INTRO TO PYTHON

Shannon Turner

Twitter: @svt827
GitHub: @shannonturner

HEAR ME CODE

OBJECTIVE

- Learn about variables
- Learn what strings are
- Learn how to display and format them
- Learn what conditionals are
- Learn how to use conditionals to change how your program behaves

VARIABLES

- Variables are containers for information; you can store text, numbers, or any other type of thing!

```
twitter = "@hearmecode"
members = 571
```

THE PRINT COMMAND

- Use the **`print`** command to show some information to the screen.

```
print "Welcome to Hear Me Code!"
print 'No boys allowed!'
```

- Since we created a variable on the previous slide, we can use it now:

```
print twitter
```

THE PRINT COMMAND

- Let's take a closer look at the difference between these two:

```
print twitter
```

```
print "twitter"
```

STRINGS: THE BASICS

- Strings are a way to store information
  - Addresses
  - Email addresses
  - URLs
  - Names (people, places, …)
  - Phone Numbers
  - so much more (anything with text!)

STRINGS: THE BASICS

- Strings are combinations of characters
  - Letters
  - Numbers
  - Punctuation
  - Basically anything you can make on the keyboard and then some
  - Special characters, like tabs and newlines

STRINGS: THE BASICS

- How to spot a string: it has quotes around it

```
"This is a string"
'This is also a string'
```

- You can mix and match single and double quotes – but they're not completely interchangeable.

```
"This will give you an error'
'But "this" is entirely okay'
```

STRINGS: THE BASICS

- Using single or double quotes comes down to personal preference

- Sometimes it can make a difference:
  - `'My governor's name is Martin O'Malley'`
  - `'She said, "Testing, 1-2-3"'`

STRINGS: THE BASICS

- If you have a really long string, use a triple quote (`"""` or `'''`) to start your string and the same triple quote to end it

- **`"""Even though this string will span multiple lines, Python isn't going to yell at me - and I can use things like "double and 'single' quotes" without problems."""`**

HEAR ME CODE

STRINGS: THE BASICS

- Special Characters

  \n Newline

  \t Tab

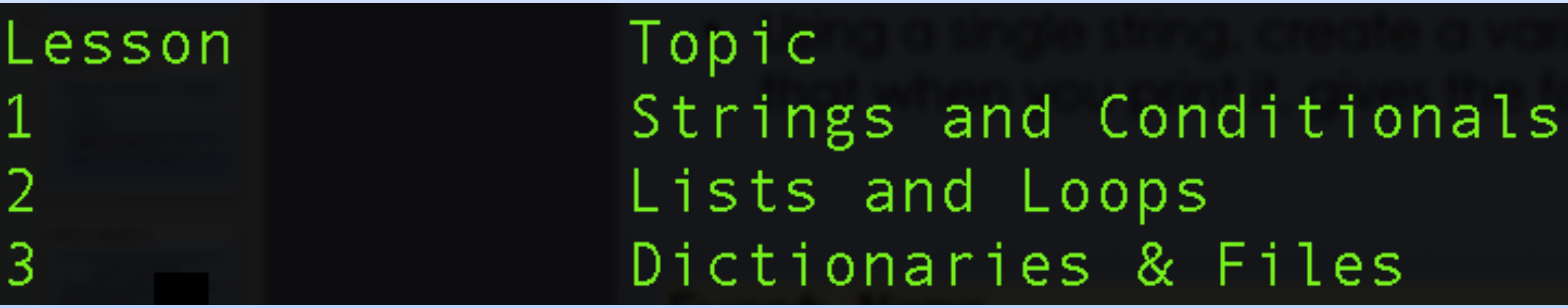
- Escape Characters

  \" Literal Double Quote

  \' Literal Single Quote

STRINGS: QUICK EXERCISE

- Print the following string:



```
Lesson          Topic
1               Strings and Conditionals
2               Lists and Loops
3               Dictionaries & Files
```

- Keep in mind you'll need to use special characters!

HOW LONG IS MY STRING?

- `twitter = "@hearmecode"`
- `len(twitter)`
- `len()` works on lists, too! We'll work with **lists** in Lesson 2.

STRINGS: SLICING

- `twitter = "@hearmecode"`

- Slicing lets you see individual pieces or "slices" of your string*

*Slicing also works with **lists** in the same way.

STRINGS: SLICING

- **`twitter = "@hearmecode"`**
- Simple slices: **`twitter[0]`**
  - Here, 0 refers to the **index** that you want to see
  - Slicing on first_name and last_name can give us a person’s initials; slicing on phone number can give area code

0	1	2	3	4	5	6	7	8	9	10
@	h	e	a	r	m	e	c	o	d	e
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

HEAR ME CODE

STRINGS: SLICING

- You can return more than one item in a slice
  - `twitter[1:5]`
    - The **index** on the left (1) is where you start
    - The **index** on the right (5) is where you end, but Python **stops short** and doesn't include it

0	1	2	3	4	5	6	7	8	9	10
@	h	e	a	r	m	e	c	o	d	e
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

STRINGS: SLICING

- The indices you provide are optional!

```
twitter[:5]
```

The left index is not provided, so Python assumes you want to start at the beginning and stop just short of item 5

0	1	2	3	4	5	6	7	8	9	10
@	h	e	a	r	m	e	c	o	d	e
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

HEAR ME CODE

STRINGS: SLICING

- The indices you provide are optional!

```
twitter[1:]
```

The right index is not provided, so Python assumes you want to start at item 1 and go to the end

0	1	2	3	4	5	6	7	8	9	10
@	h	e	a	r	m	e	c	o	d	e
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

HEAR ME CODE

STRINGS: SLICING EXERCISE

- `phone = "(202) 456-7890"`

- Use slicing to print out the area code
- And then the middle three numbers

STRINGS: STRING FORMATTING

- **`"My name is: {0} and my age is: {1} ".format(first_name, age)`**

- Think of the numbers as placeholders for your variables (or like Mad libs!)

- Remember, Python starts counting at zero. So zero is the first variable, which corresponds to **`first_name`**

STRINGS: FORMATTING EXERCISE

- **`phone = "202-555-9876"`**
- Using '**`.format()`**' and slicing, print out that phone number in these formats:
  - **Your number is: 202-555-9876**
  - **Local: 555-9876**
  - **Domestic: (202) 555-9876**
  - **International: +1-202-555-9876**

STRINGS: STRING METHODS

- String methods let you perform special actions on your strings
  - Replace one part of a string with another
  - Find one part of a string within the string
  - Count the number of times one part of a string appears within the string
  - … and many more!

STRINGS: STRING METHODS

- `email_address = "shannon@hearmecode.com"`
- `email_address.find("@")`

- Similar to Ctrl+F in most programs
- Remember slicing? The number you get back is the index where you found the item! (Unless it's -1)

STRINGS: STRING METHODS

- `twitter = "@hearmecode"`
- `twitter.replace("@", "#")`
- Similar to Ctrl+H in Word, Excel, other programs (find and replace)

HEAR ME CODE

STRINGS: STRING METHODS

- `twitter = "@hearmecode"`
- `twitter.replace("@", "#")`
- `print twitter`

- Making it stick:

```
twitter = twitter.replace("@", "#")
print twitter
```

HEAR ME CODE

HOW FUNCTIONS WORK

- **Arguments/parameters** tell a **function** or **method** how to do their action, or what to do it to.

- **`len(tweet)`**

- Function (action): **`len()`**
- Argument/parameter: **`tweet`**

HOW FUNCTIONS WORK

- **Arguments/parameters** tell a **function** or **method** how to do their action.

- **`twitter.replace("@", "#")`**

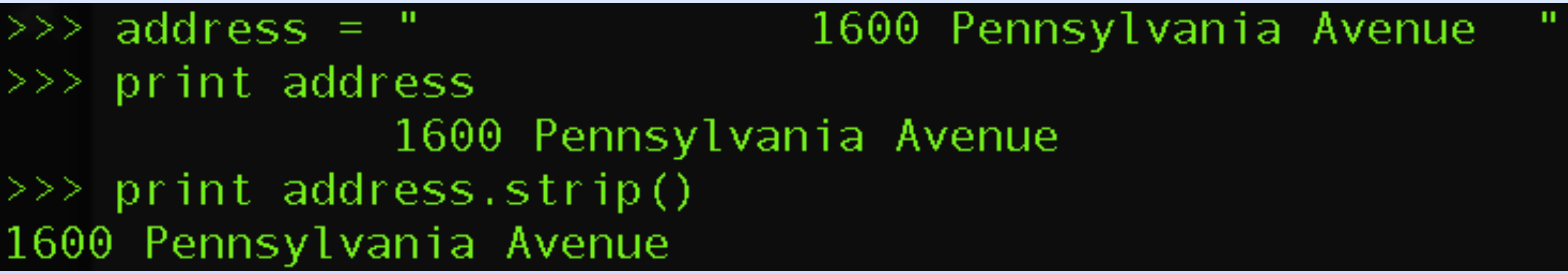
- Function (action): **`.replace()`**
- Argument/parameter: **`"@"`** and **`"#"`**
- Where does Python perform the find/replace? On the string that comes before the dot!

STRINGS: STRING METHODS

- Some functions and methods give you **return values** when they're finished so you know what happened.

- `length = len(tweet)`
- `position = phone.find("(")`

STRINGS: STRING METHODS



```
>>> address = "                  1600 Pennsylvania Avenue   "
>>> print address
          1600 Pennsylvania Avenue
>>> print address.strip()
1600 Pennsylvania Avenue
```

`.strip()`

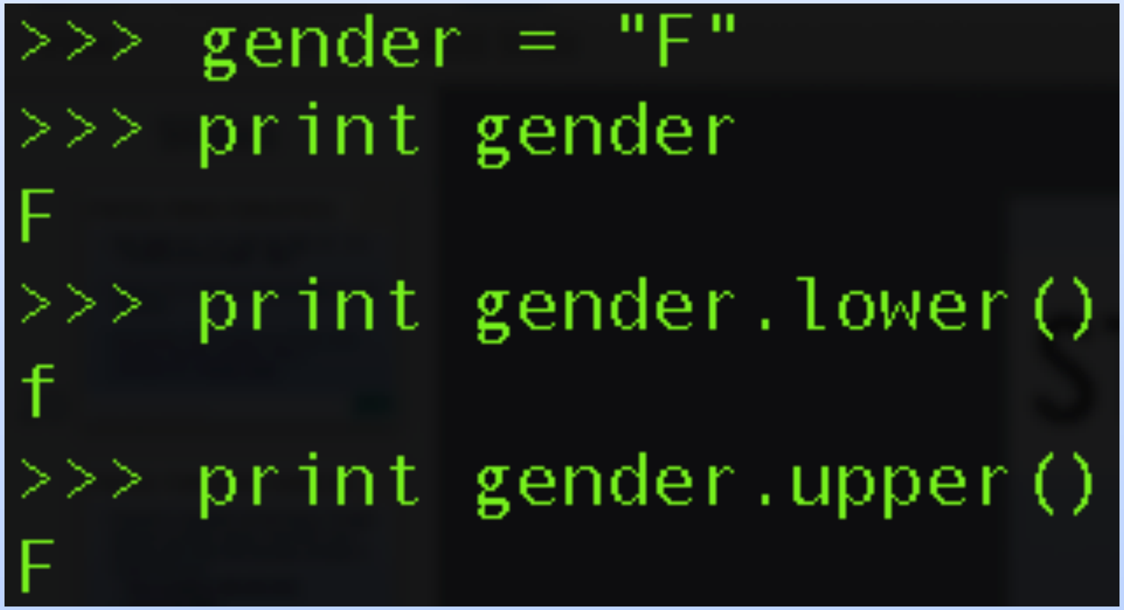
- Removes whitespace from the beginning and end of a string (not the middle)

HEAR ME CODE

STRINGS: STRING METHODS

`.lower()`

`.upper()`



```
>>> gender = "F"
>>> print gender
F
>>> print gender.lower()
f
>>> print gender.upper()
F
```

- Converts a string to all lowercase or all uppercase

STRINGS: STRING METHODS

```
article.count(" she said")
article.count(" he said")
```

- How many times was a woman quoted in this article?
- How many times was a man quoted?

AGENDA

- Conditionals
  - The basics
  - Operators
  - Compound conditionals
  - Using conditionals to change program behavior

HEAR ME CODE

CONDITIONALS

- Conditional: just a fancy name for a yes or no question
- Conditionals are ways to compare things and use that information to make decisions
- Conditionals can let you change the behavior of your program

CONDITIONALS

- Ways to think about conditionals

- Is this a valid email address?
- Does my phone number have enough digits?
- Are more people signed up for my event than the room can hold?

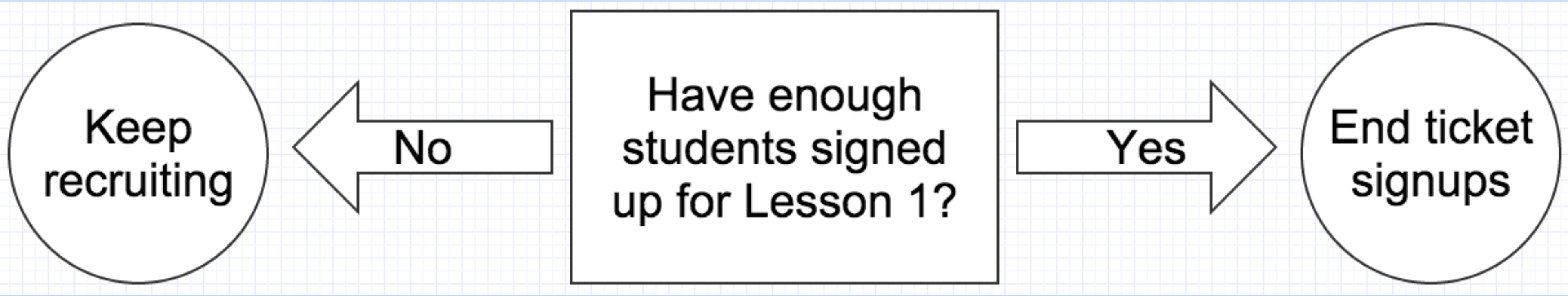
CONDITIONALS

- Conditionals are paired with if statements, which ask whether or not the conditional is true

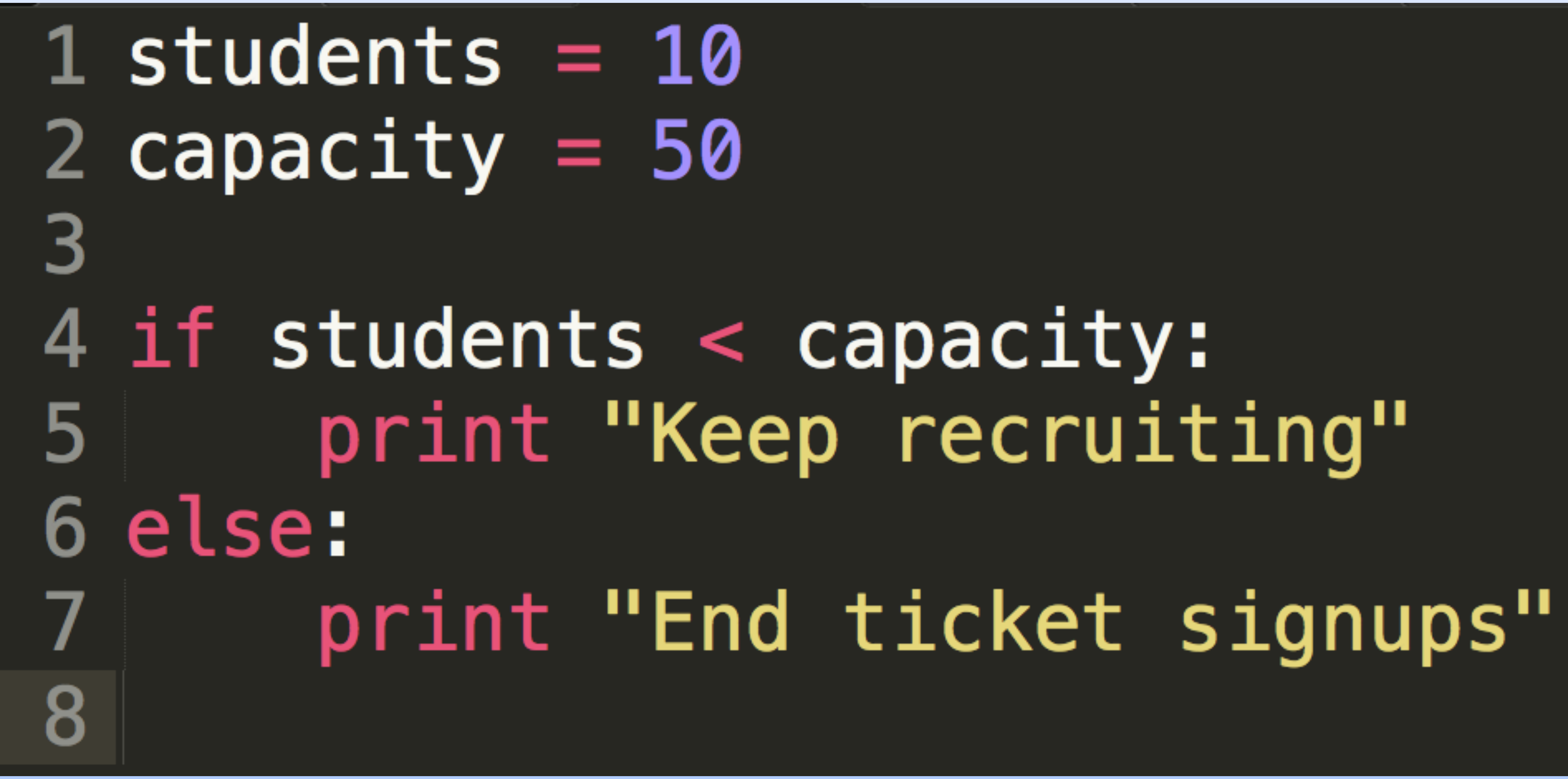
```
if gender == 'f':
  print "Welcome to Hear Me Code!"
```

  - True or False, is gender equal to "f"?
  - If so, they can join Hear Me Code. Otherwise, they can't!

CONDITIONALS



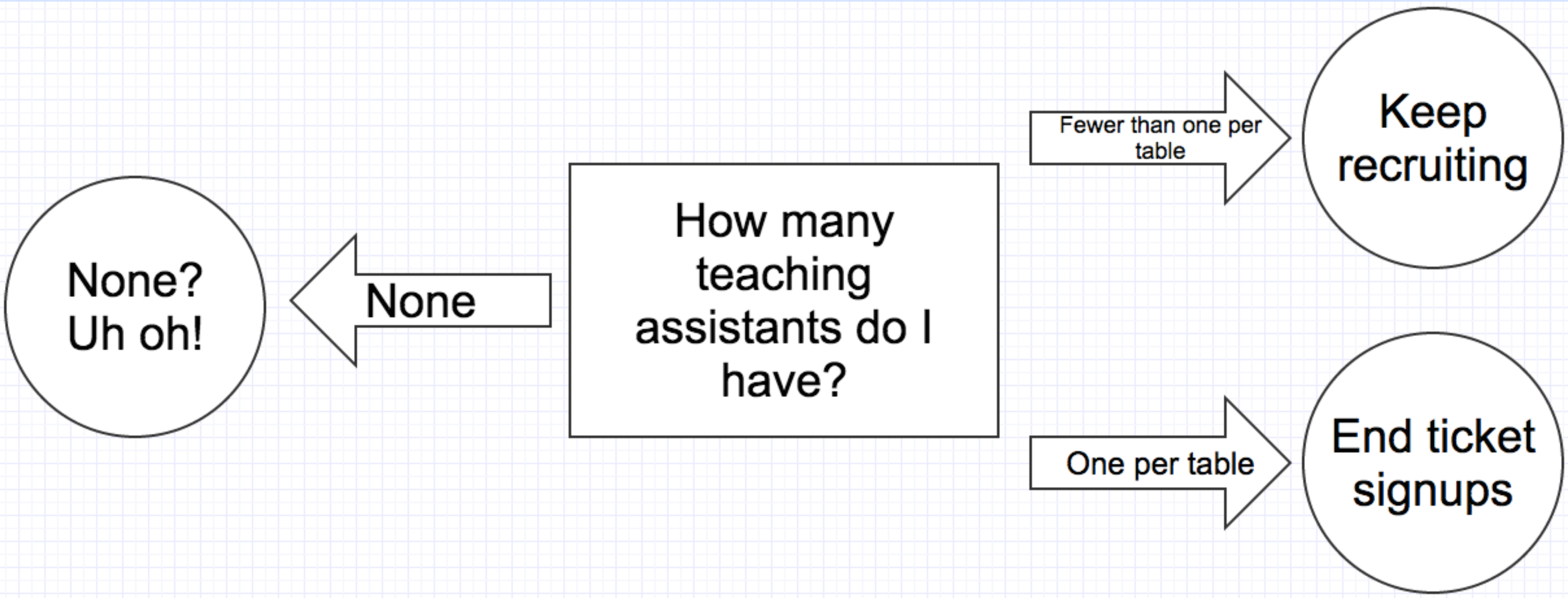
CONDITIONALS



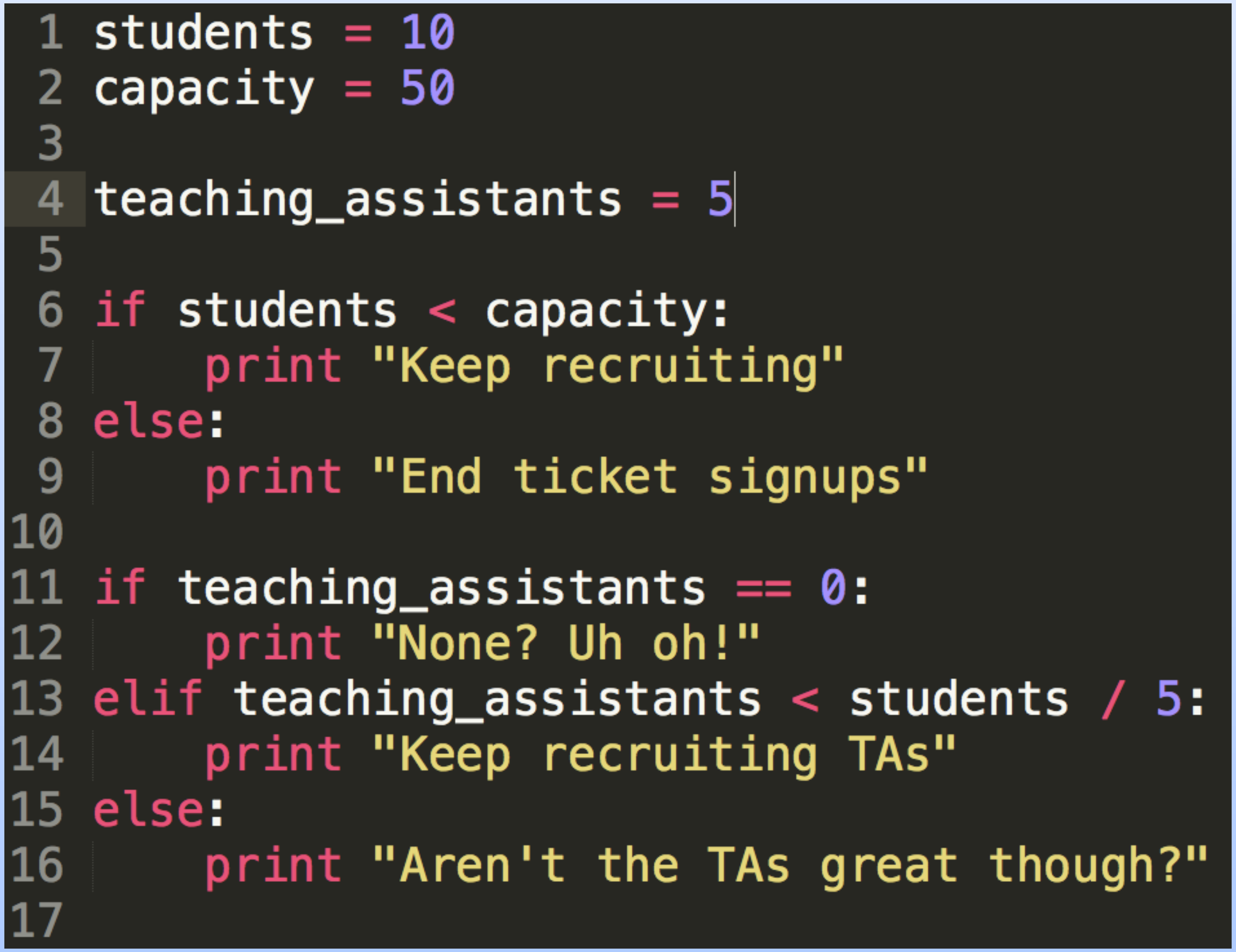
```
students = 10
capacity = 50

if students < capacity:
    print "Keep recruiting"
else:
    print "End ticket signups"

```

@hearmecode #hearmecode

CONDITIONALS



CONDITIONALS



```
students = 10
capacity = 50

teaching_assistants = 5

if students < capacity:
    print "Keep recruiting"
else:
    print "End ticket signups"

if teaching_assistants == 0:
    print "None? Uh oh!"
elif teaching_assistants < students / 5:
    print "Keep recruiting TAs"
else:
    print "Aren't the TAs great though?"
```

CONDITIONALS

Operators (ways to compare two things)

== Equality operator (don’t confuse with a single equals sign)

```
5 == 7 # Python says: False
5 == 5 # Python says: True
```

HEAR ME CODE

CONDITIONALS

Operators (ways to compare two things)

> Greater than operator

```
5 > 7 # Python says: False
5 > 2 # Python says: True
```

CONDITIONALS

Operators (ways to compare two things)

`==`	Equality
`!=`	Not equal to
`>`	Greater than
`<`	Less than
`>=`	Greater than or equal to
`<=`	Less than or equal to

HEAR ME CODE

CONDITIONALS

```
if times_volunteered >= 5:
  # send them a special thank-you

if donation >= 1000:
   # add to the major donors list
```

HEAR ME CODE

CONDITIONALS QUICK EXERCISE

Create a quick calculator program with two variables (**goal**, **current_volunteers**) and tells the user whether they are at, below, or above their recruitment goal.

Example:

Volunteer Recruitment Goal: 100

Current Volunteers: 95

> You are behind, work on recruiting!

HEAR ME CODE

COMPLEX CONDITIONALS

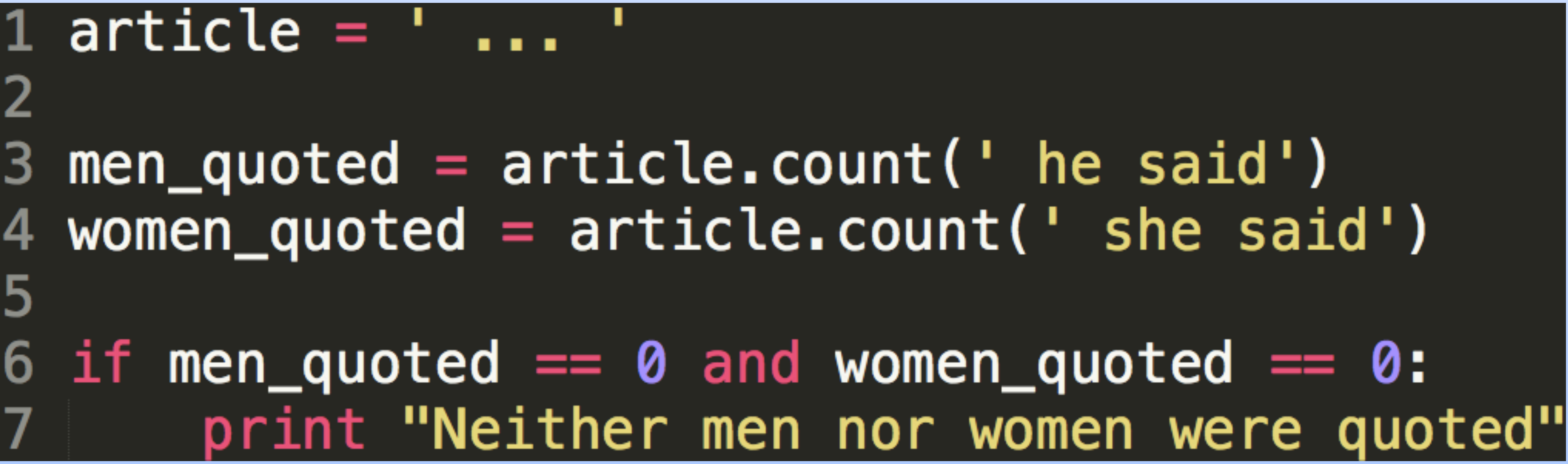
You can use your string methods as part of the conditional!

```
if gender.lower() == "f":
  # No matter how it's capitalized

if email_address.count("@") > 1:
  # this isn’t a valid email
```

HEAR ME CODE

COMPOUND CONDITIONALS

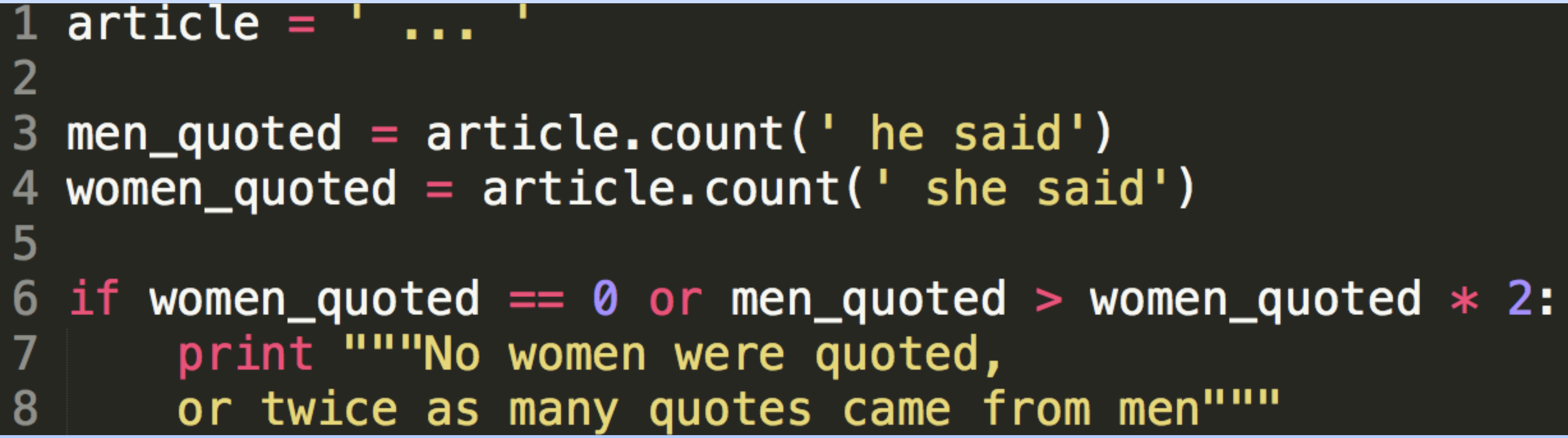
Using the **and** keyword, both conditions must be true for the print statement at line 7 to run.



```
1 article = ' ... '
2
3 men_quoted = article.count(' he said')
4 women_quoted = article.count(' she said')
5
6 if men_quoted == 0 and women_quoted == 0:
7     print "Neither men nor women were quoted"
```

COMPOUND CONDITIONALS

Using the **`or`** keyword, either condition could be true for the print statement at lines 7-8 to run.

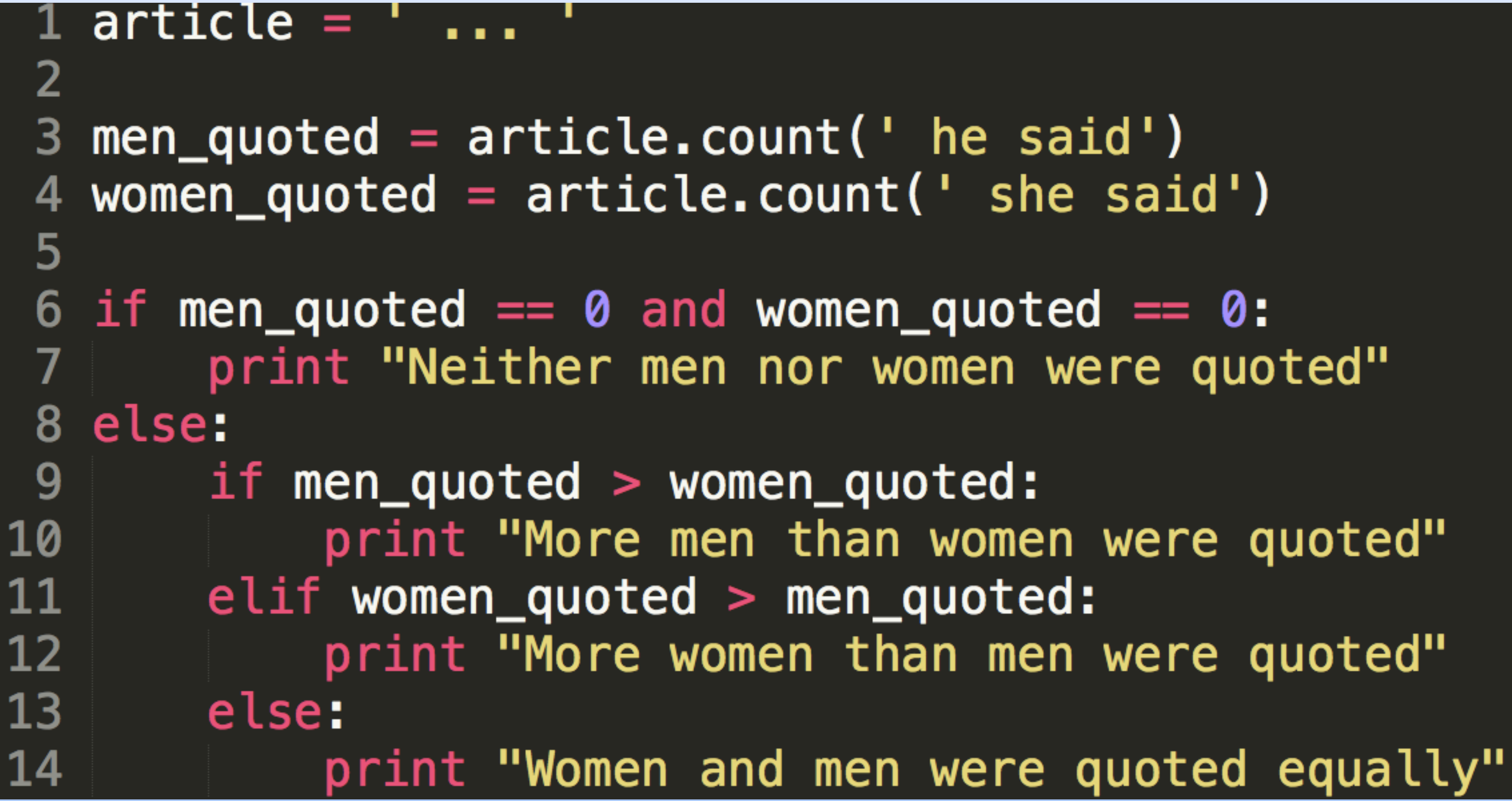


```
1 article = ' ... '
2 
3 men_quoted = article.count(' he said')
4 women_quoted = article.count(' she said')
5 
6 if women_quoted == 0 or men_quoted > women_quoted * 2:
7     print """No women were quoted,
8     or twice as many quotes came from men"""
```

@hearmecode #hearmecode

HEAR ME CODE

NESTED CONDITIONALS



```
article = ' ... '

men_quoted = article.count(' he said')
women_quoted = article.count(' she said')

if men_quoted == 0 and women_quoted == 0:
    print "Neither men nor women were quoted"
else:
    if men_quoted > women_quoted:
        print "More men than women were quoted"
    elif women_quoted > men_quoted:
        print "More women than men were quoted"
    else:
        print "Women and men were quoted equally"
```

PLAYTIME!

Head to https://github.com/shannonturner/python-lessons, which has code samples for the lesson:

Variables, math, and basics recap

Strings recap

Conditionals recap

See the playtime folder for the playtime exercise for this lesson.