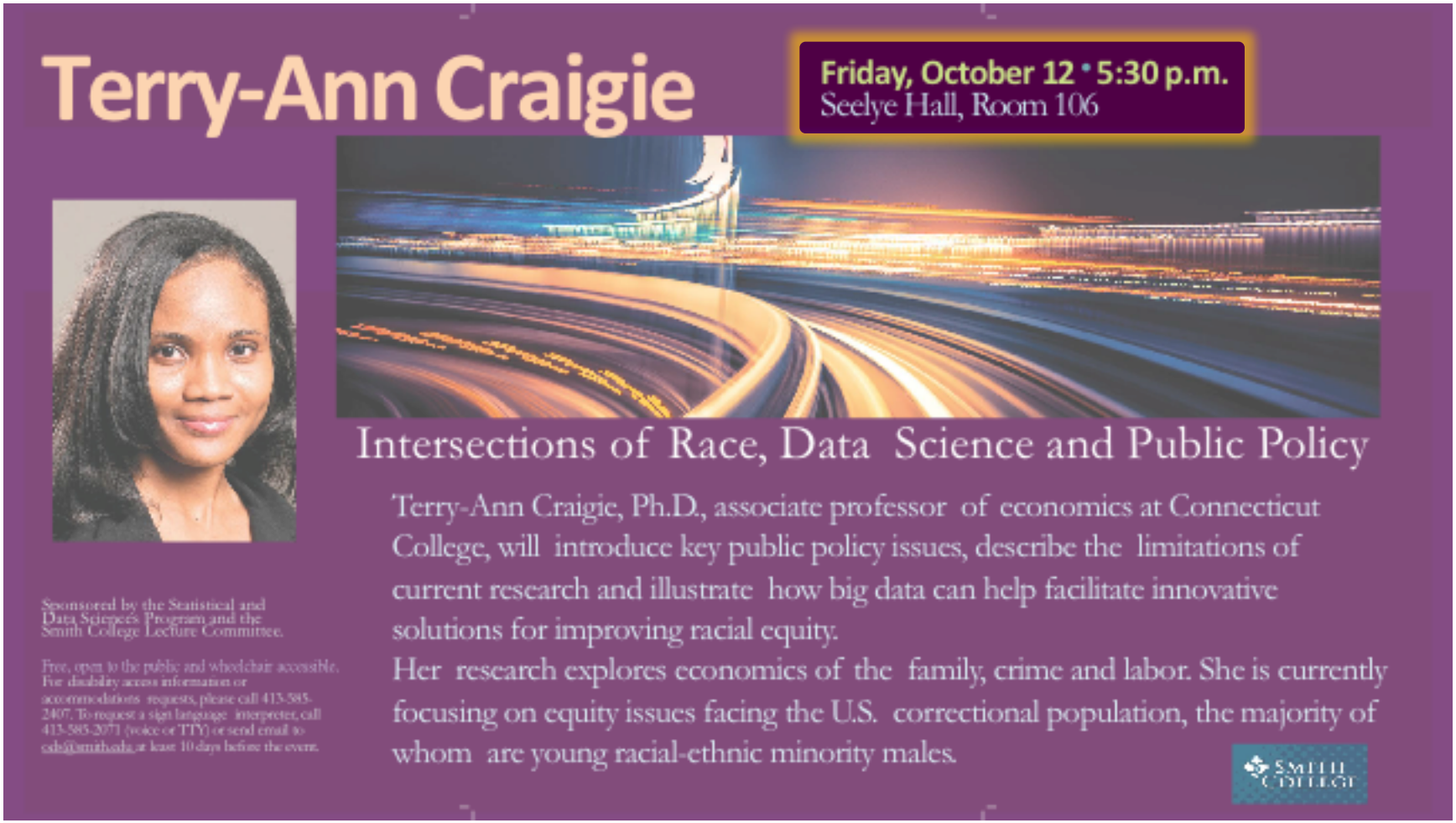
Lecture 15:

FUNCTIONS PT. 2

CSC111: Introduction to CS through Programming

R. Jordan Crouser
Assistant Professor of Computer Science
Smith College

Announcements 1/2



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”

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 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    5

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

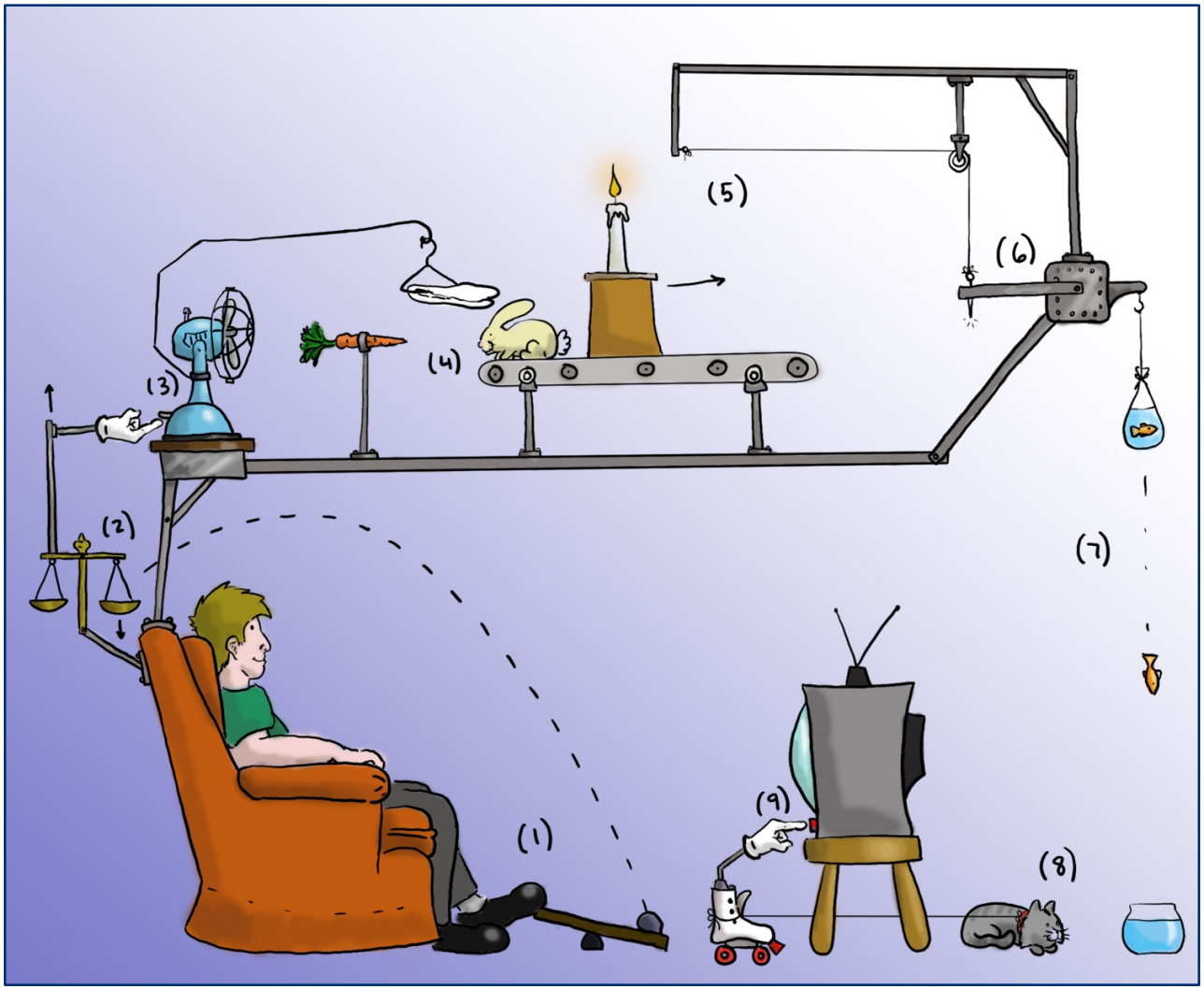
to `return` a value?

And what's the difference

between `return` and `print(…)`?

DEMO TIME

Demo: human Rube Goldberg machine



“Remote” - by philister

`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