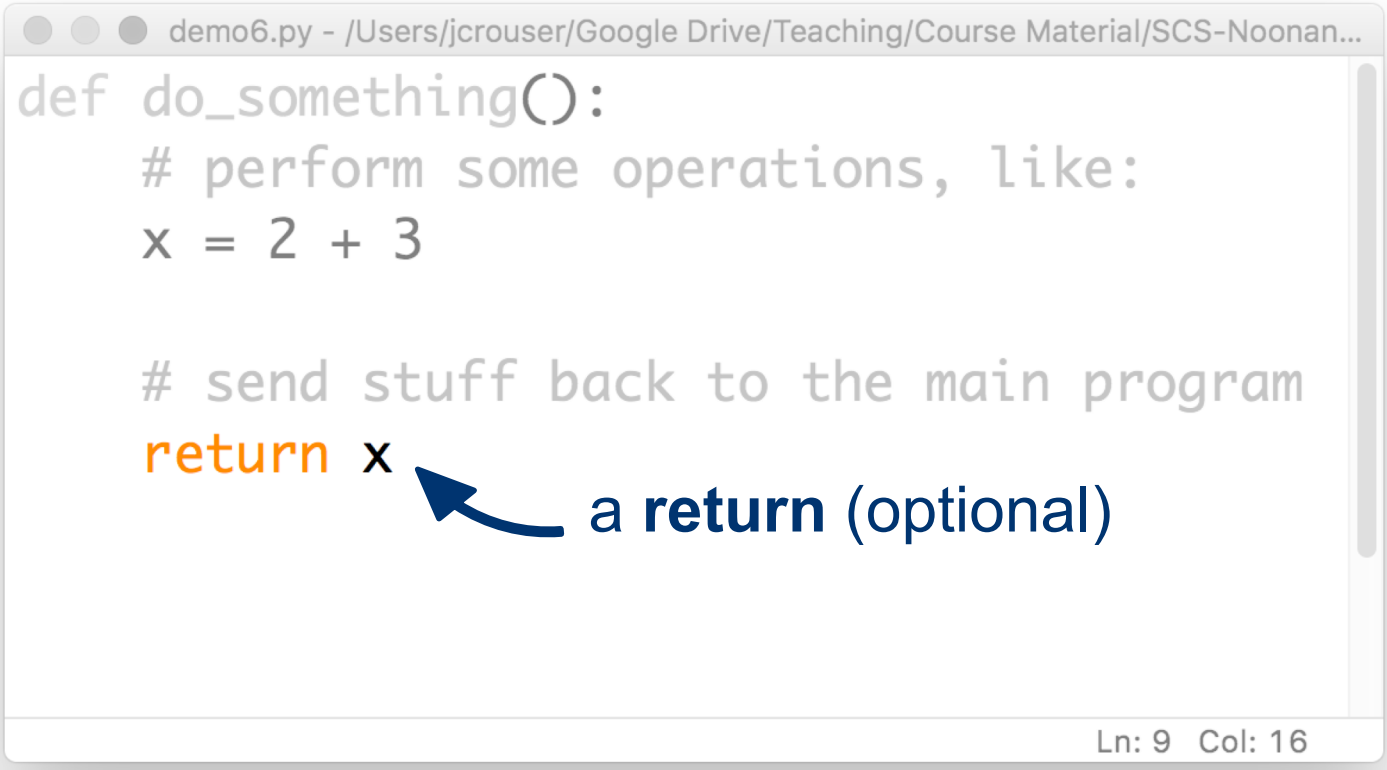


Basic components of a function

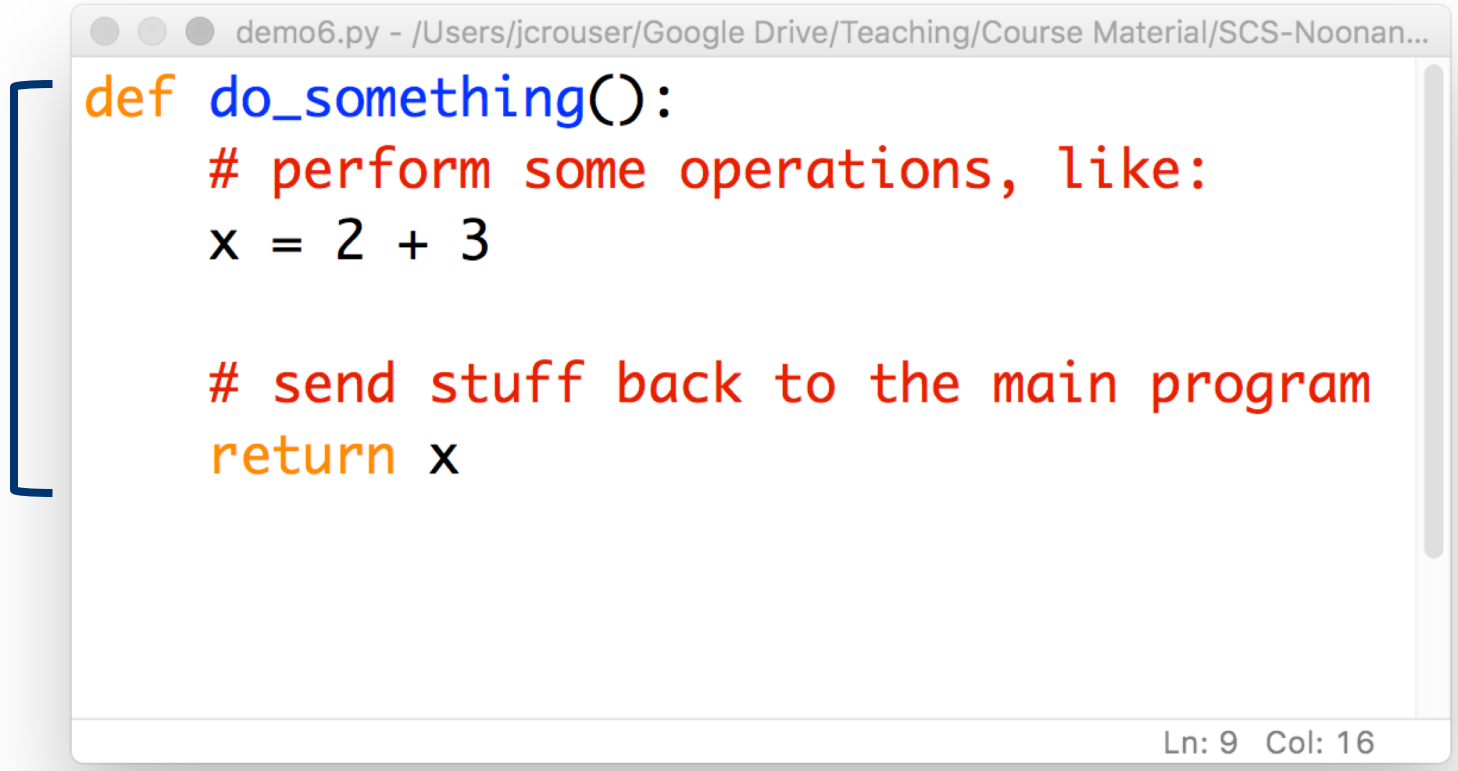
a **body** (indented)



Basic components of a function



A “function definition”



Discussion

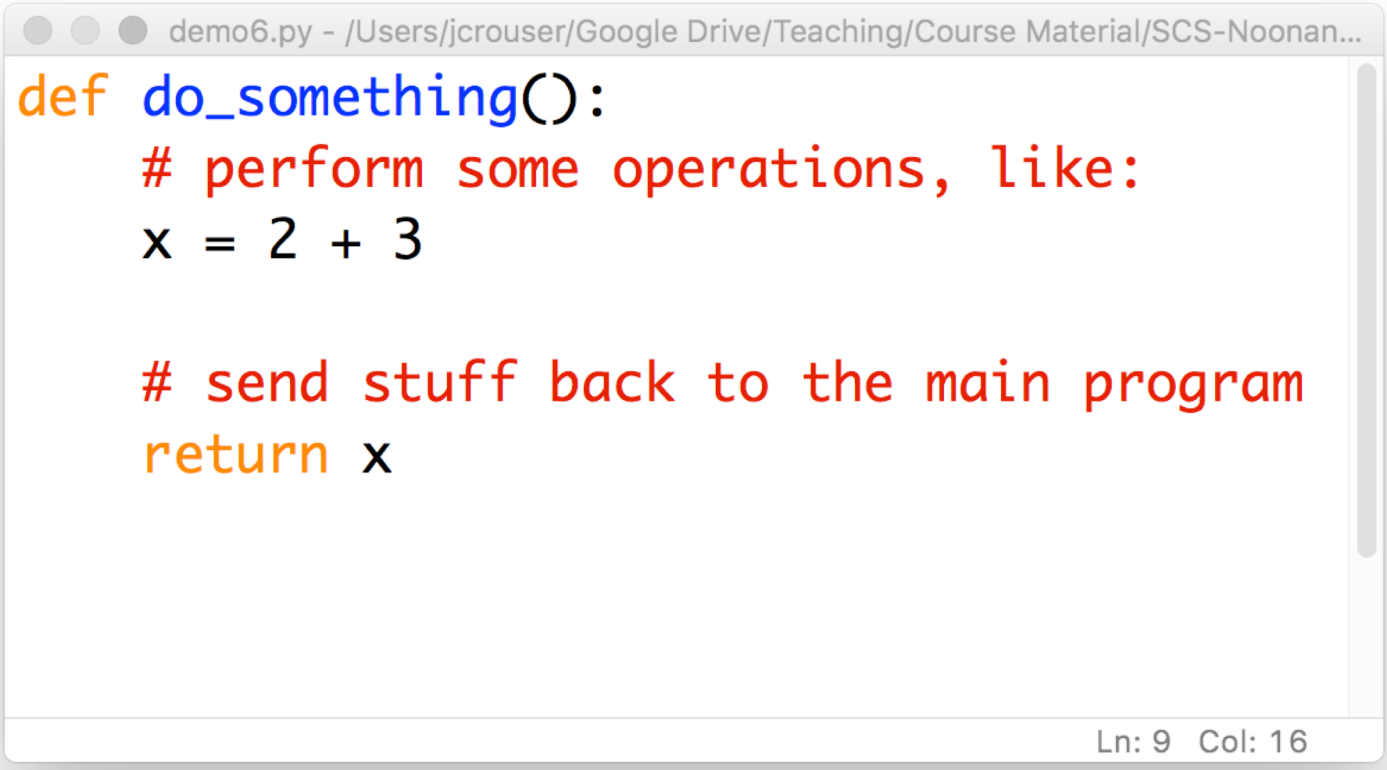
What happens if we **run** this program?

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
```

Col: 16

A “function definition” is a **description**



(but not a **directive**)

Function calls: “hey, Python! do this”

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

Function calls: “hey, Python! do this”

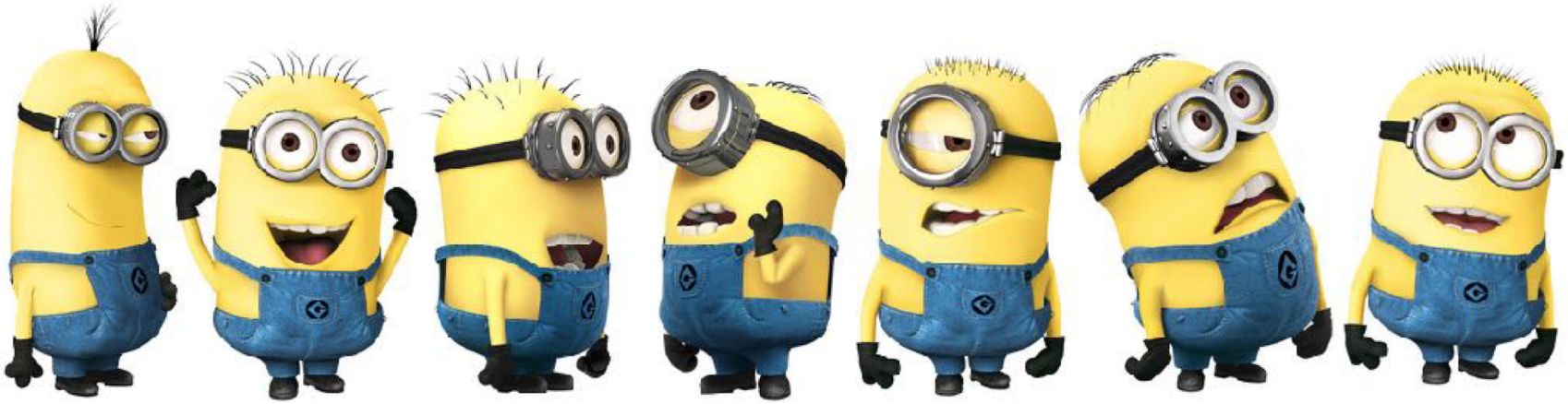
demo6.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan...

```
def do_something():
    # perform some operations, like:
    x = 2 + 3

    # send stuff back to the main program
    return x

y =
```

5

Ln: 9 Col: 16

An analogy



`functions` are your **MINIONS**

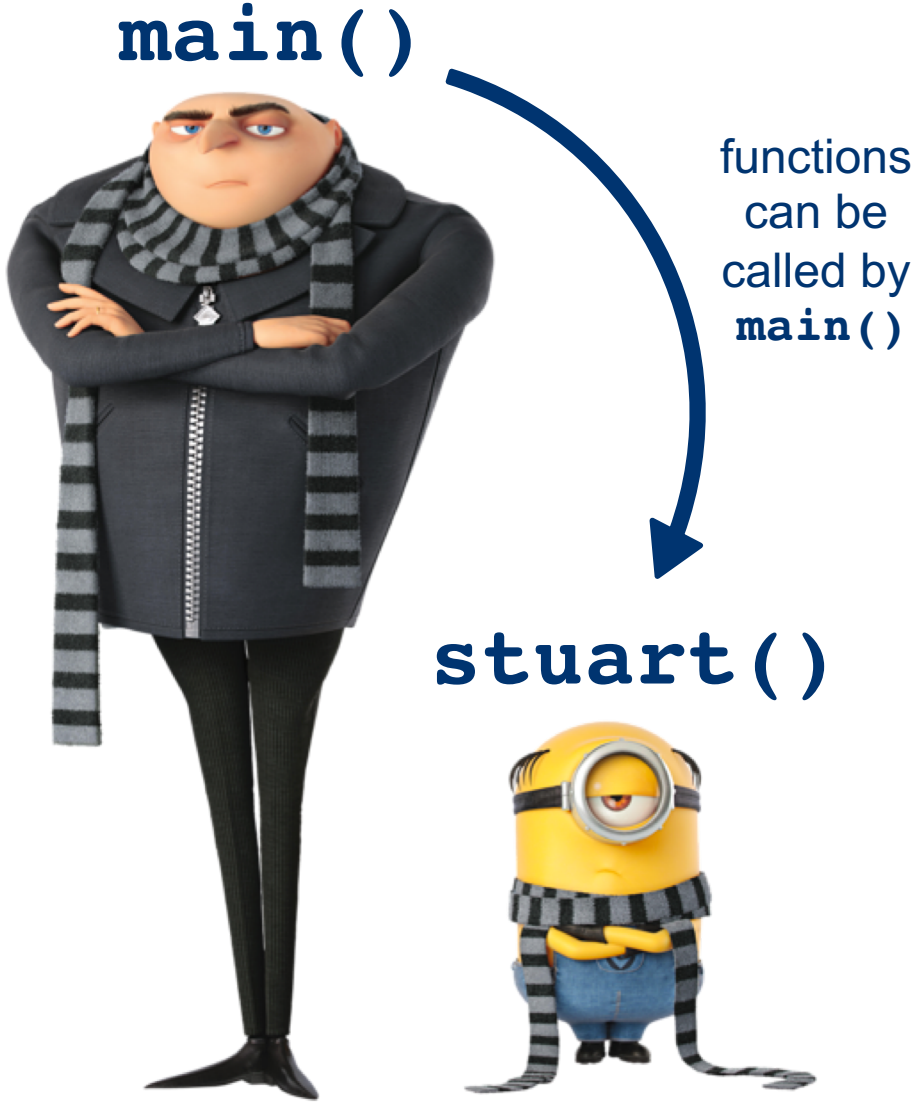
An analogy

`functions` have **NAMES**

An analogy

they only work when you **CALL** them

An analogy



An analogy

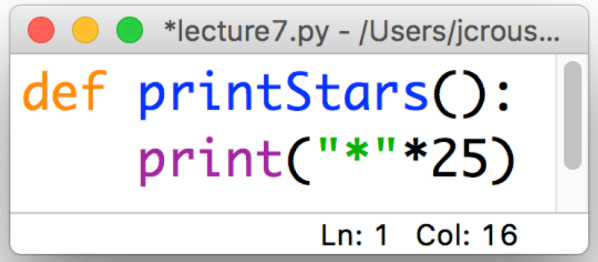
`main()`

functions
can also be
called by
one another

`stuart()` **`jerry()`**

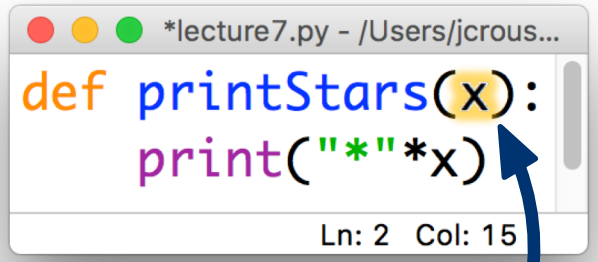
Two kinds of functions

Some functions always **do the same thing**




```
printStars()
printStars()
printStars()
```

Others adjust their behavior based on **what we give them**



"parameter"

```
printStars(5)
printStars(32)
printStars(1527)
```

15-minute exercise: Happy Birthday

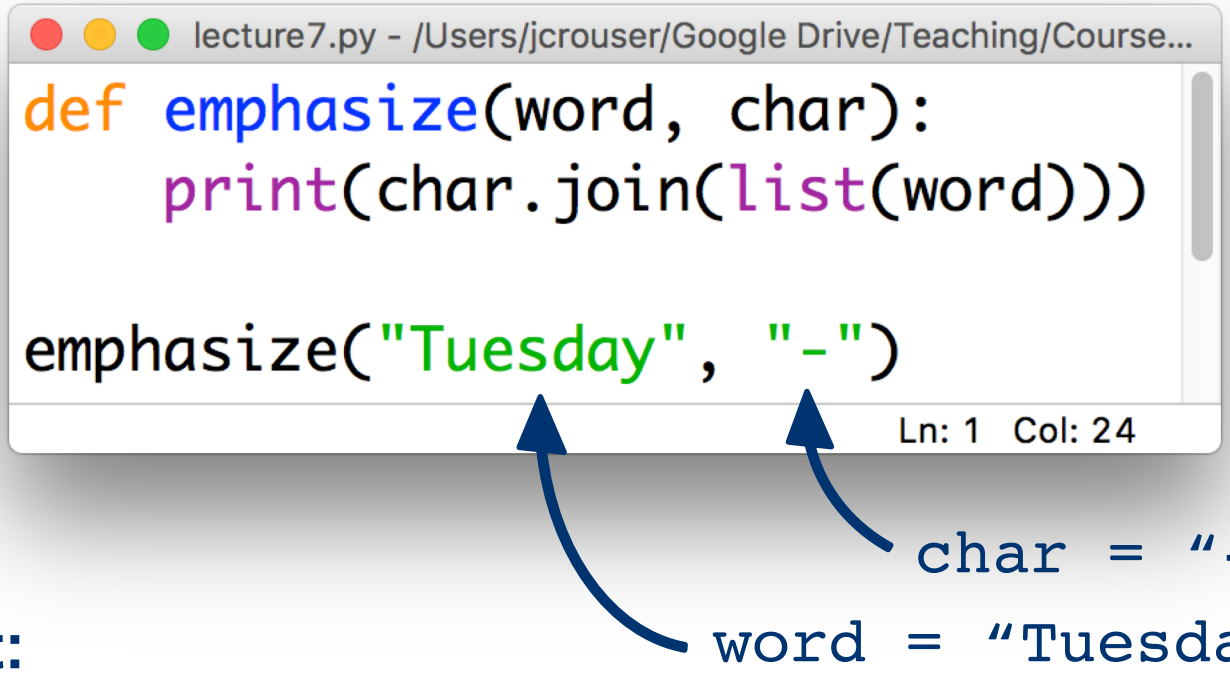
- Write a function called **`happyBirthday(name)`** that takes in a string **`name`** and prints out the lyrics to the song ”Happy Birthday” with the name inserted:

```
Happy birthday to you!
Happy birthday to you!
Happy birthday, dear NAME
Happy birthday to you!
```

- Call this function from inside the **`main()`** function, and use **`input(…)`** to get the value of **`name`** from the user

Parameters

- Functions can be defined to take in **multiple** parameters:

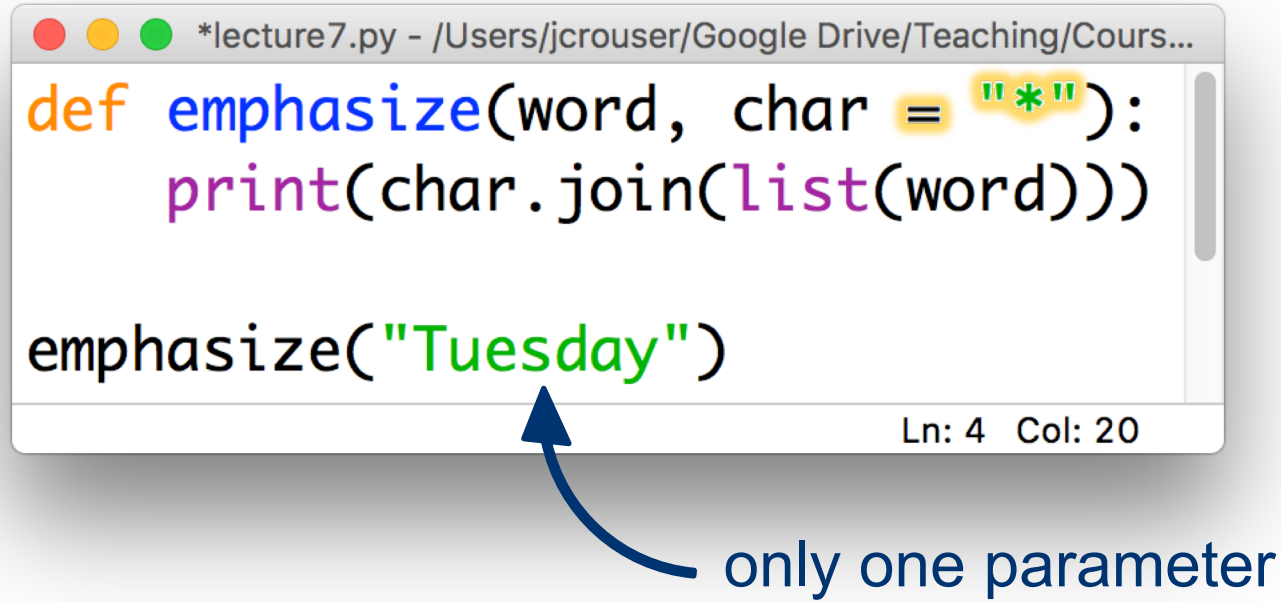


- **Result:**

```
T-u-e-s-d-a-y
```

Default parameters

- We can include a "default" value for some (or all) of them:

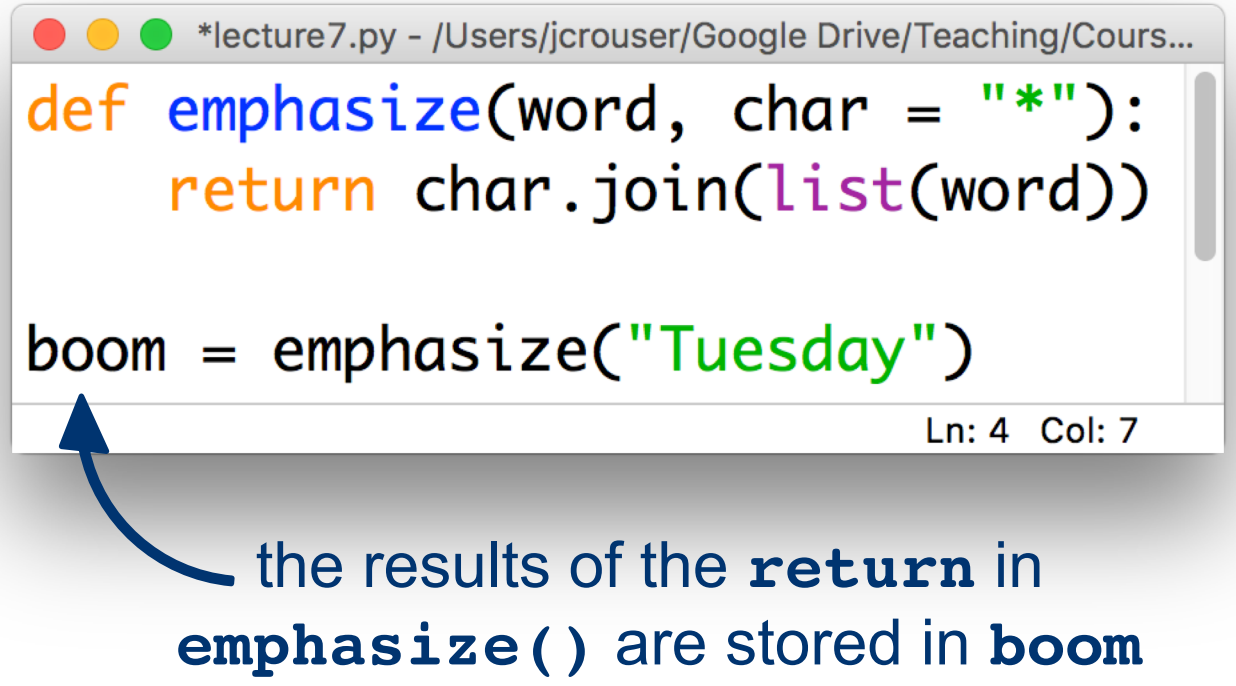


- **Result:**

```
T*u*e*s*d*a*y
```

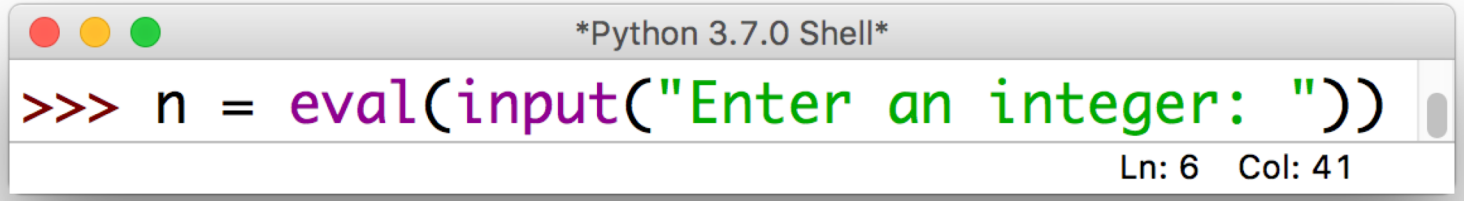
Returning values

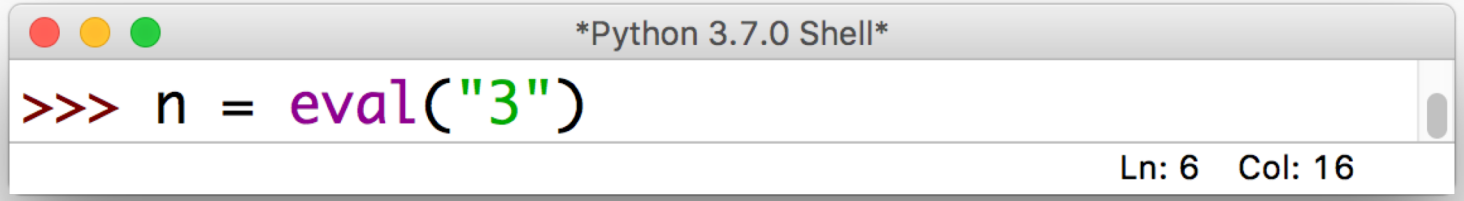
- We may want to **return** the results rather than print them:



Advanced: chaining functions

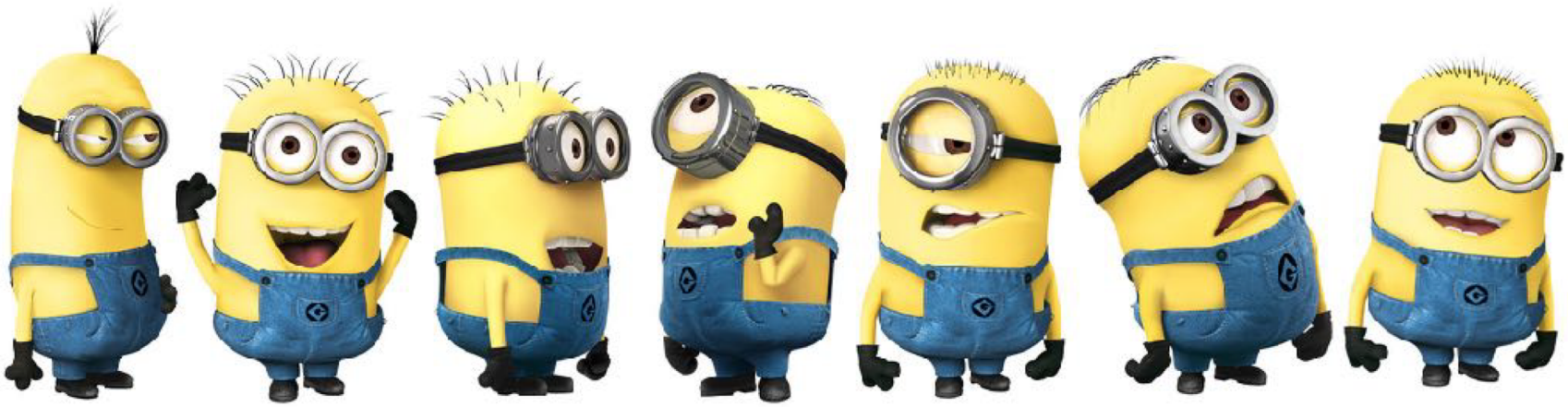
- Return values allow us to call functions **inside** other function calls:





Recap: functions

- If you have to do something **multiple times**, then you probably want a function: this helps to “modularize” code (i.e. organize it for easy reuse)
- **Define** once, **call** as many times as necessary
- Naming convention: use **`camelCase`**
- **Important**: one function = one task



(mini) Assignment 5: encrypt/decrypt

For this assignment, you'll write 3 functions:

- **`encrypt(…)`** will take in a string containing a message, and encrypt it using a variation of the PIG LATIN cipher*
- **`decrypt(…)`** will undo the effects of **`encrypt(…)`**, enabling us to recover the original message
- **`main(…)`** will ask the user to enter a message, then call **`encrypt(…)`** and print the results, and finally call **`decrypt(…)`** on the encrypted message and print the decrypted original message

* for specific details, see assignment on Moodle

Outline

- ✓Monday: FALL BREAK
- ✓Wednesday: Functions
  - ✓basic components
  - ✓definition vs. call
  - ✓an analogy
  - ✓parameters
  - ✓returning values
- LAB: MadLibs
- Friday: Catching Exceptions

MAD LIBS™

World's Greatest Word Game

It was a(n) _____ October day. I awoke to the

adjective

smell of _____ roasting in the _____.

type of food room in house

I _____ down the stairs and much to my

verb (past tense, -ed)

_____, I ran directly into a(n) _____

emotion adjective

_____. I found this very strange, because...

noun