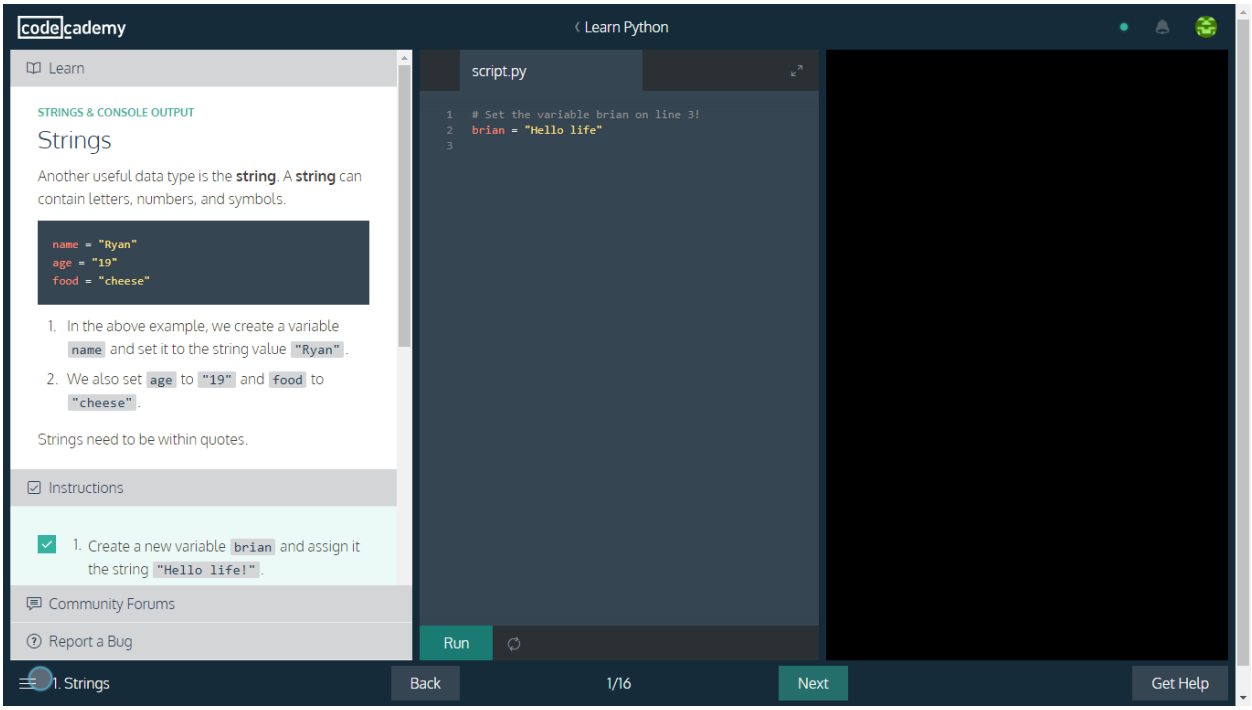
MSSV: 1560278

TÊN: LÊ HUY KHƯƠNG

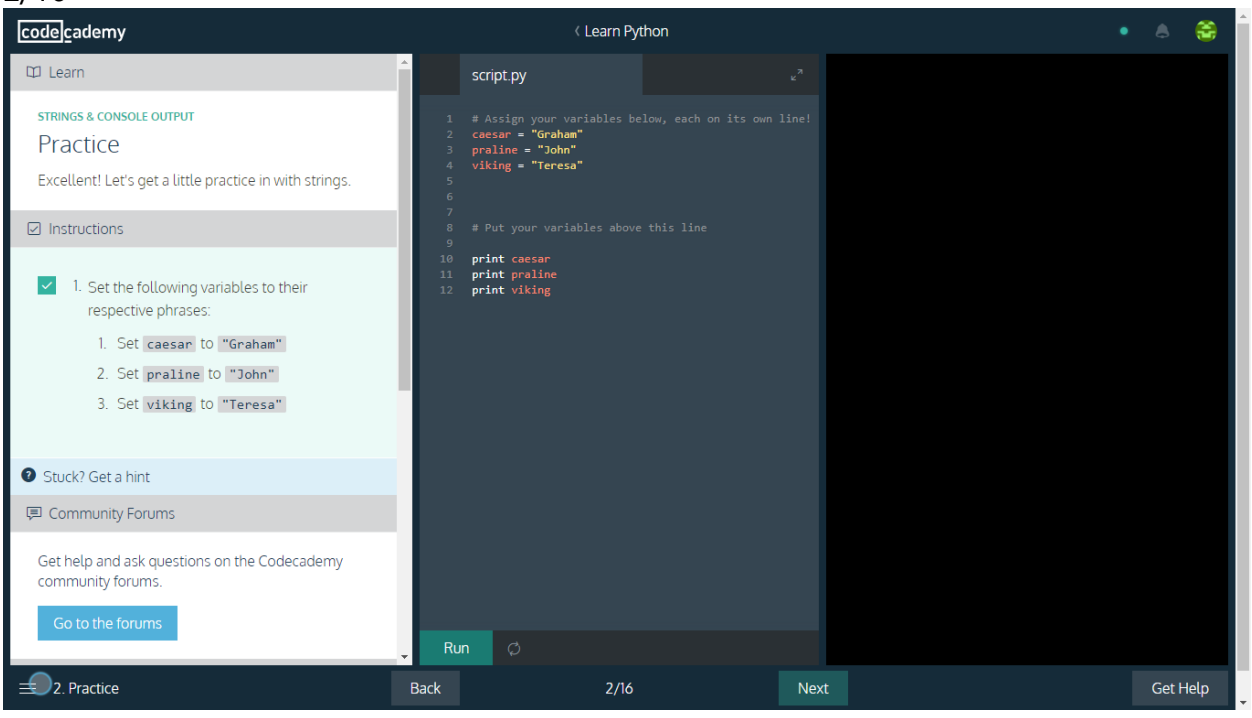
REPORT

2 Strings and Console Output

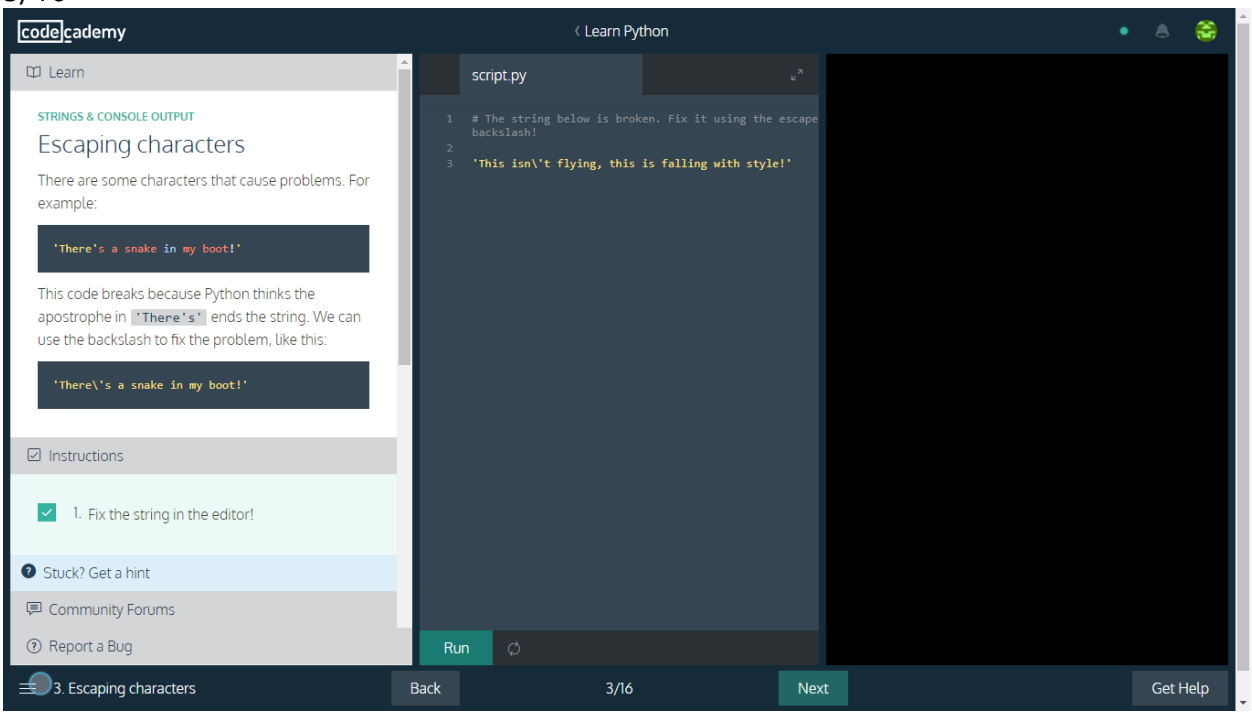
1/16



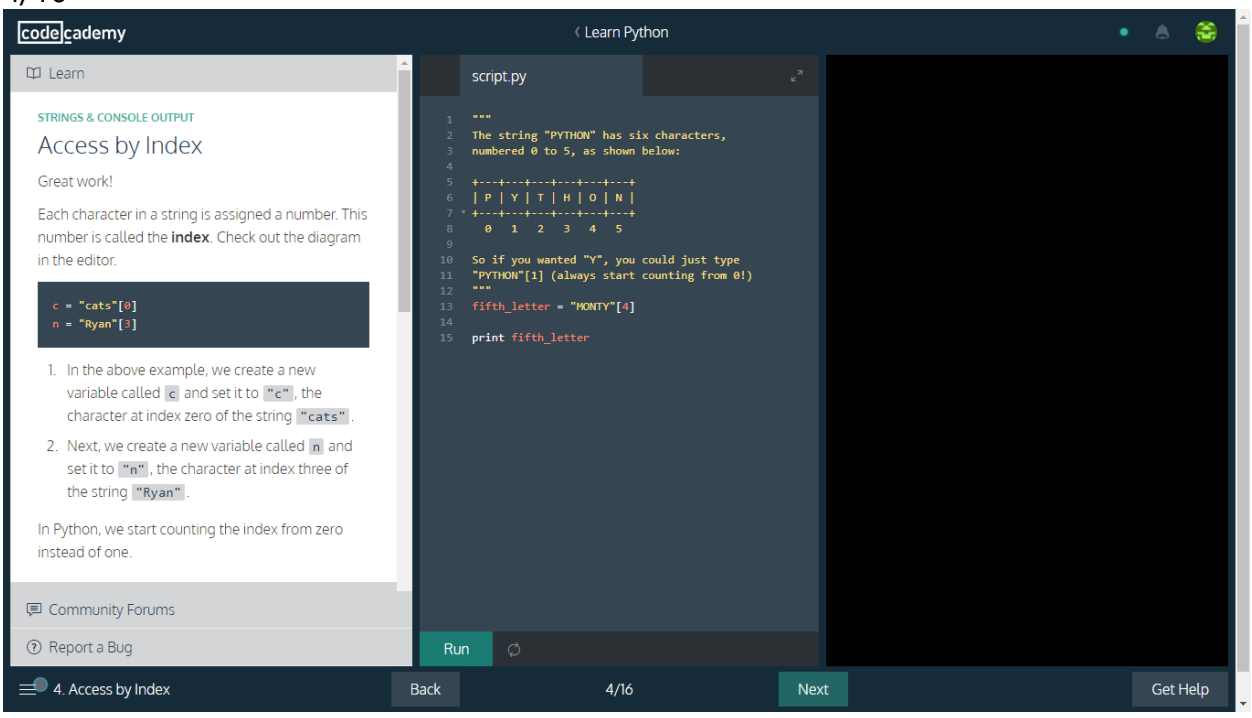
2/16



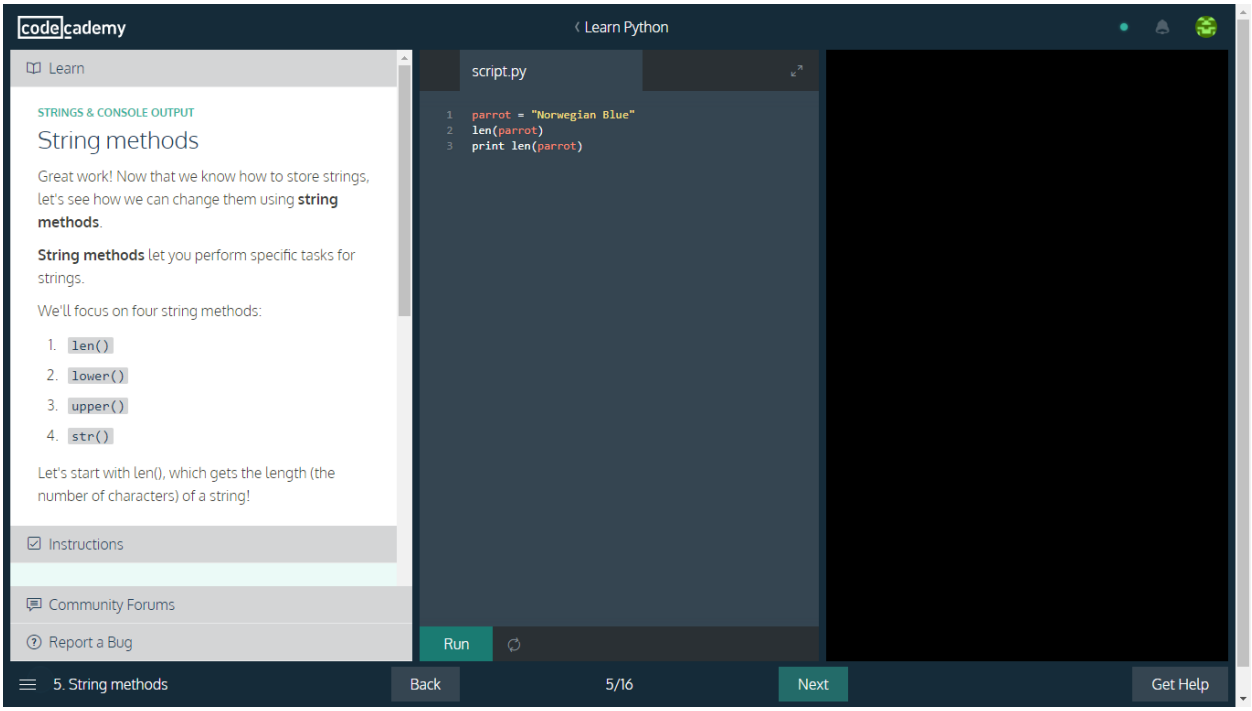
3/16


codecademy
Learn Python
Learn
STRINGS & CONSOLE OUTPUT
Escaping characters
There are some characters that cause problems. For example:
'There's a snake in my boot!'
This code breaks because Python thinks the apostrophe in 'There's' ends the string. We can use the backslash to fix the problem, like this:
'There\'s a snake in my boot!'
Instructions
1. Fix the string in the editor!
Stuck? Get a hint
Community Forums
Report a Bug
script.py
The string below is broken. Fix it using the escape backslash!
'This isn\'t flying, this is falling with style!'
Run
3. Escaping characters
Back
3/16
Next
Get Help

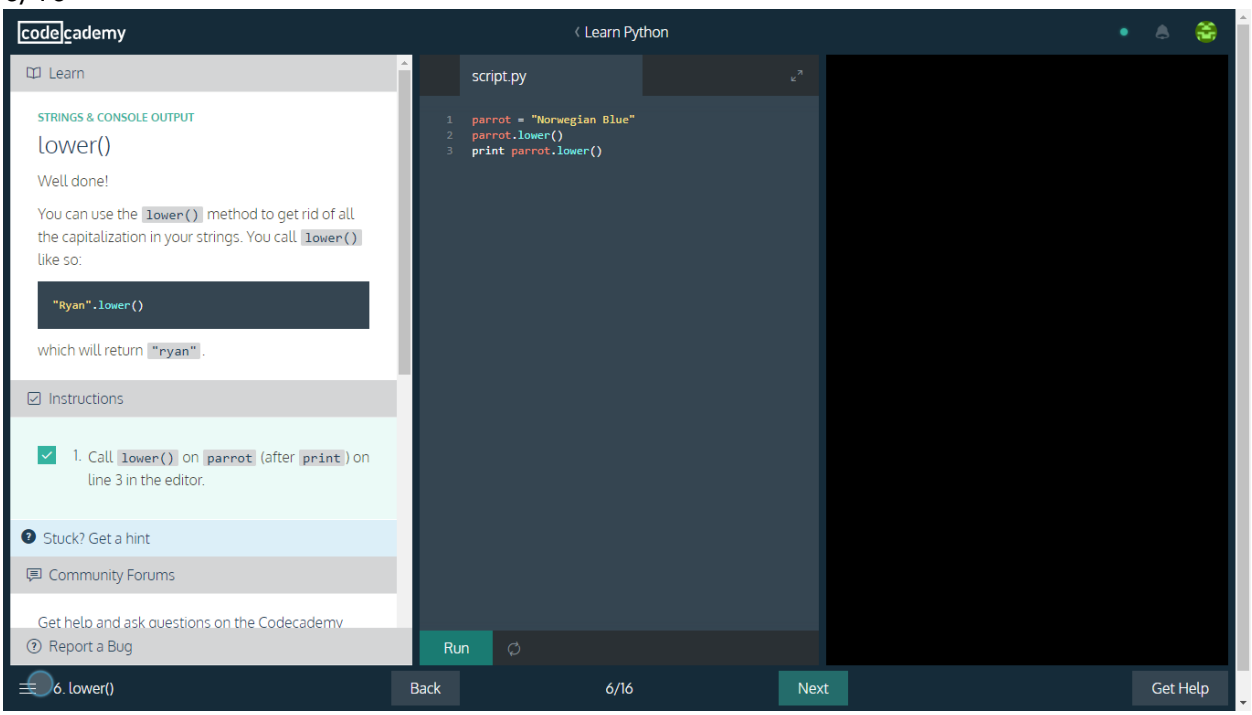
4/16


codecademy
Learn Python
Learn
STRINGS & CONSOLE OUTPUT
Access by Index
Great work!
Each character in a string is assigned a number. This number is called the index. Check out the diagram in the editor.
c = "cats"[0]
n = "Ryan"[3]
1. In the above example, we create a new variable called c and set it to "c", the character at index zero of the string "cats".
2. Next, we create a new variable called n and set it to "n", the character at index three of the string "Ryan".
In Python, we start counting the index from zero instead of one.
Community Forums
Report a Bug
script.py
"""
The string "PYTHON" has six characters,
numbered 0 to 5, as shown below:
+---+---+---+---+---+---+
| P | Y | T | H | O | N |
+---+---+---+---+---+---+
0 1 2 3 4 5
So if you wanted "Y", you could just type
"PYTHON"[1] (always start counting from 0!)
"""
fifth_letter = "MONTY"[4]
print fifth_letter
Run
4. Access by Index
Back
4/16
Next
Get Help

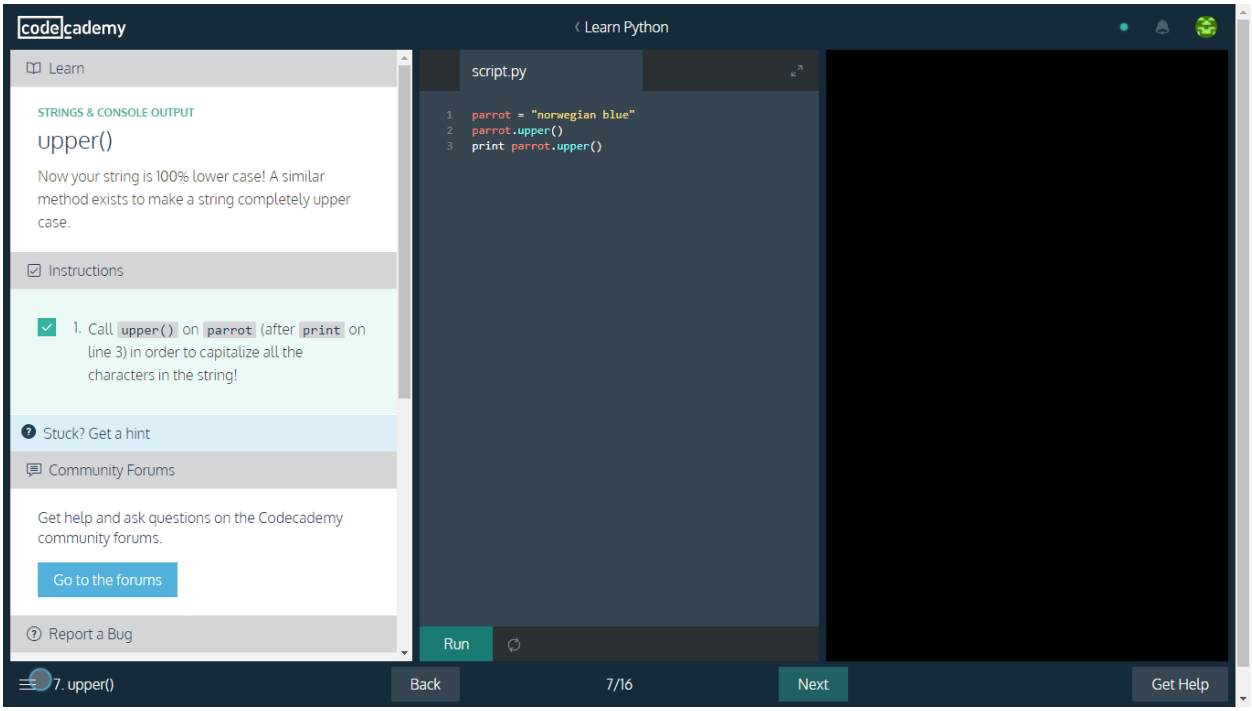
5/16


codecademy
Learn Python
Learn
STRINGS & CONSOLE OUTPUT
String methods
Great work! Now that we know how to store strings, let's see how we can change them using string methods.
String methods let you perform specific tasks for strings.
We'll focus on four string methods:
1. len()
2. lower()
3. upper()
4. str()
Let's start with len(), which gets the length (the number of characters) of a string!
Instructions
Community Forums
Report a Bug
script.py
parrot = "Norwegian Blue"
len(parrot)
print len(parrot)
Run
5. String methods
Back
5/16
Next
Get Help

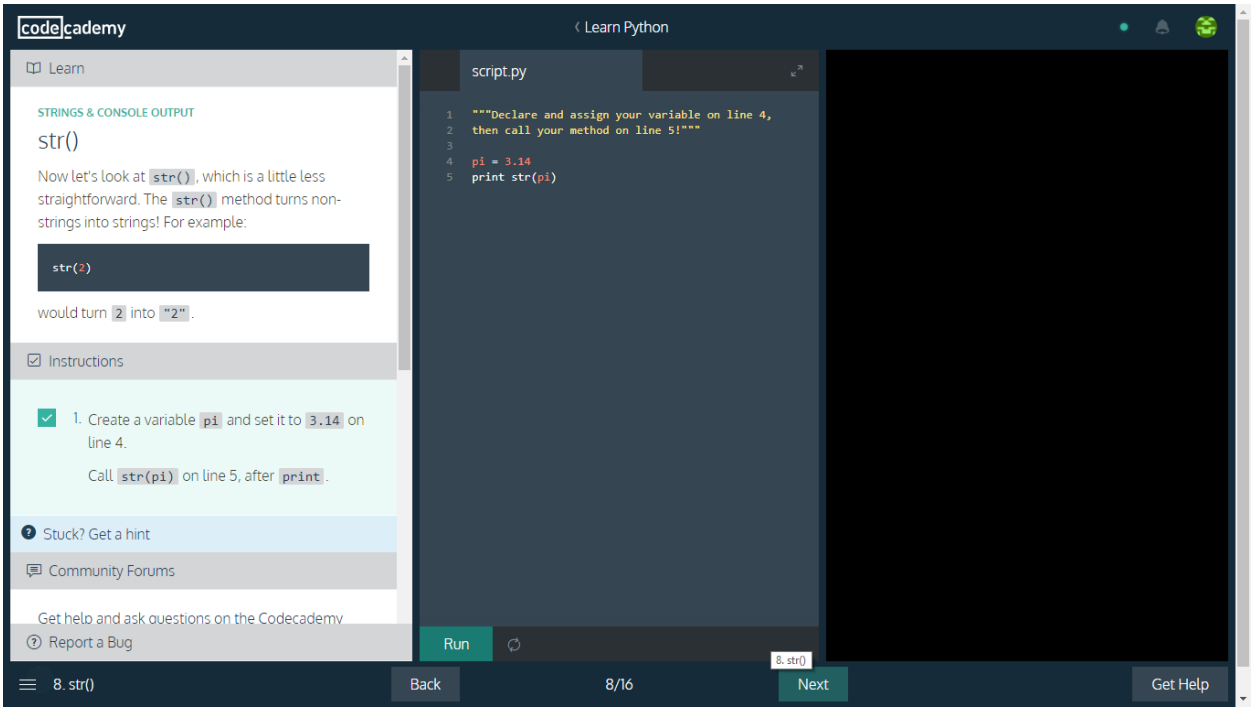
6/16


codecademy
Learn Python
Learn
STRINGS & CONSOLE OUTPUT
lower()
Well done!
You can use the lower() method to get rid of all the capitalization in your strings. You call lower() like so:
"Ryan".lower()
which will return "ryan".
Instructions
1. Call lower() on parrot (after print) on line 3 in the editor.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy
Report a Bug
script.py
parrot = "Norwegian Blue"
parrot.lower()
print parrot.lower()
Run
6. lower()
Back
6/16
Next
Get Help

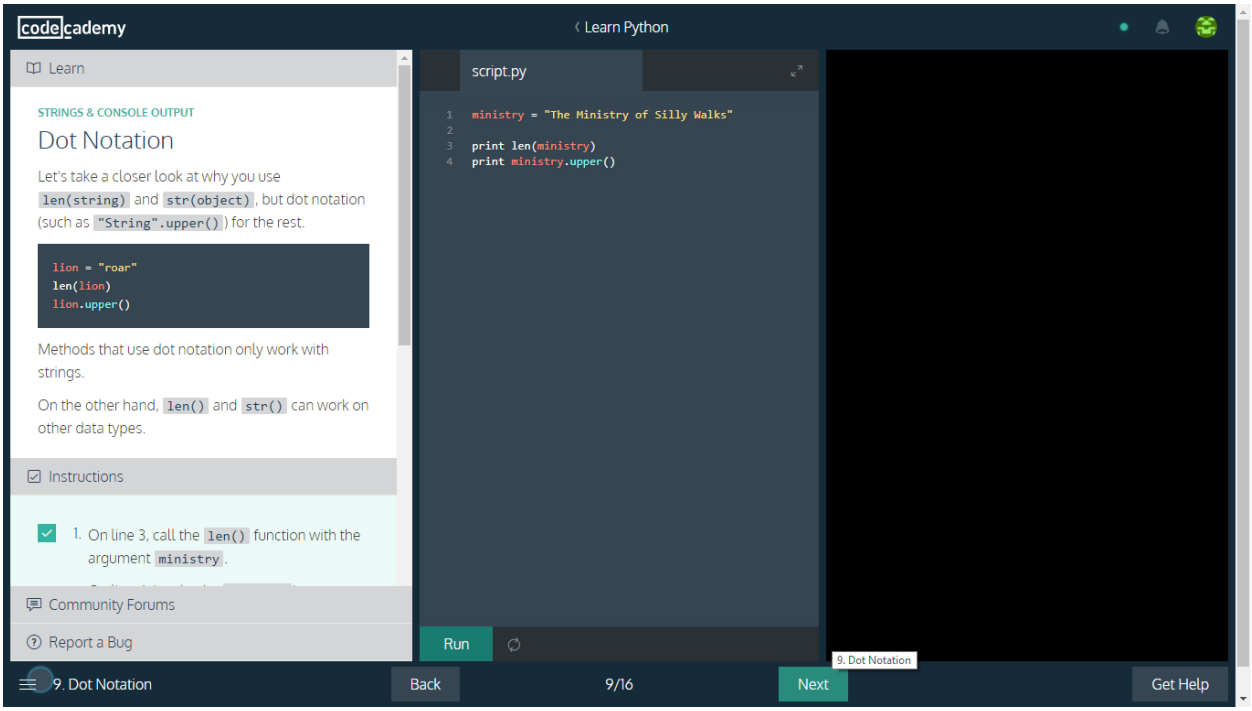
7/16


codecademy
Learn Python
Learn
STRINGS & CONSOLE OUTPUT
upper()
Now your string is 100% lower case! A similar method exists to make a string completely upper case.
Instructions
1. Call upper() on parrot (after print on line 3) in order to capitalize all the characters in the string!
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
Report a Bug
script.py
parrot = "norwegian blue"
parrot.upper()
print parrot.upper()
Run
7. upper()
Back
7/16
Next
Get Help

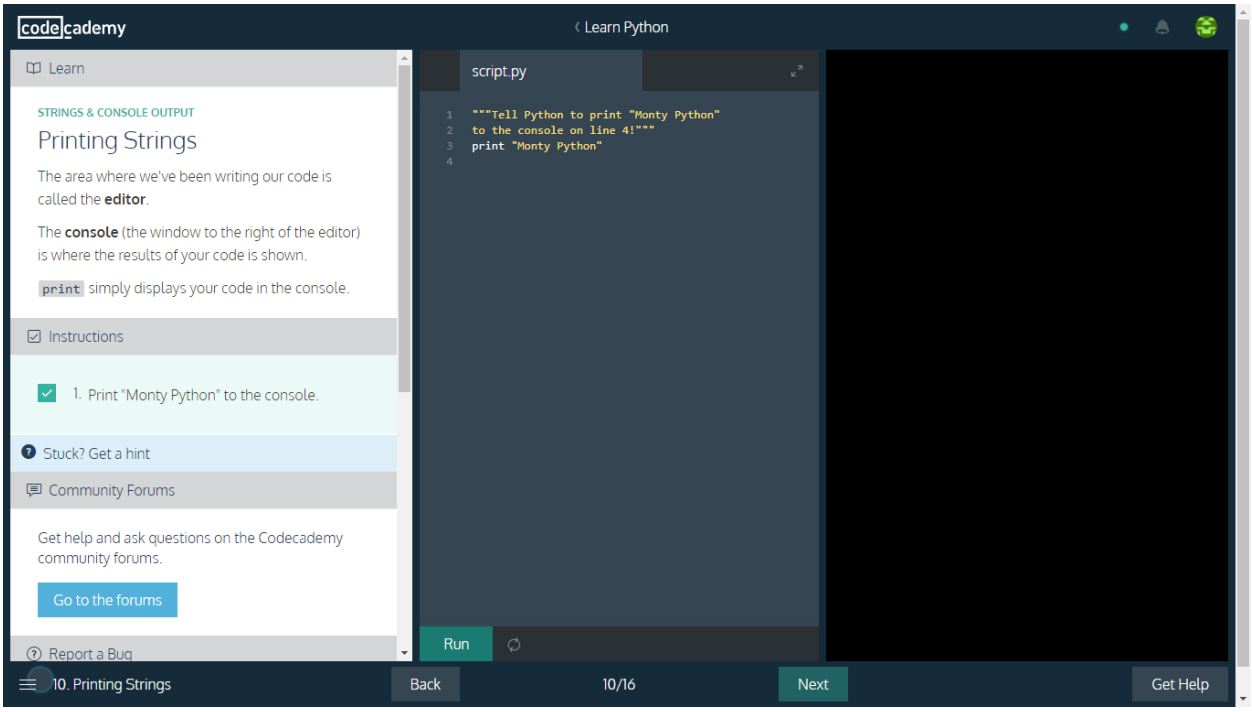
8/16


codecademy
Learn Python
Learn
STRINGS & CONSOLE OUTPUT
str()
Now let's look at str(), which is a little less straightforward. The str() method turns non-strings into strings! For example:
str(2)
would turn 2 into "2".
Instructions
1. Create a variable pi and set it to 3.14 on line 4.
Call str(pi) on line 5, after print.
Stuck? Get a hint
Community Forums
Report a Bug
script.py
"""Declare and assign your variable on line 4,
then call your method on line 5!"""
pi = 3.14
print str(pi)
Run
8. str()
Back
8/16
Next
Get Help

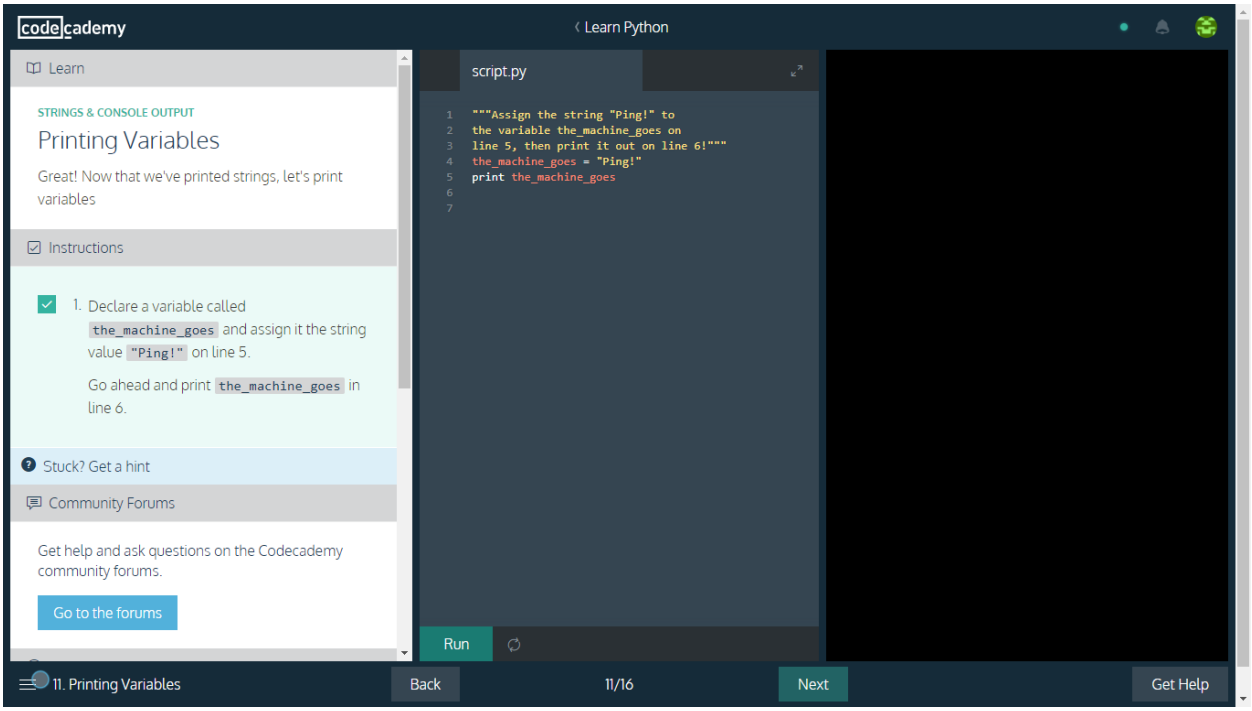
9/16


codecademy
Learn Python
Learn
STRINGS & CONSOLE OUTPUT
Dot Notation
Let's take a closer look at why you use len(string) and str(object), but dot notation (such as "String".upper()) for the rest.
lion = "roar"
len(lion)
lion.upper()
Methods that use dot notation only work with strings.
On the other hand, len() and str() can work on other data types.
Instructions
1. On line 3, call the len() function with the argument ministry.
Community Forums
Report a Bug
script.py
ministry = "The Ministry of Silly Walks"
print len(ministry)
print ministry.upper()
Run
9. Dot Notation
9. Dot Notation
Back
9/16
Next
Get Help

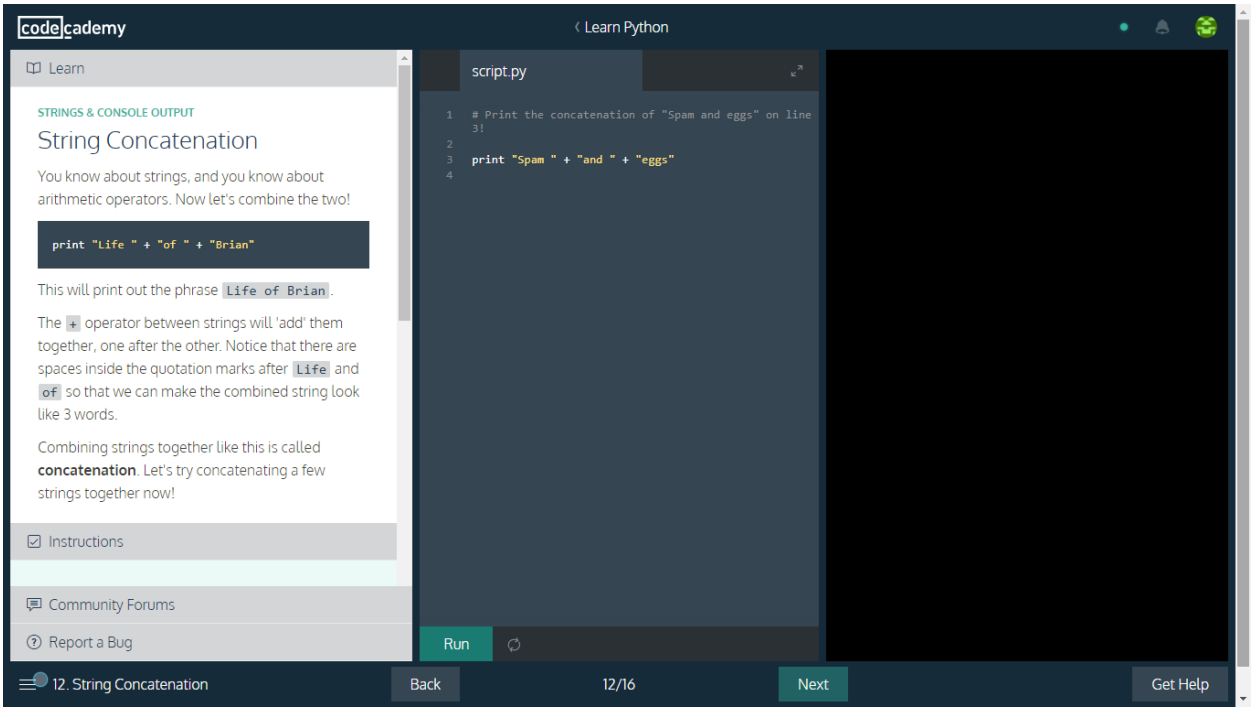
10/16


codecademy
Learn Python
Learn
STRINGS & CONSOLE OUTPUT
Printing Strings
The area where we've been writing our code is called the editor.
The console (the window to the right of the editor) is where the results of your code is shown.
print simply displays your code in the console.
Instructions
1. Print "Monty Python" to the console.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
Report a Bug
script.py
"""Tell Python to print "Monty Python"
to the console on line 4!"""
print "Monty Python"
Run
10. Printing Strings
Back
10/16
Next
Get Help

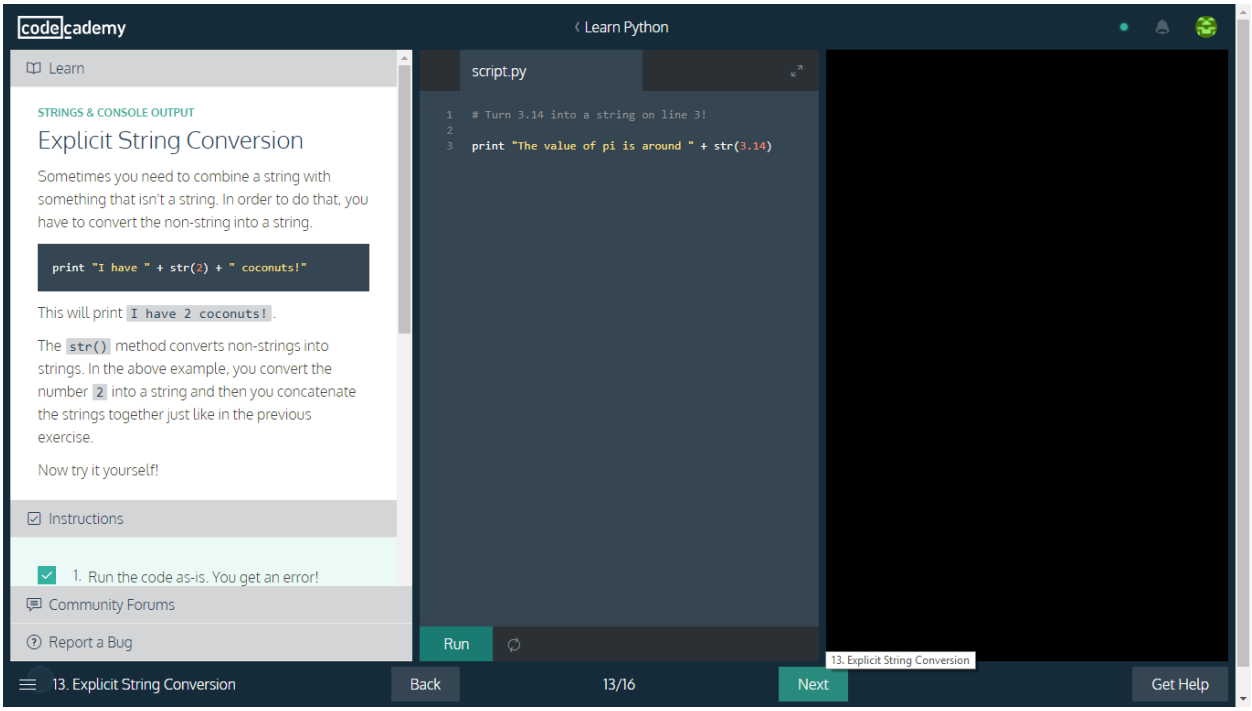
11/16


codecademy
Learn Python
Learn
STRINGS & CONSOLE OUTPUT
Printing Variables
Great! Now that we've printed strings, let's print variables
Instructions
1. Declare a variable called the_machine_goes and assign it the string value "Ping!" on line 5.
Go ahead and print the_machine_goes in line 6.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
script.py
"""Assign the string "Ping!" to the variable the_machine_goes on line 5, then print it out on line 6!"""
the_machine_goes = "Ping!"
print the_machine_goes
Run
11. Printing Variables
Back
11/16
Next
Get Help

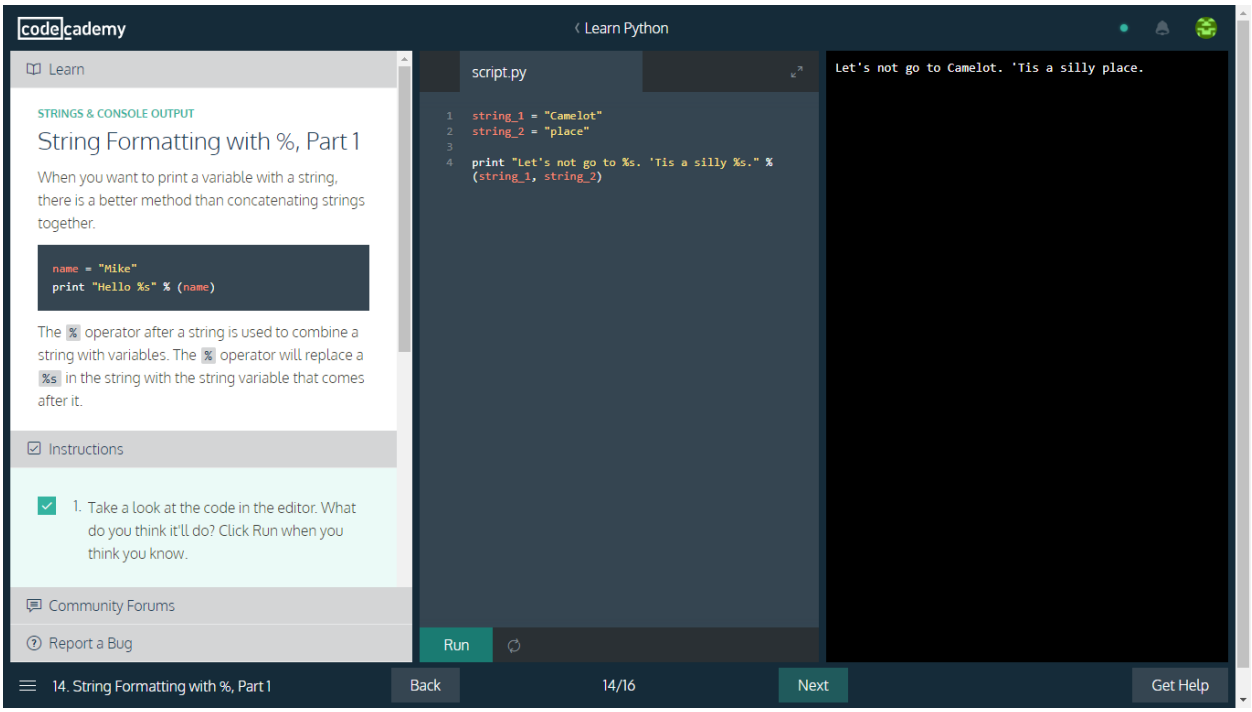
12/16


codecademy
Learn Python
Learn
STRINGS & CONSOLE OUTPUT
String Concatenation
You know about strings, and you know about arithmetic operators. Now let's combine the two!
print "Life " + "of " + "Brian"
This will print out the phrase Life of Brian.
The + operator between strings will 'add' them together, one after the other. Notice that there are spaces inside the quotation marks after Life and of so that we can make the combined string look like 3 words.
Combining strings together like this is called concatenation. Let's try concatenating a few strings together now!
Instructions
Community Forums
Report a Bug
script.py
Print the concatenation of "Spam and eggs" on line 3!
print "Spam " + "and " + "eggs"
Run
12. String Concatenation
Back
12/16
Next
Get Help

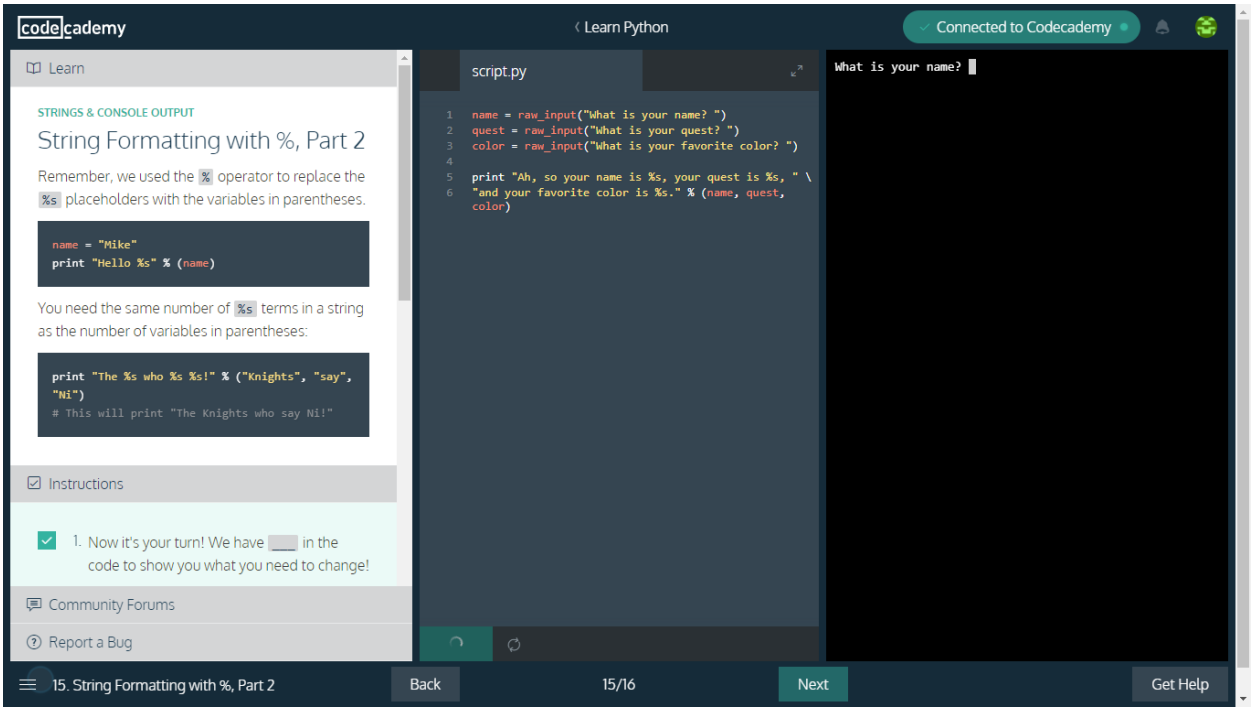
13/16


codecademy
Learn Python
Learn
STRINGS & CONSOLE OUTPUT
Explicit String Conversion
Sometimes you need to combine a string with something that isn't a string. In order to do that, you have to convert the non-string into a string.
print "I have " + str(2) + " coconuts!"
This will print I have 2 coconuts!.
The str() method converts non-strings into strings. In the above example, you convert the number 2 into a string and then you concatenate the strings together just like in the previous exercise.
Now try it yourself!
Instructions
1. Run the code as-is. You get an error!
Community Forums
Report a Bug
script.py
Turn 3.14 into a string on line 3!
print "The value of pi is around " + str(3.14)
Run
13. Explicit String Conversion
13. Explicit String Conversion
Back
13/16
Next
Get Help

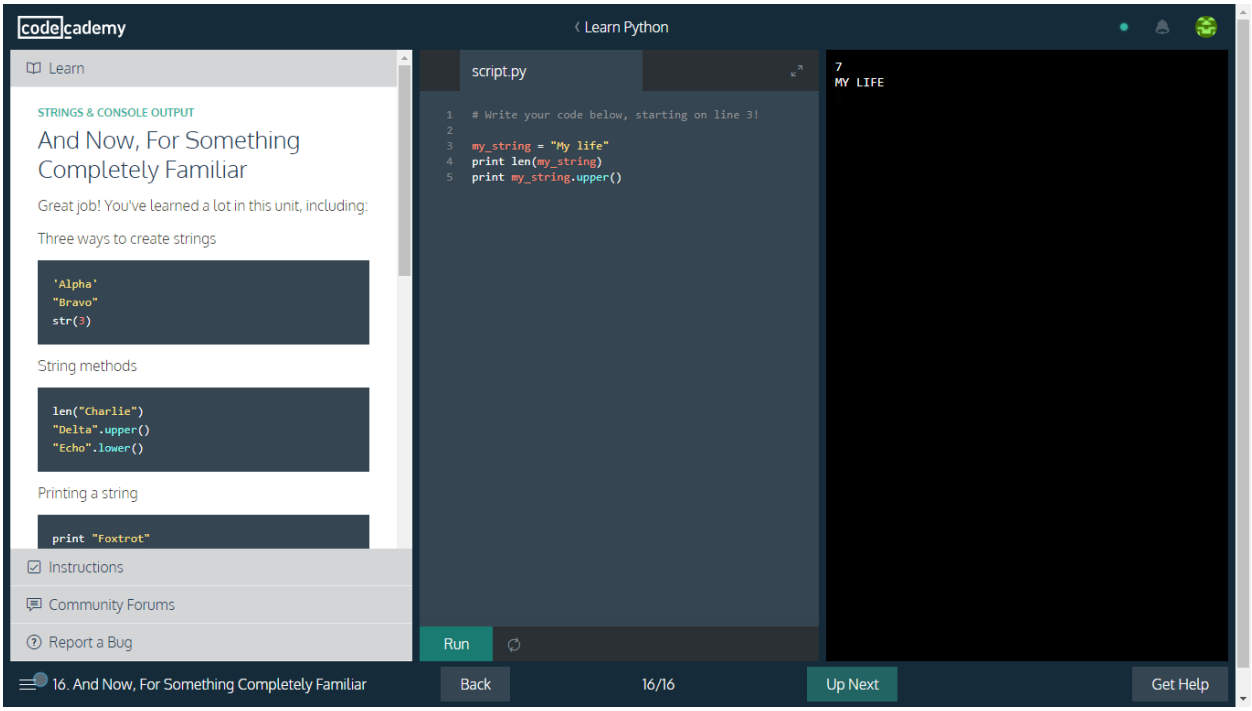
14/16


codecademy
Learn Python
Learn
STRINGS & CONSOLE OUTPUT
String Formatting with %, Part 1
When you want to print a variable with a string, there is a better method than concatenating strings together.
name = "Mike"
print "Hello %s" % (name)
The % operator after a string is used to combine a string with variables. The % operator will replace a %s in the string with the string variable that comes after it.
Instructions
1. Take a look at the code in the editor. What do you think it'll do? Click Run when you think you know.
Community Forums
Report a Bug
script.py
string_1 = "Camelot"
string_2 = "place"
print "Let's not go to %s. 'Tis a silly %s." % (string_1, string_2)
Run
Let's not go to Camelot. 'Tis a silly place.
14. String Formatting with %, Part 1
Back
14/16
Next
Get Help

15/16



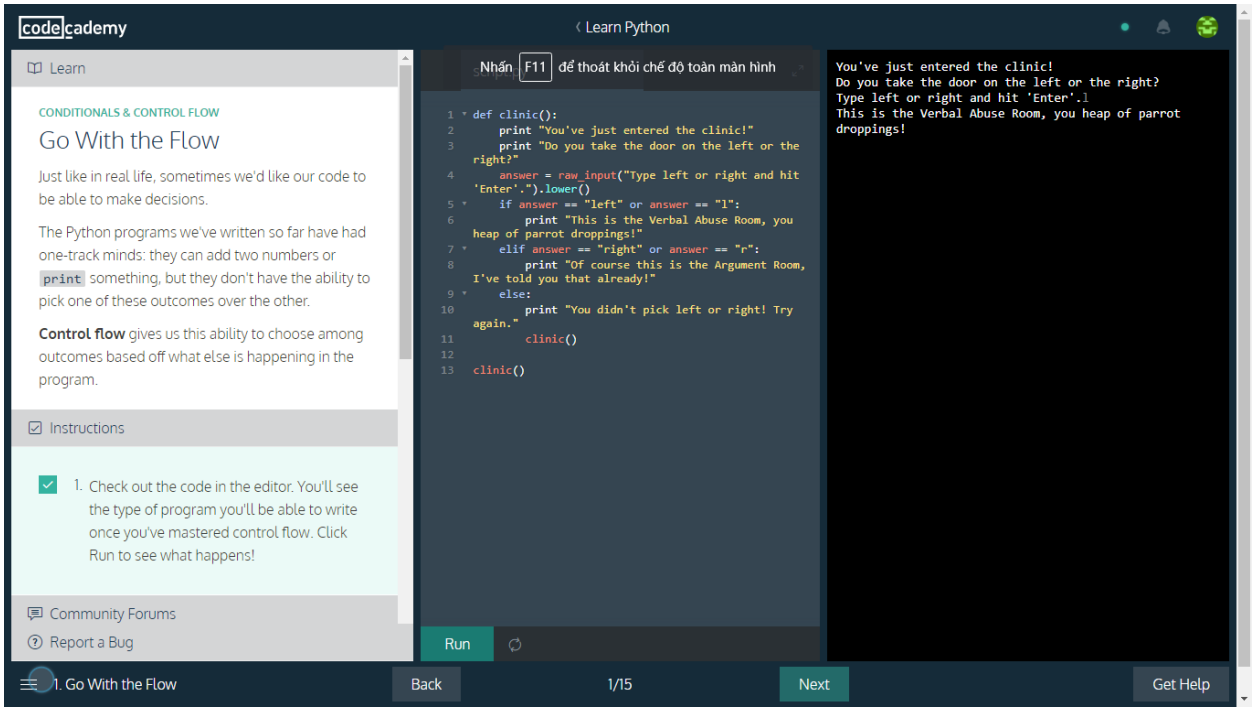
16/16



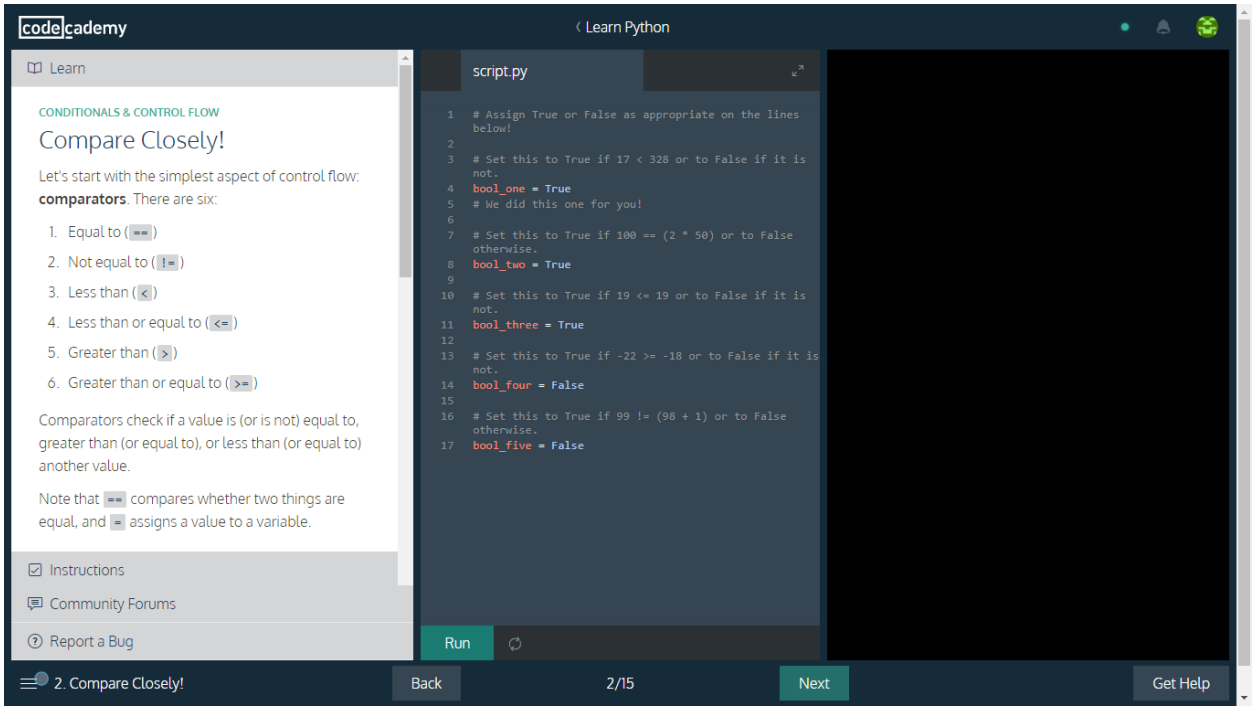
3 Conditionals and Control Flow

A. Conditionals and Control Flow

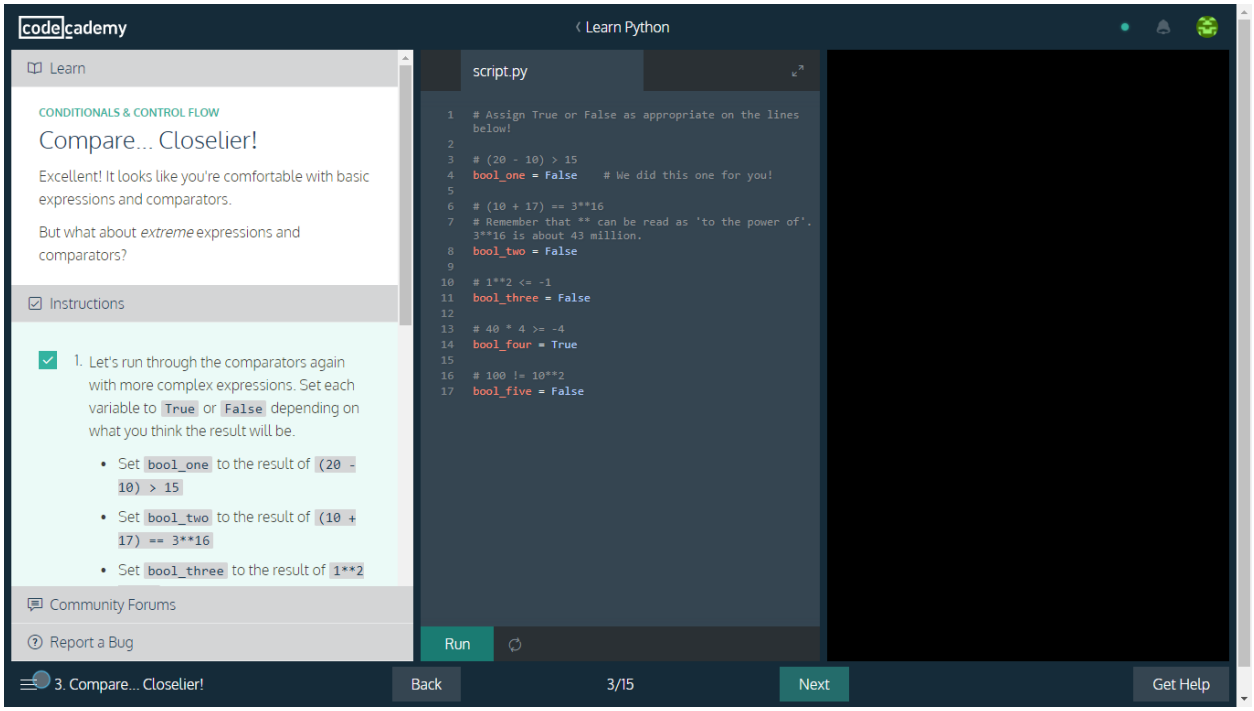
1/15


codecademy
Learn Python
Learn
CONDITIONALS & CONTROL FLOW
Go With the Flow
Just like in real life, sometimes we'd like our code to be able to make decisions.
The Python programs we've written so far have had one-track minds: they can add two numbers or print something, but they don't have the ability to pick one of these outcomes over the other.
Control flow gives us this ability to choose among outcomes based off what else is happening in the program.
Instructions
1. Check out the code in the editor. You'll see the type of program you'll be able to write once you've mastered control flow. Click Run to see what happens!
Community Forums
Report a Bug
Nhấn F11 để thoát khỏi chế độ toàn màn hình
def clinic():
print "You've just entered the clinic!"
print "Do you take the door on the left or the right?"
answer = raw_input("Type left or right and hit 'Enter'.").lower()
if answer == "left" or answer == "l":
print "This is the Verbal Abuse Room, you heap of parrot droppings!"
elif answer == "right" or answer == "r":
print "Of course this is the Argument Room, I've told you that already!"
else:
print "You didn't pick left or right! Try again."
clinic()
clinic()
Run
You've just entered the clinic!
Do you take the door on the left or the right?
Type left or right and hit 'Enter'.l
This is the Verbal Abuse Room, you heap of parrot droppings!
1. Go With the Flow
Back
1/15
Next
Get Help

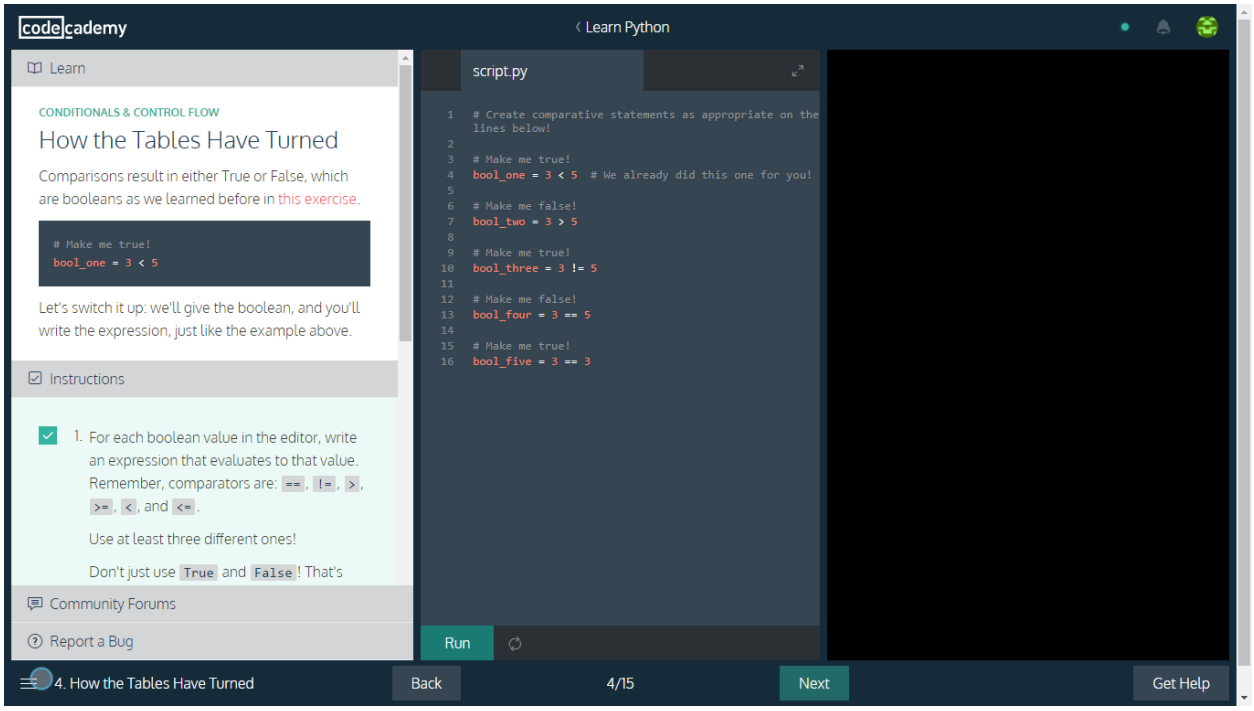
2/15


codecademy
Learn Python
Learn
CONDITIONALS & CONTROL FLOW
Compare Closely!
Let's start with the simplest aspect of control flow: comparators. There are six:
1. Equal to (==)
2. Not equal to (!=)
3. Less than (<)
4. Less than or equal to (<=)
5. Greater than (>)
6. Greater than or equal to (>=)
Comparators check if a value is (or is not) equal to, greater than (or equal to), or less than (or equal to) another value.
Note that == compares whether two things are equal, and = assigns a value to a variable.
Instructions
Community Forums
Report a Bug
script.py
# Assign True or False as appropriate on the lines below!
# Set this to True if 17 < 328 or to False if it is not.
bool_one = True
# We did this one for you!
# Set this to True if 100 == (2 * 50) or to False otherwise.
bool_two = True
# Set this to True if 19 <= 19 or to False if it is not.
bool_three = True
# Set this to True if -22 >= -18 or to False if it is not.
bool_four = False
# Set this to True if 99 != (98 + 1) or to False otherwise.
bool_five = False
Run
2. Compare Closely!
Back
2/15
Next
Get Help

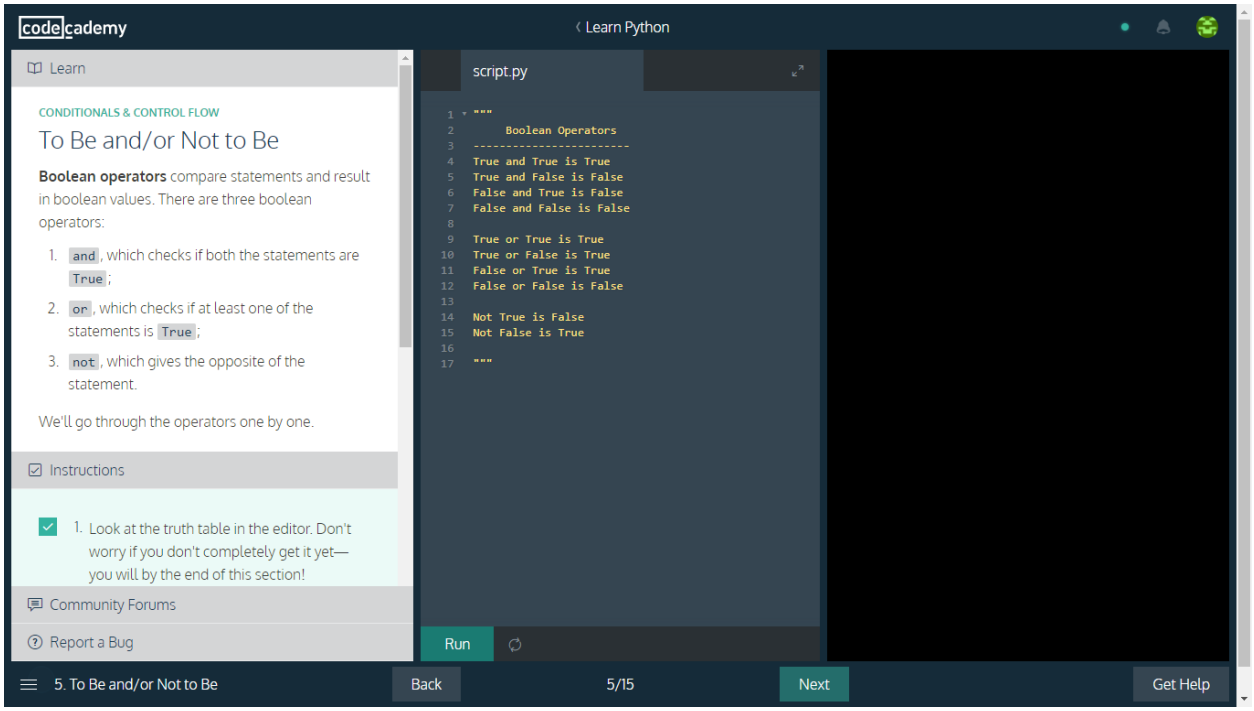
3/15


codecademy
Learn Python
Learn
CONDITIONALS & CONTROL FLOW
Compare... Closelier!
Excellent! It looks like you're comfortable with basic expressions and comparators.
But what about extreme expressions and comparators?
Instructions
1. Let's run through the comparators again with more complex expressions. Set each variable to True or False depending on what you think the result will be.
• Set bool_one to the result of (20 - 10) > 15
• Set bool_two to the result of (10 + 17) == 3**16
• Set bool_three to the result of 1**2
Community Forums
Report a Bug
script.py
# Assign True or False as appropriate on the lines below!
# (20 - 10) > 15
bool_one = False # We did this one for you!
# (10 + 17) == 3**16
# Remember that ** can be read as 'to the power of'. 3**16 is about 43 million.
bool_two = False
# 1**2 <= -1
bool_three = False
# 40 * 4 >= -4
bool_four = True
# 100 != 10**2
bool_five = False
Run
3. Compare... Closelier!
Back
3/15
Next
Get Help

4/15

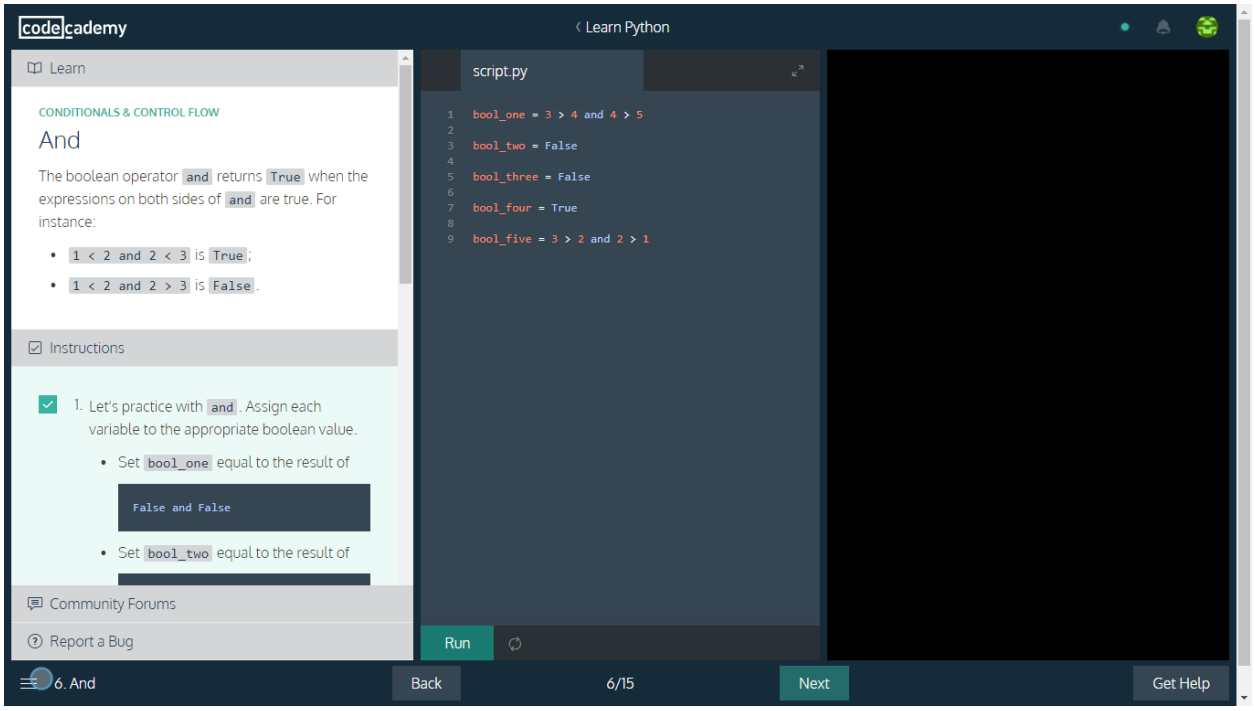

codecademy
Learn Python
Learn
CONDITIONALS & CONTROL FLOW
How the Tables Have Turned
Comparisons result in either True or False, which are booleans as we learned before in this exercise.
# Make me true!
bool_one = 3 < 5
Let's switch it up: we'll give the boolean, and you'll write the expression, just like the example above.
Instructions
1. For each boolean value in the editor, write an expression that evaluates to that value. Remember, comparators are: ==, !=, >, >=, <, and <=.
Use at least three different ones!
Don't just use True and False! That's
Community Forums
Report a Bug
script.py
# Create comparative statements as appropriate on the lines below!
# Make me true!
bool_one = 3 < 5 # We already did this one for you!
# Make me false!
bool_two = 3 > 5
# Make me true!
bool_three = 3 != 5
# Make me false!
bool_four = 3 == 5
# Make me true!
bool_five = 3 == 3
Run
4. How the Tables Have Turned
Back
4/15
Next
Get Help

5/15

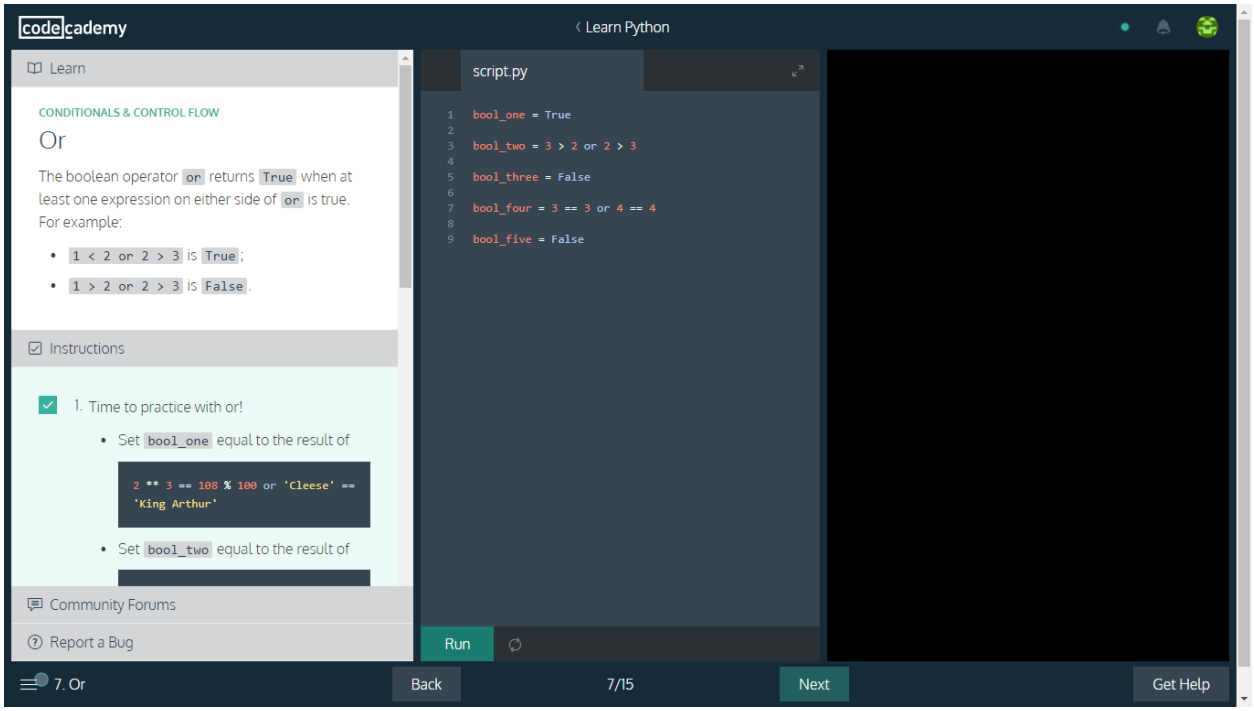

codecademy
Learn Python
Learn
CONDITIONALS & CONTROL FLOW
To Be and/or Not to Be
Boolean operators compare statements and result in boolean values. There are three boolean operators:
1. and, which checks if both the statements are True;
2. or, which checks if at least one of the statements is True;
3. not, which gives the opposite of the statement.
We'll go through the operators one by one.
Instructions
1. Look at the truth table in the editor. Don't worry if you don't completely get it yet—you will by the end of this section!
Community Forums
Report a Bug
script.py
"""
Boolean Operators

True and True is True
True and False is False
False and True is False
False and False is False
True or True is True
True or False is True
False or True is True
False or False is False
Not True is False
Not False is True
"""
Run
5. To Be and/or Not to Be
Back
5/15
Next
Get Help

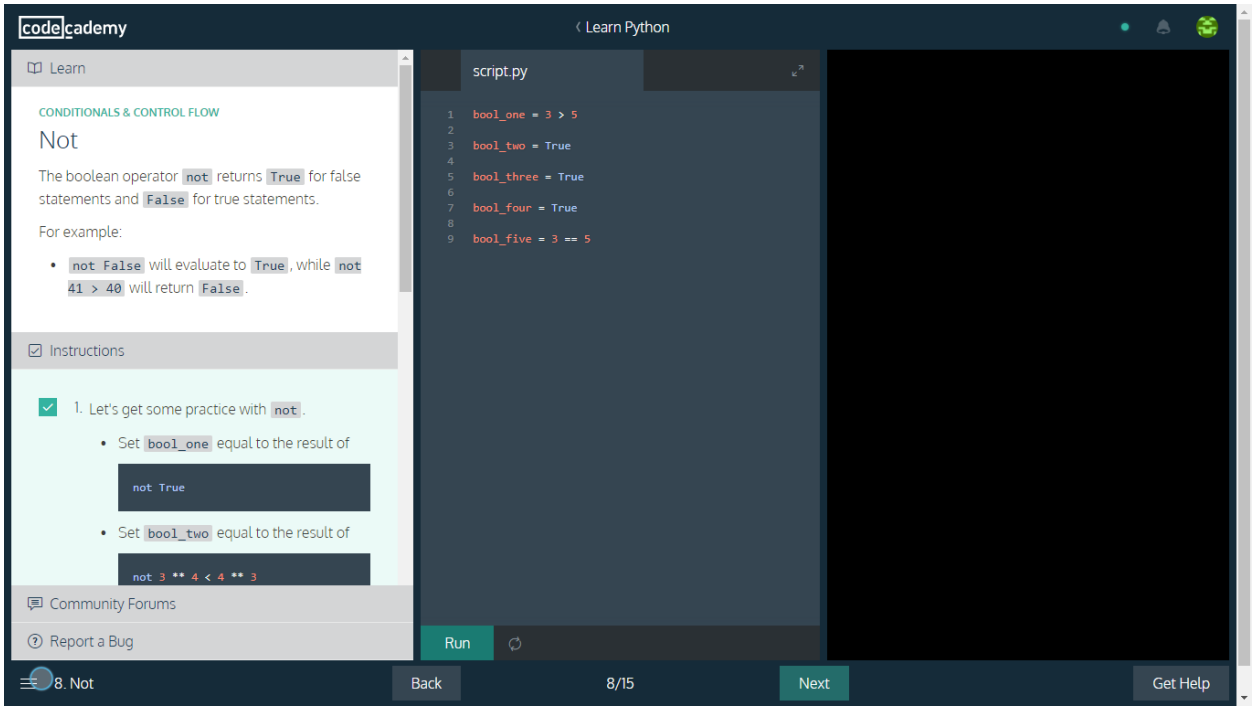
6/15


codecademy
Learn Python
Learn
CONDITIONALS & CONTROL FLOW
And
The boolean operator and returns True when the expressions on both sides of and are true. For instance:
• 1 < 2 and 2 < 3 is True;
• 1 < 2 and 2 > 3 is False.
Instructions
1. Let's practice with and. Assign each variable to the appropriate boolean value.
• Set bool_one equal to the result of
False and False
• Set bool_two equal to the result of
Community Forums
Report a Bug
script.py
bool_one = 3 > 4 and 4 > 5
bool_two = False
bool_three = False
bool_four = True
bool_five = 3 > 2 and 2 > 1
Run
6. And
Back
6/15
Next
Get Help

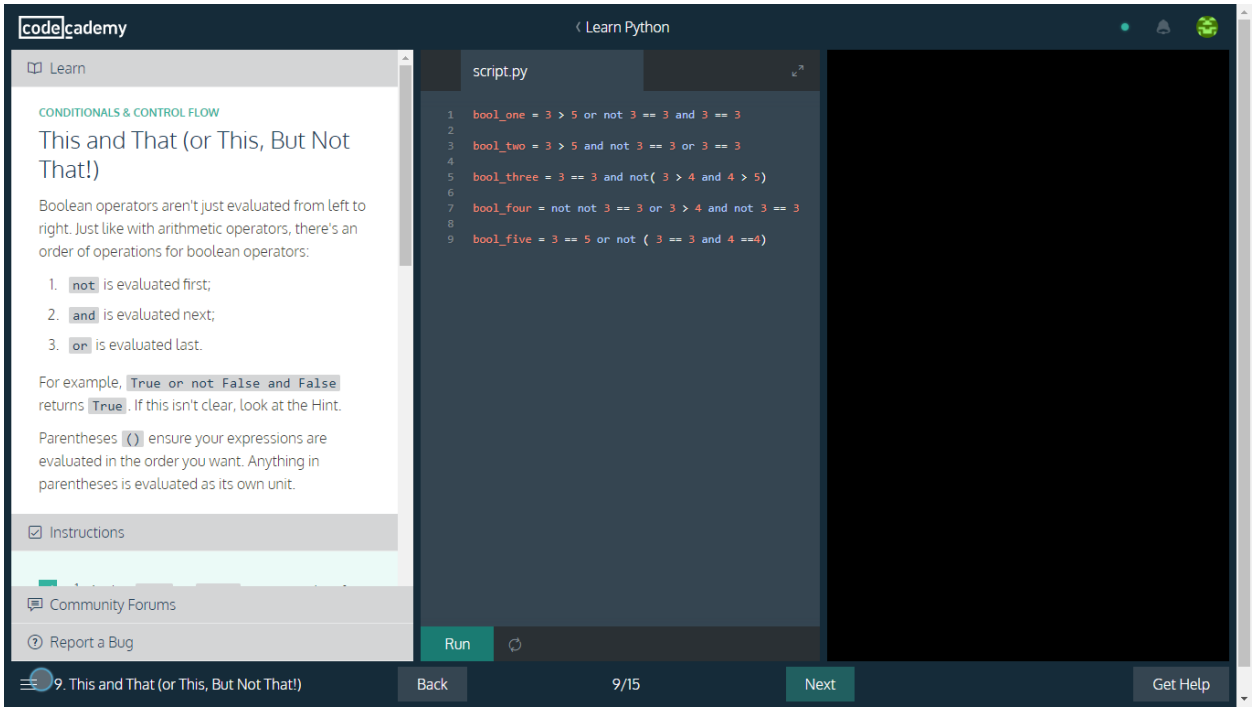
7/15


codecademy
Learn Python
Learn
CONDITIONALS & CONTROL FLOW
Or
The boolean operator or returns True when at least one expression on either side of or is true. For example:
1 < 2 or 2 > 3 is True;
1 > 2 or 2 > 3 is False.
Instructions
1. Time to practice with or!
Set bool_one equal to the result of
2 ** 3 == 108 % 100 or 'Cleese' == 'King Arthur'
Set bool_two equal to the result of
Community Forums
Report a Bug
script.py
bool_one = True
bool_two = 3 > 2 or 2 > 3
bool_three = False
bool_four = 3 == 3 or 4 == 4
bool_five = False
Run
7. Or
Back
7/15
Next
Get Help

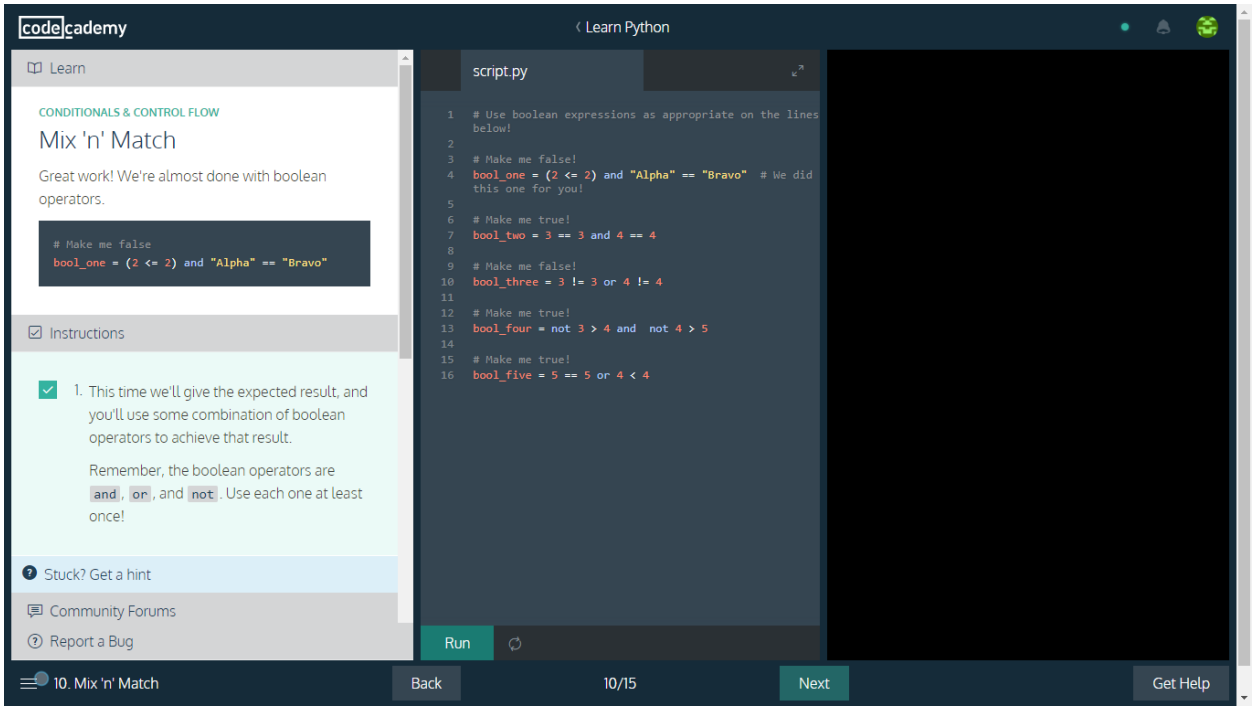
8/15


codecademy
Learn Python
Learn
CONDITIONALS & CONTROL FLOW
Not
The boolean operator not returns True for false statements and False for true statements.
For example:
not False will evaluate to True, while not 41 > 40 will return False.
Instructions
1. Let's get some practice with not.
Set bool_one equal to the result of
not True
Set bool_two equal to the result of
not 3 ** 4 < 4 ** 3
Community Forums
Report a Bug
script.py
bool_one = 3 > 5
bool_two = True
bool_three = True
bool_four = True
bool_five = 3 == 5
Run
8. Not
Back
8/15
Next
Get Help

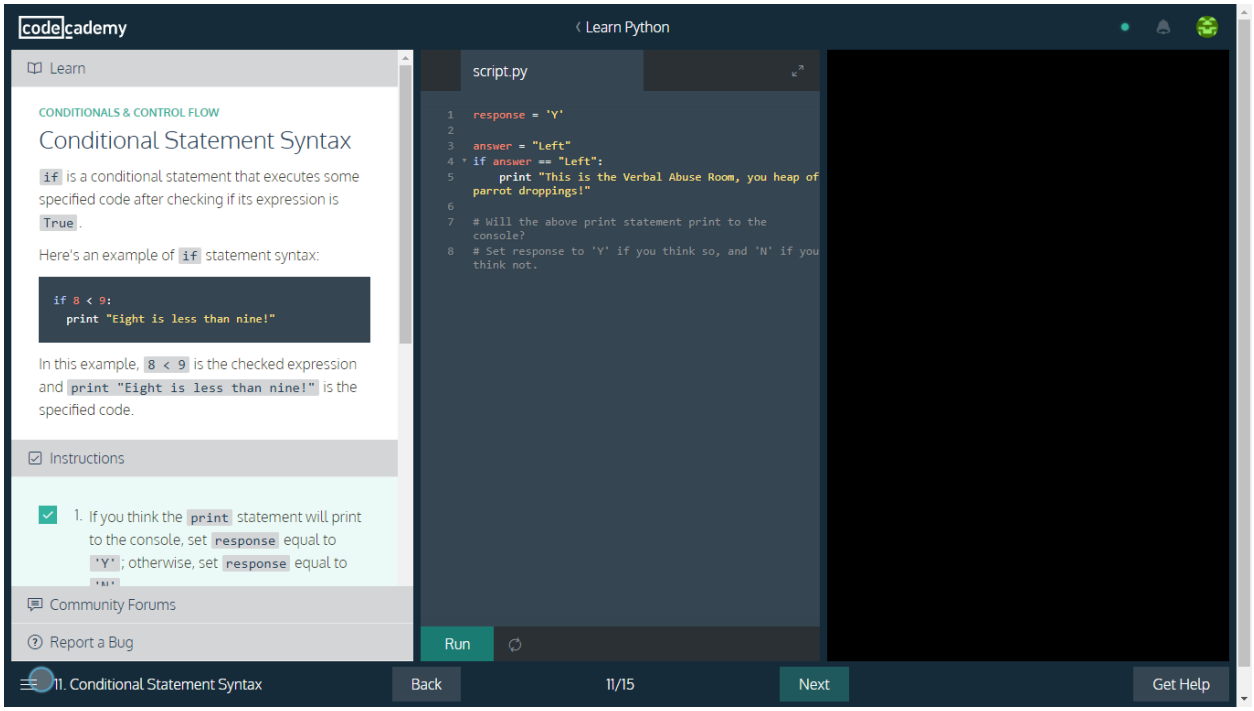
9/15


codecademy
Learn Python
Learn
CONDITIONALS & CONTROL FLOW
This and That (or This, But Not That!)
Boolean operators aren't just evaluated from left to right. Just like with arithmetic operators, there's an order of operations for boolean operators:
1. not is evaluated first;
2. and is evaluated next;
3. or is evaluated last.
For example, True or not False and False returns True. If this isn't clear, look at the Hint.
Parentheses () ensure your expressions are evaluated in the order you want. Anything in parentheses is evaluated as its own unit.
Instructions
Community Forums
Report a Bug
script.py
bool_one = 3 > 5 or not 3 == 3 and 3 == 3
bool_two = 3 > 5 and not 3 == 3 or 3 == 3
bool_three = 3 == 3 and not(3 > 4 and 4 > 5)
bool_four = not not 3 == 3 or 3 > 4 and not 3 == 3
bool_five = 3 == 5 or not (3 == 3 and 4 ==4)
Run
9. This and That (or This, But Not That!)
Back
9/15
Next
Get Help

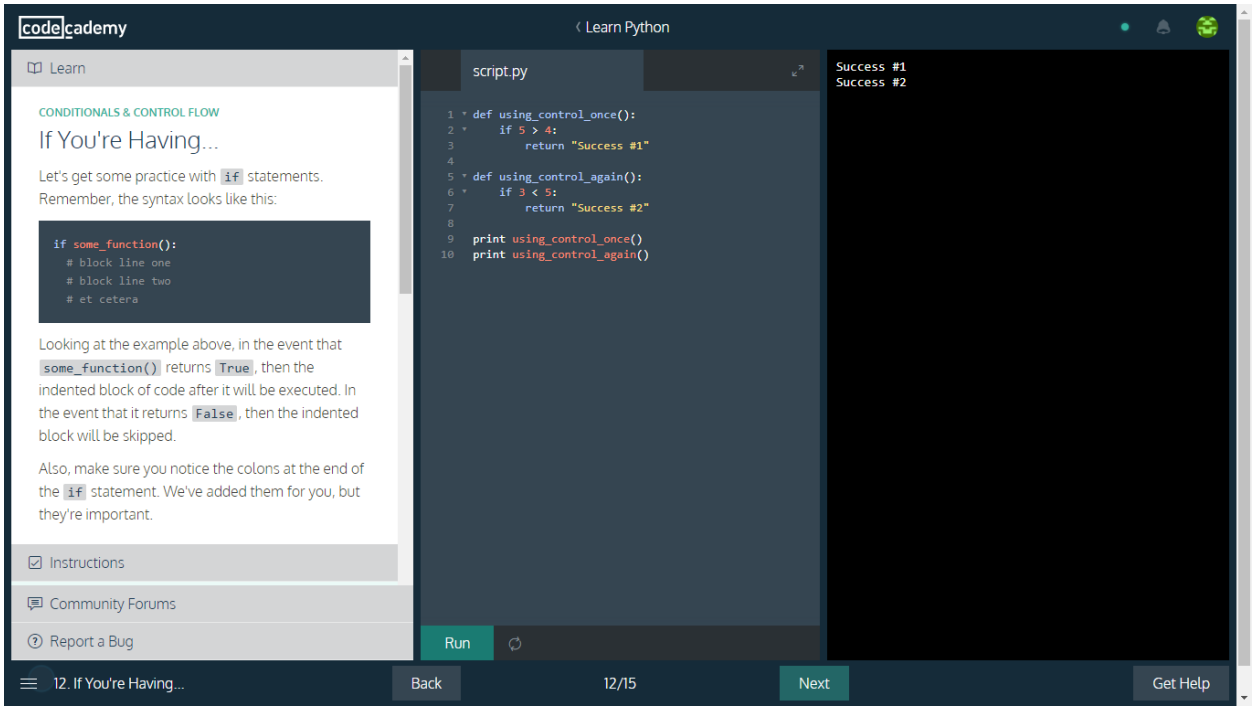
10/15


codecademy
Learn Python
Learn
CONDITIONALS & CONTROL FLOW
Mix 'n' Match
Great work! We're almost done with boolean operators.
# Make me false
bool_one = (2 <= 2) and "Alpha" == "Bravo"
Instructions
1. This time we'll give the expected result, and you'll use some combination of boolean operators to achieve that result.
Remember, the boolean operators are and, or, and not. Use each one at least once!
Stuck? Get a hint
Community Forums
Report a Bug
script.py
# Use boolean expressions as appropriate on the lines below!
# Make me false!
bool_one = (2 <= 2) and "Alpha" == "Bravo" # We did this one for you!
# Make me true!
bool_two = 3 == 3 and 4 == 4
# Make me false!
bool_three = 3 != 3 or 4 != 4
# Make me true!
bool_four = not 3 > 4 and not 4 > 5
# Make me true!
bool_five = 5 == 5 or 4 < 4
Run
10. Mix 'n' Match
Back
10/15
Next
Get Help

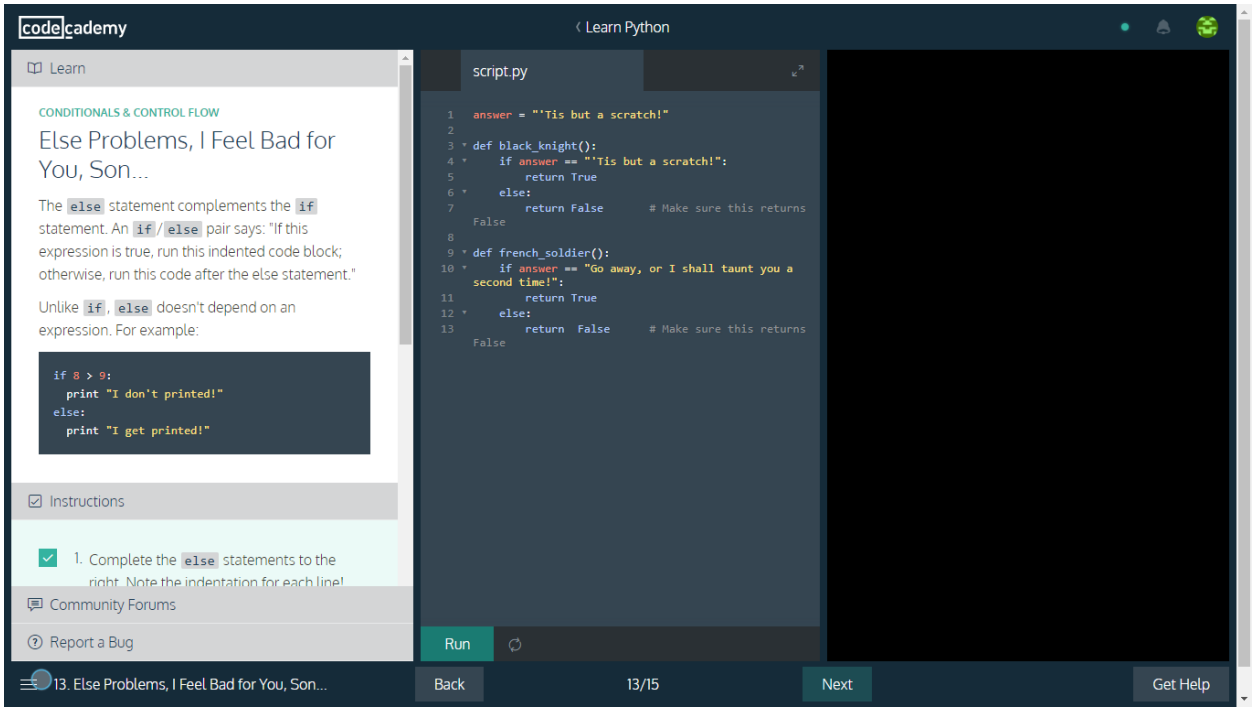
11/15


codecademy
Learn Python
Learn
CONDITIONALS & CONTROL FLOW
Conditional Statement Syntax
if is a conditional statement that executes some specified code after checking if its expression is True.
Here's an example of if statement syntax:
if 8 < 9:
print "Eight is less than nine!"
In this example, 8 < 9 is the checked expression and print "Eight is less than nine!" is the specified code.
Instructions
1. If you think the print statement will print to the console, set response equal to 'Y'; otherwise, set response equal to
Community Forums
Report a Bug
script.py
response = 'Y'
answer = "Left"
if answer == "Left":
print "This is the Verbal Abuse Room, you heap of parrot droppings!"
Will the above print statement print to the console?
Set response to 'Y' if you think so, and 'N' if you think not.
Run
11. Conditional Statement Syntax
Back
11/15
Next
Get Help

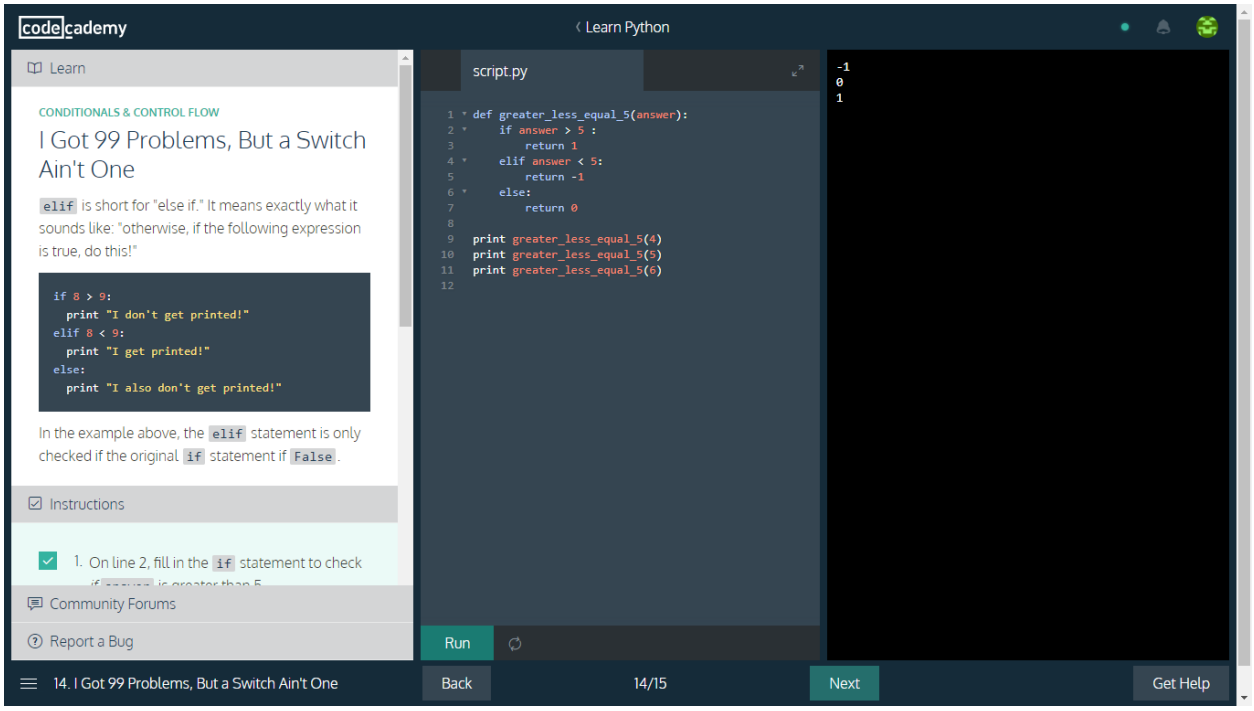
12/15


codecademy
Learn Python
Learn
CONDITIONALS & CONTROL FLOW
If You're Having...
Let's get some practice with if statements. Remember, the syntax looks like this:
if some_function():
# block line one
# block line two
# et cetera
Looking at the example above, in the event that some_function() returns True, then the indented block of code after it will be executed. In the event that it returns False, then the indented block will be skipped.
Also, make sure you notice the colons at the end of the if statement. We've added them for you, but they're important.
Instructions
Community Forums
Report a Bug
script.py
def using_control_once():
if 5 > 4:
return "Success #1"
def using_control_again():
if 3 < 5:
return "Success #2"
print using_control_once()
print using_control_again()
Success #1
Success #2
Run
12. If You're Having...
Back
12/15
Next
Get Help

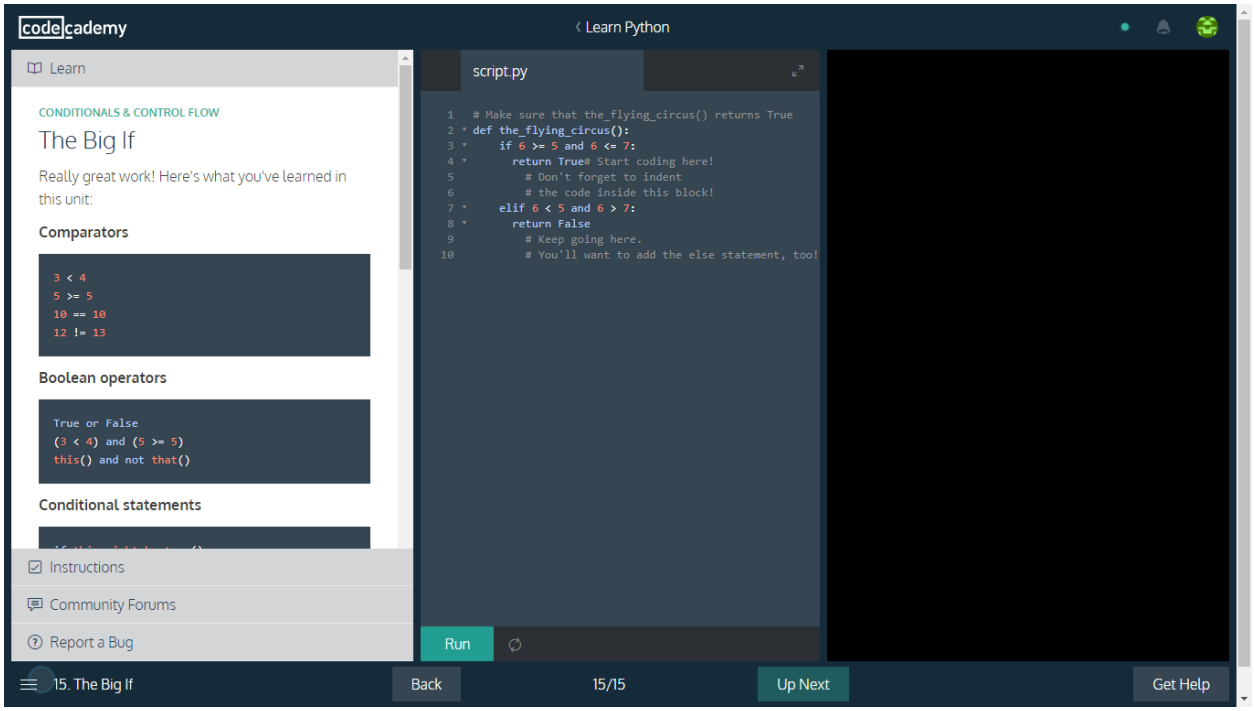
13/15


codecademy
Learn Python
Learn
CONDITIONALS & CONTROL FLOW
Else Problems, I Feel Bad for You, Son...
The else statement complements the if statement. An if / else pair says: "If this expression is true, run this indented code block; otherwise, run this code after the else statement."
Unlike if, else doesn't depend on an expression. For example:
if 8 > 9:
print "I don't printed!"
else:
print "I get printed!"
Instructions
1. Complete the else statements to the
Community Forums
Report a Bug
script.py
answer = "'Tis but a scratch!"
def black_knight():
if answer == "'Tis but a scratch!":
return True
else:
return False # Make sure this returns False
def french_soldier():
if answer == "Go away, or I shall taunt you a second time!":
return True
else:
return False # Make sure this returns False
Run
13. Else Problems, I Feel Bad for You, Son...
Back
13/15
Next
Get Help

14/15

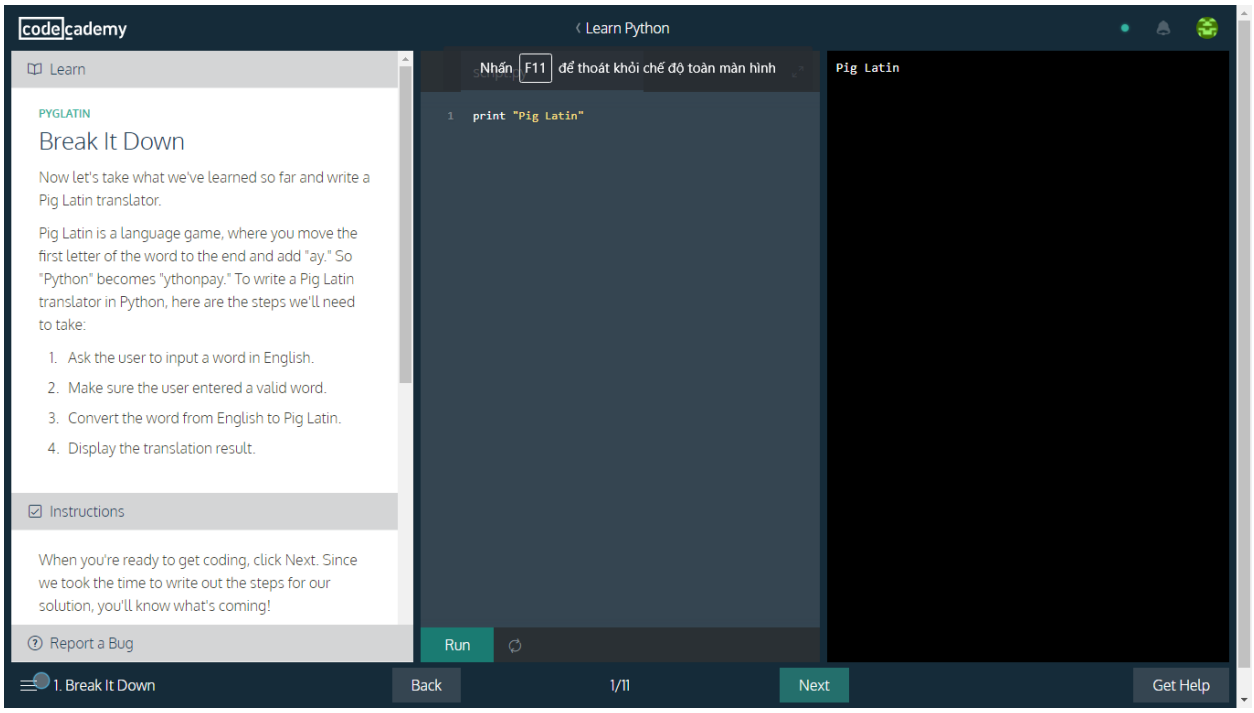

codecademy
Learn Python
Learn
CONDITIONALS & CONTROL FLOW
I Got 99 Problems, But a Switch Ain't One
elif is short for "else if." It means exactly what it sounds like: "otherwise, if the following expression is true, do this!"
if 8 > 9:
print "I don't get printed!"
elif 8 < 9:
print "I get printed!"
else:
print "I also don't get printed!"
In the example above, the elif statement is only checked if the original if statement if False.
Instructions
1. On line 2, fill in the if statement to check
Community Forums
Report a Bug
script.py
def greater_less_equal_5(answer):
if answer > 5 :
return 1
elif answer < 5:
return -1
else:
return 0
print greater_less_equal_5(4)
print greater_less_equal_5(5)
print greater_less_equal_5(6)
-1
0
1
Run
14. I Got 99 Problems, But a Switch Ain't One
Back
14/15
Next
Get Help

15/15

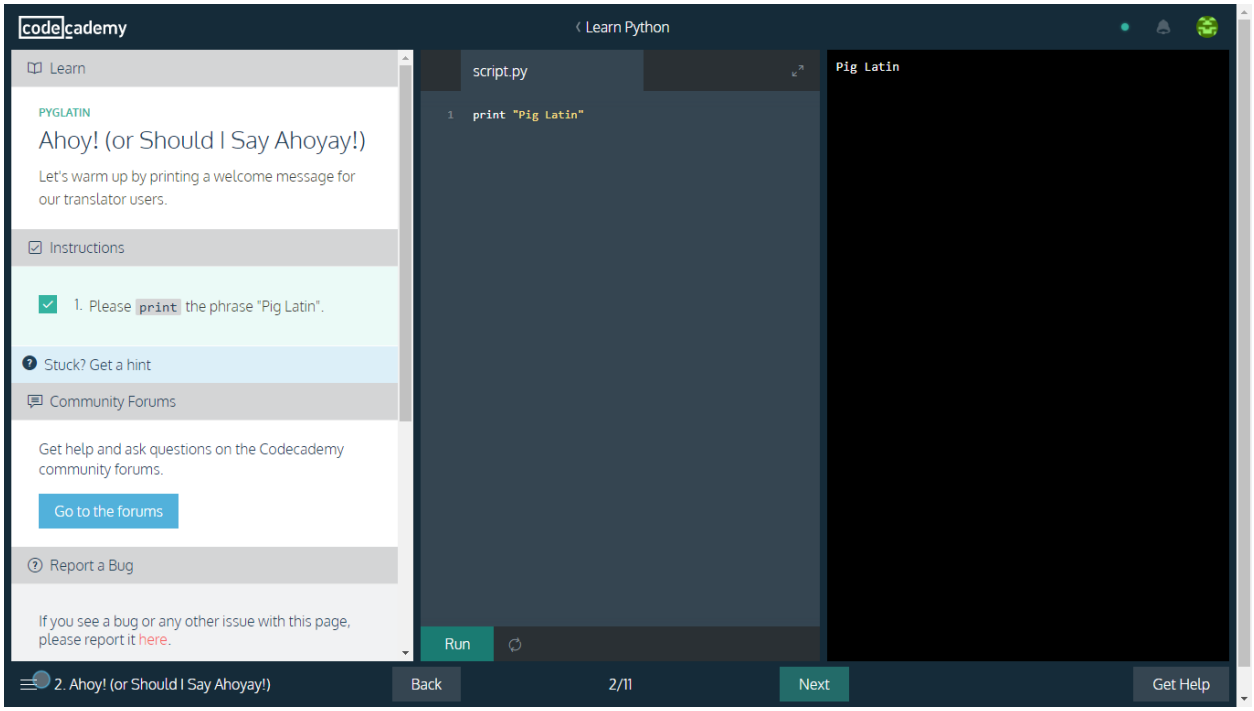


B.Pig Latin

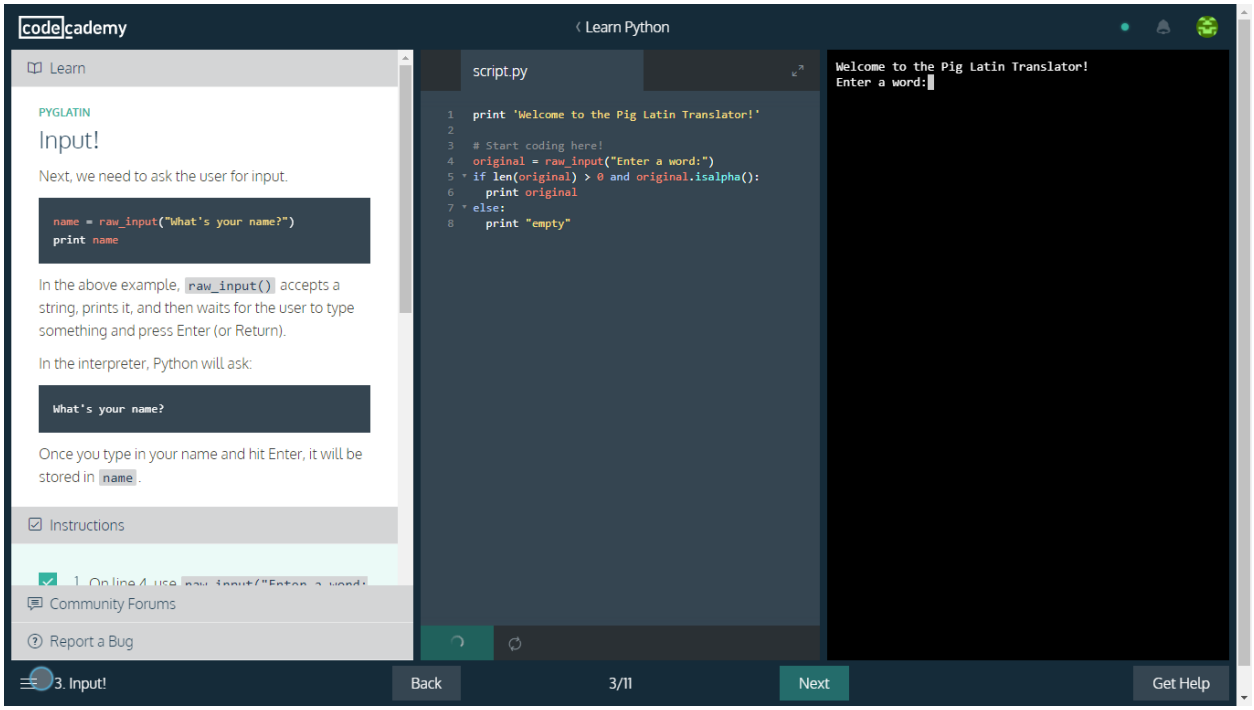
1/11



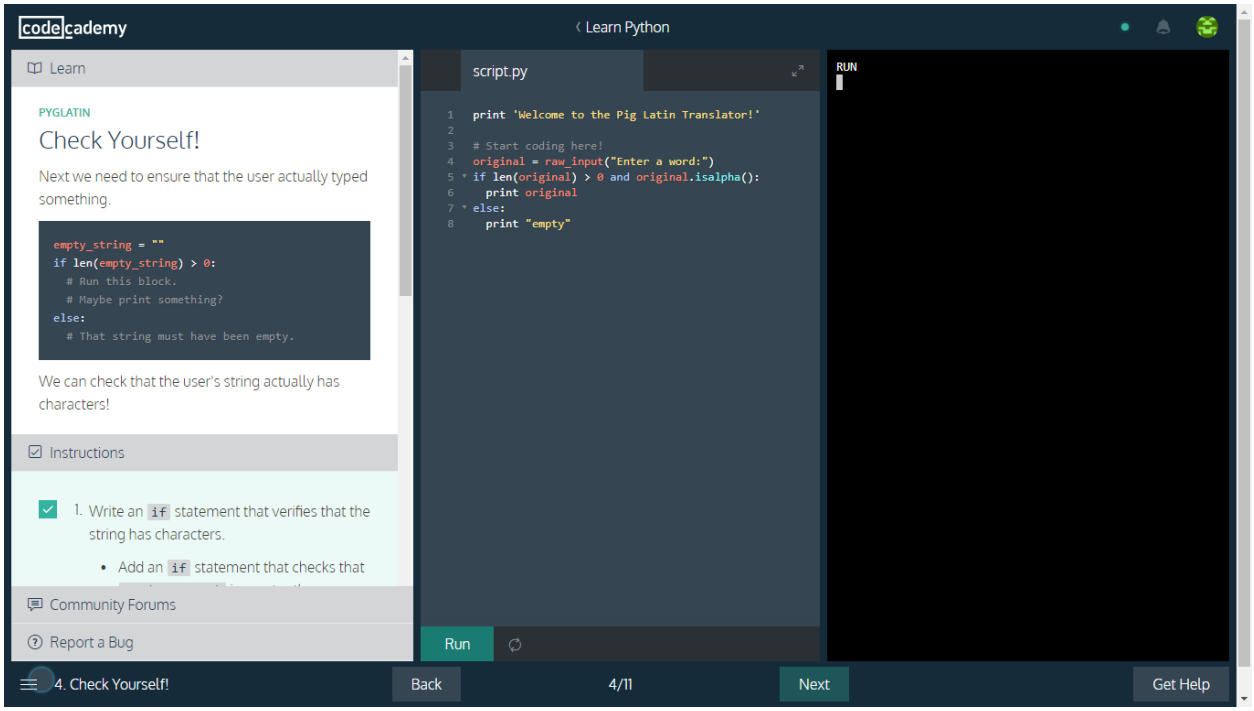
2/11


codecademy
Learn Python
Learn
PYGLATIN
Ahoy! (or Should I Say Ahoyay!)
Let's warm up by printing a welcome message for our translator users.
Instructions
1. Please print the phrase "Pig Latin".
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
Report a Bug
If you see a bug or any other issue with this page, please report it here.
script.py
1 print "Pig Latin"
Run
Pig Latin
2. Ahoy! (or Should I Say Ahoyay!)
Back
2/11
Next
Get Help

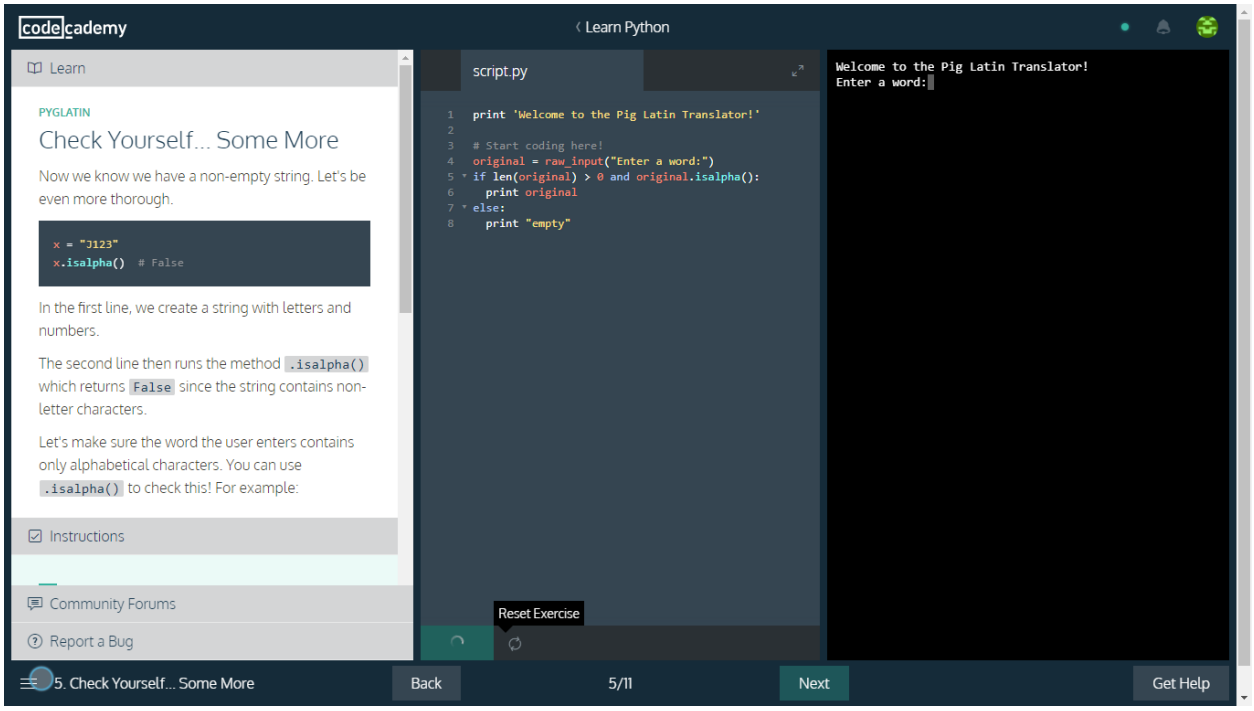
3/11


codecademy
Learn Python
Learn
PYGLATIN
Input!
Next, we need to ask the user for input.
name = raw_input("What's your name?")
print name
In the above example, raw_input() accepts a string, prints it, and then waits for the user to type something and press Enter (or Return).
In the interpreter, Python will ask:
What's your name?
Once you type in your name and hit Enter, it will be stored in name.
Instructions
Community Forums
Report a Bug
script.py
1 print 'Welcome to the Pig Latin Translator!'
2
3 # Start coding here!
4 original = raw_input("Enter a word:")
5 if len(original) > 0 and original.isalpha():
6 print original
7 else:
8 print "empty"
Welcome to the Pig Latin Translator!
Enter a word:
3. Input!
Back
3/11
Next
Get Help

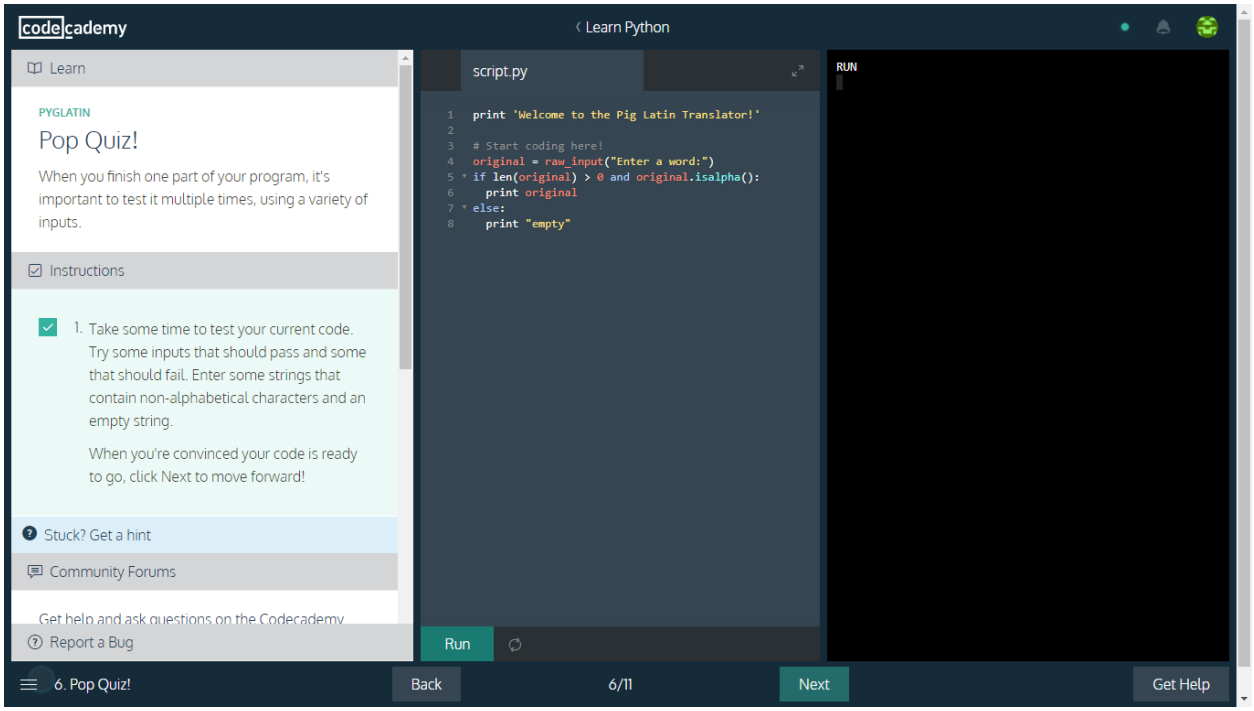
4/11


codecademy
Learn Python
Learn
PYGLATIN
Check Yourself!
Next we need to ensure that the user actually typed something.
empty_string = ""
if len(empty_string) > 0:
# Run this block.
# Maybe print something?
else:
# That string must have been empty.
We can check that the user's string actually has characters!
Instructions
1. Write an if statement that verifies that the string has characters.
Add an if statement that checks that
Community Forums
Report a Bug
script.py
print 'Welcome to the Pig Latin Translator!'
# Start coding here!
original = raw_input("Enter a word:")
if len(original) > 0 and original.isalpha():
print original
else:
print "empty"
Run
RUN
4. Check Yourself!
Back
4/11
Next
Get Help

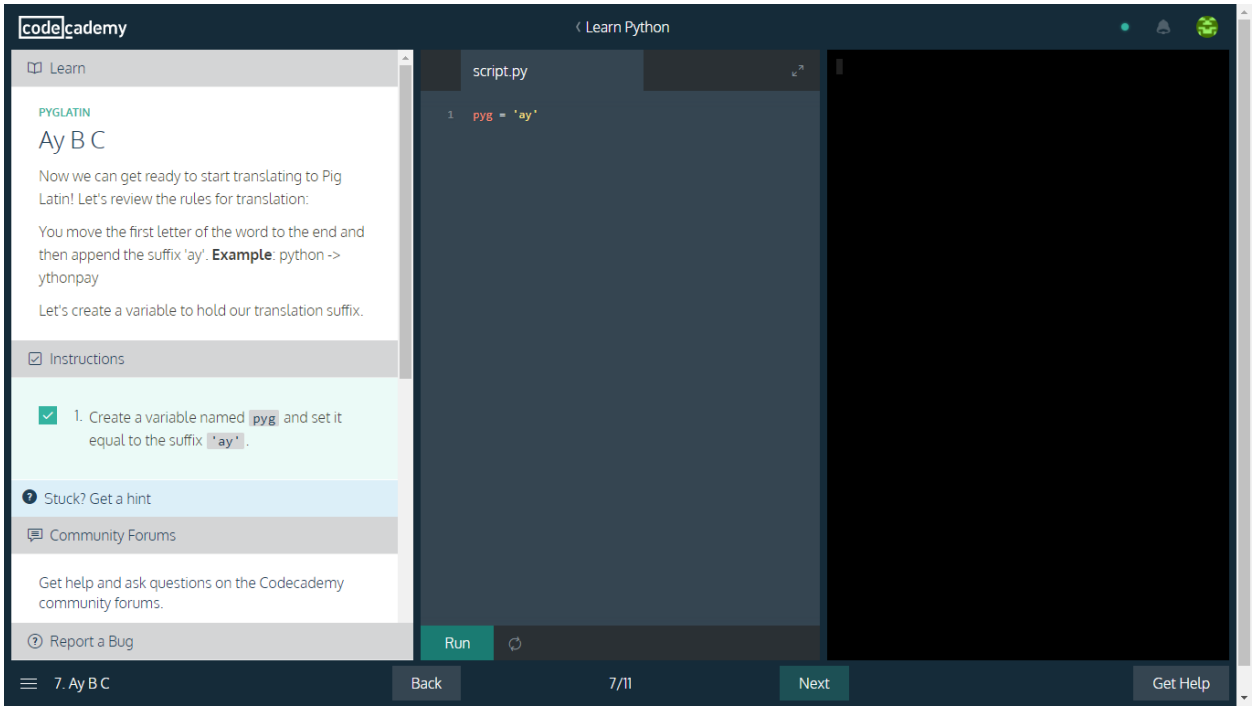
5/11


codecademy
Learn Python
Learn
PYGLATIN
Check Yourself... Some More
Now we know we have a non-empty string. Let's be even more thorough.
x = "J123"
x.isalpha() # False
In the first line, we create a string with letters and numbers.
The second line then runs the method .isalpha() which returns False since the string contains non-letter characters.
Let's make sure the word the user enters contains only alphabetical characters. You can use .isalpha() to check this! For example:
Instructions
Community Forums
Report a Bug
script.py
print 'Welcome to the Pig Latin Translator!'
# Start coding here!
original = raw_input("Enter a word:")
if len(original) > 0 and original.isalpha():
print original
else:
print "empty"
Reset Exercise
Welcome to the Pig Latin Translator!
Enter a word:
5. Check Yourself... Some More
Back
5/11
Next
Get Help

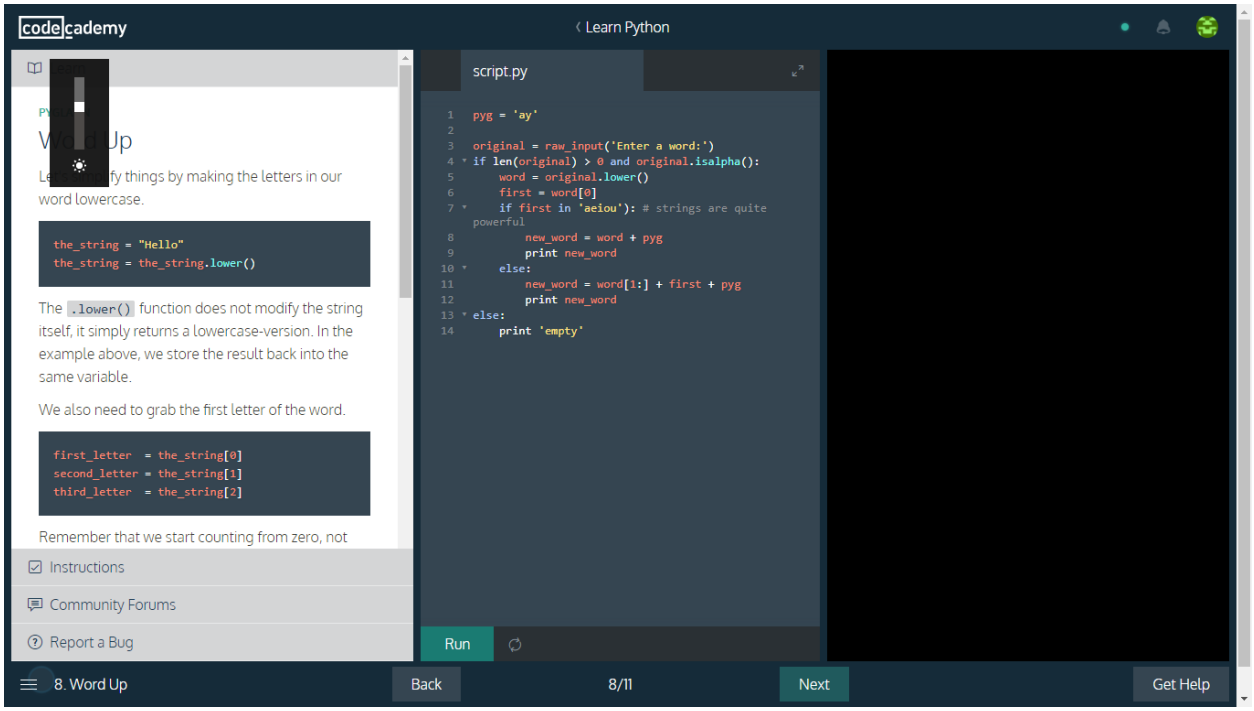
6/11


codecademy
Learn Python
Learn
PYGLATIN
Pop Quiz!
When you finish one part of your program, it's important to test it multiple times, using a variety of inputs.
Instructions
1. Take some time to test your current code. Try some inputs that should pass and some that should fail. Enter some strings that contain non-alphabetical characters and an empty string.
When you're convinced your code is ready to go, click Next to move forward!
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy
Report a Bug
script.py
print 'Welcome to the Pig Latin Translator!'
# Start coding here!
original = raw_input("Enter a word:")
if len(original) > 0 and original.isalpha():
print original
else:
print "empty"
RUN
Run
6. Pop Quiz!
Back
6/11
Next
Get Help

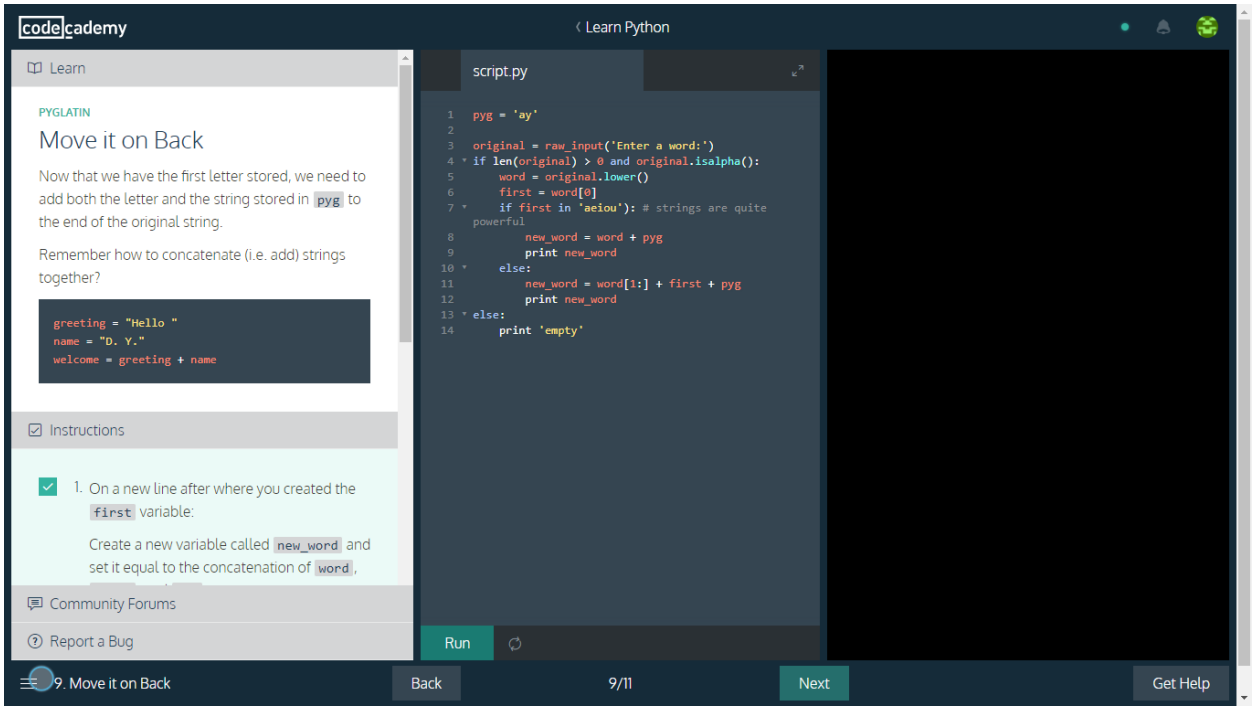
7/11


codecademy
Learn Python
Learn
PYGLATIN
Ay B C
Now we can get ready to start translating to Pig Latin! Let's review the rules for translation:
You move the first letter of the word to the end and then append the suffix 'ay'. Example: python -> ythonpay
Let's create a variable to hold our translation suffix.
Instructions
1. Create a variable named pyg and set it equal to the suffix 'ay'.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Report a Bug
script.py
pyg = 'ay'
Run
7. Ay B C
Back
7/11
Next
Get Help

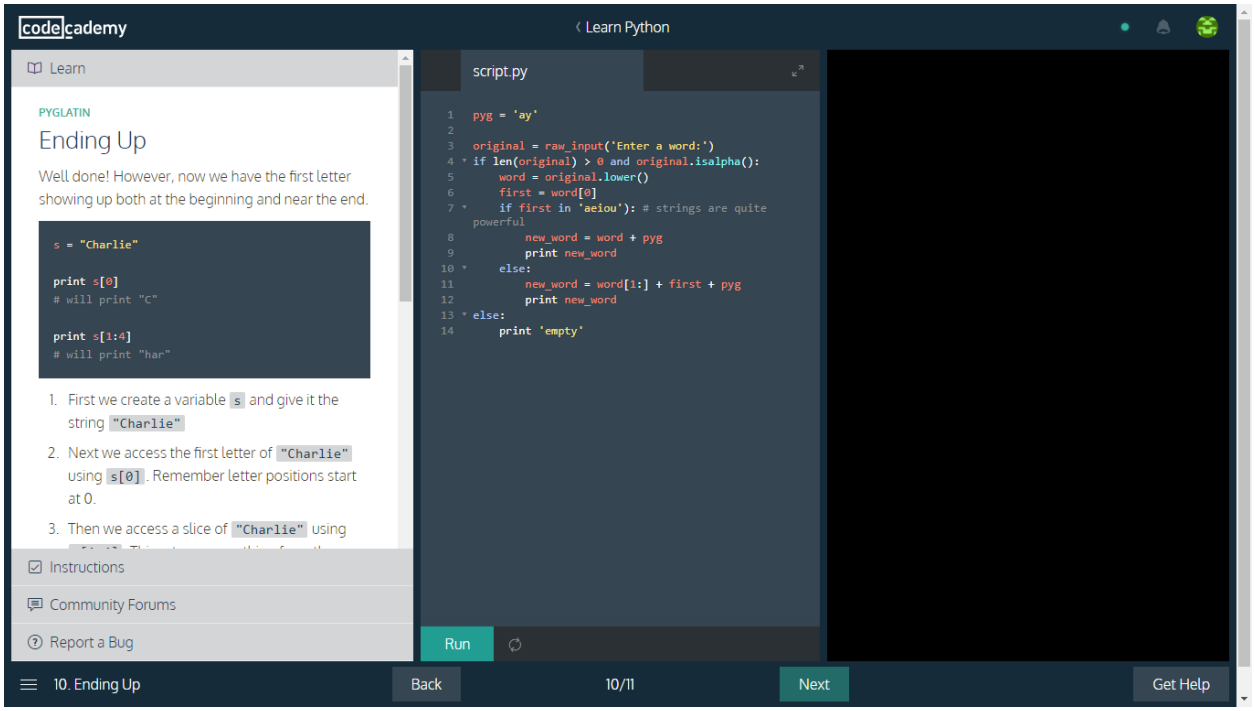
8/11


codecademy
Learn Python
Learn
Word Up
Let's jazzify things by making the letters in our word lowercase.
the_string = "Hello"
the_string = the_string.lower()
The .lower() function does not modify the string itself, it simply returns a lowercase-version. In the example above, we store the result back into the same variable.
We also need to grab the first letter of the word.
first_letter = the_string[0]
second_letter = the_string[1]
third_letter = the_string[2]
Remember that we start counting from zero, not
Instructions
Community Forums
Report a Bug
script.py
pyg = 'ay'
original = raw_input('Enter a word:')
if len(original) > 0 and original.isalpha():
word = original.lower()
first = word[0]
if first in 'aeiou'): # strings are quite powerful
new_word = word + pyg
print new_word
else:
new_word = word[1:] + first + pyg
print new_word
else:
print 'empty'
Run
8. Word Up
Back
8/11
Next
Get Help

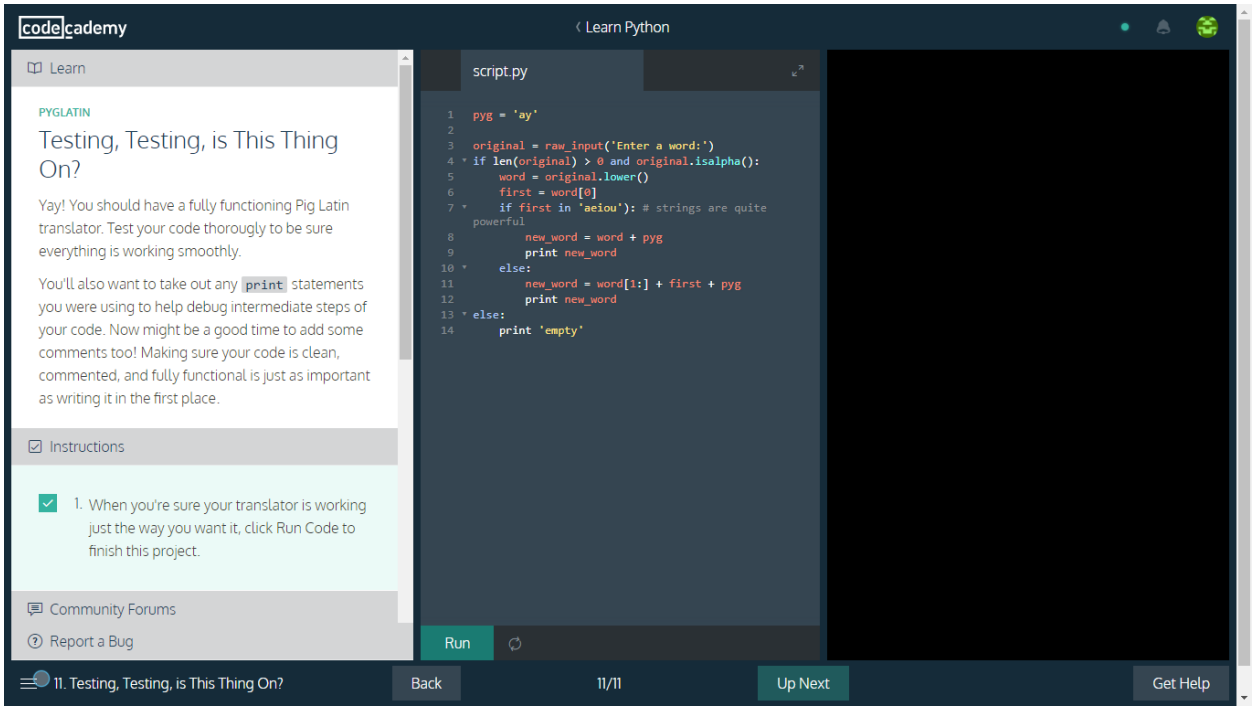
9/11


codecademy
Learn Python
Learn
PYGLATIN
Move it on Back
Now that we have the first letter stored, we need to add both the letter and the string stored in pyg to the end of the original string.
Remember how to concatenate (i.e. add) strings together?
greeting = "Hello "
name = "D. Y."
welcome = greeting + name
Instructions
1. On a new line after where you created the first variable:
Create a new variable called new_word and set it equal to the concatenation of word,
Community Forums
Report a Bug
script.py
pyg = 'ay'
original = raw_input('Enter a word:')
if len(original) > 0 and original.isalpha():
word = original.lower()
first = word[0]
if first in 'aeiou'): # strings are quite powerful
new_word = word + pyg
print new_word
else:
new_word = word[1:] + first + pyg
print new_word
else:
print 'empty'
Run
9. Move it on Back
Back
9/11
Next
Get Help

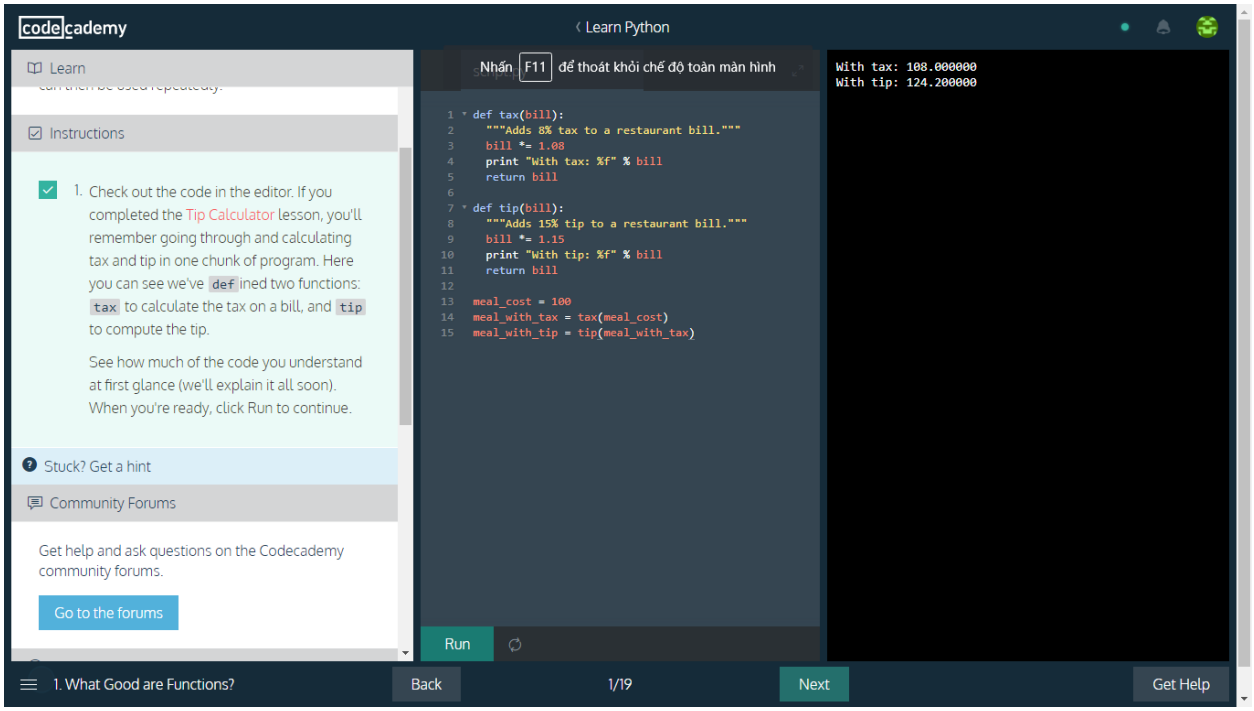
10/11



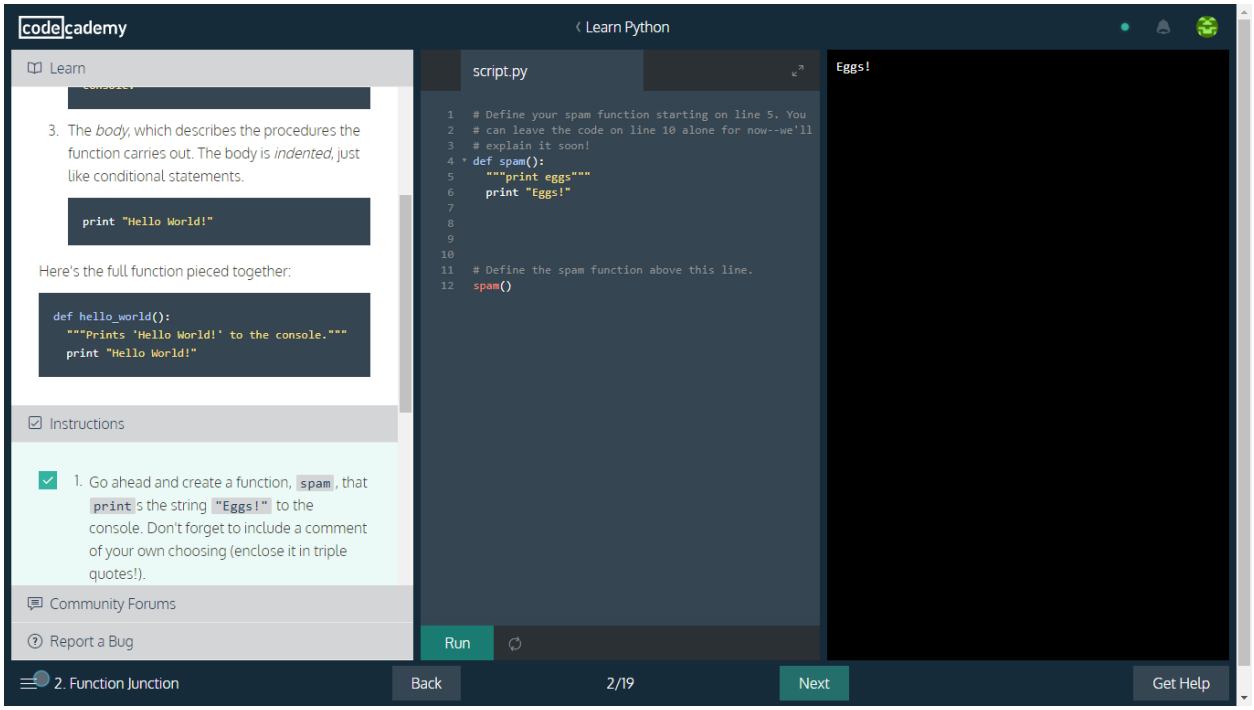
11/11



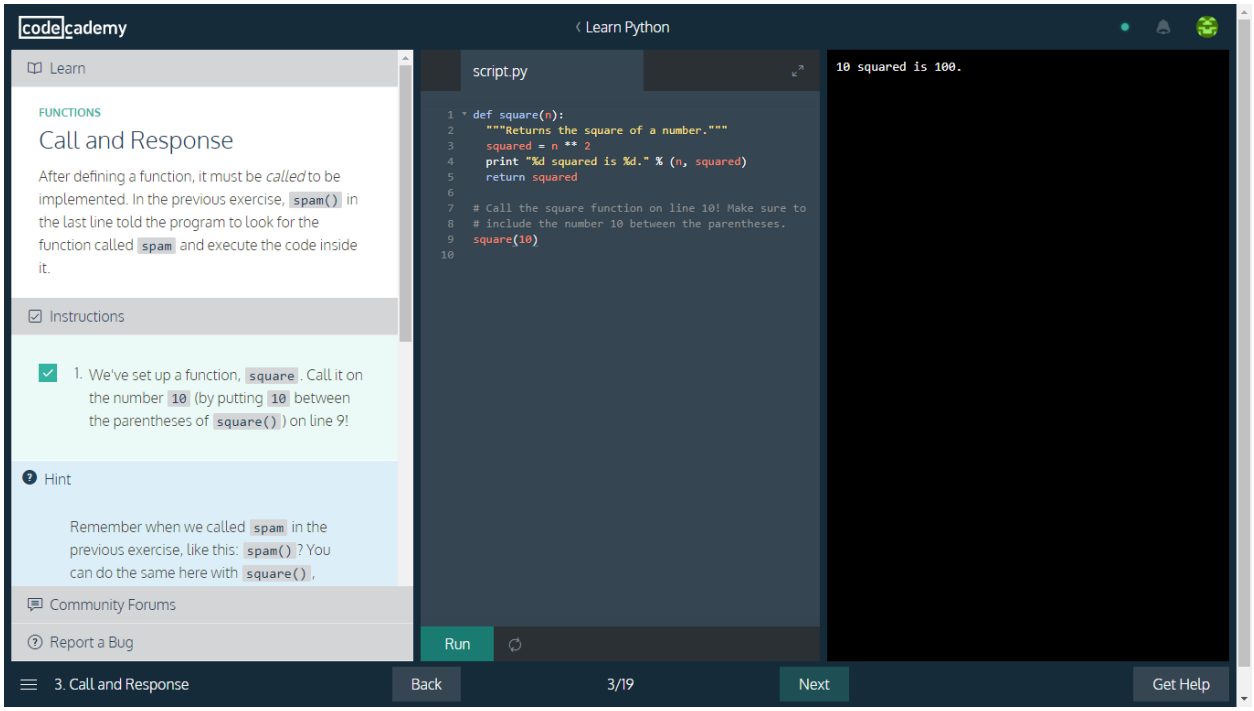
Functions
1/19


codecademy
Learn Python
Learn
Instructions
1. Check out the code in the editor. If you completed the Tip Calculator lesson, you'll remember going through and calculating tax and tip in one chunk of program. Here you can see we've defined two functions: tax to calculate the tax on a bill, and tip to compute the tip.
See how much of the code you understand at first glance (we'll explain it all soon). When you're ready, click Run to continue.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
Nhấn F11 để thoát khỏi chế độ toàn màn hình
def tax(bill):
"""Adds 8% tax to a restaurant bill."""
bill *= 1.08
print "With tax: %f" % bill
return bill
def tip(bill):
"""Adds 15% tip to a restaurant bill."""
bill *= 1.15
print "With tip: %f" % bill
return bill
meal_cost = 100
meal_with_tax = tax(meal_cost)
meal_with_tip = tip(meal_with_tax)
With tax: 108.000000
With tip: 124.200000
Run
1. What Good are Functions?
Back
1/19
Next
Get Help

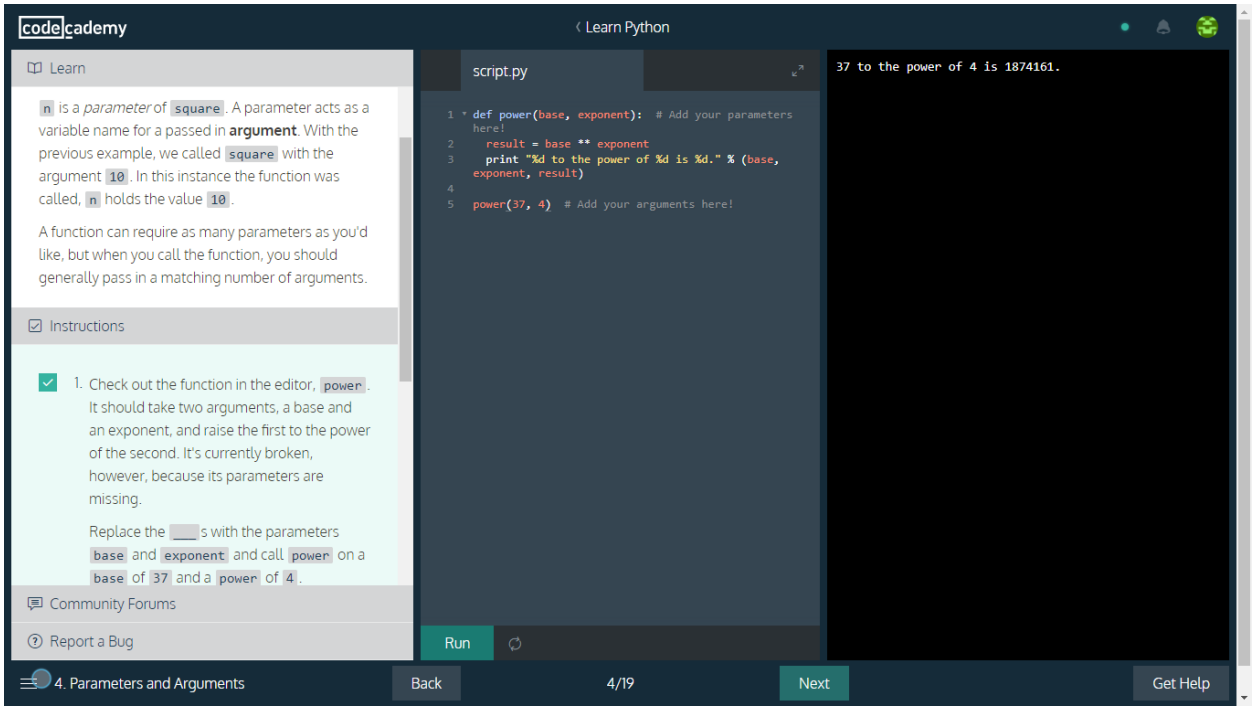
2/19


codecademy
Learn Python
Learn
3. The body, which describes the procedures the function carries out. The body is indented, just like conditional statements.
print "Hello World!"
Here's the full function pieced together:
def hello_world():
"""Prints 'Hello World!' to the console."""
print "Hello World!"
Instructions
1. Go ahead and create a function, spam, that prints the string "Eggs!" to the console. Don't forget to include a comment of your own choosing (enclose it in triple quotes!).
Community Forums
Report a Bug
script.py
# Define your spam function starting on line 5. You
# can leave the code on line 10 alone for now--we'll
# explain it soon!
def spam():
"""print eggs"""
print "Eggs!"
# Define the spam function above this line.
spam()
Eggs!
Run
2. Function Junction
Back
2/19
Next
Get Help

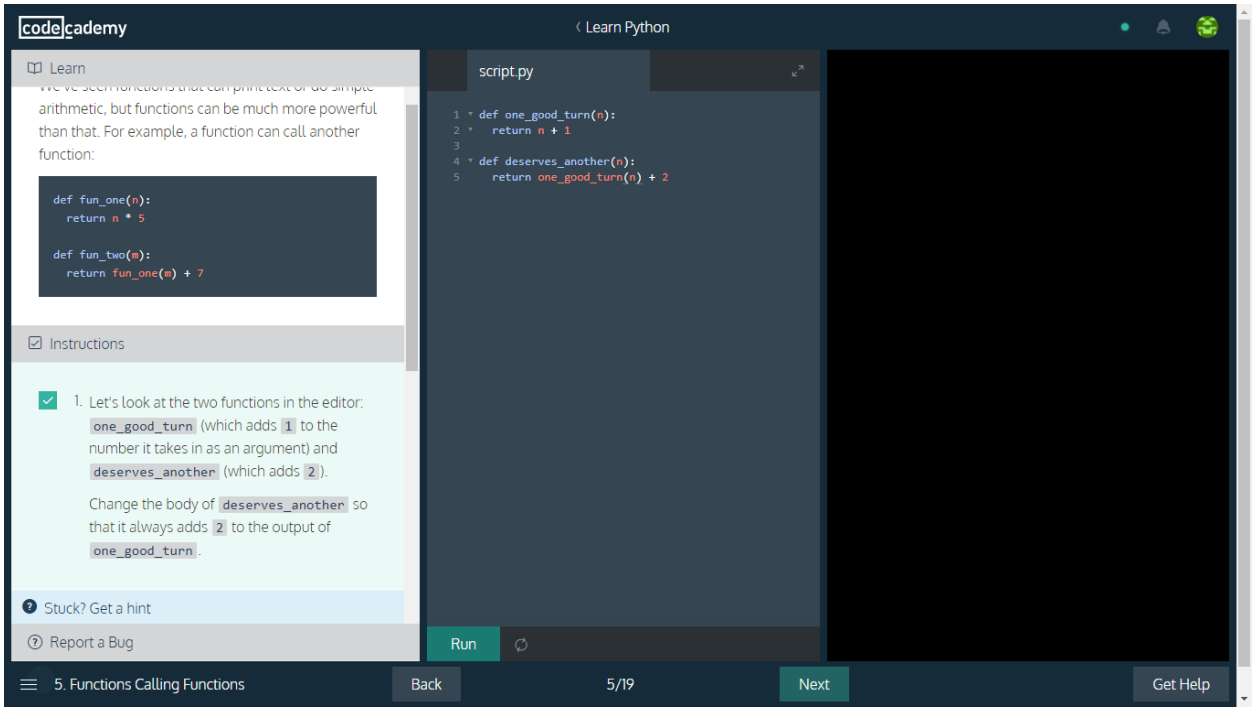
3/19


codecademy
Learn Python
Learn
FUNCTIONS
Call and Response
After defining a function, it must be called to be implemented. In the previous exercise, spam() in the last line told the program to look for the function called spam and execute the code inside it.
Instructions
1. We've set up a function, square. Call it on the number 10 (by putting 10 between the parentheses of square()) on line 9!
Hint
Remember when we called spam in the previous exercise, like this: spam()? You can do the same here with square(),
Community Forums
Report a Bug
script.py
def square(n):
"""Returns the square of a number."""
squared = n ** 2
print "%d squared is %d." % (n, squared)
return squared
# Call the square function on line 10! Make sure to
# include the number 10 between the parentheses.
square(10)
Run
10 squared is 100.
3. Call and Response
Back
3/19
Next
Get Help

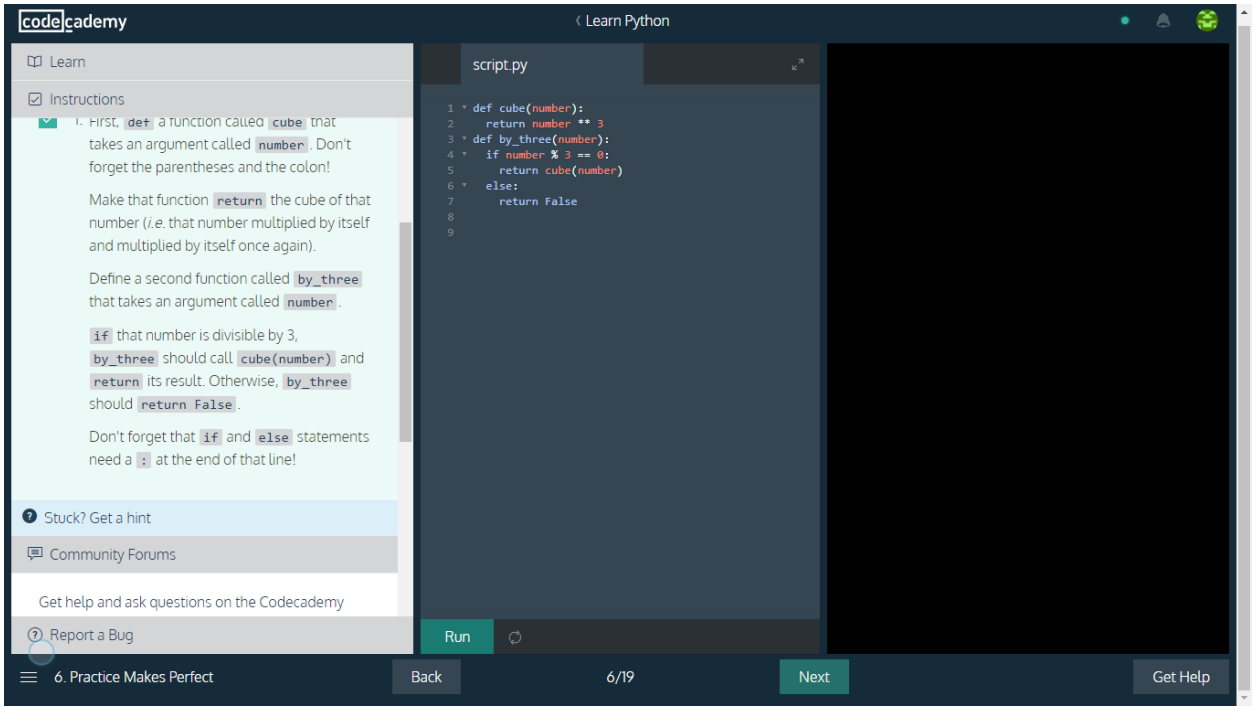
4/19


codecademy
Learn Python
Learn
n is a parameter of square. A parameter acts as a variable name for a passed in argument. With the previous example, we called square with the argument 10. In this instance the function was called, n holds the value 10.
A function can require as many parameters as you'd like, but when you call the function, you should generally pass in a matching number of arguments.
Instructions
1. Check out the function in the editor, power. It should take two arguments, a base and an exponent, and raise the first to the power of the second. It's currently broken, however, because its parameters are missing.
Replace the ___s with the parameters base and exponent and call power on a base of 37 and a power of 4.
Community Forums
Report a Bug
script.py
def power(base, exponent): # Add your parameters here!
result = base ** exponent
print "%d to the power of %d is %d." % (base, exponent, result)
power(37, 4) # Add your arguments here!
Run
37 to the power of 4 is 1874161.
4. Parameters and Arguments
Back
4/19
Next
Get Help

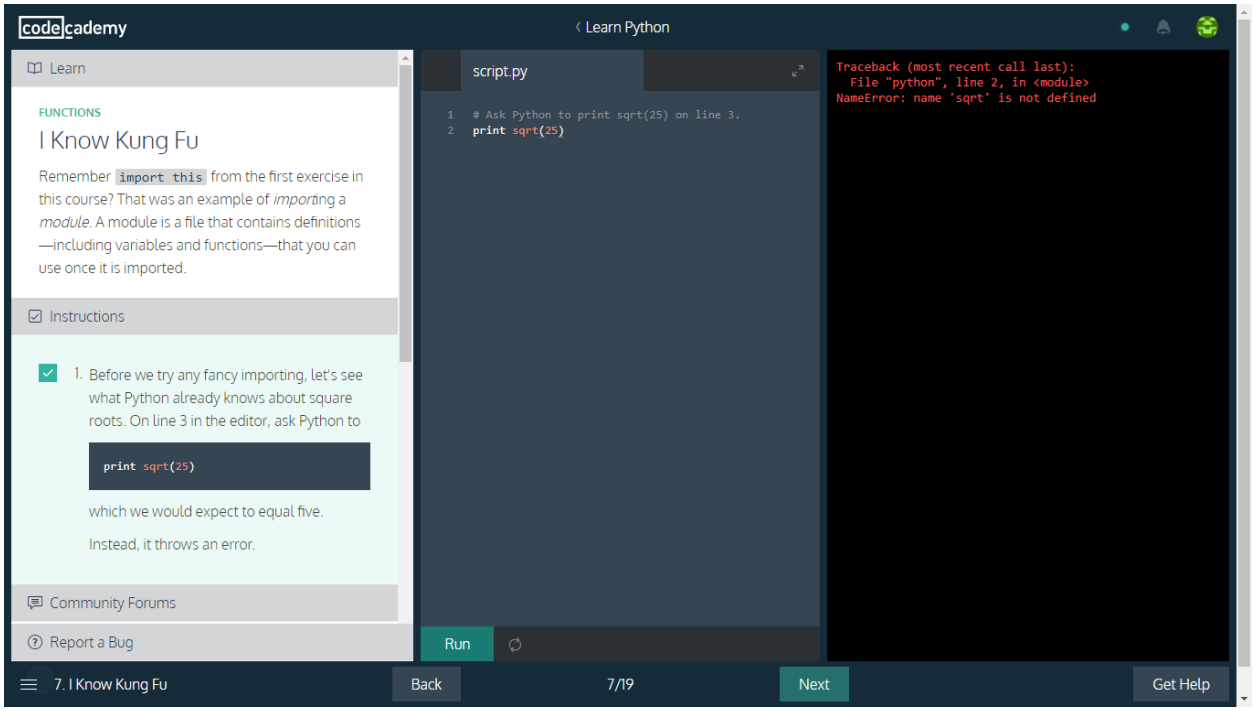
5/19


codecademy
Learn Python
Learn
arithmetic, but functions can be much more powerful than that. For example, a function can call another function:
def fun_one(n):
return n * 5
def fun_two(m):
return fun_one(m) + 7
Instructions
1. Let's look at the two functions in the editor: one_good_turn (which adds 1 to the number it takes in as an argument) and deserves_another (which adds 2).
Change the body of deserves_another so that it always adds 2 to the output of one_good_turn.
Stuck? Get a hint
Report a Bug
script.py
def one_good_turn(n):
return n + 1
def deserves_another(n):
return one_good_turn(n) + 2
Run
5. Functions Calling Functions
Back
5/19
Next
Get Help

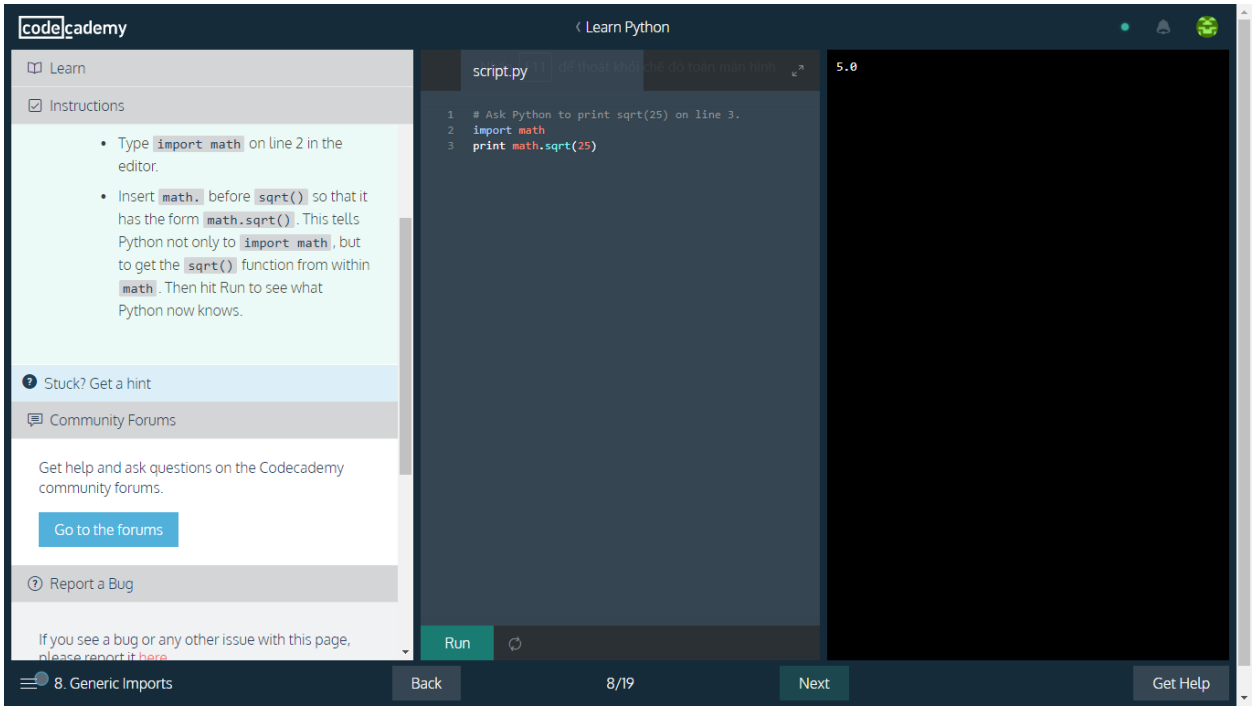
6/19


codecademy
Learn Python
Learn
Instructions
1. First, def a function called cube that takes an argument called number. Don't forget the parentheses and the colon!
Make that function return the cube of that number (i.e. that number multiplied by itself and multiplied by itself once again).
Define a second function called by_three that takes an argument called number.
if that number is divisible by 3, by_three should call cube(number) and return its result. Otherwise, by_three should return False.
Don't forget that if and else statements need a : at the end of that line!
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy
Report a Bug
script.py
def cube(number):
return number ** 3
def by_three(number):
if number % 3 == 0:
return cube(number)
else:
return False
Run
6. Practice Makes Perfect
Back
6/19
Next
Get Help

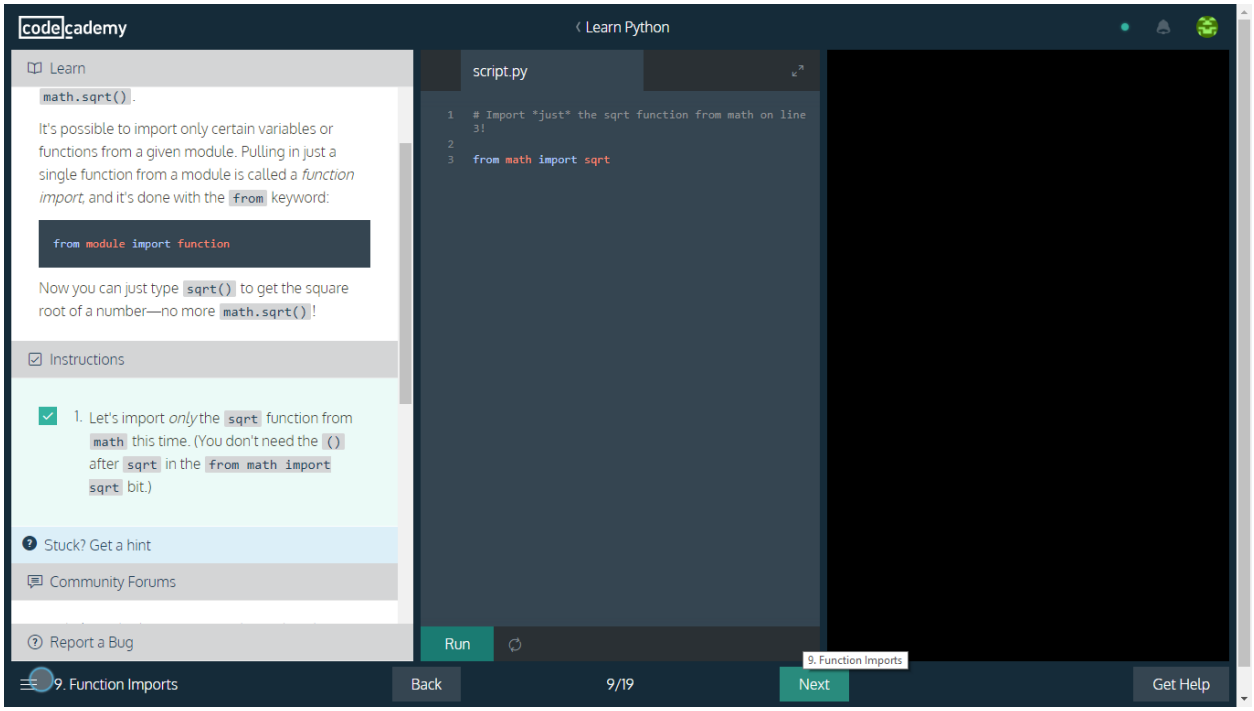
7/19


codecademy
Learn Python
Learn
FUNCTIONS
I Know Kung Fu
Remember import this from the first exercise in this course? That was an example of importing a module. A module is a file that contains definitions—including variables and functions—that you can use once it is imported.
Instructions
1. Before we try any fancy importing, let's see what Python already knows about square roots. On line 3 in the editor, ask Python to
print sqrt(25)
which we would expect to equal five.
Instead, it throws an error.
Community Forums
Report a Bug
script.py
Ask Python to print sqrt(25) on line 3.
print sqrt(25)
Traceback (most recent call last):
File "python", line 2, in <module>
NameError: name 'sqrt' is not defined
Run
7. I Know Kung Fu
Back
7/19
Next
Get Help

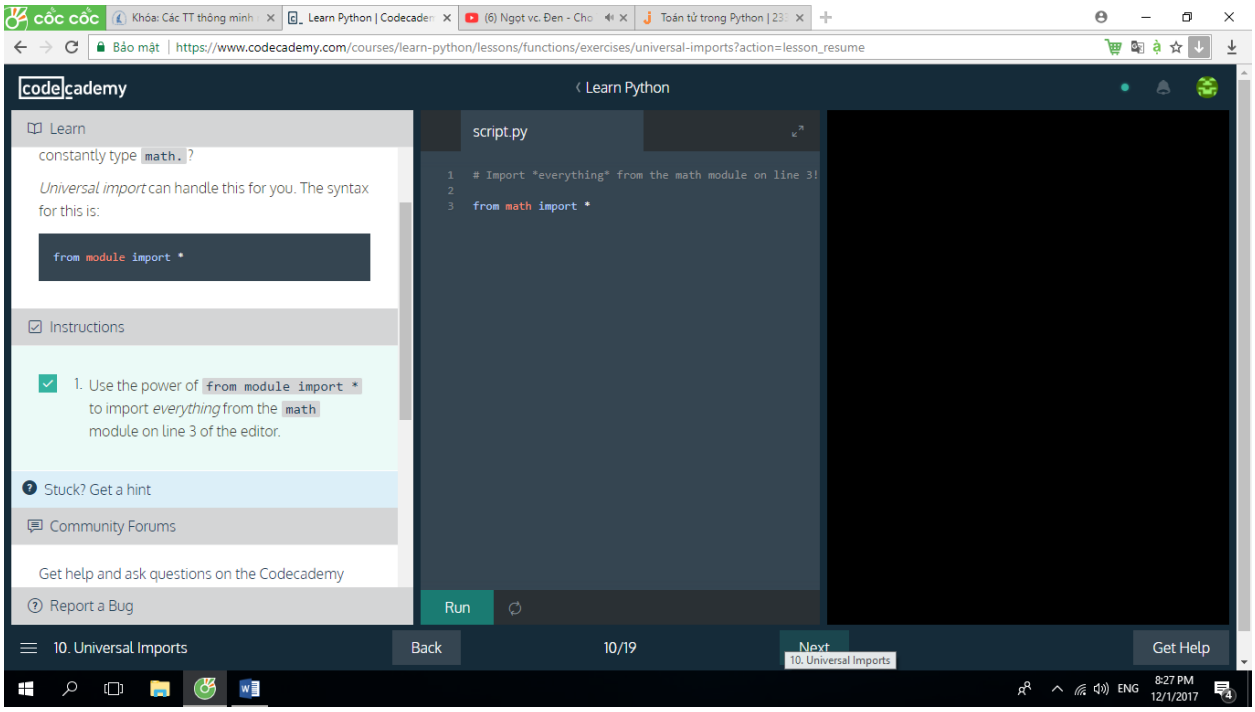
8/19


codecademy
Learn Python
Learn
Instructions
Type import math on line 2 in the editor.
Insert math. before sqrt() so that it has the form math.sqrt(). This tells Python not only to import math, but to get the sqrt() function from within math. Then hit Run to see what Python now knows.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
Report a Bug
If you see a bug or any other issue with this page, please report it here
script.py
# Ask Python to print sqrt(25) on line 3.
import math
print math.sqrt(25)
5.0
Run
8. Generic Imports
Back
8/19
Next
Get Help

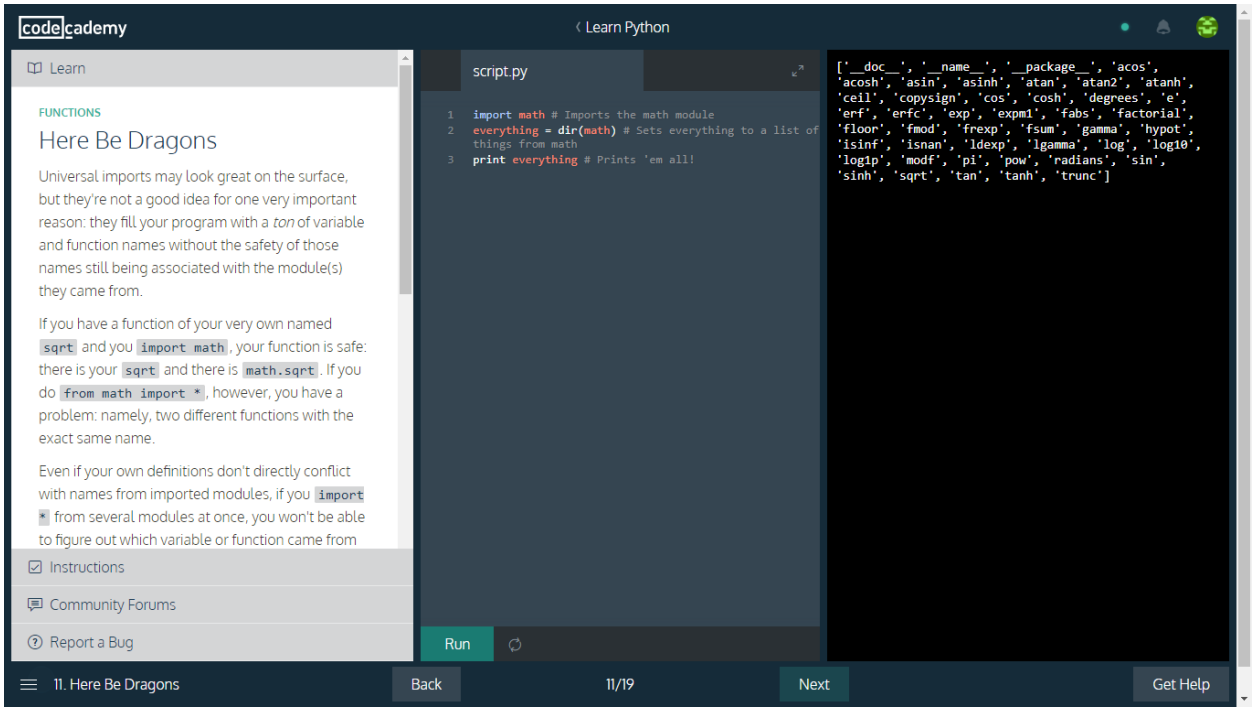
9/19


codecademy
Learn Python
Learn
math.sqrt().
It's possible to import only certain variables or functions from a given module. Pulling in just a single function from a module is called a function import, and it's done with the from keyword:
from module import function
Now you can just type sqrt() to get the square root of a number—no more math.sqrt()!
Instructions
1. Let's import only the sqrt function from math this time. (You don't need the () after sqrt in the from math import sqrt bit.)
Stuck? Get a hint
Community Forums
Report a Bug
script.py
# Import *just* the sqrt function from math on line 3!
from math import sqrt
Run
9. Function Imports
Back
9/19
Next
Get Help

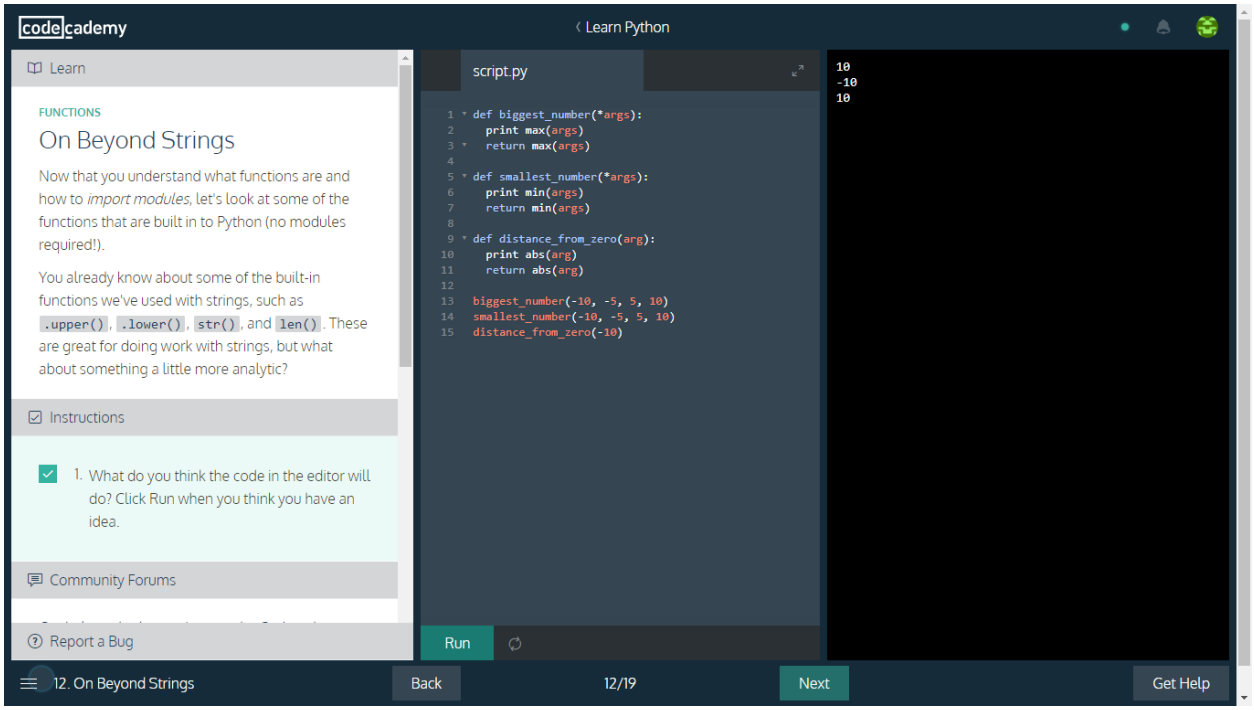
10/19


cốc cốc
Bảo mật | https://www.codecademy.com/courses/learn-python/lessons/functions/exercises/universal-imports?action=lesson_resume
codecademy
Learn Python
Learn
constantly type math.?
Universal import can handle this for you. The syntax for this is:
from module import *
Instructions
1. Use the power of from module import * to import everything from the math module on line 3 of the editor.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy
Report a Bug
script.py
# Import *everything* from the math module on line 3!
from math import *
Run
10. Universal Imports
Back
10/19
Next
Get Help
ENG
8:27 PM
12/1/2017

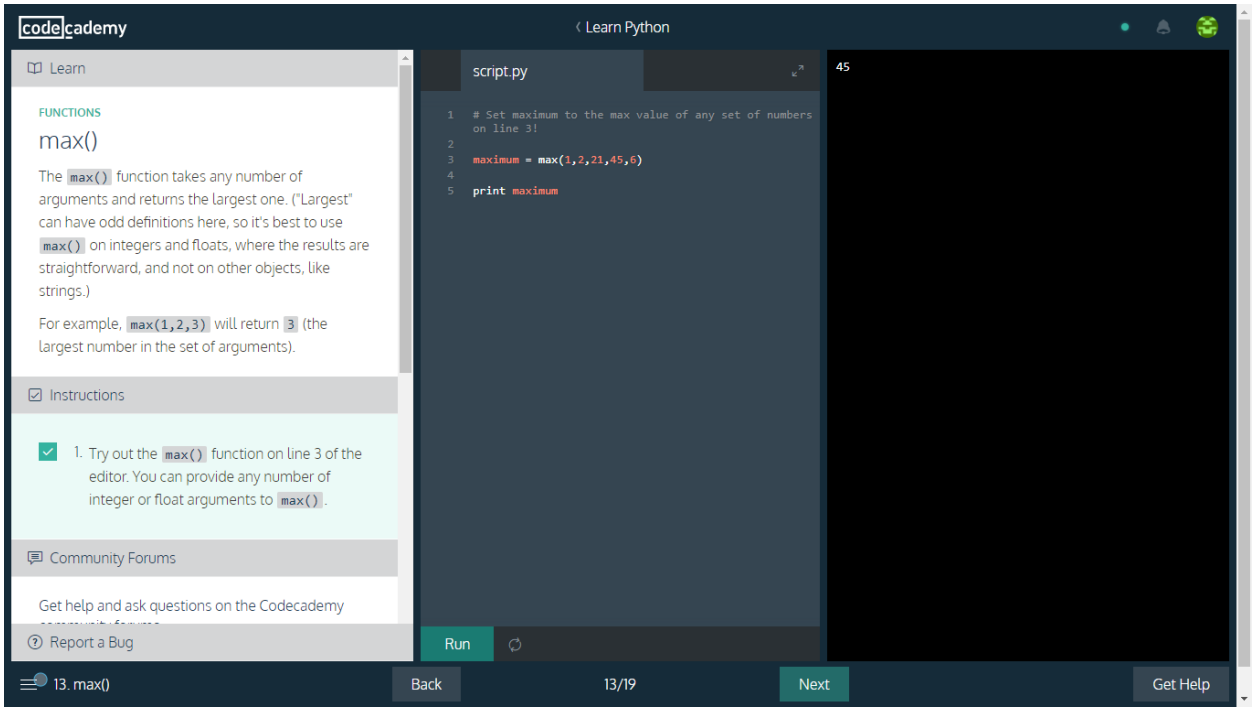
11/19


codecademy
Learn Python
Learn
FUNCTIONS
Here Be Dragons
Universal imports may look great on the surface, but they're not a good idea for one very important reason: they fill your program with a ton of variable and function names without the safety of those names still being associated with the module(s) they came from.
If you have a function of your very own named sqrt and you import math, your function is safe: there is your sqrt and there is math.sqrt. If you do from math import *, however, you have a problem: namely, two different functions with the exact same name.
Even if your own definitions don't directly conflict with names from imported modules, if you import * from several modules at once, you won't be able to figure out which variable or function came from
Instructions
Community Forums
Report a Bug
script.py
import math # Imports the math module
everything = dir(math) # Sets everything to a list of things from math
print everything # Prints 'em all!
['__doc__', '__name__', '__package__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'hypot', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
Run
11. Here Be Dragons
Back
11/19
Next
Get Help

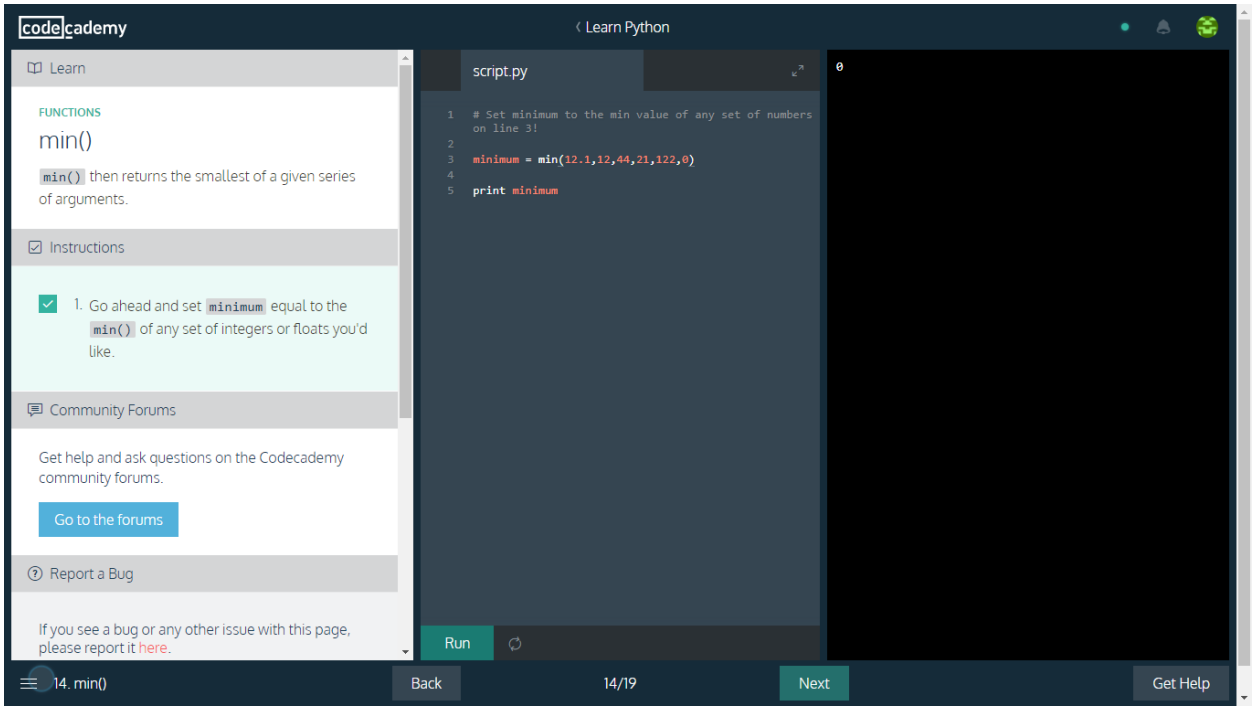
12/19


codecademy
Learn Python
Learn
FUNCTIONS
On Beyond Strings
Now that you understand what functions are and how to import modules, let's look at some of the functions that are built in to Python (no modules required!).
You already know about some of the built-in functions we've used with strings, such as .upper(), .lower(), str(), and len(). These are great for doing work with strings, but what about something a little more analytic?
Instructions
1. What do you think the code in the editor will do? Click Run when you think you have an idea.
Community Forums
Report a Bug
script.py
def biggest_number(*args):
print max(args)
return max(args)
def smallest_number(*args):
print min(args)
return min(args)
def distance_from_zero(arg):
print abs(arg)
return abs(arg)
biggest_number(-10, -5, 5, 10)
smallest_number(-10, -5, 5, 10)
distance_from_zero(-10)
10
-10
10
Run
12. On Beyond Strings
Back
12/19
Next
Get Help

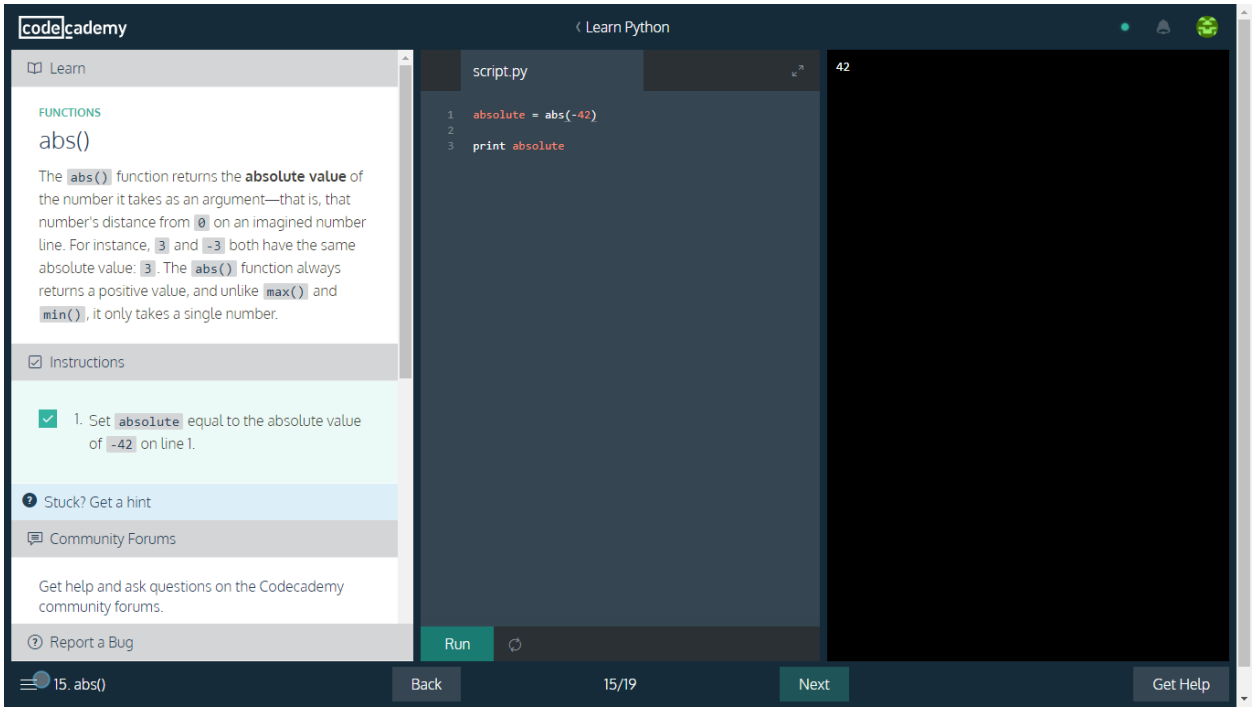
13/19


codecademy
Learn Python
Learn
FUNCTIONS
max()
The max() function takes any number of arguments and returns the largest one. ("Largest" can have odd definitions here, so it's best to use max() on integers and floats, where the results are straightforward, and not on other objects, like strings.)
For example, max(1,2,3) will return 3 (the largest number in the set of arguments).
Instructions
1. Try out the max() function on line 3 of the editor. You can provide any number of integer or float arguments to max().
Community Forums
Get help and ask questions on the Codecademy
Report a Bug
script.py
# Set maximum to the max value of any set of numbers on line 3!
maximum = max(1,2,21,45,6)
print maximum
Run
45
13. max()
Back
13/19
Next
Get Help

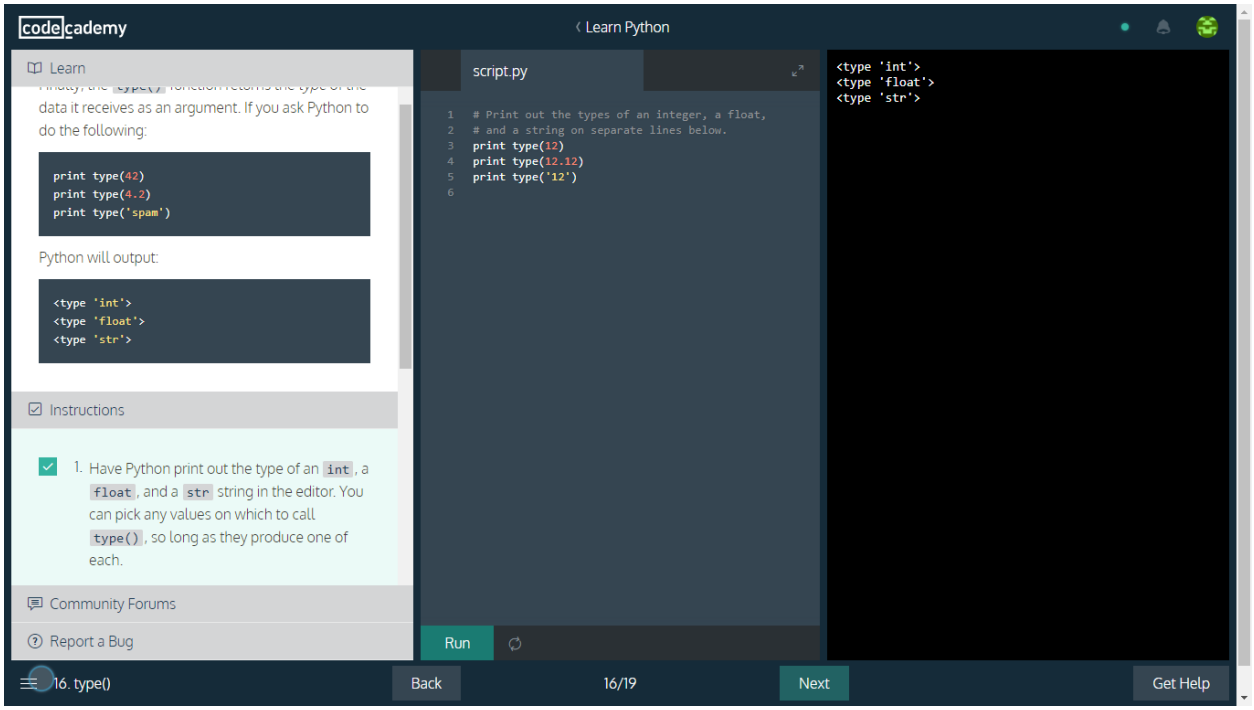
14/19


codecademy
Learn Python
Learn
FUNCTIONS
min()
min() then returns the smallest of a given series of arguments.
Instructions
1. Go ahead and set minimum equal to the min() of any set of integers or floats you'd like.
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
Report a Bug
If you see a bug or any other issue with this page, please report it here.
script.py
# Set minimum to the min value of any set of numbers on line 3!
minimum = min(12.1,12,44,21,122,0)
print minimum
Run
0
14. min()
Back
14/19
Next
Get Help

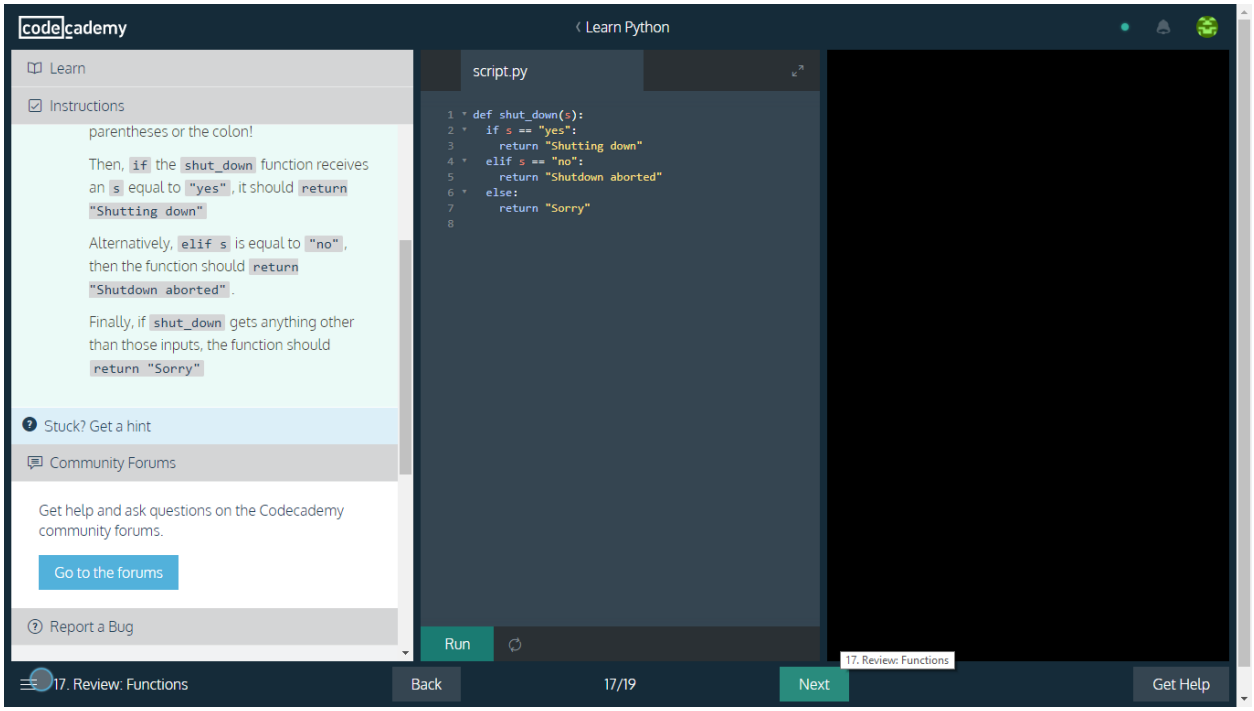
15/19


codecademy
Learn Python
Learn
FUNCTIONS
abs()
The abs() function returns the absolute value of the number it takes as an argument—that is, that number's distance from 0 on an imagined number line. For instance, 3 and -3 both have the same absolute value: 3. The abs() function always returns a positive value, and unlike max() and min(), it only takes a single number.
Instructions
1. Set absolute equal to the absolute value of -42 on line 1.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Report a Bug
script.py
absolute = abs(-42)
print absolute
42
Run
15. abs()
Back
15/19
Next
Get Help

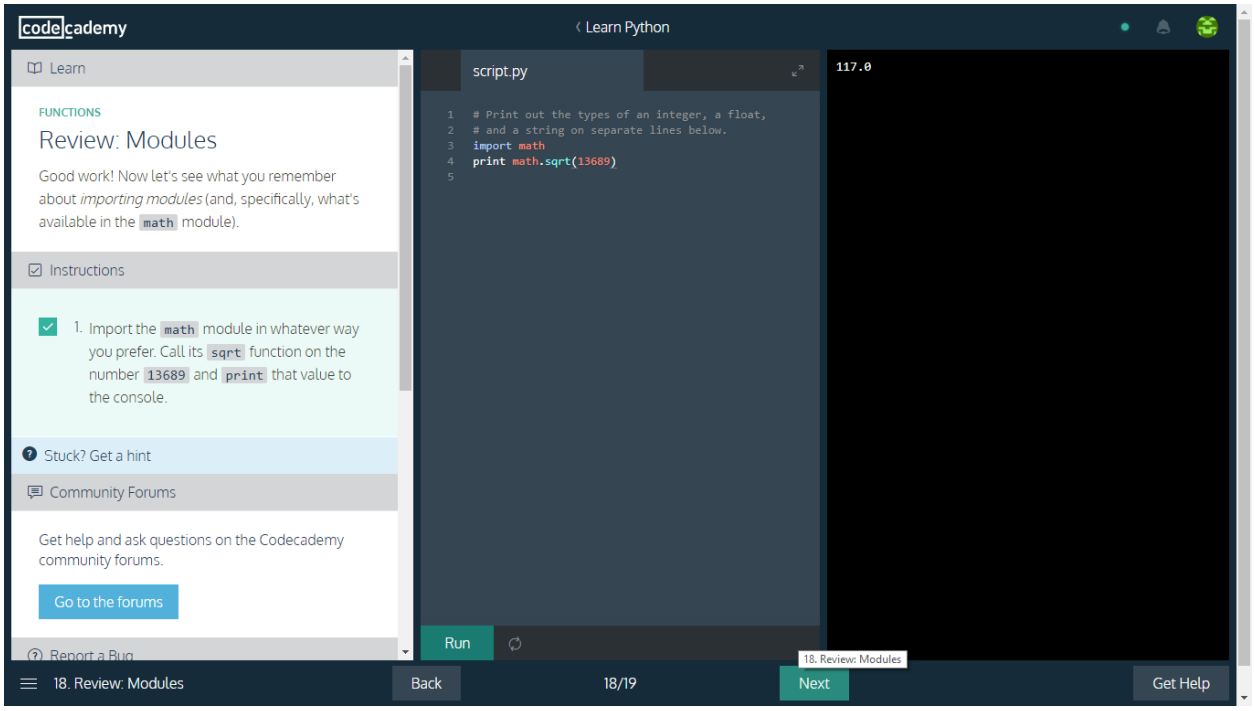
16/19


codecademy
Learn Python
Learn
data it receives as an argument. If you ask Python to do the following:
print type(42)
print type(4.2)
print type('spam')
Python will output:
<type 'int'>
<type 'float'>
<type 'str'>
Instructions
1. Have Python print out the type of an int, a float, and a str string in the editor. You can pick any values on which to call type(), so long as they produce one of each.
Community Forums
Report a Bug
script.py
Print out the types of an integer, a float,
and a string on separate lines below.
print type(12)
print type(12.12)
print type('12')
<type 'int'>
<type 'float'>
<type 'str'>
Run
16. type()
Back
16/19
Next
Get Help

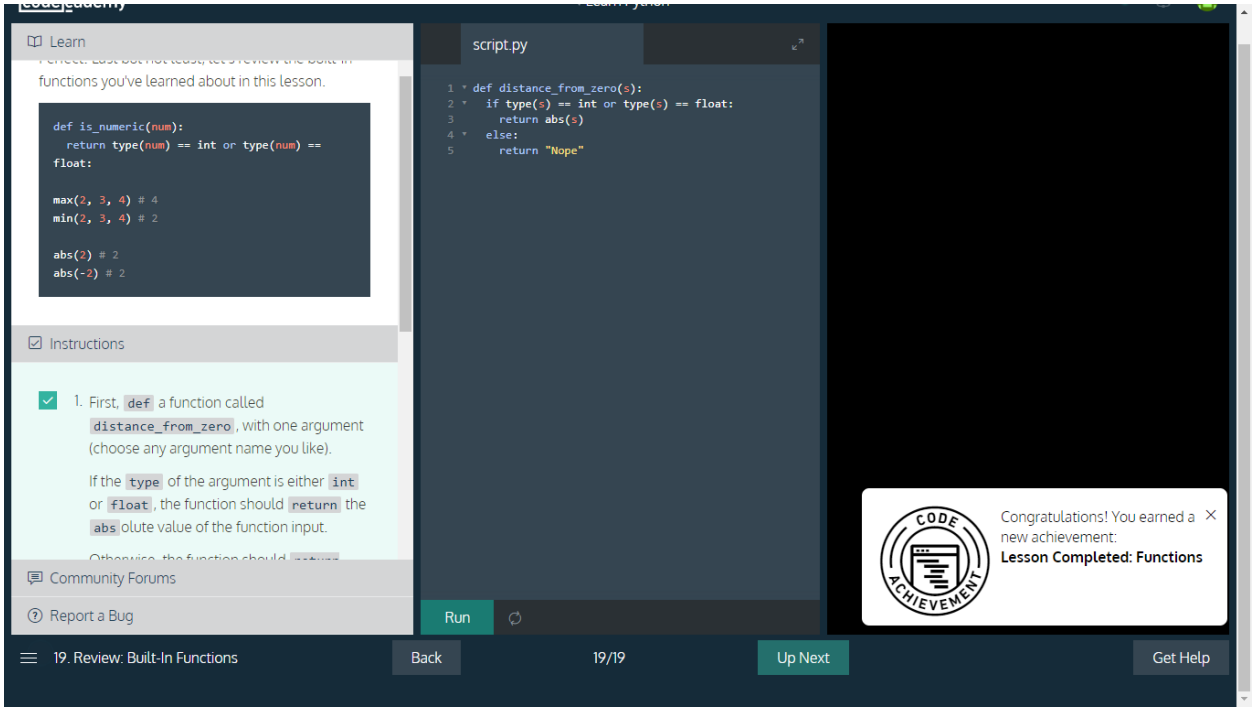
17/19


codecademy
Learn Python
Learn
Instructions
parentheses or the colon!
Then, if the shut_down function receives an s equal to "yes", it should return "Shutting down"
Alternatively, elif s is equal to "no", then the function should return "Shutdown aborted".
Finally, if shut_down gets anything other than those inputs, the function should return "Sorry"
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
Report a Bug
script.py
def shut_down(s):
if s == "yes":
return "Shutting down"
elif s == "no":
return "Shutdown aborted"
else:
return "Sorry"
Run
17. Review: Functions
17. Review: Functions
Back
17/19
Next
Get Help

18/19

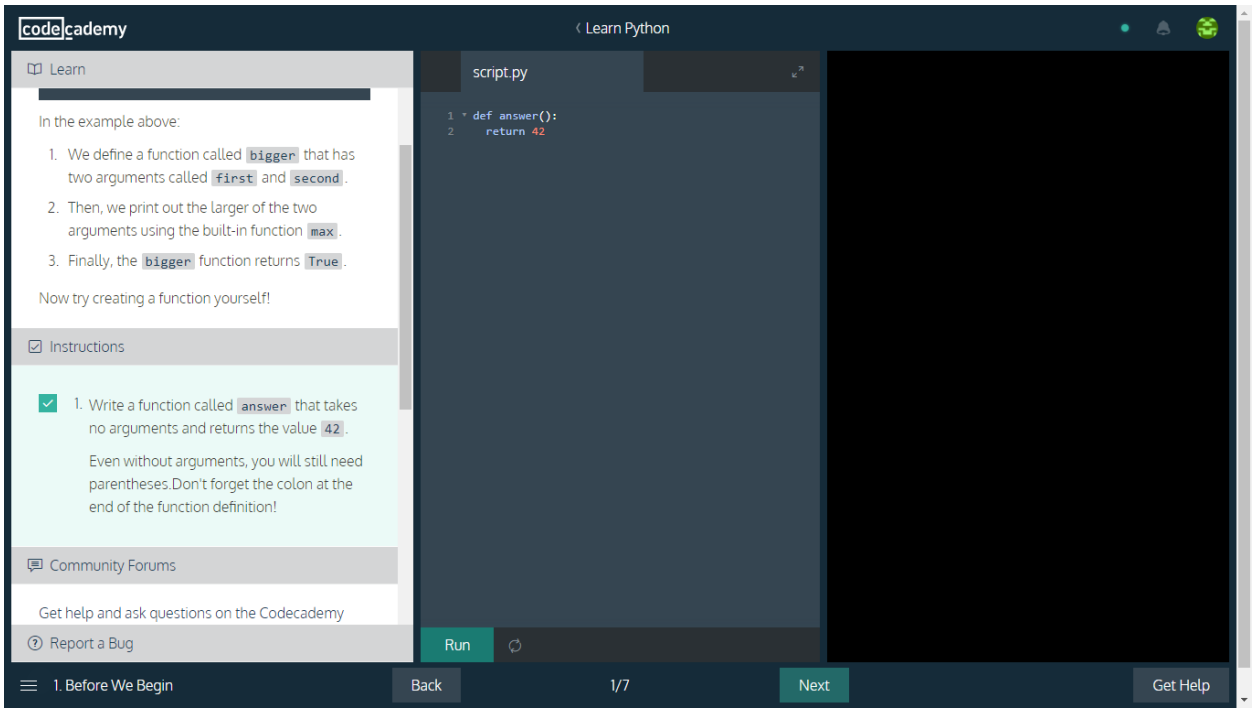

codecademy
Learn Python
Learn
FUNCTIONS
Review: Modules
Good work! Now let's see what you remember about importing modules (and, specifically, what's available in the math module).
Instructions
1. Import the math module in whatever way you prefer. Call its sqrt function on the number 13689 and print that value to the console.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
Report a Bug
script.py
# Print out the types of an integer, a float,
# and a string on separate lines below.
import math
print math.sqrt(13689)
117.0
Run
18. Review: Modules
18. Review: Modules
Back
18/19
Next
Get Help

19/19

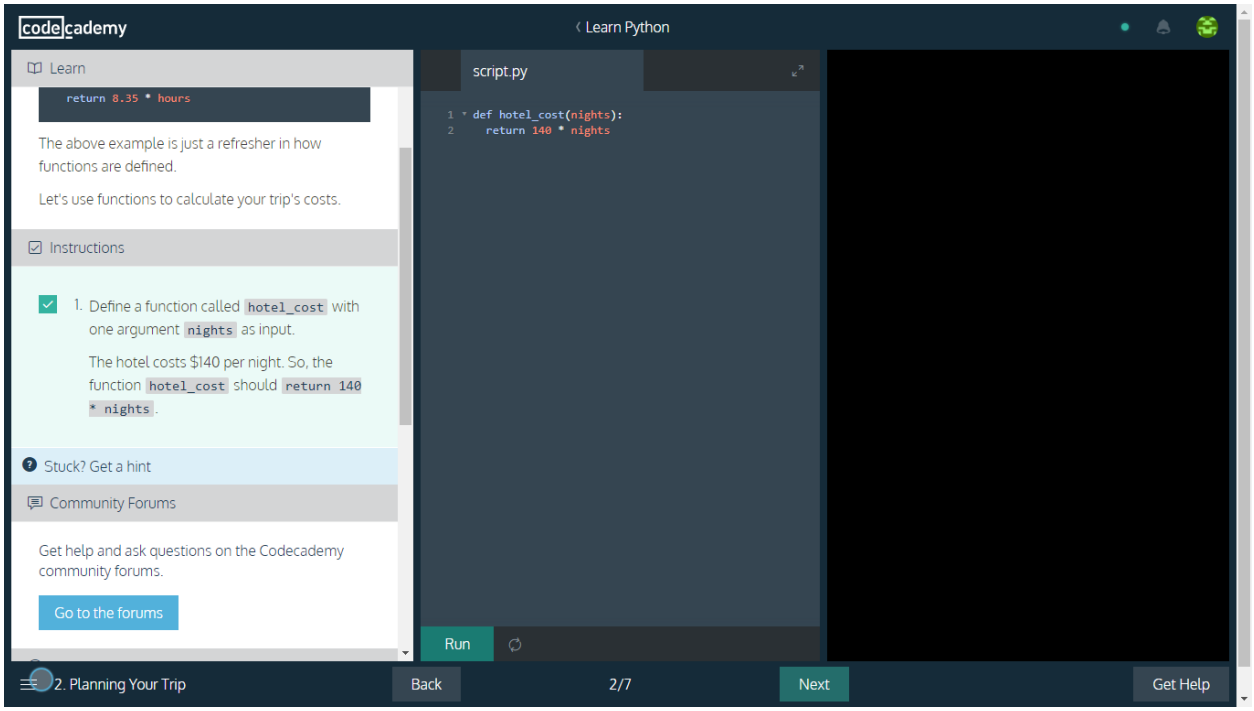


Taking a vacation

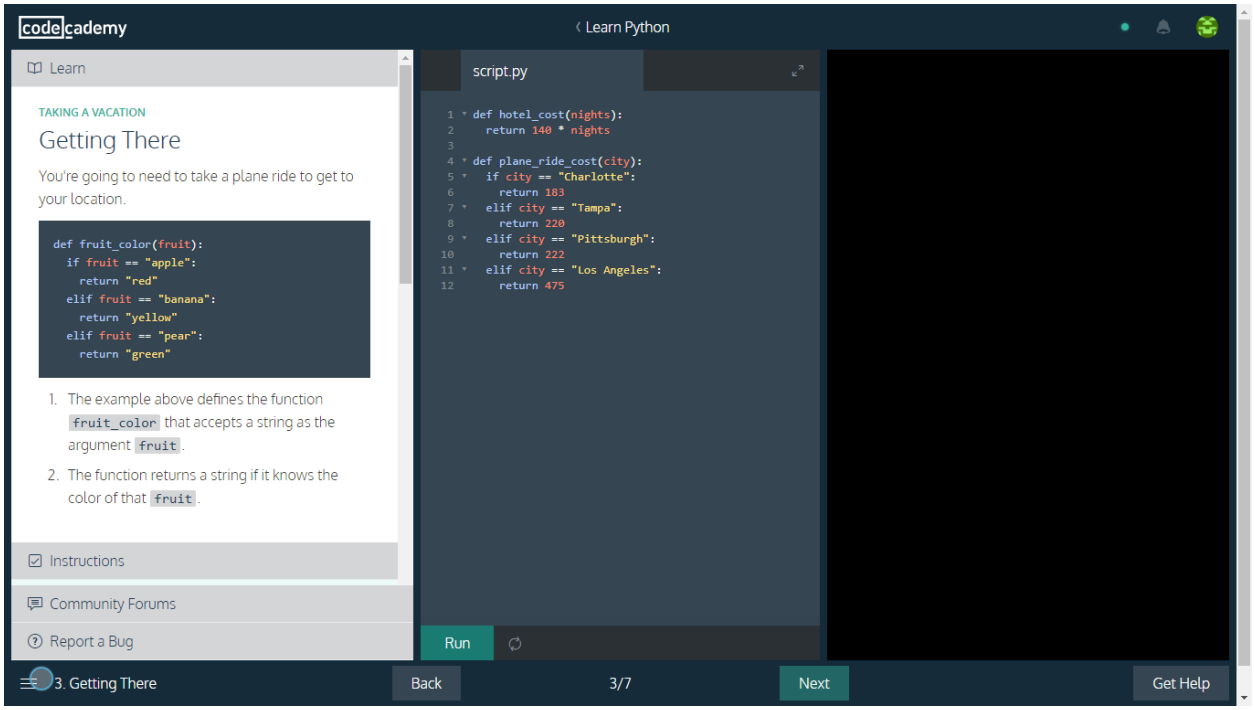
1/7



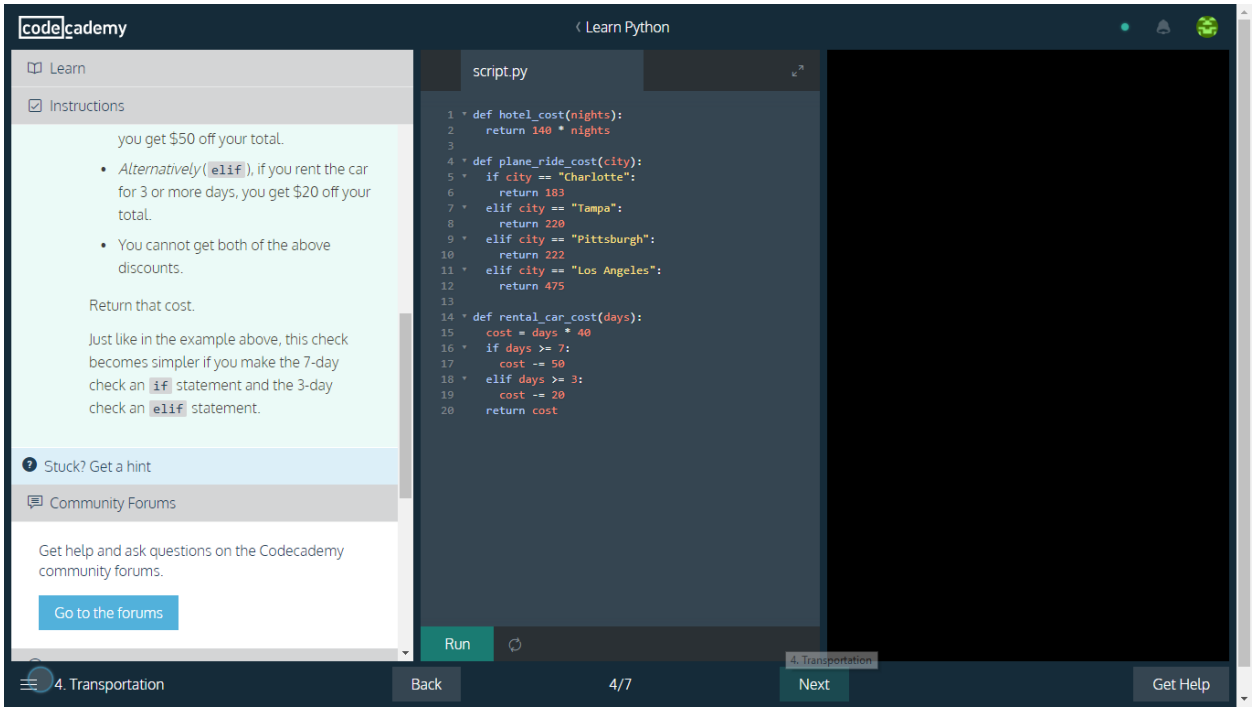
2/7


codecademy
Learn Python
Learn
return 8.35 * hours
The above example is just a refresher in how functions are defined.
Let's use functions to calculate your trip's costs.
Instructions
1. Define a function called hotel_cost with one argument nights as input.
The hotel costs $140 per night. So, the function hotel_cost should return 140 * nights.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
script.py
def hotel_cost(nights):
return 140 * nights
Run
2. Planning Your Trip
Back
2/7
Next
Get Help

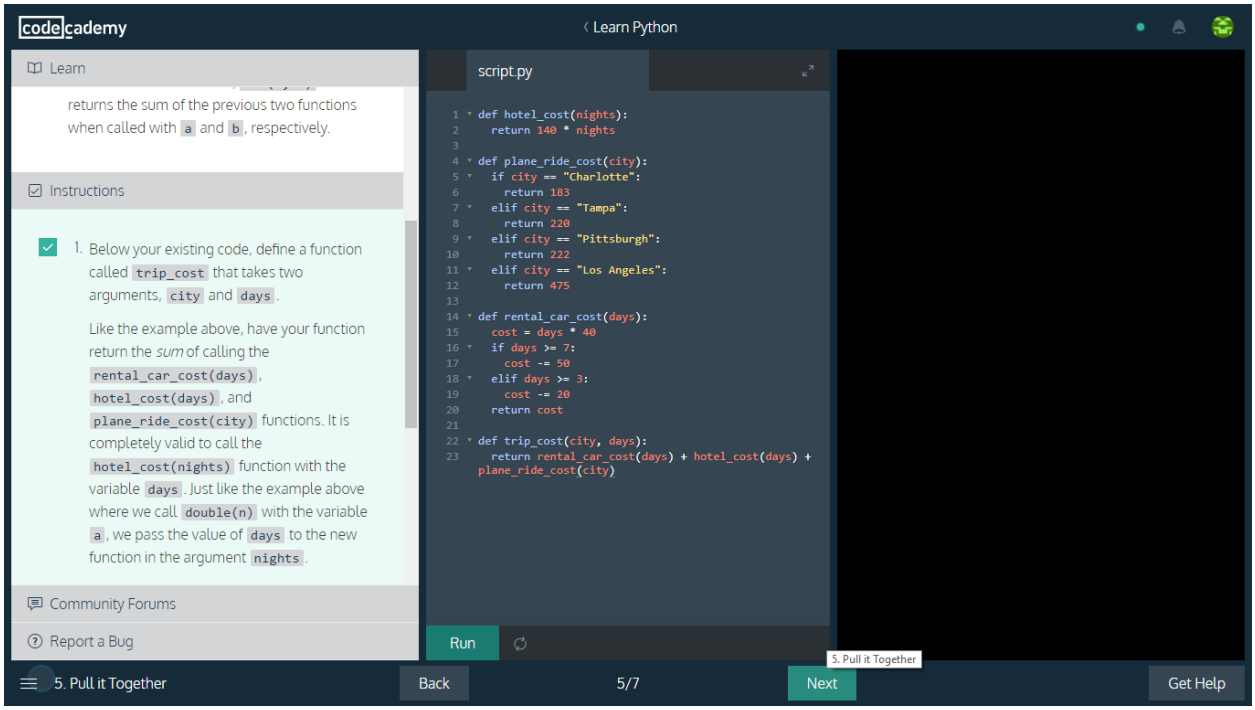
3/7


codecademy
Learn Python
Learn
TAKING A VACATION
Getting There
You're going to need to take a plane ride to get to your location.
def fruit_color(fruit):
if fruit == "apple":
return "red"
elif fruit == "banana":
return "yellow"
elif fruit == "pear":
return "green"
1. The example above defines the function fruit_color that accepts a string as the argument fruit.
2. The function returns a string if it knows the color of that fruit.
Instructions
Community Forums
Report a Bug
script.py
def hotel_cost(nights):
return 140 * nights
def plane_ride_cost(city):
if city == "Charlotte":
return 183
elif city == "Tampa":
return 220
elif city == "Pittsburgh":
return 222
elif city == "Los Angeles":
return 475
Run
3. Getting There
Back
3/7
Next
Get Help

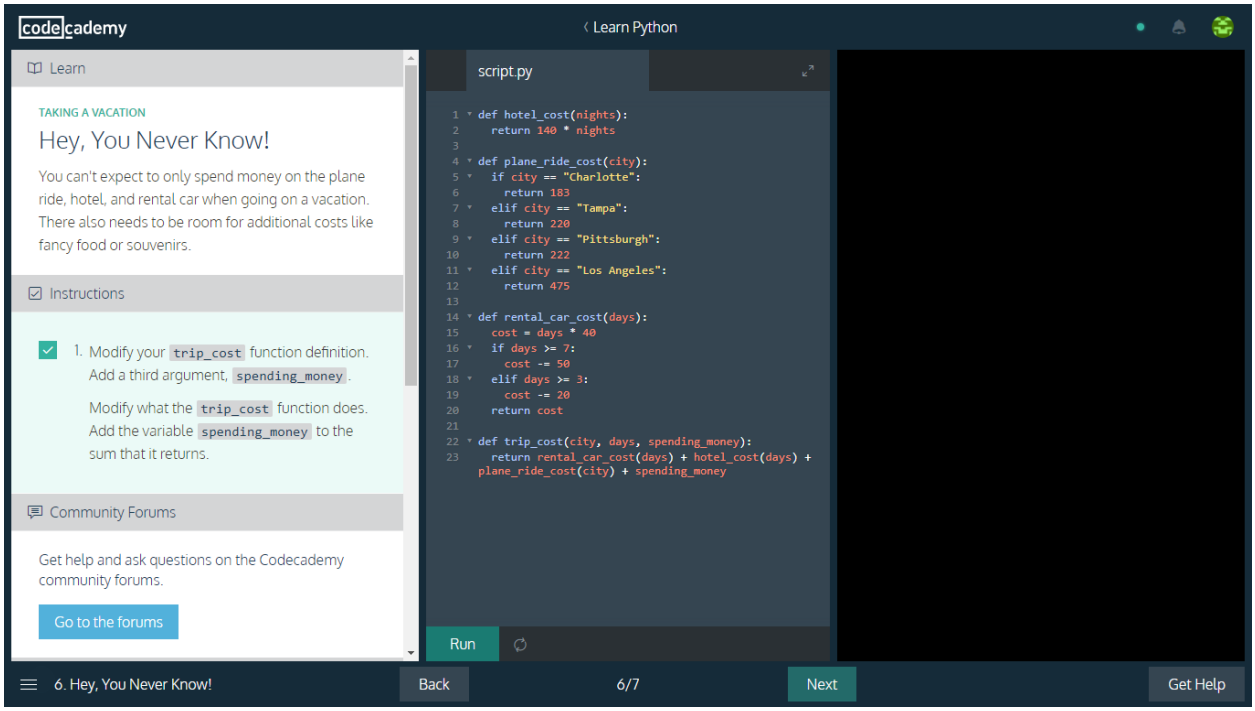
4/7


codecademy
Learn Python
Learn
Instructions
you get $50 off your total.
Alternatively (elif), if you rent the car for 3 or more days, you get $20 off your total.
You cannot get both of the above discounts.
Return that cost.
Just like in the example above, this check becomes simpler if you make the 7-day check an if statement and the 3-day check an elif statement.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
script.py
def hotel_cost(nights):
return 140 * nights
def plane_ride_cost(city):
if city == "Charlotte":
return 183
elif city == "Tampa":
return 220
elif city == "Pittsburgh":
return 222
elif city == "Los Angeles":
return 475
def rental_car_cost(days):
cost = days * 40
if days >= 7:
cost -= 50
elif days >= 3:
cost -= 20
return cost
Run
4. Transportation
Back
4/7
Next
Get Help

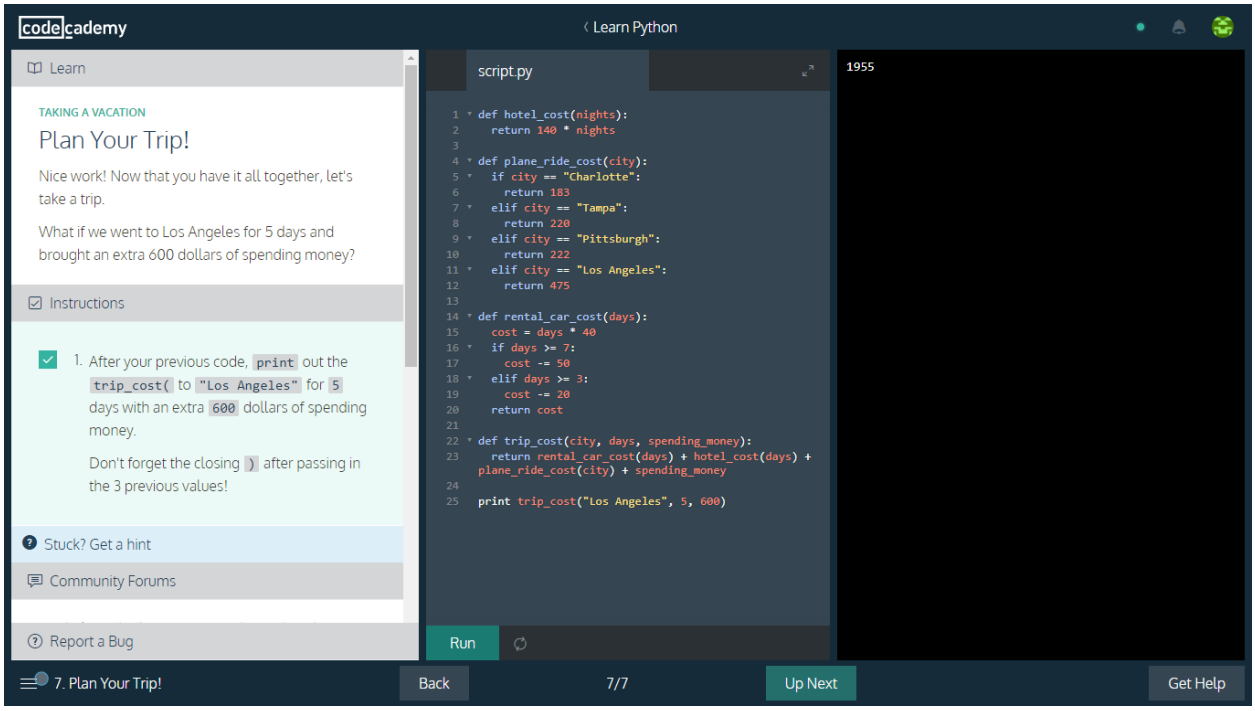
5/7


codecademy
Learn Python
Learn
returns the sum of the previous two functions when called with a and b, respectively.
Instructions
1. Below your existing code, define a function called trip_cost that takes two arguments, city and days.
Like the example above, have your function return the sum of calling the rental_car_cost(days), hotel_cost(days), and plane_ride_cost(city) functions. It is completely valid to call the hotel_cost(nights) function with the variable days. Just like the example above where we call double(n) with the variable a, we pass the value of days to the new function in the argument nights.
Community Forums
Report a Bug
script.py
def hotel_cost(nights):
return 140 * nights
def plane_ride_cost(city):
if city == "Charlotte":
return 183
elif city == "Tampa":
return 220
elif city == "Pittsburgh":
return 222
elif city == "Los Angeles":
return 475
def rental_car_cost(days):
cost = days * 40
if days >= 7:
cost -= 50
elif days >= 3:
cost -= 20
return cost
def trip_cost(city, days):
return rental_car_cost(days) + hotel_cost(days) + plane_ride_cost(city)
Run
5. Pull it Together
Back
5/7
Next
Get Help

6/7



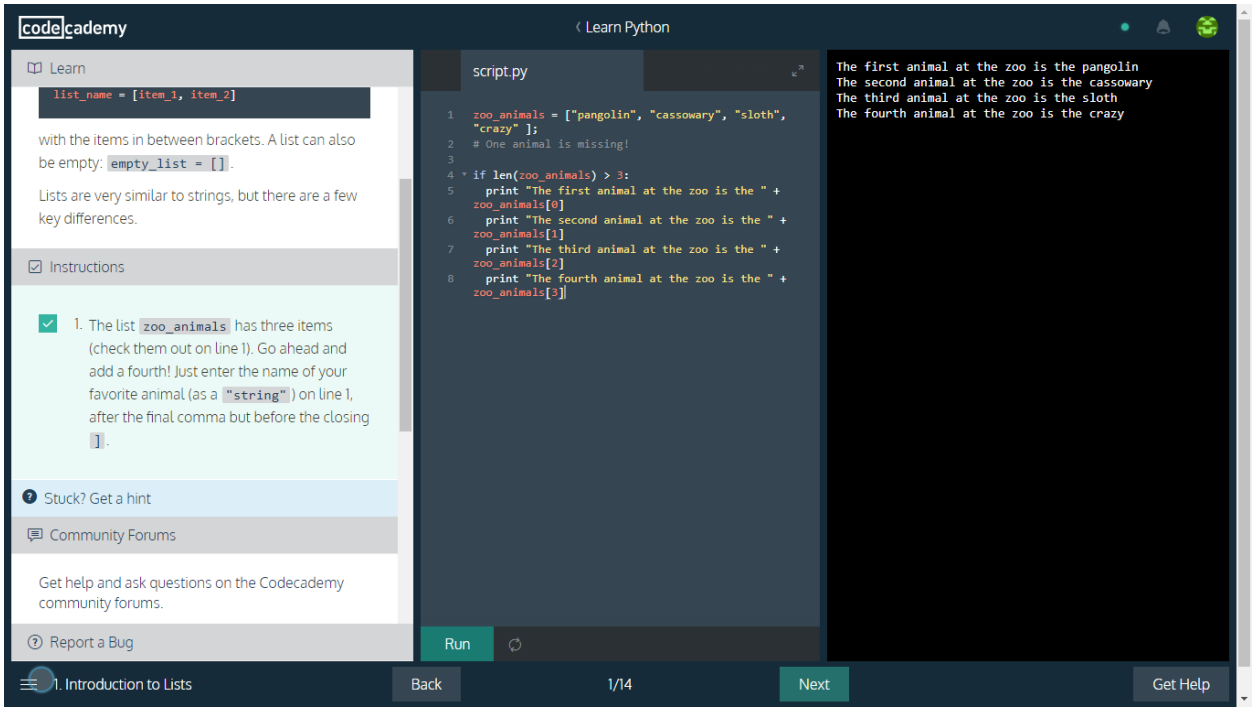
7/7



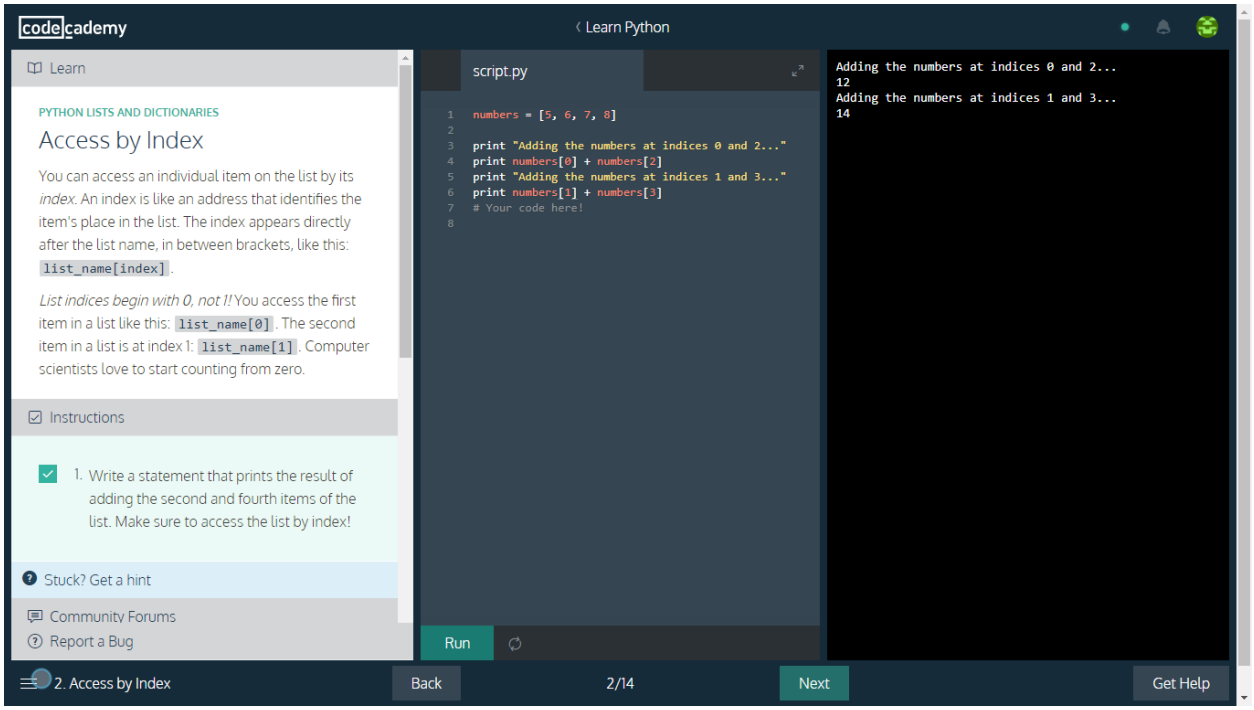
5. Lists & Dictionaries

A

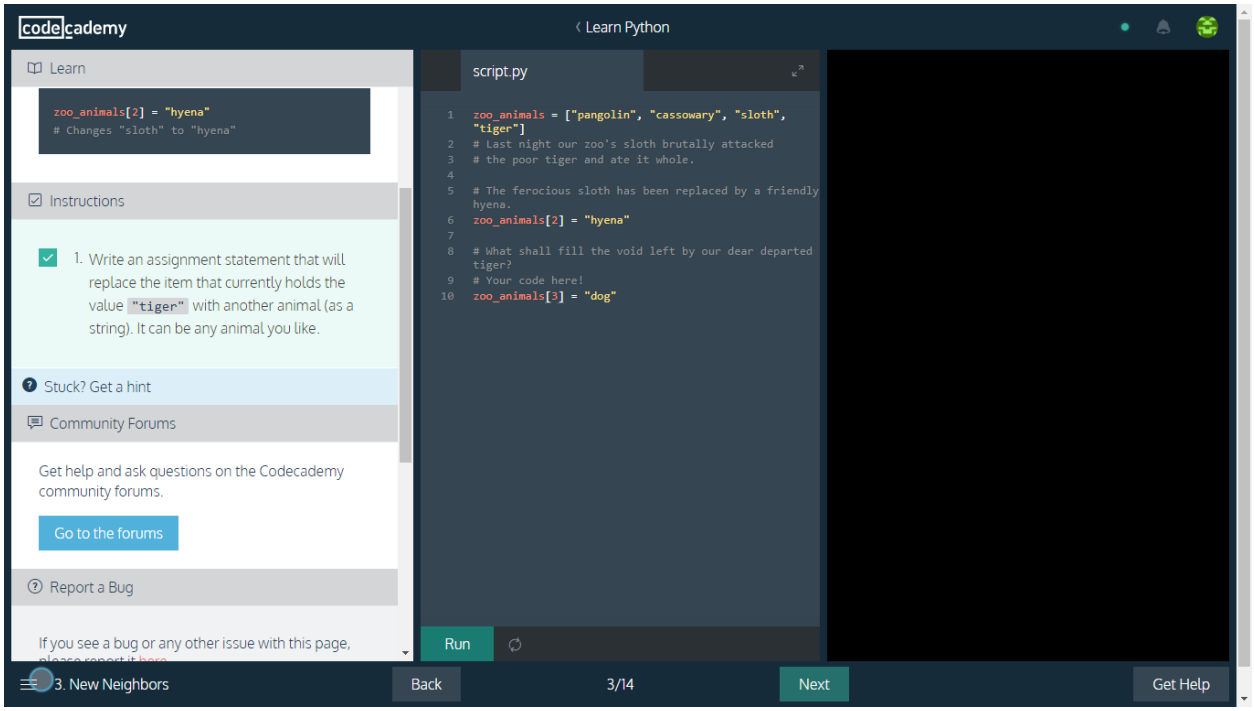
1/14


codecademy
Learn Python
Learn
list_name = [item_1, item_2]
with the items in between brackets. A list can also be empty: empty_list = [].
Lists are very similar to strings, but there are a few key differences.
Instructions
1. The list zoo_animals has three items (check them out on line 1). Go ahead and add a fourth! Just enter the name of your favorite animal (as a "string") on line 1, after the final comma but before the closing].
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Report a Bug
script.py
zoo_animals = ["pangolin", "cassowary", "sloth", "crazy"];
# One animal is missing!
if len(zoo_animals) > 3:
print "The first animal at the zoo is the " + zoo_animals[0]
print "The second animal at the zoo is the " + zoo_animals[1]
print "The third animal at the zoo is the " + zoo_animals[2]
print "The fourth animal at the zoo is the " + zoo_animals[3]
Run
The first animal at the zoo is the pangolin
The second animal at the zoo is the cassowary
The third animal at the zoo is the sloth
The fourth animal at the zoo is the crazy
1. Introduction to Lists
Back
1/14
Next
Get Help

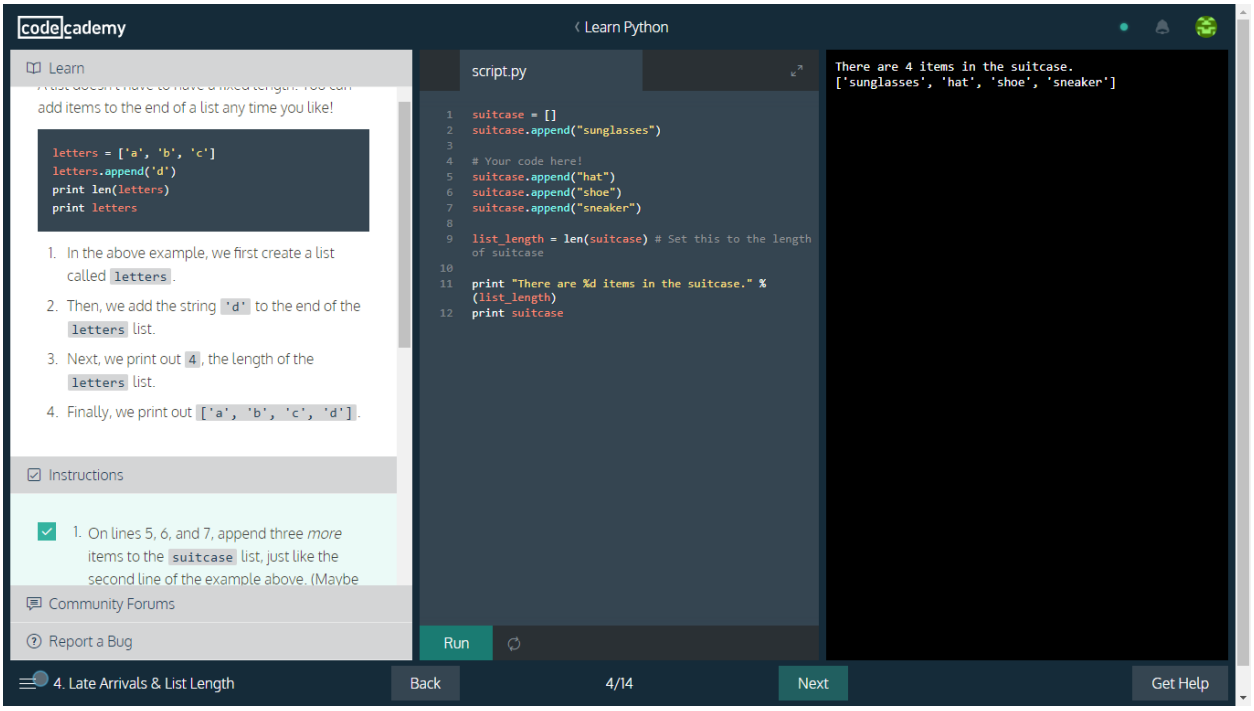
2/14


codecademy
Learn Python
Learn
PYTHON LISTS AND DICTIONARIES
Access by Index
You can access an individual item on the list by its index. An index is like an address that identifies the item's place in the list. The index appears directly after the list name, in between brackets, like this: list_name[index].
List indices begin with 0, not 1! You access the first item in a list like this: list_name[0]. The second item in a list is at index 1: list_name[1]. Computer scientists love to start counting from zero.
Instructions
1. Write a statement that prints the result of adding the second and fourth items of the list. Make sure to access the list by index!
Stuck? Get a hint
Community Forums
Report a Bug
script.py
numbers = [5, 6, 7, 8]
print "Adding the numbers at indices 0 and 2..."
print numbers[0] + numbers[2]
print "Adding the numbers at indices 1 and 3..."
print numbers[1] + numbers[3]
Your code here!
Run
Adding the numbers at indices 0 and 2...
12
Adding the numbers at indices 1 and 3...
14
2. Access by Index
Back
2/14
Next
Get Help

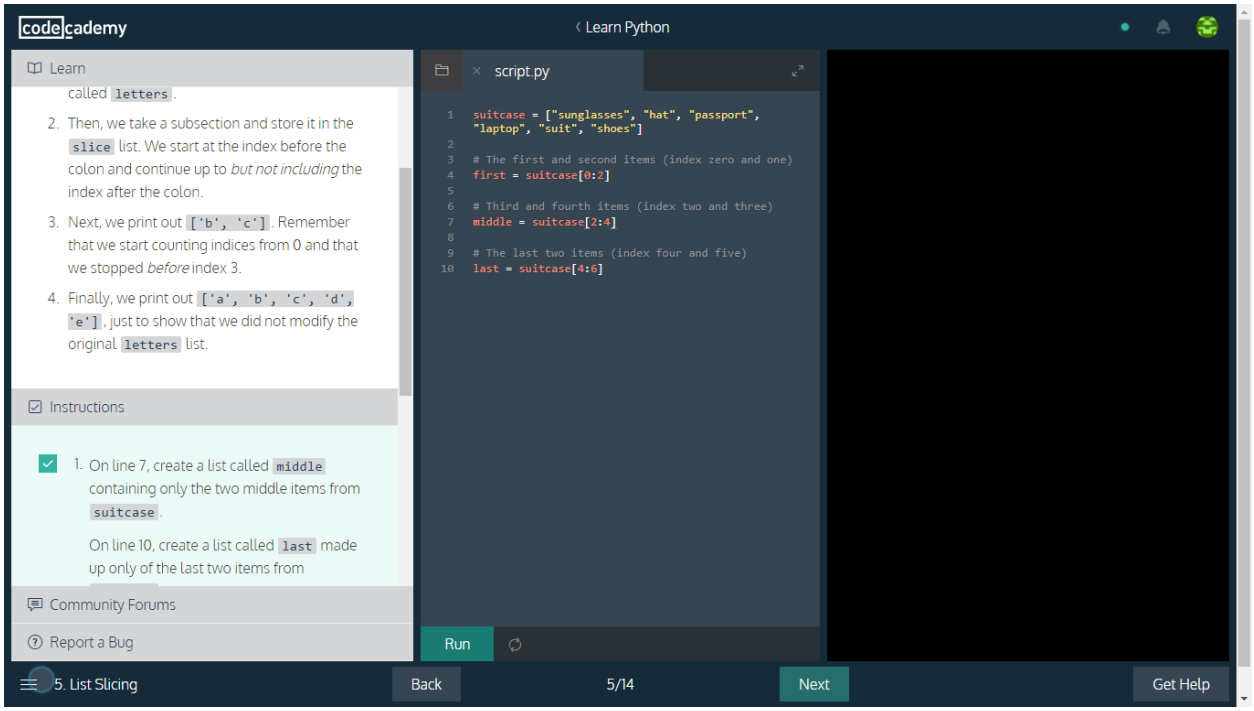
3/14


codecademy
Learn Python
Learn
zoo_animals[2] = "hyena"
# Changes "sloth" to "hyena"
Instructions
1. Write an assignment statement that will replace the item that currently holds the value "tiger" with another animal (as a string). It can be any animal you like.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
Report a Bug
If you see a bug or any other issue with this page,
script.py
zoo_animals = ["pangolin", "cassowary", "sloth", "tiger"]
# Last night our zoo's sloth brutally attacked
# the poor tiger and ate it whole.
# The ferocious sloth has been replaced by a friendly hyena.
zoo_animals[2] = "hyena"
# What shall fill the void left by our dear departed tiger?
# Your code here!
zoo_animals[3] = "dog"
Run
3. New Neighbors
Back
3/14
Next
Get Help

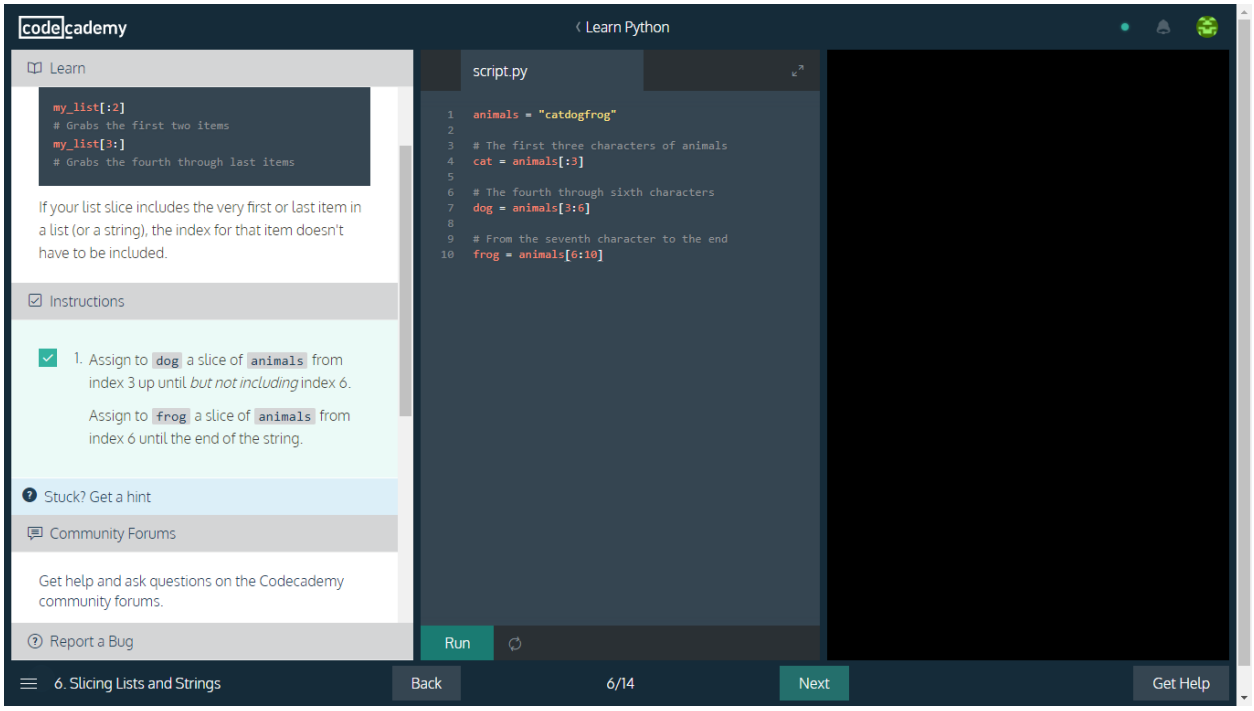
4/14


codecademy
Learn Python
Learn
add items to the end of a list any time you like!
letters = ['a', 'b', 'c']
letters.append('d')
print len(letters)
print letters
1. In the above example, we first create a list called letters.
2. Then, we add the string 'd' to the end of the letters list.
3. Next, we print out 4, the length of the letters list.
4. Finally, we print out ['a', 'b', 'c', 'd'].
Instructions
1. On lines 5, 6, and 7, append three more items to the suitcase list, just like the second line of the example above. (Maybe
Community Forums
Report a Bug
script.py
suitcase = []
suitcase.append("sunglasses")
# Your code here!
suitcase.append("hat")
suitcase.append("shoe")
suitcase.append("sneaker")
list_length = len(suitcase) # Set this to the length of suitcase
print "There are %d items in the suitcase." % (list_length)
print suitcase
There are 4 items in the suitcase.
['sunglasses', 'hat', 'shoe', 'sneaker']
Run
4. Late Arrivals & List Length
Back
4/14
Next
Get Help

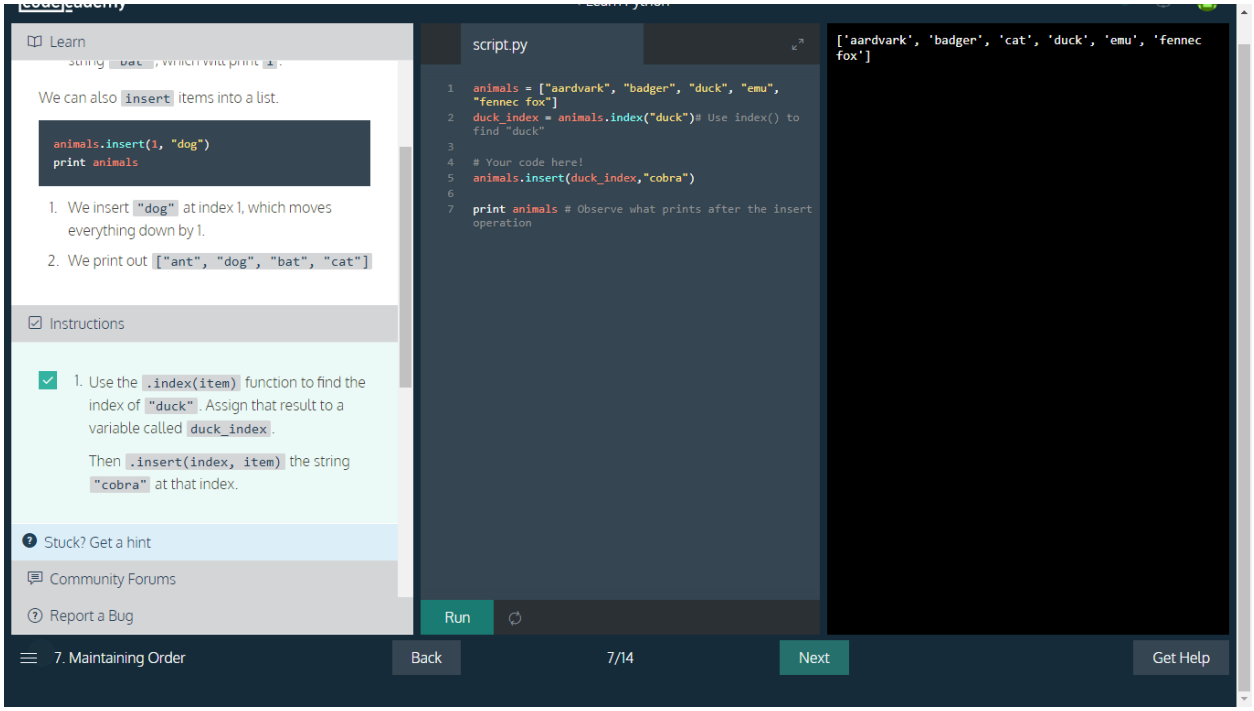
5/14


codecademy
Learn Python
Learn
called letters.
2. Then, we take a subsection and store it in the slice list. We start at the index before the colon and continue up to but not including the index after the colon.
3. Next, we print out ['b', 'c']. Remember that we start counting indices from 0 and that we stopped before index 3.
4. Finally, we print out ['a', 'b', 'c', 'd', 'e'], just to show that we did not modify the original letters list.
Instructions
1. On line 7, create a list called middle containing only the two middle items from suitcase.
On line 10, create a list called last made up only of the last two items from
Community Forums
Report a Bug
script.py
suitcase = ["sunglasses", "hat", "passport", "laptop", "suit", "shoes"]
# The first and second items (index zero and one)
first = suitcase[0:2]
# Third and fourth items (index two and three)
middle = suitcase[2:4]
# The last two items (index four and five)
last = suitcase[4:6]
Run
5. List Slicing
Back
5/14
Next
Get Help

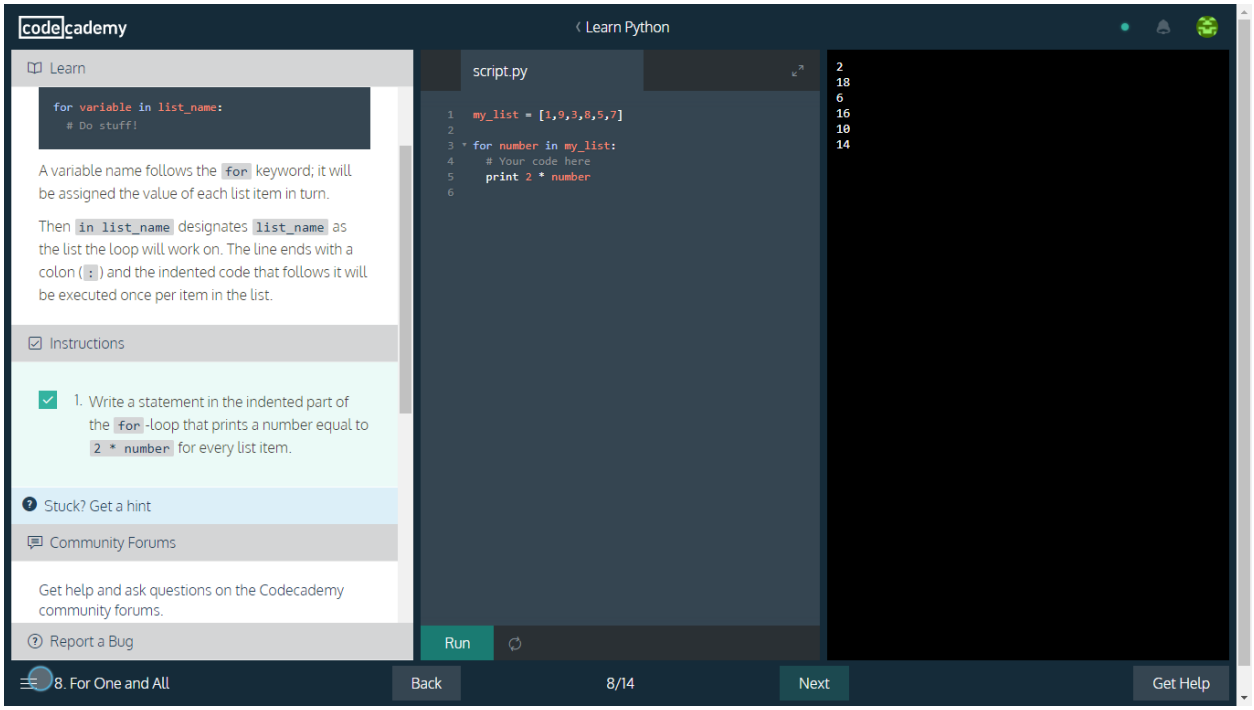
6/14


codecademy
Learn Python
Learn
my_list[:2]
Grabs the first two items
my_list[3:]
Grabs the fourth through last items
If your list slice includes the very first or last item in a list (or a string), the index for that item doesn't have to be included.
Instructions
1. Assign to dog a slice of animals from index 3 up until but not including index 6.
Assign to frog a slice of animals from index 6 until the end of the string.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Report a Bug
script.py
animals = "catdogfrog"
# The first three characters of animals
cat = animals[:3]
# The fourth through sixth characters
dog = animals[3:6]
# From the seventh character to the end
frog = animals[6:10]
Run
6. Slicing Lists and Strings
Back
6/14
Next
Get Help

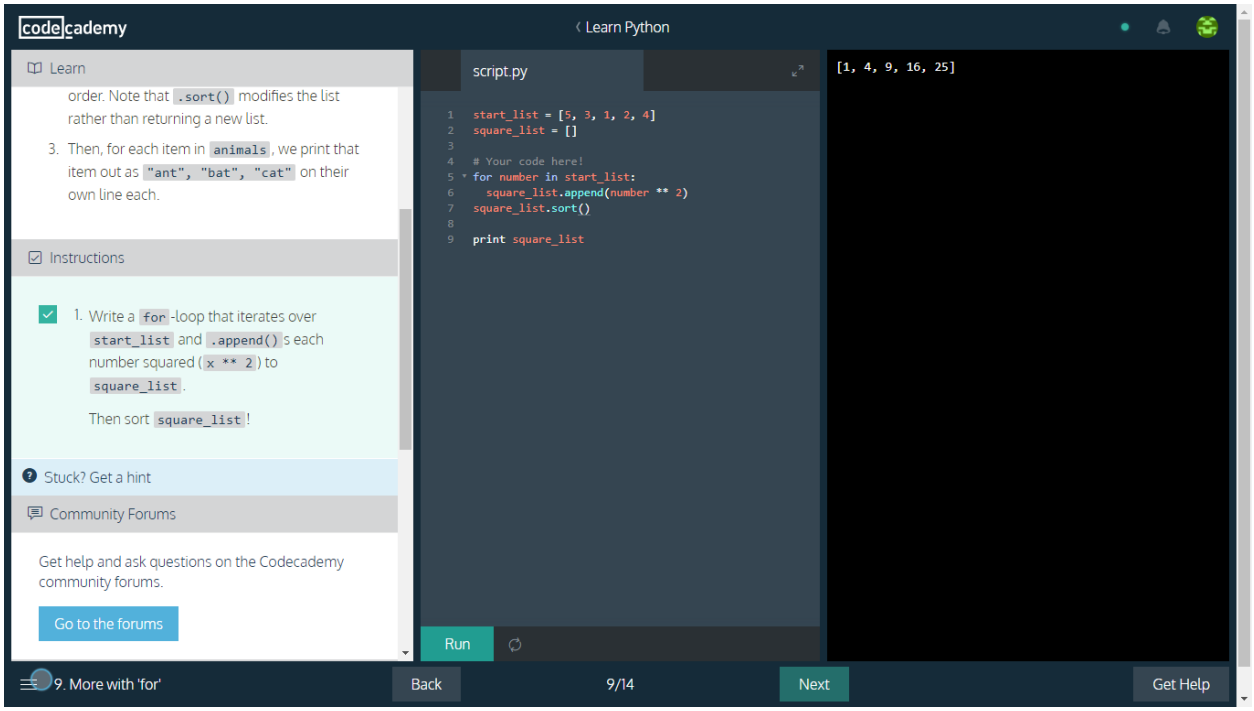
7/14


Learn
We can also insert items into a list.
animals.insert(1, "dog")
print animals
1. We insert "dog" at index 1, which moves everything down by 1.
2. We print out ["ant", "dog", "bat", "cat"]
Instructions
1. Use the .index(item) function to find the index of "duck". Assign that result to a variable called duck_index.
Then .insert(index, item) the string "cobra" at that index.
Stuck? Get a hint
Community Forums
Report a Bug
script.py
animals = ["aardvark", "badger", "duck", "emu", "fennec fox"]
duck_index = animals.index("duck")# Use index() to find "duck"
# Your code here!
animals.insert(duck_index,"cobra")
print animals # Observe what prints after the insert operation
Run
['aardvark', 'badger', 'cat', 'duck', 'emu', 'fennec fox']
7. Maintaining Order
Back
7/14
Next
Get Help

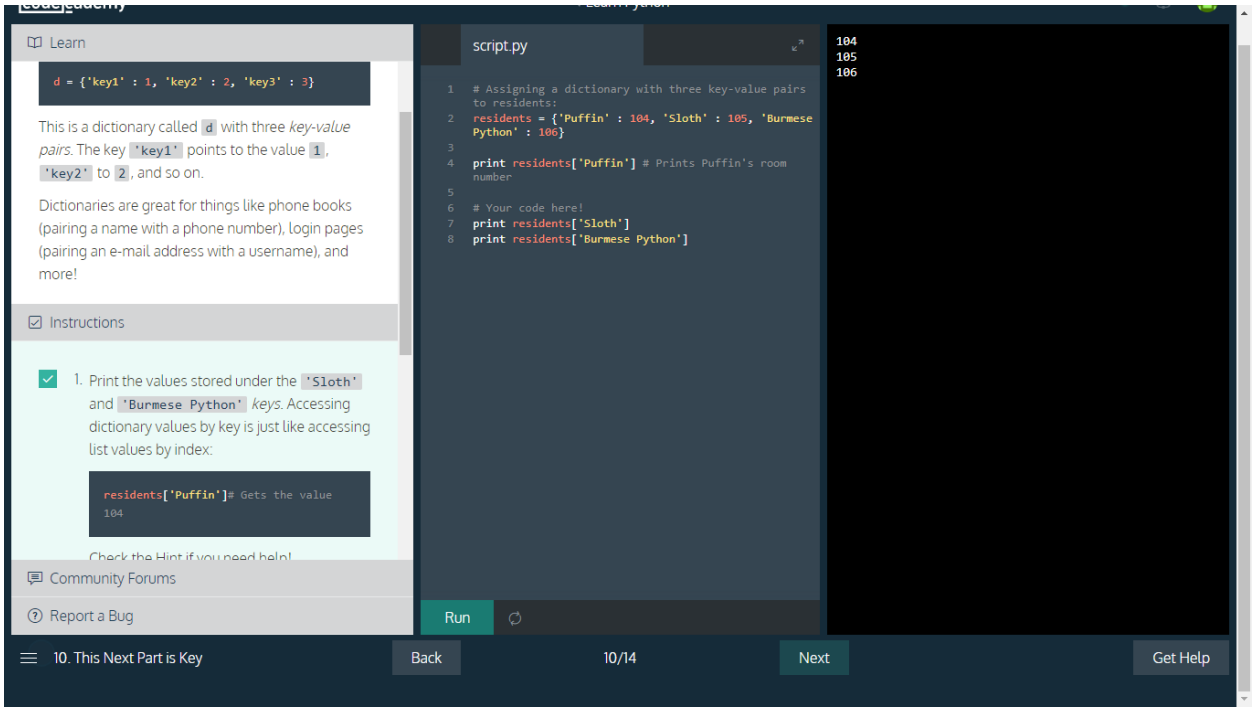
8/14


codecademy
Learn Python
Learn
for variable in list_name:
# Do stuff!
A variable name follows the for keyword; it will be assigned the value of each list item in turn.
Then in list_name designates list_name as the list the loop will work on. The line ends with a colon (:) and the indented code that follows it will be executed once per item in the list.
Instructions
1. Write a statement in the indented part of the for-loop that prints a number equal to 2 * number for every list item.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Report a Bug
script.py
my_list = [1,9,3,8,5,7]
for number in my_list:
# Your code here
print 2 * number
Run
2
18
6
16
10
14
8. For One and All
Back
8/14
Next
Get Help

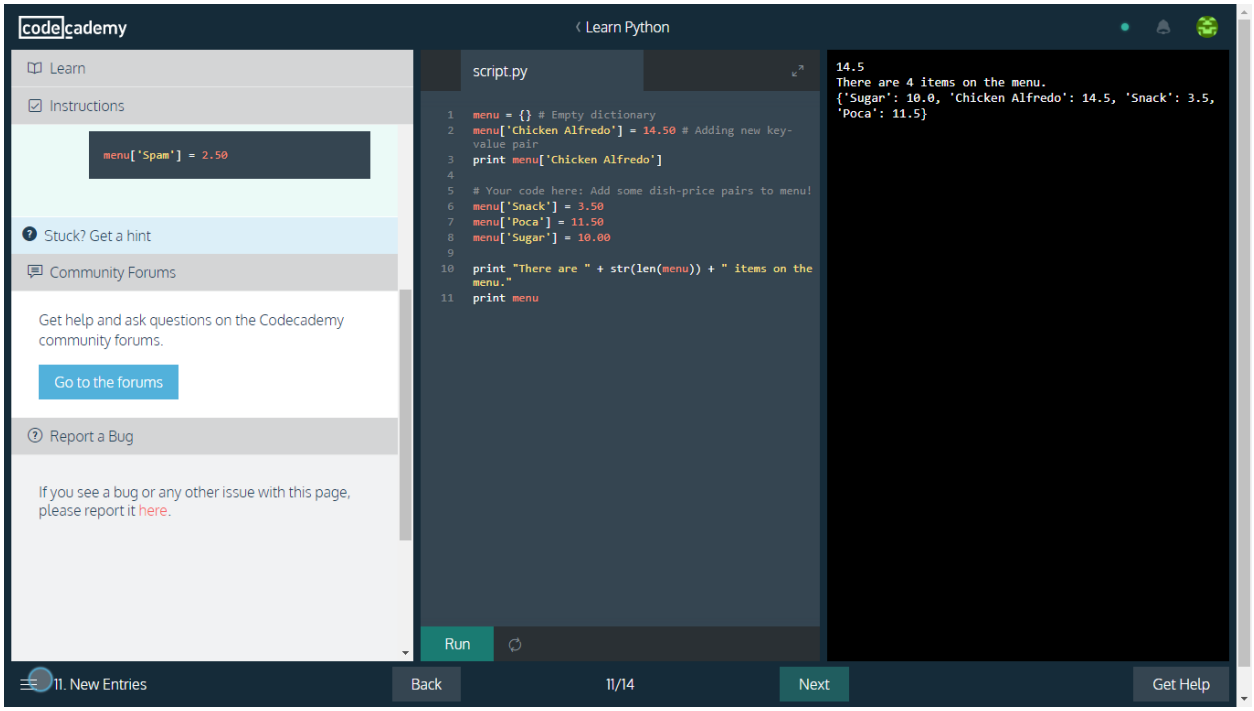
9/14


codecademy
Learn Python
Learn
order. Note that .sort() modifies the list rather than returning a new list.
3. Then, for each item in animals, we print that item out as "ant", "bat", "cat" on their own line each.
Instructions
1. Write a for -loop that iterates over start_list and .append()s each number squared (x ** 2) to square_list.
Then sort square_list!
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
script.py
start_list = [5, 3, 1, 2, 4]
square_list = []
# Your code here!
for number in start_list:
square_list.append(number ** 2)
square_list.sort()
print square_list
[1, 4, 9, 16, 25]
Run
9. More with 'for'
Back
9/14
Next
Get Help

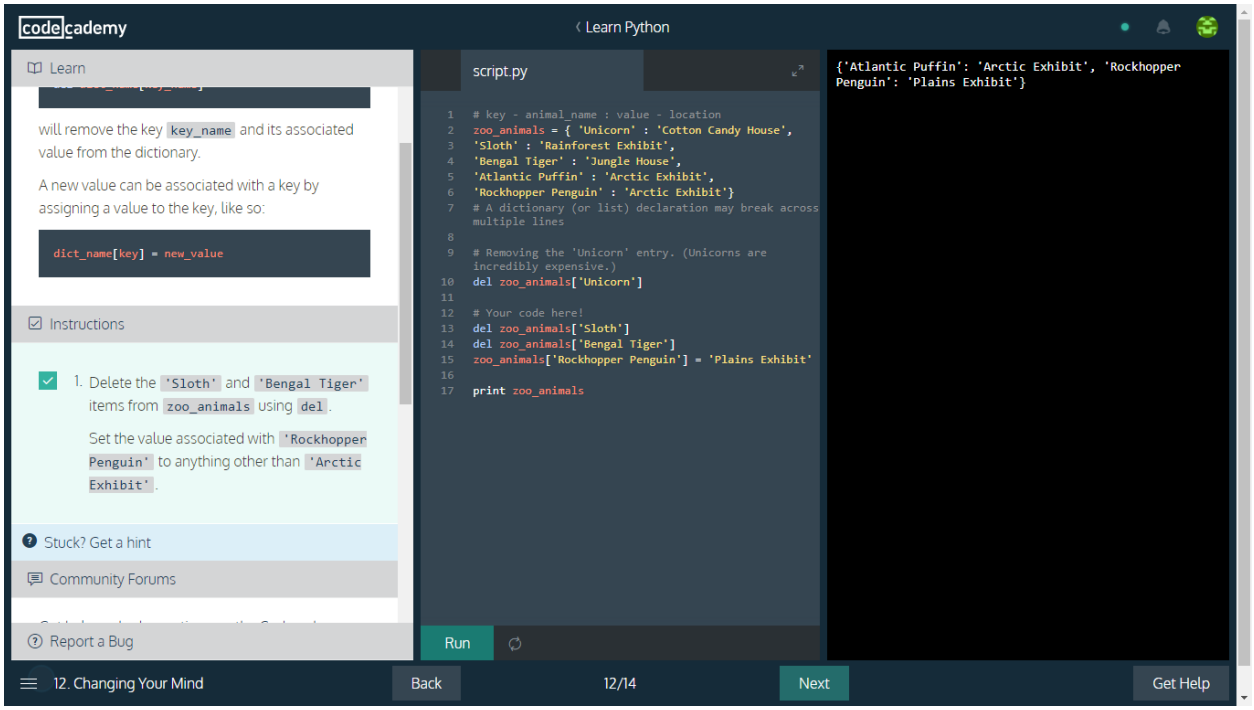
10/14


Learn
d = {'key1' : 1, 'key2' : 2, 'key3' : 3}
This is a dictionary called d with three key-value pairs. The key 'key1' points to the value 1, 'key2' to 2, and so on.
Dictionaries are great for things like phone books (pairing a name with a phone number), login pages (pairing an e-mail address with a username), and more!
Instructions
1. Print the values stored under the 'Sloth' and 'Burmese Python' keys. Accessing dictionary values by key is just like accessing list values by index:
residents['Puffin']# Gets the value 104
Community Forums
Report a Bug
script.py
# Assigning a dictionary with three key-value pairs to residents:
residents = {'Puffin' : 104, 'Sloth' : 105, 'Burmese Python' : 106}
print residents['Puffin'] # Prints Puffin's room number
Your code here!
print residents['Sloth']
print residents['Burmese Python']
104
105
106
Run
10. This Next Part is Key
Back
10/14
Next
Get Help

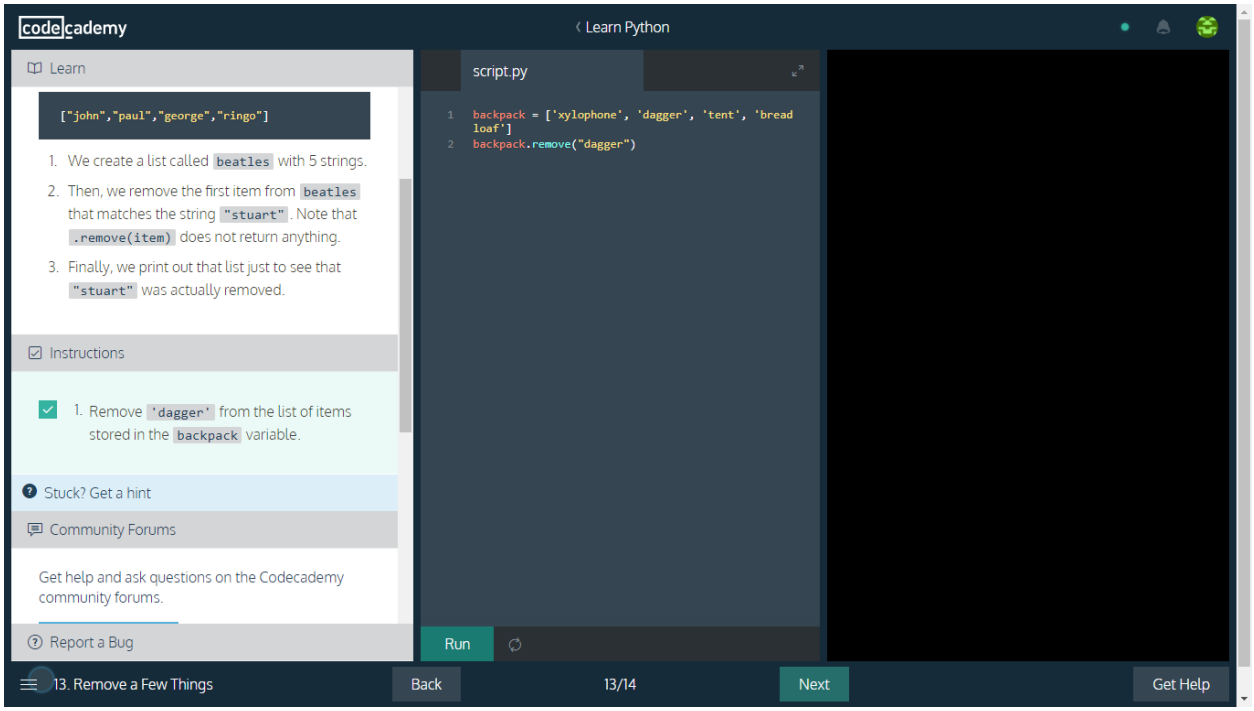
11/14


codecademy
Learn Python
Learn
Instructions
menu['Spam'] = 2.50
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
Report a Bug
If you see a bug or any other issue with this page, please report it here.
script.py
menu = {} # Empty dictionary
menu['Chicken Alfredo'] = 14.50 # Adding new key-value pair
print menu['Chicken Alfredo']
Your code here: Add some dish-price pairs to menu!
menu['Snack'] = 3.50
menu['Poca'] = 11.50
menu['Sugar'] = 10.00
print "There are " + str(len(menu)) + " items on the menu."
print menu
Run
14.5
There are 4 items on the menu.
{'Sugar': 10.0, 'Chicken Alfredo': 14.5, 'Snack': 3.5, 'Poca': 11.5}
11. New Entries
Back
11/14
Next
Get Help

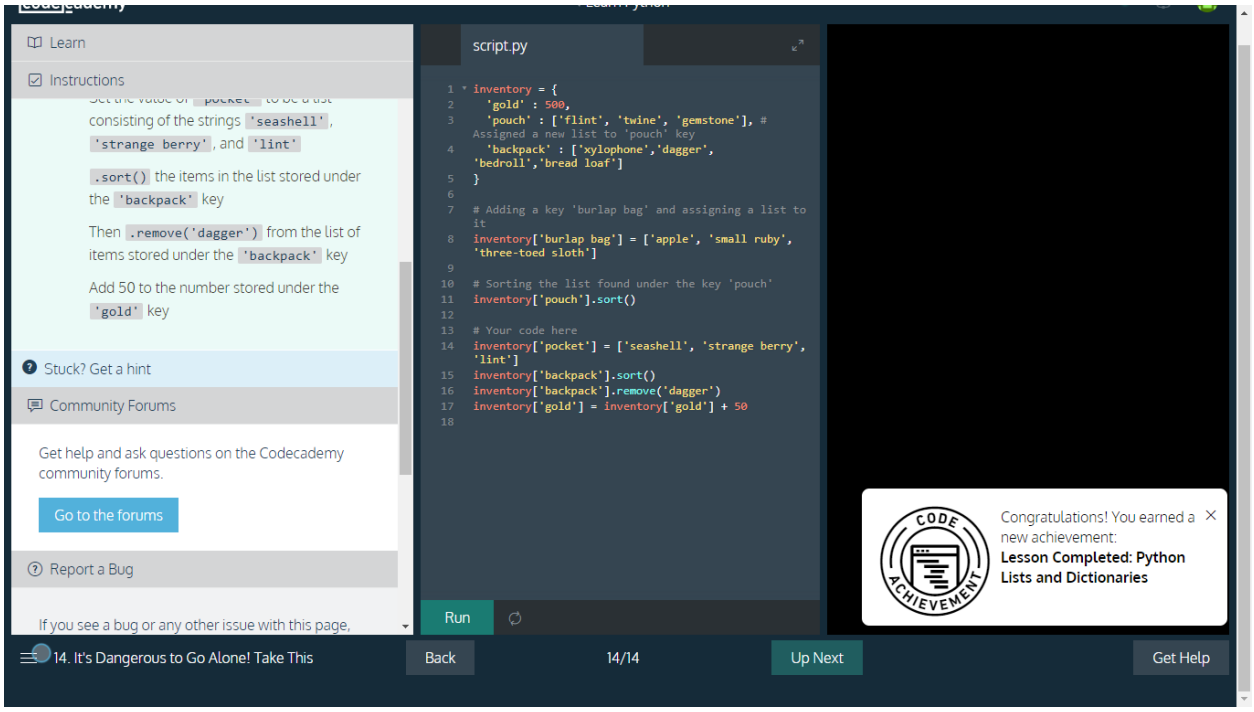
12/14


codecademy
Learn Python
Learn
will remove the key key_name and its associated value from the dictionary.
A new value can be associated with a key by assigning a value to the key, like so:
dict_name[key] = new_value
Instructions
1. Delete the 'Sloth' and 'Bengal Tiger' items from zoo_animals using del.
Set the value associated with 'Rockhopper Penguin' to anything other than 'Arctic Exhibit'.
Stuck? Get a hint
Community Forums
Report a Bug
script.py
# key - animal_name : value - location
zoo_animals = { 'Unicorn' : 'Cotton Candy House',
'Sloth' : 'Rainforest Exhibit',
'Bengal Tiger' : 'Jungle House',
'Atlantic Puffin' : 'Arctic Exhibit',
'Rockhopper Penguin' : 'Arctic Exhibit'}
A dictionary (or list) declaration may break across multiple lines
Removing the 'Unicorn' entry. (Unicorns are incredibly expensive.)
del zoo_animals['Unicorn']
Your code here!
del zoo_animals['Sloth']
del zoo_animals['Bengal Tiger']
zoo_animals['Rockhopper Penguin'] = 'Plains Exhibit'
print zoo_animals
Run
{'Atlantic Puffin': 'Arctic Exhibit', 'Rockhopper Penguin': 'Plains Exhibit'}
12. Changing Your Mind
Back
12/14
Next
Get Help

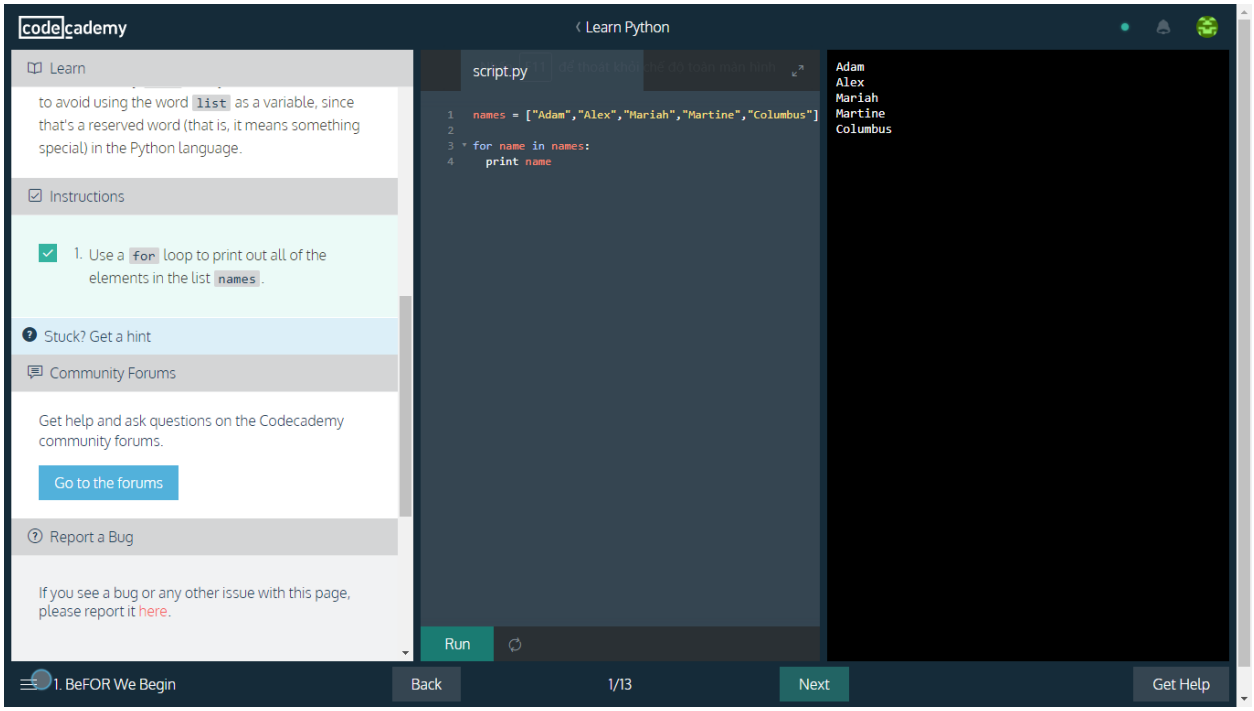
13/14


codecademy
Learn Python
Learn
["john","paul","george","ringo"]
1. We create a list called beatles with 5 strings.
2. Then, we remove the first item from beatles that matches the string "stuart". Note that .remove(item) does not return anything.
3. Finally, we print out that list just to see that "stuart" was actually removed.
Instructions
1. Remove 'dagger' from the list of items stored in the backpack variable.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Report a Bug
script.py
backpack = ['xylophone', 'dagger', 'tent', 'bread loaf']
backpack.remove("dagger")
Run
13. Remove a Few Things
Back
13/14
Next
Get Help

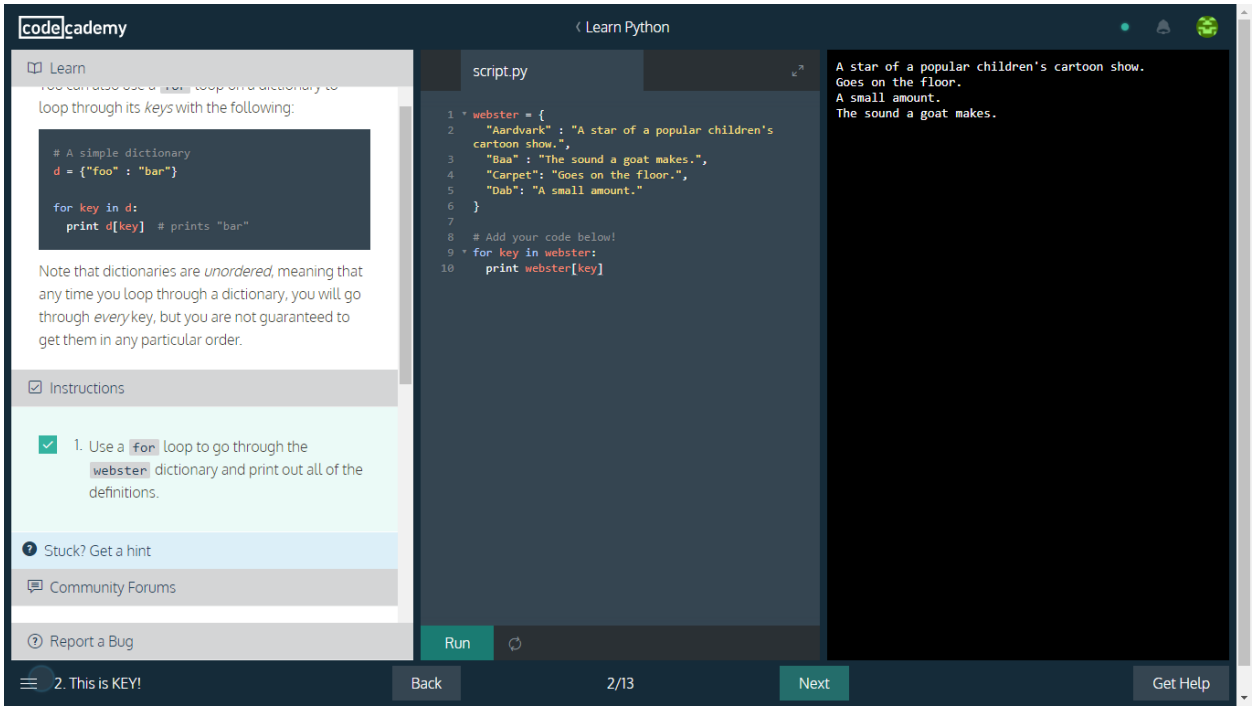
14/14


Learn
Instructions
consisting of the strings 'seashell', 'strange berry', and 'lint'
.sort() the items in the list stored under the 'backpack' key
Then .remove('dagger') from the list of items stored under the 'backpack' key
Add 50 to the number stored under the 'gold' key
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
Report a Bug
If you see a bug or any other issue with this page,
script.py
inventory = {
'gold' : 500,
'pouch' : ['flint', 'twine', 'gemstone'], # Assigned a new list to 'pouch' key
'backpack' : ['xylophone','dagger', 'bedroll','bread loaf']
}
# Adding a key 'burlap bag' and assigning a list to it
inventory['burlap bag'] = ['apple', 'small ruby', 'three-toed sloth']
Sorting the list found under the key 'pouch'
inventory['pouch'].sort()
Your code here
inventory['pocket'] = ['seashell', 'strange berry', 'lint']
inventory['backpack'].sort()
inventory['backpack'].remove('dagger')
inventory['gold'] = inventory['gold'] + 50
Run
CODE ACHIEVEMENT
Congratulations! You earned a new achievement:
Lesson Completed: Python Lists and Dictionaries
14. It's Dangerous to Go Alone! Take This
Back
14/14
Up Next
Get Help

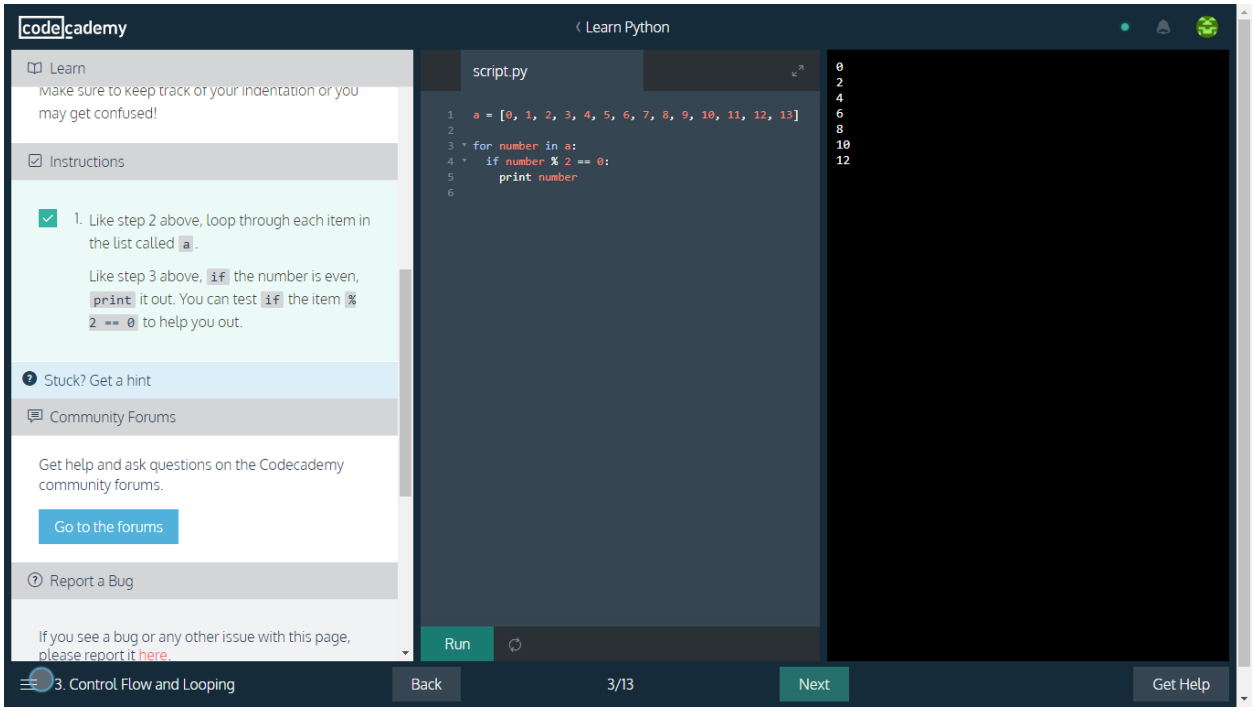
B
1/13


codecademy
Learn Python
Learn
to avoid using the word list as a variable, since that's a reserved word (that is, it means something special) in the Python language.
Instructions
1. Use a for loop to print out all of the elements in the list names.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
Report a Bug
If you see a bug or any other issue with this page, please report it here.
script.py
names = ["Adam","Alex","Mariah","Martine","Columbus"]
for name in names:
print name
Run
Adam
Alex
Mariah
Martine
Columbus
1. BeFOR We Begin
Back
1/13
Next
Get Help

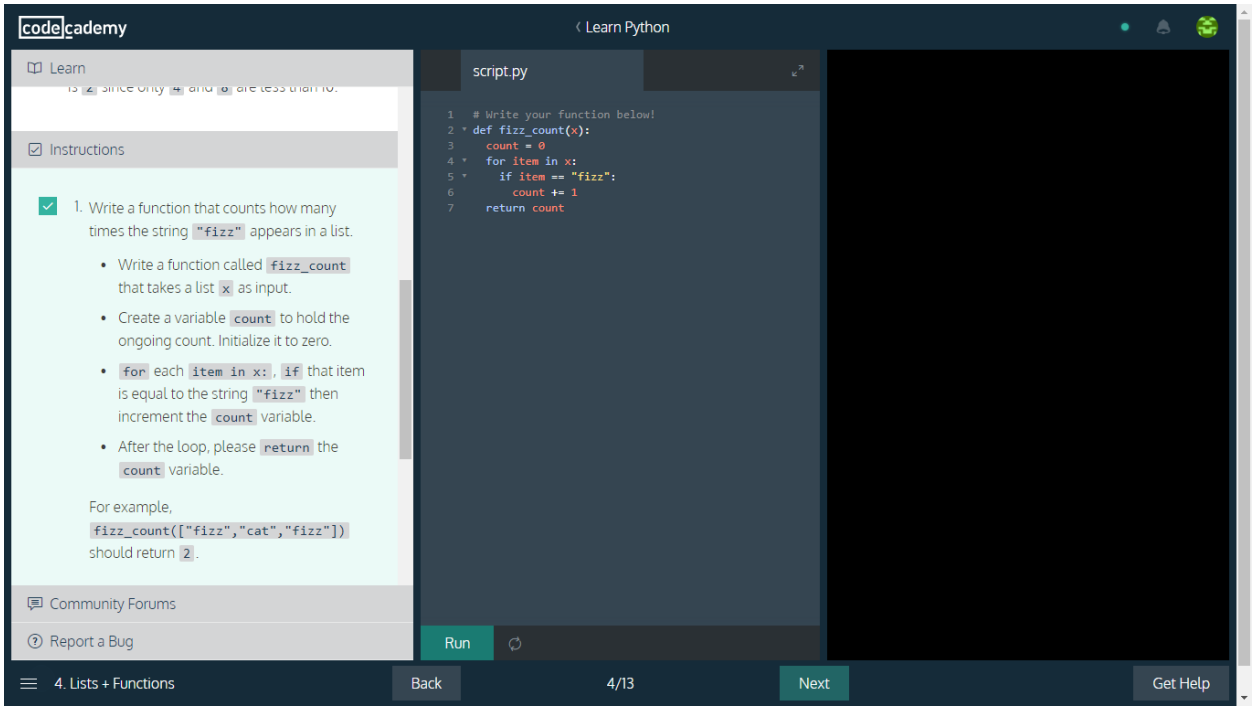
2/13


codecademy
Learn Python
Learn
loop through its keys with the following:
# A simple dictionary
d = {"foo" : "bar"}
for key in d:
print d[key] # prints "bar"
Note that dictionaries are unordered, meaning that any time you loop through a dictionary, you will go through every key, but you are not guaranteed to get them in any particular order.
Instructions
1. Use a for loop to go through the webster dictionary and print out all of the definitions.
Stuck? Get a hint
Community Forums
Report a Bug
script.py
webster = {
"Aardvark" : "A star of a popular children's cartoon show.",
"Baa" : "The sound a goat makes.",
"Carpet": "Goes on the floor.",
"Dab": "A small amount."
}
# Add your code below!
for key in webster:
print webster[key]
Run
A star of a popular children's cartoon show.
Goes on the floor.
A small amount.
The sound a goat makes.
2. This is KEY!
Back
2/13
Next
Get Help

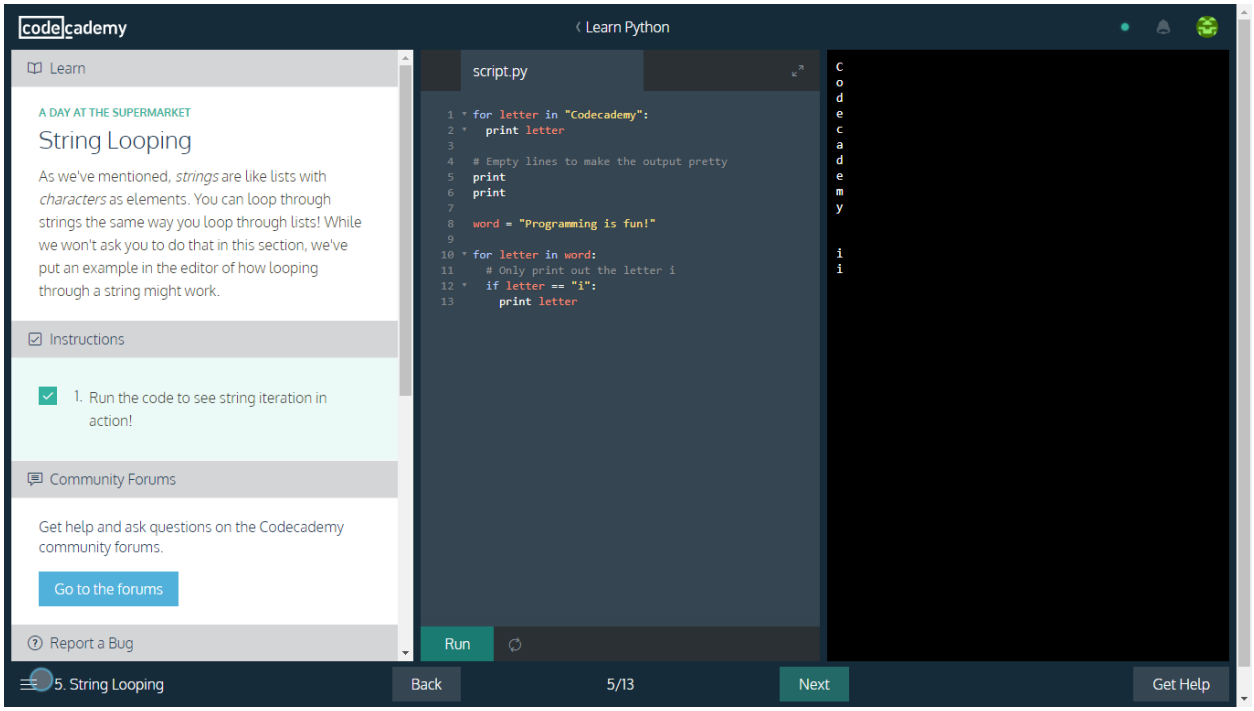
3/13


codecademy
Learn Python
Learn
Make sure to keep track of your indentation or you may get confused!
Instructions
1. Like step 2 above, loop through each item in the list called a.
Like step 3 above, if the number is even, print it out. You can test if the item % 2 == 0 to help you out.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
Report a Bug
If you see a bug or any other issue with this page, please report it here.
script.py
a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
for number in a:
if number % 2 == 0:
print number
0
2
4
6
8
10
12
Run
3. Control Flow and Looping
Back
3/13
Next
Get Help

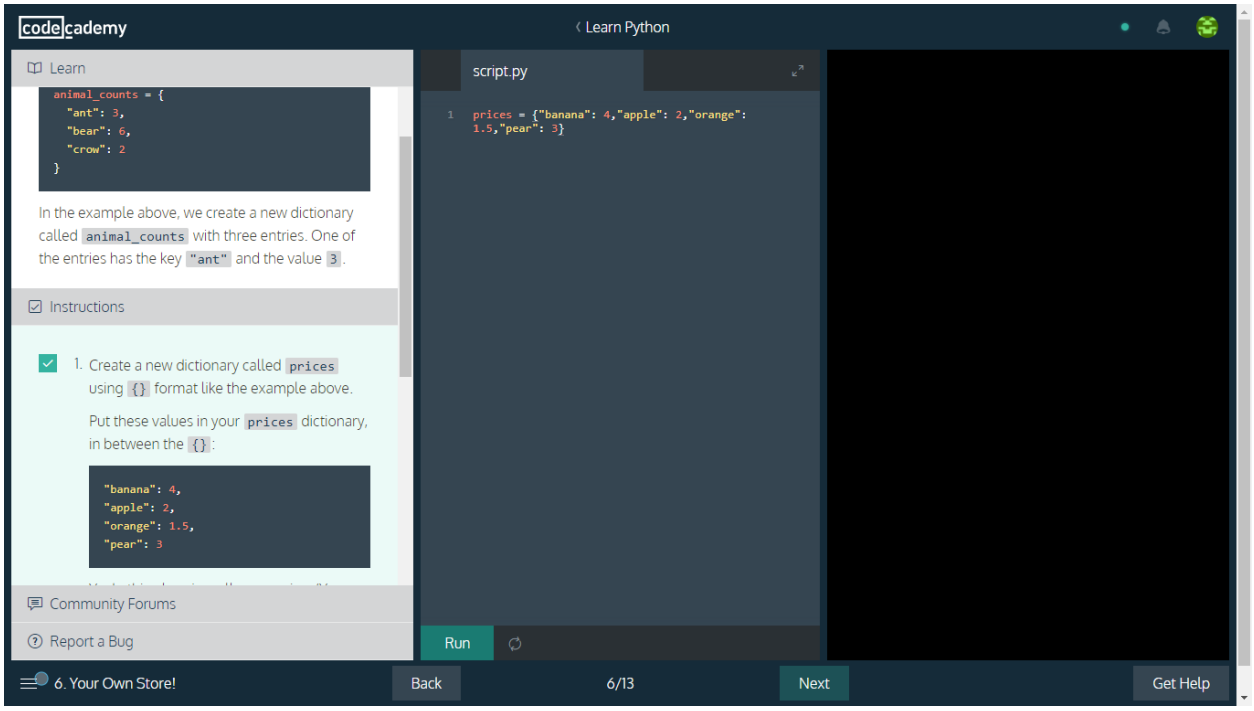
4/13


codecademy
Learn Python
Learn
Instructions
1. Write a function that counts how many times the string "fizz" appears in a list.
Write a function called fizz_count that takes a list x as input.
Create a variable count to hold the ongoing count. Initialize it to zero.
for each item in x:, if that item is equal to the string "fizz" then increment the count variable.
After the loop, please return the count variable.
For example, fizz_count(["fizz","cat","fizz"]) should return 2.
Community Forums
Report a Bug
script.py
# Write your function below!
def fizz_count(x):
count = 0
for item in x:
if item == "fizz":
count += 1
return count
Run
4. Lists + Functions
Back
4/13
Next
Get Help

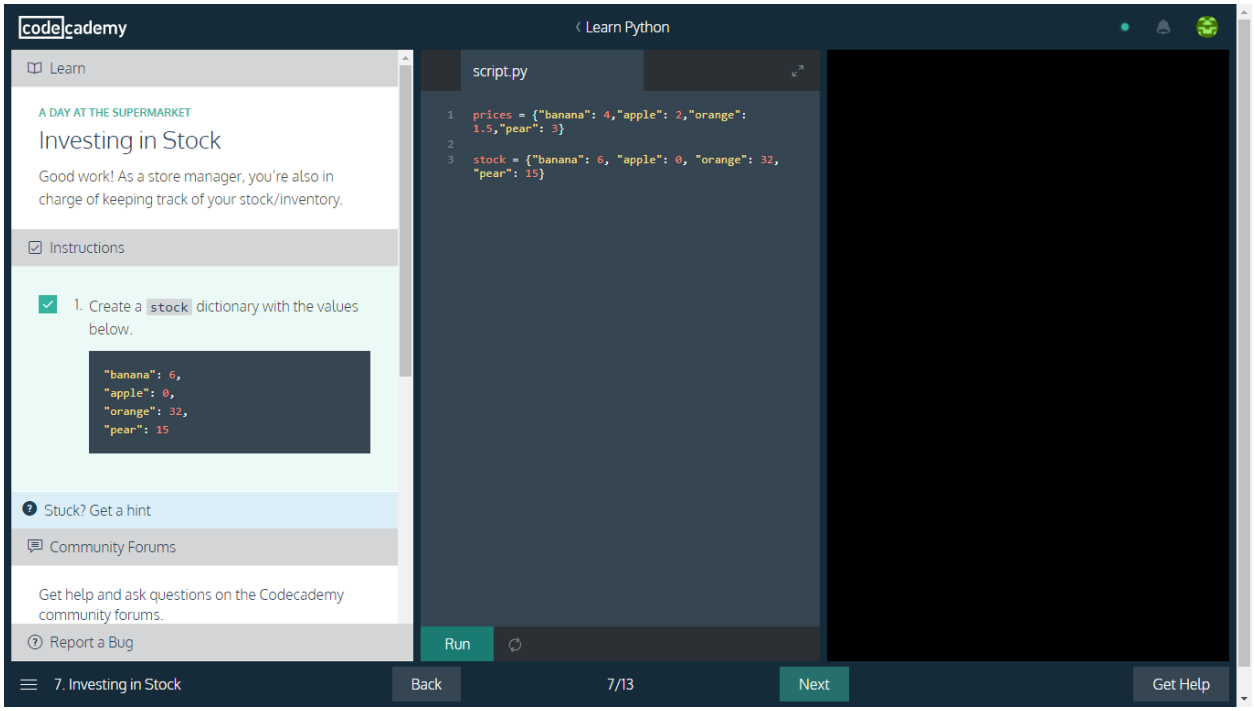
5/13


codecademy
Learn Python
Learn
A DAY AT THE SUPERMARKET
String Looping
As we've mentioned, strings are like lists with characters as elements. You can loop through strings the same way you loop through lists! While we won't ask you to do that in this section, we've put an example in the editor of how looping through a string might work.
Instructions
1. Run the code to see string iteration in action!
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
Report a Bug
script.py
for letter in "Codecademy":
print letter
Empty lines to make the output pretty
print
print
word = "Programming is fun!"
for letter in word:
# Only print out the letter i
if letter == "i":
print letter
C o d e c a d e m y
i i
Run
5. String Looping
Back
5/13
Next
Get Help

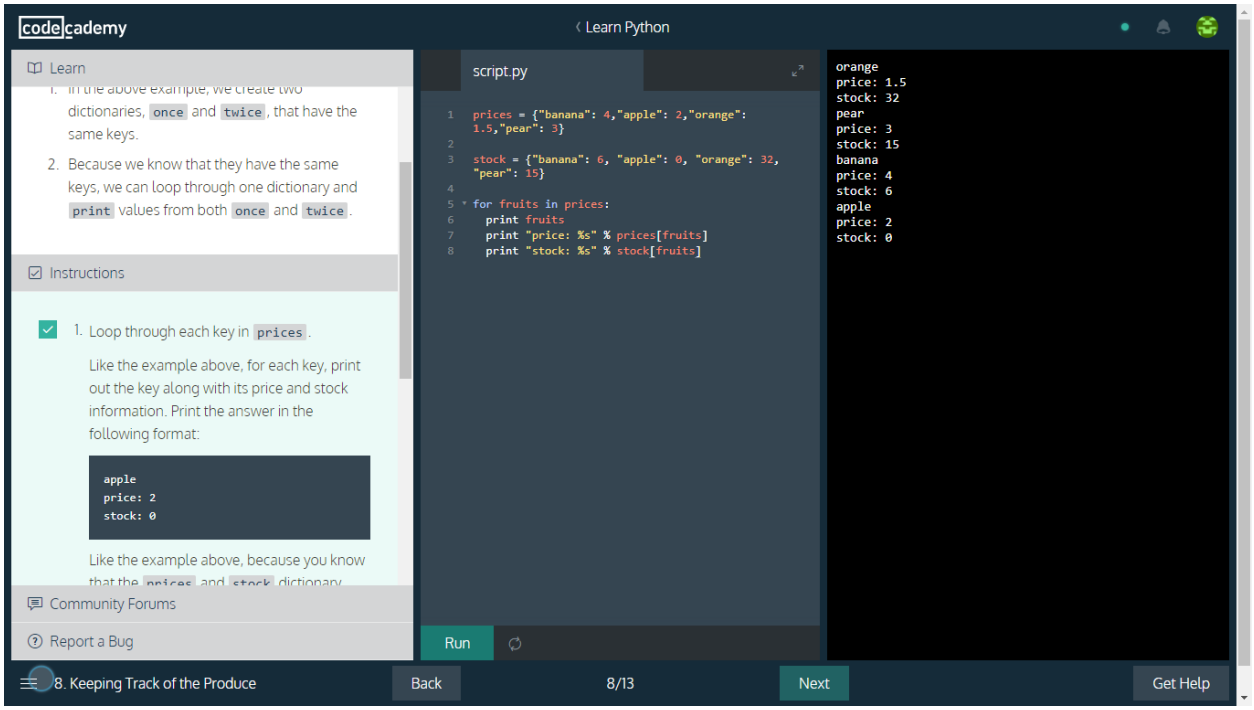
6/13


codecademy
Learn Python
Learn
animal_counts = {
"ant": 3,
"bear": 6,
"crow": 2
}
In the example above, we create a new dictionary called animal_counts with three entries. One of the entries has the key "ant" and the value 3.
Instructions
1. Create a new dictionary called prices using {} format like the example above.
Put these values in your prices dictionary, in between the {}:
"banana": 4,
"apple": 2,
"orange": 1.5,
"pear": 3
Community Forums
Report a Bug
script.py
prices = {"banana": 4,"apple": 2,"orange": 1.5,"pear": 3}
Run
6. Your Own Store!
Back
6/13
Next
Get Help

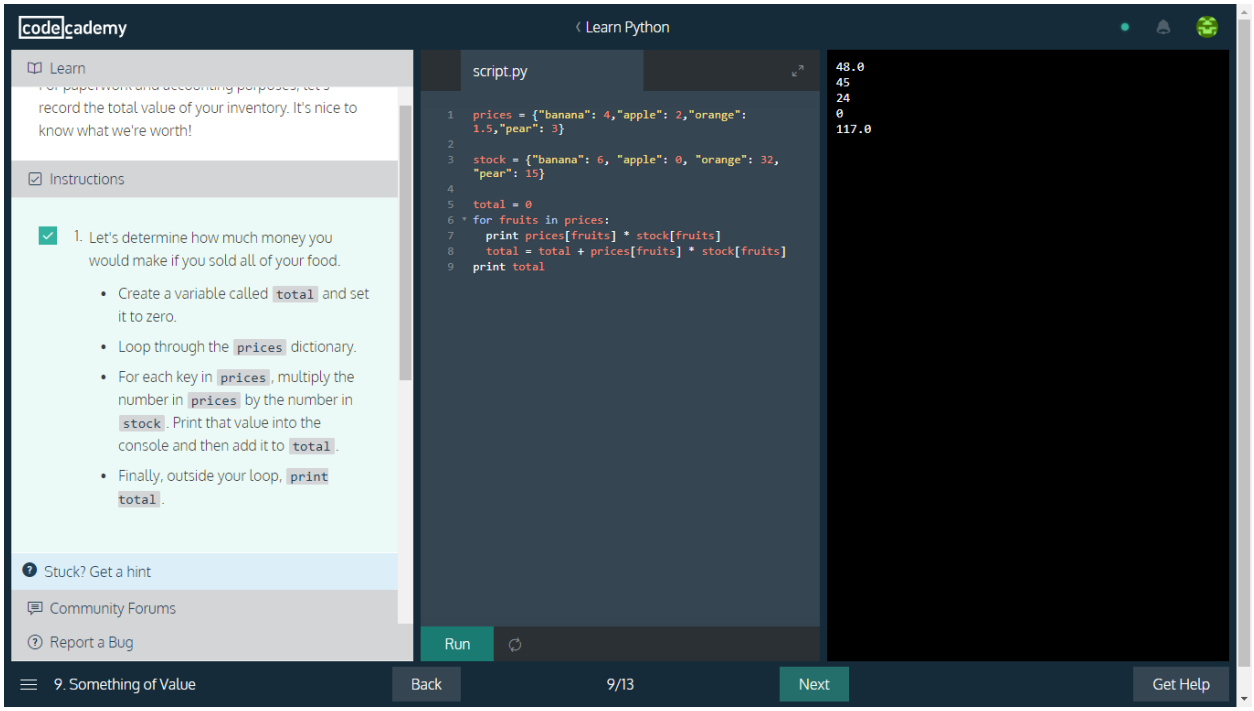
7/13


codecademy
Learn Python
Learn
A DAY AT THE SUPERMARKET
Investing in Stock
Good work! As a store manager, you're also in charge of keeping track of your stock/inventory.
Instructions
1. Create a stock dictionary with the values below.
"banana": 6,
"apple": 0,
"orange": 32,
"pear": 15
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Report a Bug
script.py
prices = {"banana": 4,"apple": 2,"orange": 1.5,"pear": 3}
stock = {"banana": 6, "apple": 0, "orange": 32, "pear": 15}
Run
7. Investing in Stock
Back
7/13
Next
Get Help

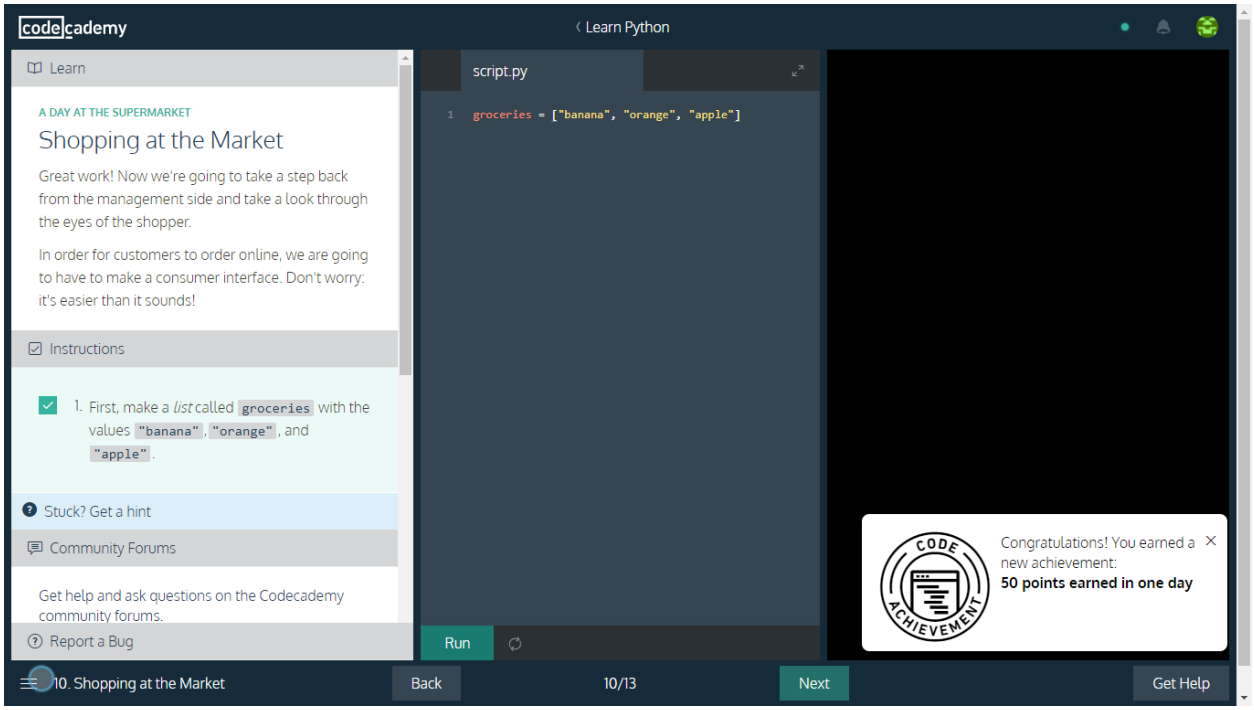
8/13


codecademy
Learn Python
Learn
dictionaries, once and twice, that have the same keys.
2. Because we know that they have the same keys, we can loop through one dictionary and print values from both once and twice.
Instructions
1. Loop through each key in prices.
Like the example above, for each key, print out the key along with its price and stock information. Print the answer in the following format:
apple
price: 2
stock: 0
Like the example above, because you know
Community Forums
Report a Bug
script.py
prices = {"banana": 4,"apple": 2,"orange": 1.5,"pear": 3}
stock = {"banana": 6, "apple": 0, "orange": 32, "pear": 15}
for fruits in prices:
print fruits
print "price: %s" % prices[fruits]
print "stock: %s" % stock[fruits]
orange
price: 1.5
stock: 32
pear
price: 3
stock: 15
banana
price: 4
stock: 6
apple
price: 2
stock: 0
Run
8. Keeping Track of the Produce
Back
8/13
Next
Get Help

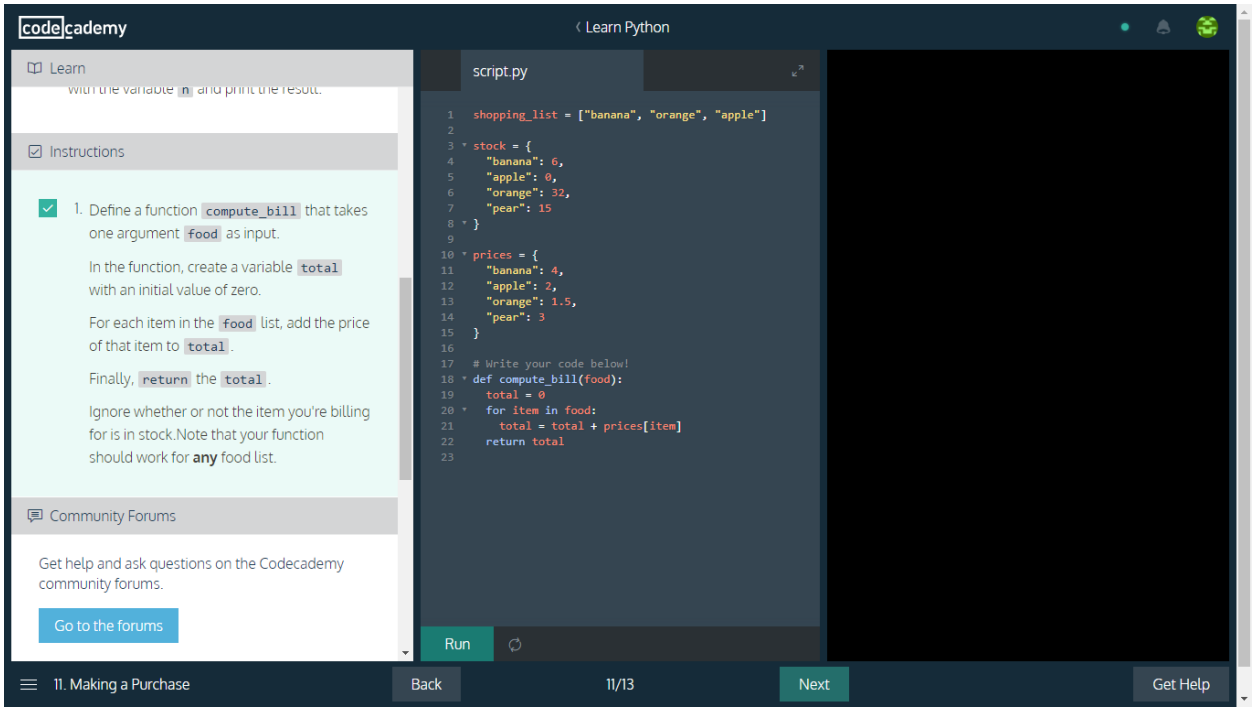
9/13


codecademy
Learn Python
Learn
record the total value of your inventory. It's nice to know what we're worth!
Instructions
1. Let's determine how much money you would make if you sold all of your food.
Create a variable called total and set it to zero.
Loop through the prices dictionary.
For each key in prices, multiply the number in prices by the number in stock. Print that value into the console and then add it to total.
Finally, outside your loop, print total.
Stuck? Get a hint
Community Forums
Report a Bug
script.py
prices = {"banana": 4,"apple": 2,"orange": 1.5,"pear": 3}
stock = {"banana": 6, "apple": 0, "orange": 32, "pear": 15}
total = 0
for fruits in prices:
print prices[fruits] * stock[fruits]
total = total + prices[fruits] * stock[fruits]
print total
Run
48.0
45
24
0
117.0
9. Something of Value
Back
9/13
Next
Get Help

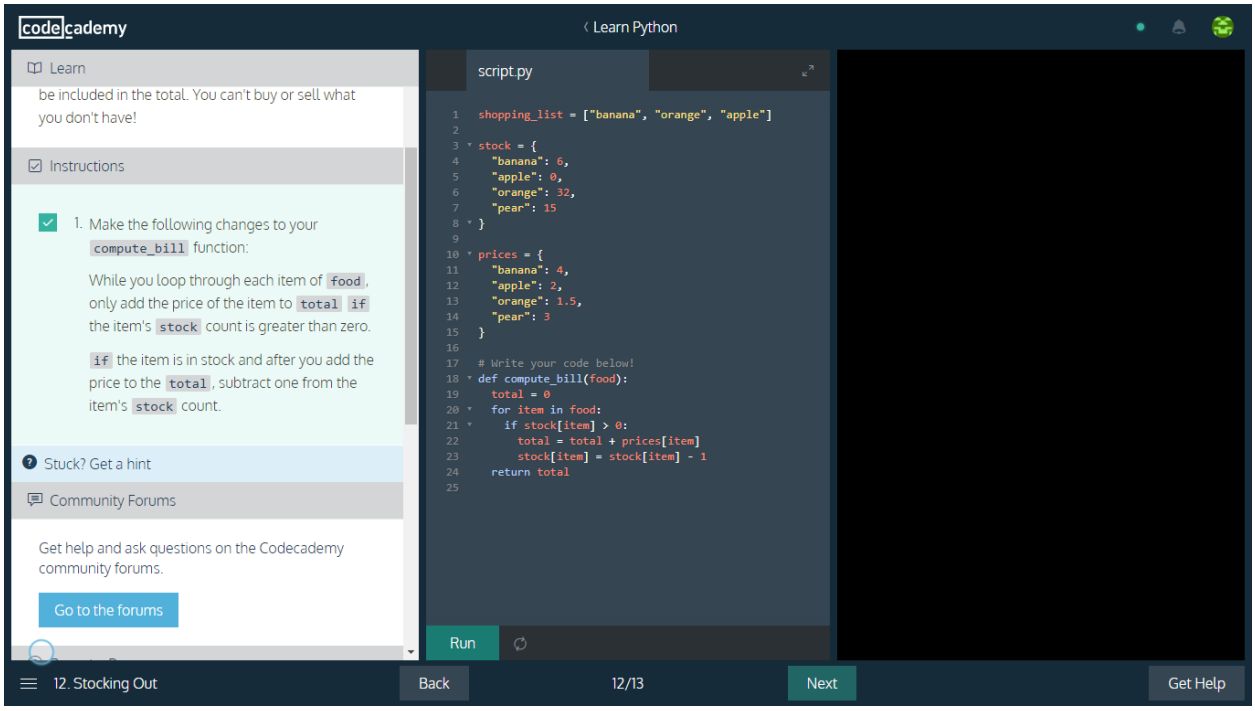
10/13


codecademy
Learn Python
Learn
A DAY AT THE SUPERMARKET
Shopping at the Market
Great work! Now we're going to take a step back from the management side and take a look through the eyes of the shopper.
In order for customers to order online, we are going to have to make a consumer interface. Don't worry: it's easier than it sounds!
Instructions
1. First, make a list called groceries with the values "banana", "orange", and "apple".
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Report a Bug
script.py
groceries = ["banana", "orange", "apple"]
Run
CODE ACHIEVEMENT
Congratulations! You earned a new achievement:
50 points earned in one day
10. Shopping at the Market
Back
10/13
Next
Get Help

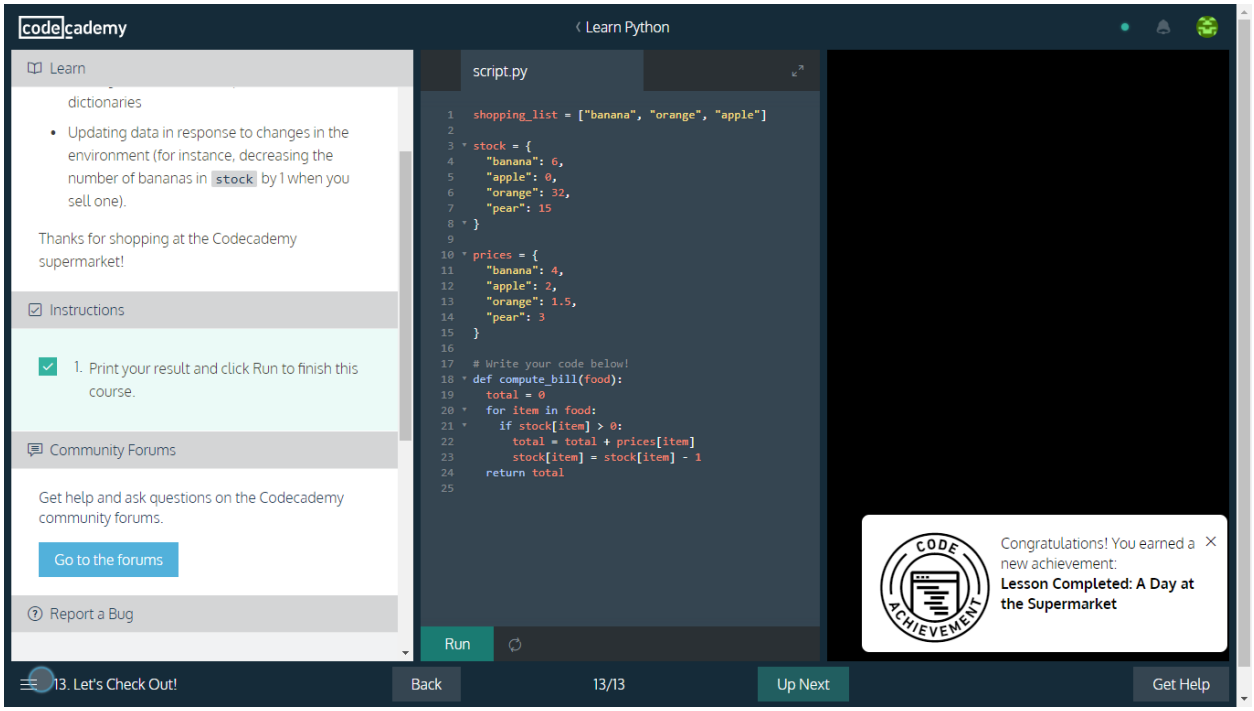
11/13


codecademy
Learn Python
Learn
with the variable n and print the result.
Instructions
1. Define a function compute_bill that takes one argument food as input.
In the function, create a variable total with an initial value of zero.
For each item in the food list, add the price of that item to total.
Finally, return the total.
Ignore whether or not the item you're billing for is in stock.Note that your function should work for any food list.
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
script.py
shopping_list = ["banana", "orange", "apple"]
stock = {
"banana": 6,
"apple": 0,
"orange": 32,
"pear": 15
}
prices = {
"banana": 4,
"apple": 2,
"orange": 1.5,
"pear": 3
}
# Write your code below!
def compute_bill(food):
total = 0
for item in food:
total = total + prices[item]
return total
Run
11. Making a Purchase
Back
11/13
Next
Get Help

12/13


codecademy
Learn Python
Learn
be included in the total. You can't buy or sell what you don't have!
Instructions
1. Make the following changes to your compute_bill function:
While you loop through each item of food, only add the price of the item to total if the item's stock count is greater than zero.
if the item is in stock and after you add the price to the total, subtract one from the item's stock count.
Stuck? Get a hint
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
script.py
shopping_list = ["banana", "orange", "apple"]
stock = {
"banana": 6,
"apple": 0,
"orange": 32,
"pear": 15
}
prices = {
"banana": 4,
"apple": 2,
"orange": 1.5,
"pear": 3
}
# Write your code below!
def compute_bill(food):
total = 0
for item in food:
if stock[item] > 0:
total = total + prices[item]
stock[item] = stock[item] - 1
return total
Run
12. Stocking Out
Back
12/13
Next
Get Help

13/13


codecademy
Learn Python
Learn
dictionaries
Updating data in response to changes in the environment (for instance, decreasing the number of bananas in stock by 1 when you sell one).
Thanks for shopping at the Codecademy supermarket!
Instructions
1. Print your result and click Run to finish this course.
Community Forums
Get help and ask questions on the Codecademy community forums.
Go to the forums
Report a Bug
script.py
shopping_list = ["banana", "orange", "apple"]
stock = {
"banana": 6,
"apple": 0,
"orange": 32,
"pear": 15
}
prices = {
"banana": 4,
"apple": 2,
"orange": 1.5,
"pear": 3
}
# Write your code below!
def compute_bill(food):
total = 0
for item in food:
if stock[item] > 0:
total = total + prices[item]
stock[item] = stock[item] - 1
return total
Run
CODE ACHIEVEMENT
Congratulations! You earned a new achievement:
Lesson Completed: A Day at the Supermarket
13. Let's Check Out!
Back
13/13
Up Next
Get Help