

Lecture 13:

LIFE SKILL: DOCUMENTATION

CSC111: Introduction to CS through Programming

R. Jordan Crouser

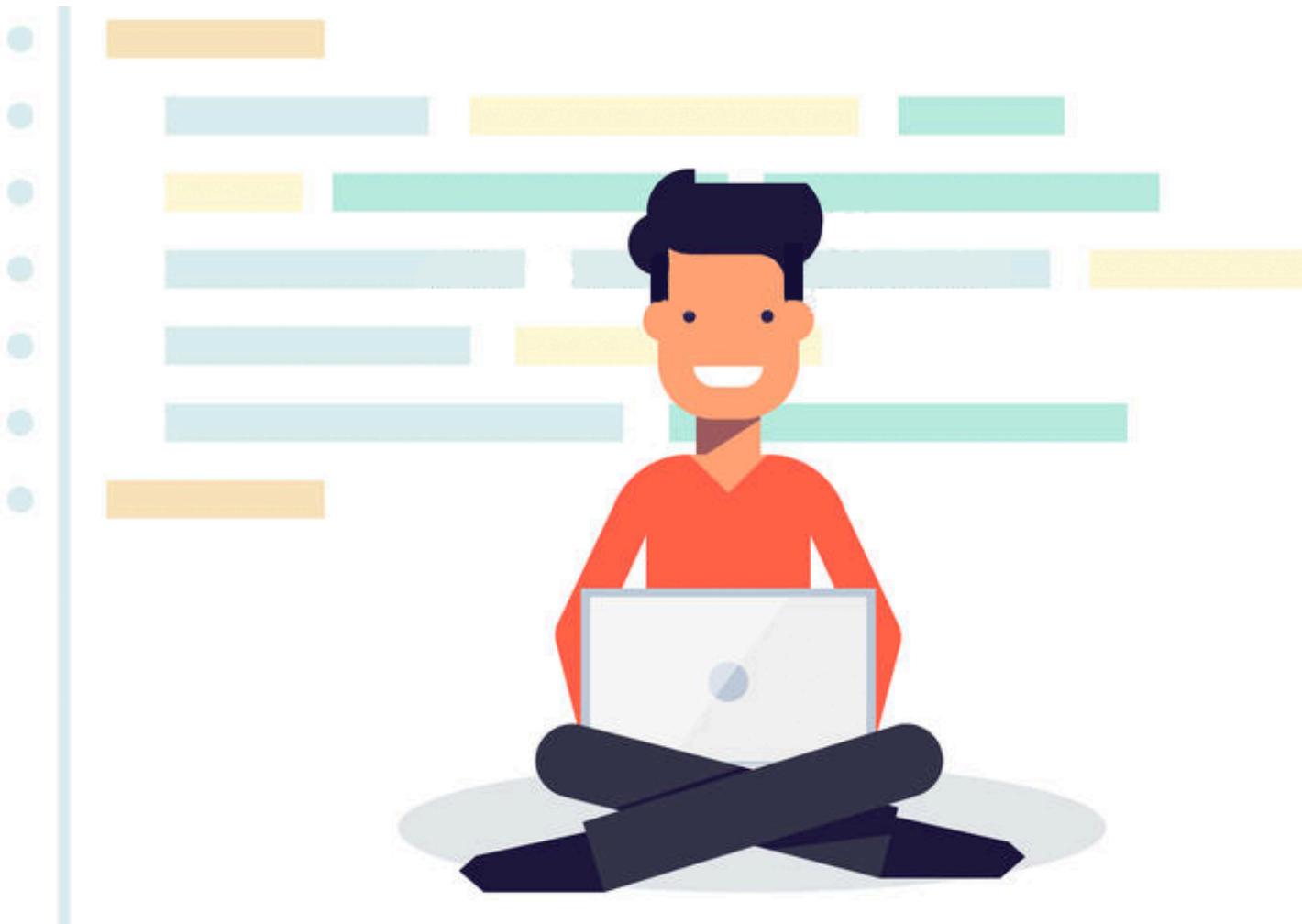
Assistant Professor of Computer Science

Smith College

Overview

- ✓ Announcements
- ✓ Debrief of “Life Skill #2: Debugging”
- ✓ Loops
 - ✓ `for...in` (looping through items in a list)
 - ✓ the `range()` function (getting a list of numbers)
 - ✓ `while` (looping until something happens)
- ✓ The `random` module
- Lab: Old MacDonald
- **Life Skill #3: Documentation**

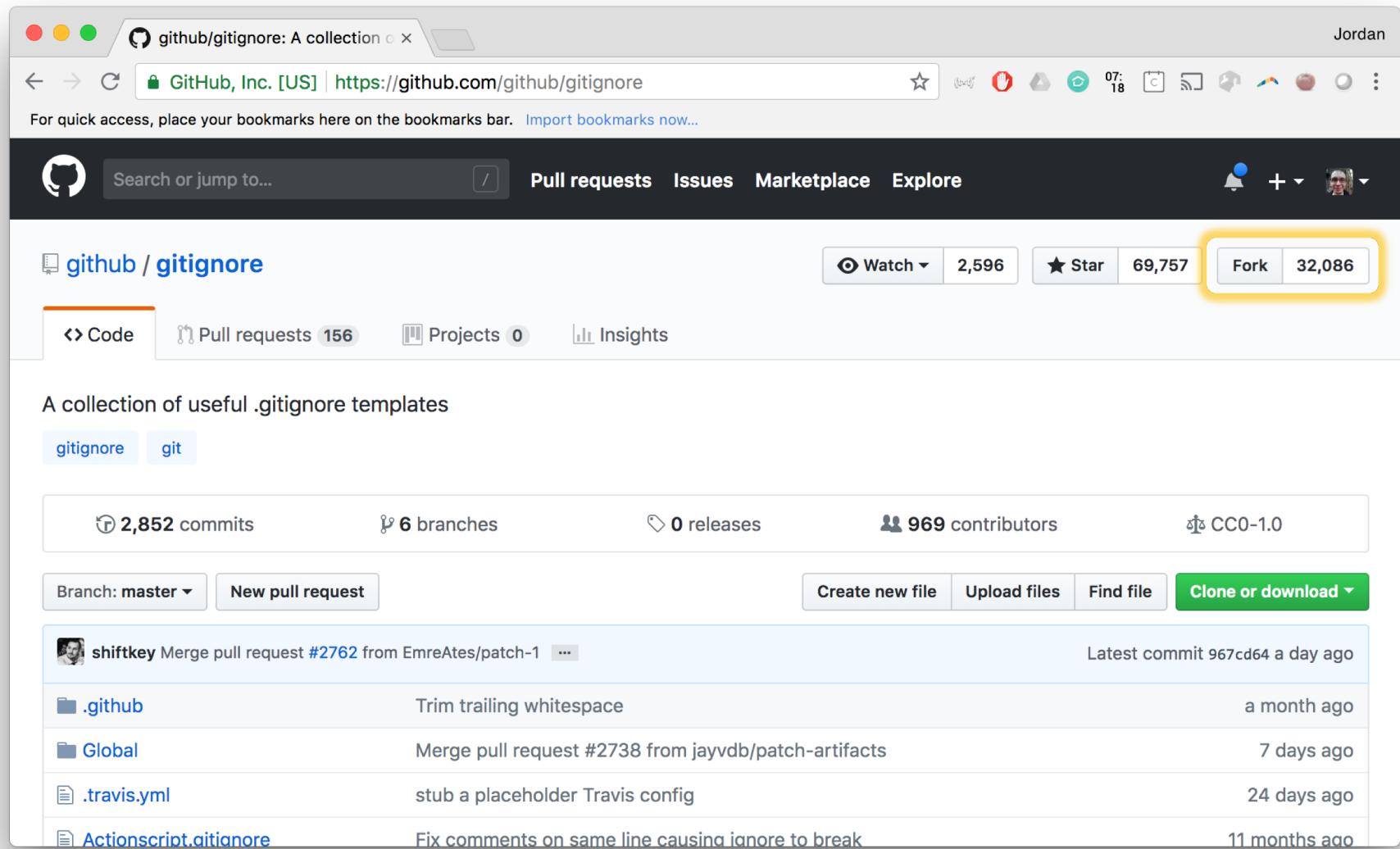
In the beginning...



Now...



Eventually...



A screenshot of a Mac OS X desktop showing a GitHub repository page for 'github/gitignore'. The browser window title is 'github/gitignore: A collection c x' and the address bar shows 'GitHub, Inc. [US] | https://github.com/github/gitignore'. The user 'Jordan' is logged in, as indicated by the profile icon in the top right. The page displays various repository statistics: 2,596 watches, 69,757 stars, and 32,086 forks. The 'Fork' button is highlighted with a yellow box. Below the stats, there are tabs for 'Code', 'Pull requests 156', 'Projects 0', and 'Insights'. The repository description is 'A collection of useful .gitignore templates'. It contains tags for 'gitignore' and 'git'. Key metrics shown include 2,852 commits, 6 branches, 0 releases, 969 contributors, and a CC0-1.0 license. A pull request from 'shiftkey' is listed as the latest commit. The repository has a history of contributions, including merges from 'EmreAtes/patch-1', 'jayvdb/patch-artifacts', and a placeholder Travis config. The last commit was 11 months ago.

github / gitignore

Watch 2,596 Star 69,757 Fork 32,086

Code Pull requests 156 Projects 0 Insights

A collection of useful .gitignore templates

gitignore git

2,852 commits 6 branches 0 releases 969 contributors CC0-1.0

Branch: master New pull request Create new file Upload files Find file Clone or download

shiftkey Merge pull request #2762 from EmreAtes/patch-1 ... Latest commit 967cd64 a day ago

.github Trim trailing whitespace a month ago

Global Merge pull request #2738 from jayvdb/patch-artifacts 7 days ago

.travis.yml stub a placeholder Travis config 24 days ago

Actionscript.gitignore Fix comments on same line causing ignore to break 11 months ago

The point

- **Other people** need to be able to understand your code
- **Future you** needs to be able to understand your code
- **Document it**

... but how?

Step 1: meaningful nouns for variables

```
*Untitled*
```

```
x = "Jordan Crouser"
y = "Professor"
z = "Smith College"
print(x, "-", y, "at", z)
```

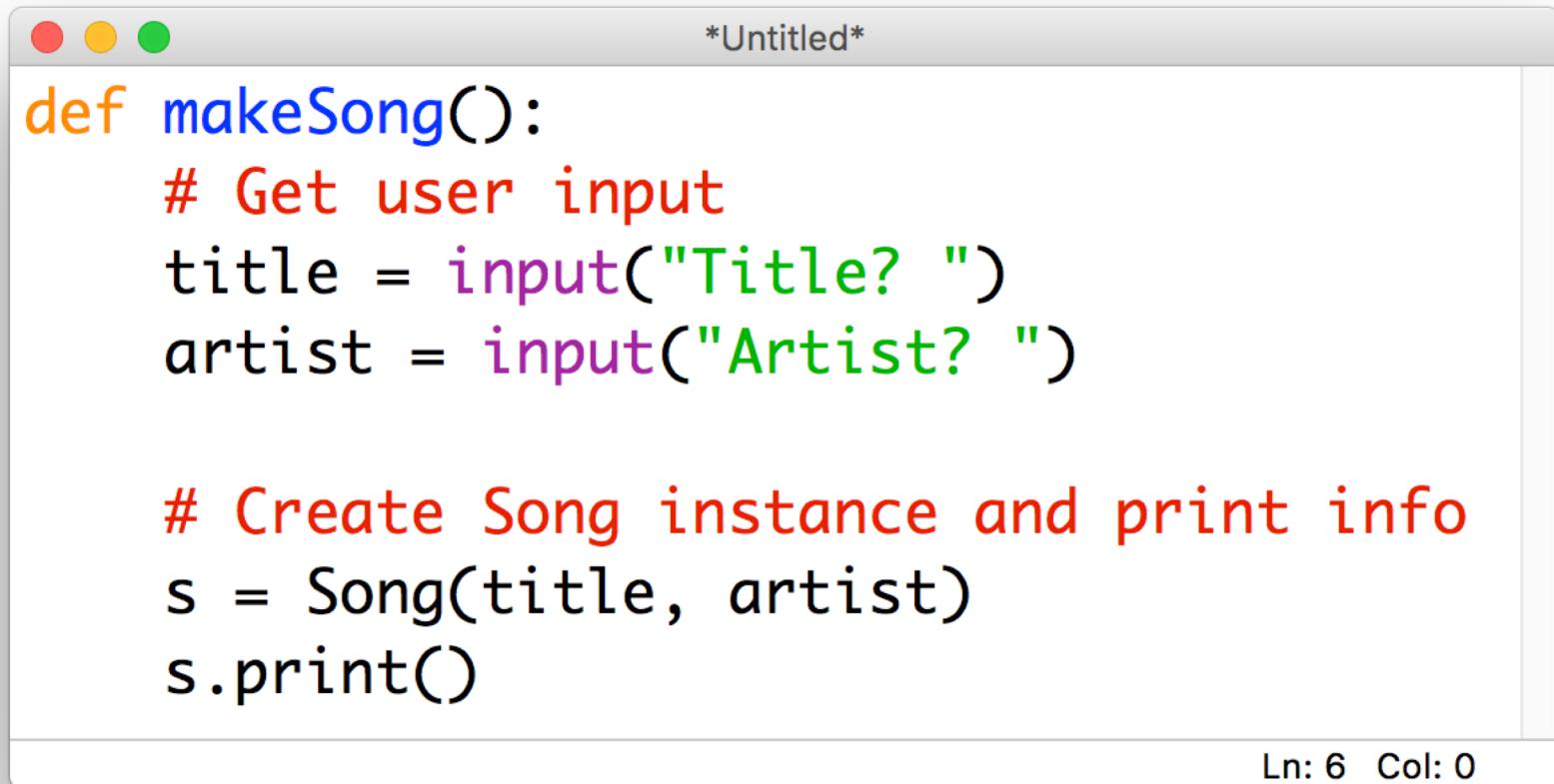
Ln: 1 Col: 1

```
*Untitled*
```

```
name = "Jordan Crouser"
title = "Professor"
institution = "Smith College"
print(name, "-", title, "at", institution)
```

Ln: 4 Col: 41

Step 2: lots of comments



The image shows a screenshot of a code editor window titled "*Untitled*". The window has three colored window control buttons (red, yellow, green) in the top-left corner. The code editor displays the following Python script:

```
def makeSong():
    # Get user input
    title = input("Title? ")
    artist = input("Artist? ")

    # Create Song instance and print info
    s = Song(title, artist)
    s.print()
```

In the bottom right corner of the code editor, there is a status bar displaying "Ln: 6 Col: 0".

A useful technique: code tracing

- **Given:** a very short, poorly-documented program
- **Your goal:** try to figure out what it's doing
- **Recommendations:**
 - walk through the program step-by-step (“**trace its execution**”) using the **whiteboard or paper** instead of typing it into IDLE
 - once you understand what’s happening, then rewrite it using **informative variable names** and **comments**

Example

```
*Untitled*
```

```
x = int(input("Enter lower bound: "))
y = int(input("Enter upper bound: "))

for z in range(x, y+1):
    if z > 1:
        p = True
        for zz in range(2, z):
            if (z % zz) == 0:
                p = False
                break
        if p:
            print(z)
```

Ln: 12 Col: 16

Debrief

This is great, but what about when
our programs get **more complicated?**



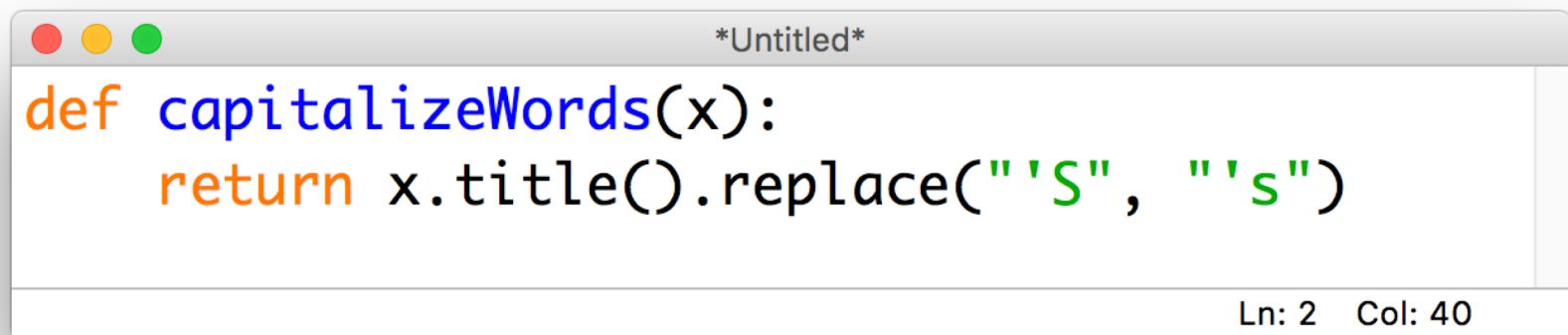
Step 3*: describe the **action** for functions



A screenshot of a Mac OS X-style code editor window titled "*Untitled*". The code editor displays the following Python function:

```
def bloop(x):
    return x.title().replace("'S", "'s")
```

The status bar at the bottom right shows "Ln: 1 Col: 9".

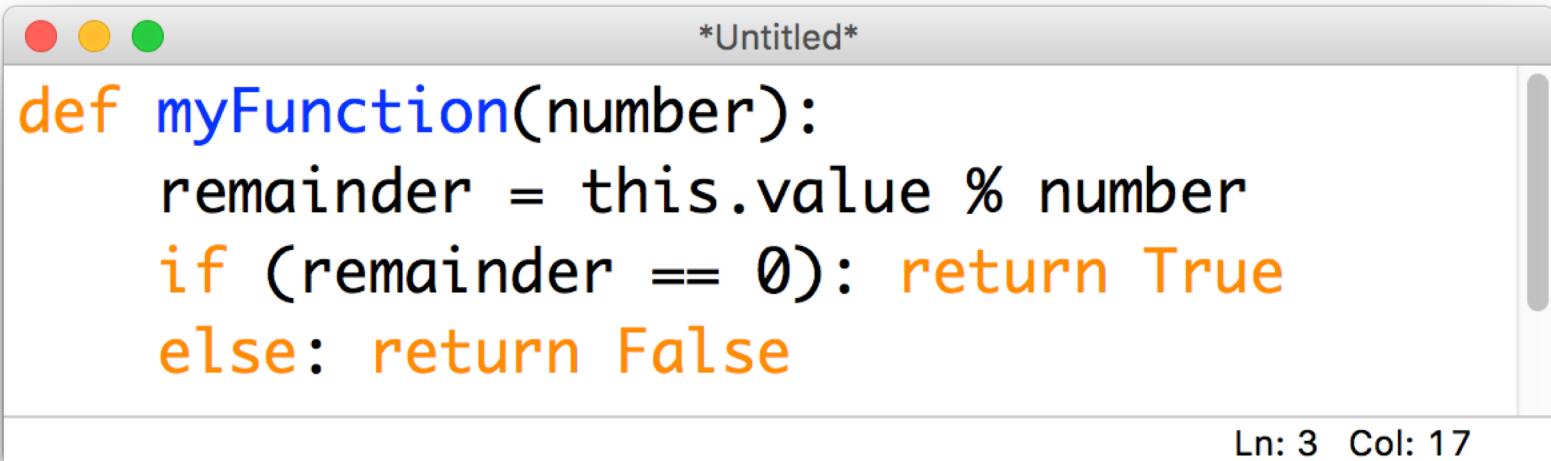


A screenshot of a Mac OS X-style code editor window titled "*Untitled*". The code editor displays the following Python function:

```
def capitalizeWords(x):
    return x.title().replace("'S", "'s")
```

The status bar at the bottom right shows "Ln: 2 Col: 40".

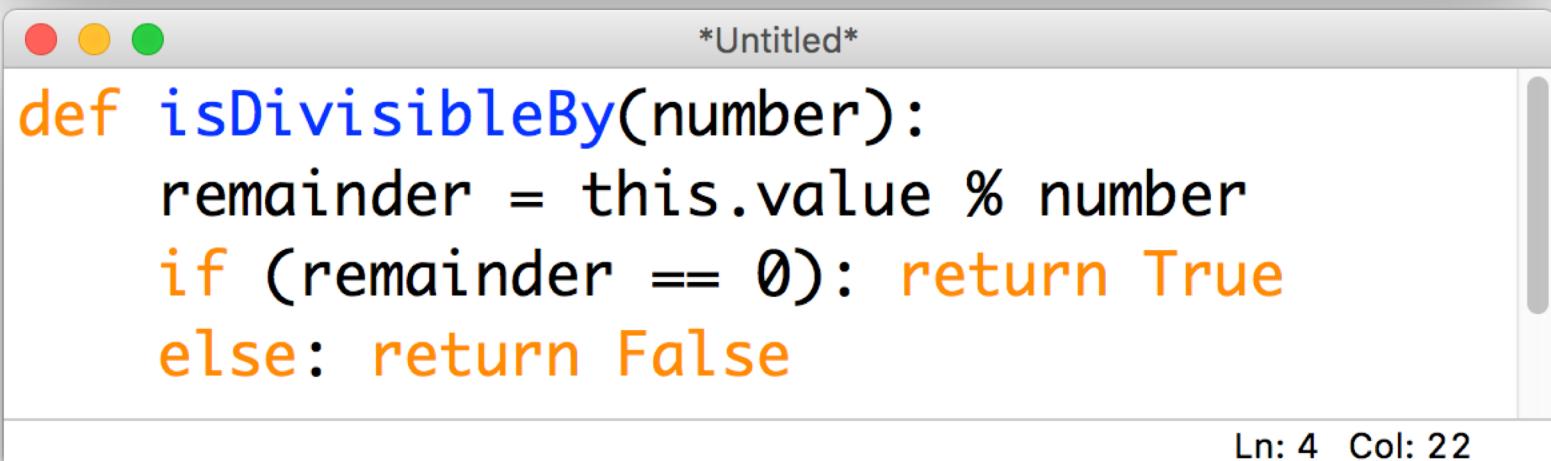
Step 3*: describe the **action** for functions



A screenshot of a code editor window titled "*Untitled*". The code defines a function named `myFunction` that takes a parameter `number`. It calculates the remainder of the current value divided by `number`. If the remainder is zero, it returns `True`; otherwise, it returns `False`. The code editor shows the following text:

```
def myFunction(number):
    remainder = this.value % number
    if (remainder == 0): return True
    else: return False
```

The status bar at the bottom right indicates "Ln: 3 Col: 17".



A screenshot of a code editor window titled "*Untitled*". The code defines a function named `isDivisibleBy` that takes a parameter `number`. It calculates the remainder of the current value divided by `number`. If the remainder is zero, it returns `True`; otherwise, it returns `False`. The code editor shows the following text:

```
def isDivisibleBy(number):
    remainder = this.value % number
    if (remainder == 0): return True
    else: return False
```

The status bar at the bottom right indicates "Ln: 4 Col: 22".

Step 4*: docstrings

```
*Untitled*
```

```
def makeSong():
    """ Creates and prints a Song instance
        from user input"""
    # Get user input
    title = input("Title? ")
    artist = input("Artist? ")

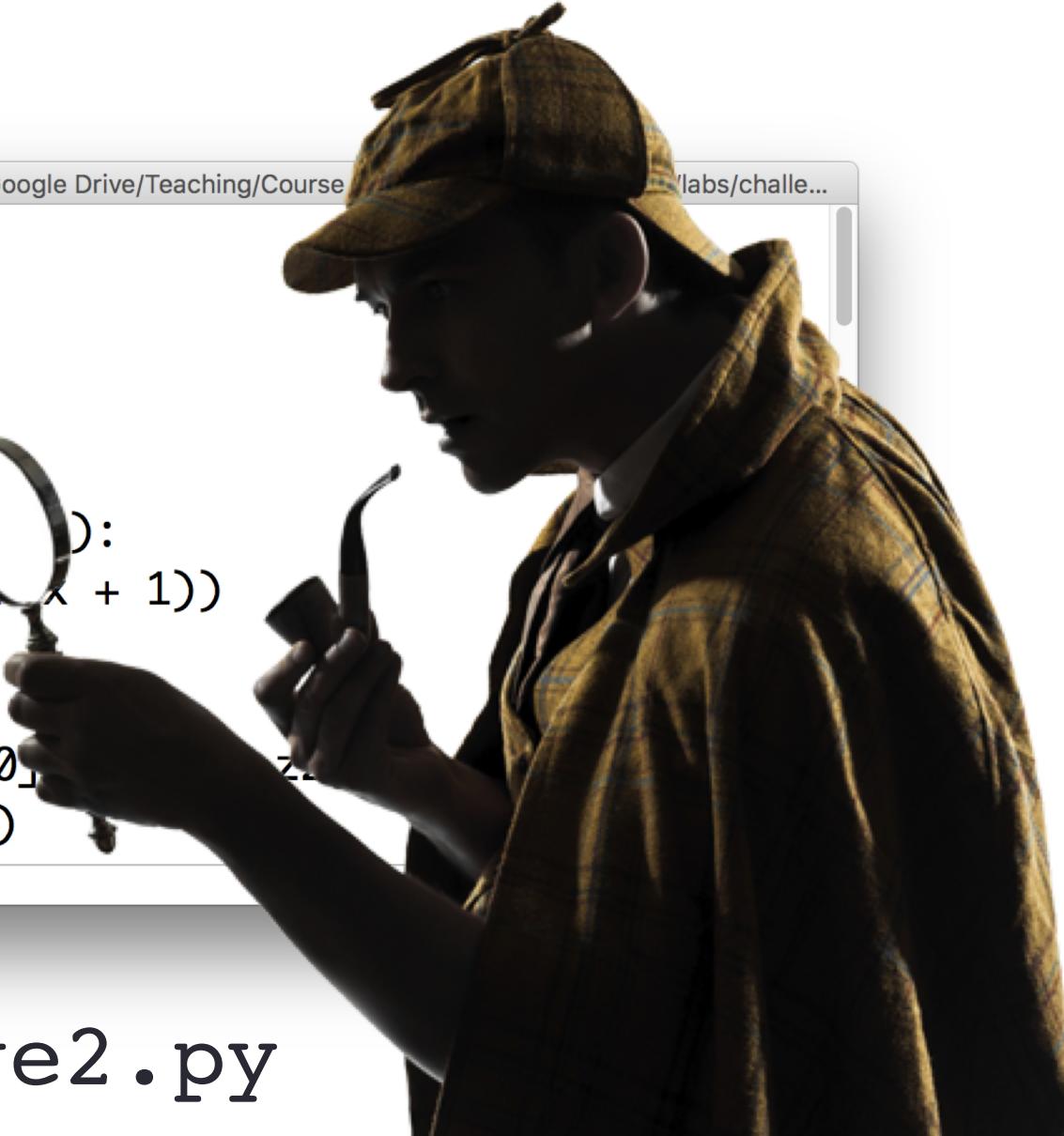
    # Create Song instance and print info
    s = Song(title, artist)
    s.print()
```

Ln: 5 Col: 0

D E M O

T I M E

Activity: “code detective”



A silhouette of a detective wearing a fedora hat and a trench coat, holding a magnifying glass over a computer screen. The screen displays a Python script named challenge2.py.

```
challenge2.py - /Users/jcrouser/Google Drive/Teaching/Course.../labs/challe...
zz = []
p1go = True
w = False

def su():
    for x in range(10):
        zz.append(str(x + 1))

def pb():
    print(' ' + zz[0])
    print('---+---')
```

challenge2.py

Debrief

What was that experience like?

What **strategies** did you use?



Coming up next

- *A4: Magic 8 Ball* is due **Sunday 11:55pm**
- **Mon 10/8: NO CLASS (FALL BREAK)**
- **Weds 10/10:** Writing functions
- **Lab:** MadLibs
- **Fri 10/5:** Working with Files