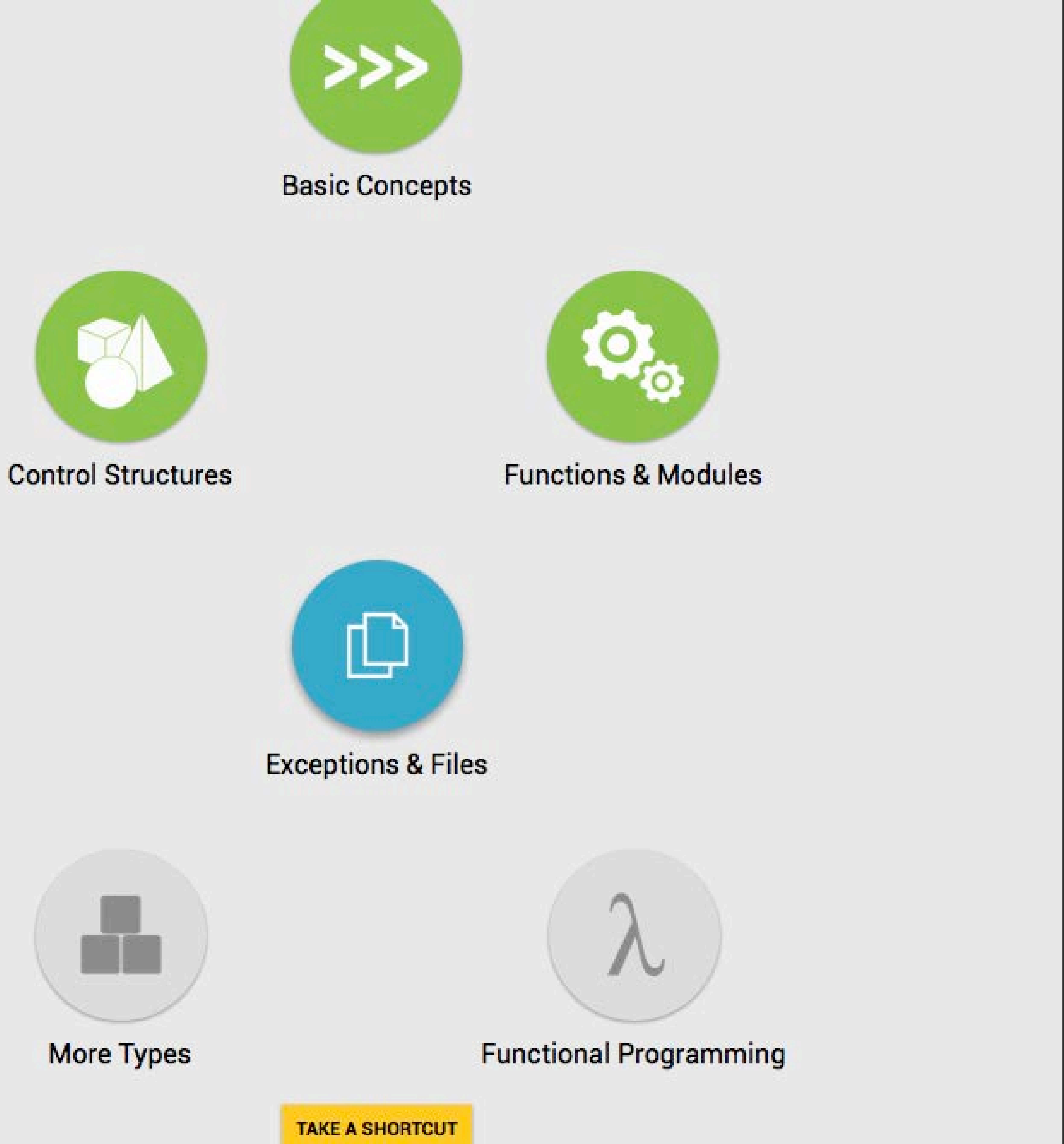
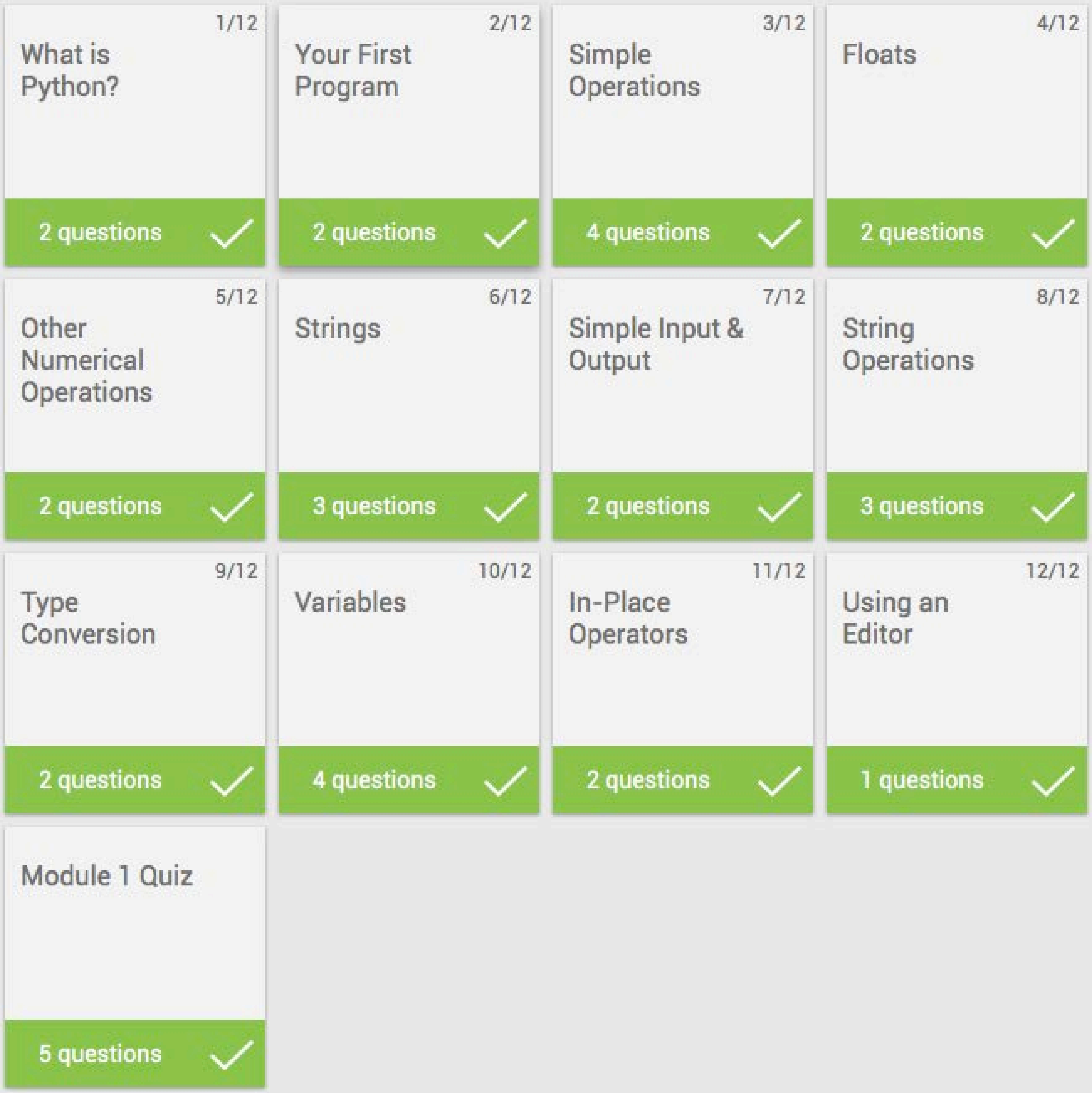

Basic Concepts
Control Structures
Functions & Modules
Exceptions & Files
More Types
Functional Programming
TAKE A SHORTCUT



1/12
What is Python?
2 questions ✓

2/12
Your First Program
2 questions ✓

3/12
Simple Operations
4 questions ✓

4/12
Floats
2 questions ✓

5/12
Other Numerical Operations
2 questions ✓

6/12
Strings
3 questions ✓

7/12
Simple Input & Output
2 questions ✓

8/12
String Operations
3 questions ✓

9/12
Type Conversion
2 questions ✓

10/12
Variables
4 questions ✓

11/12
In-Place Operators
2 questions ✓

12/12
Using an Editor
1 questions ✓

Module 1 Quiz
5 questions ✓

Welcome to Python!

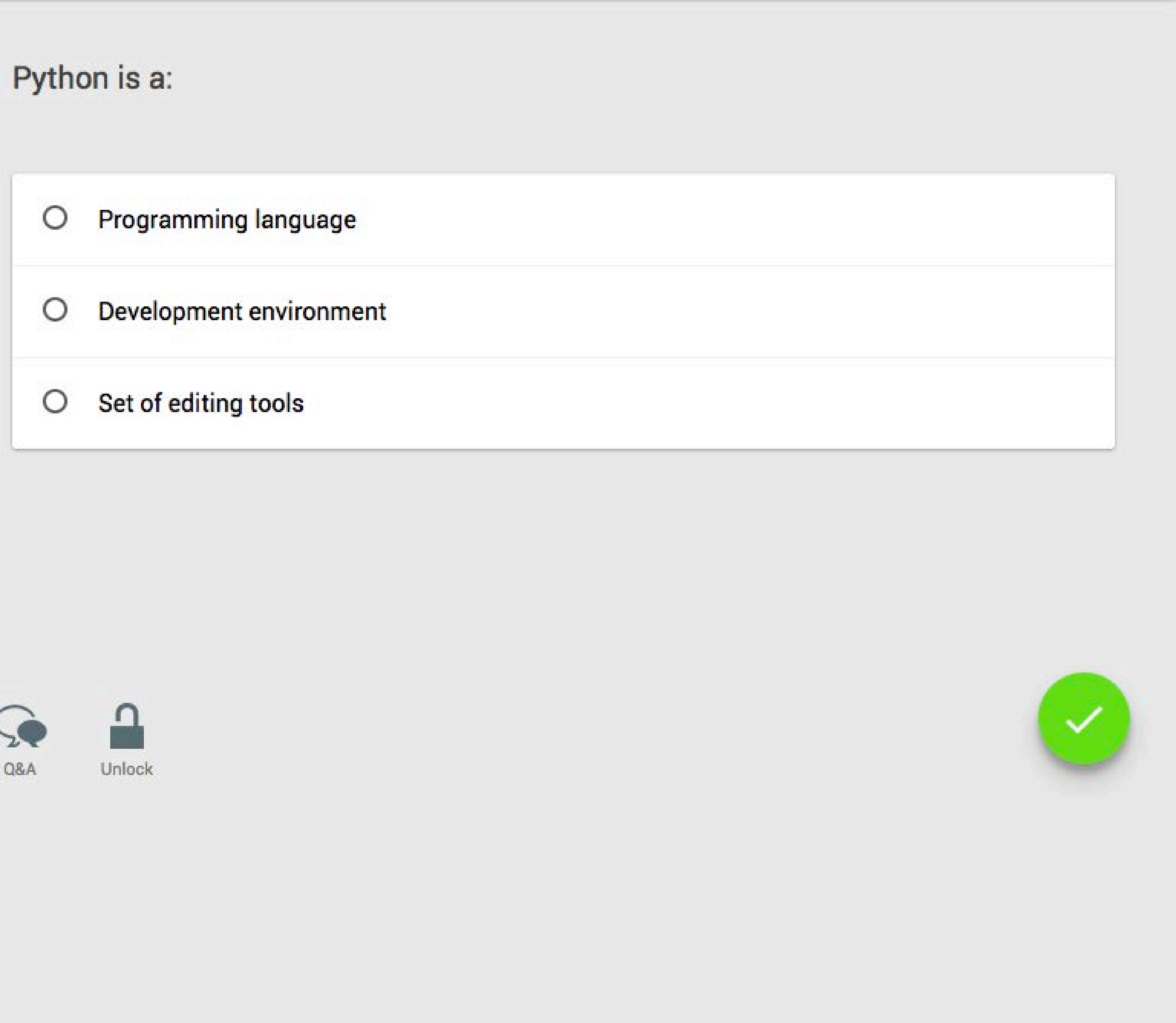
Python is a high-level programming language, with applications in numerous areas, including web programming, scripting, scientific computing, and artificial intelligence.

It is very popular and used by organizations such as Google, NASA, the CIA, and Disney.

> Python is processed at runtime by the interpreter. There is no need to compile your program before executing it.

693 COMMENTS

Q&A



Python is a:

- Programming language
- Development environment
- Set of editing tools

Welcome to Python!

The three major versions of Python are 1.x, 2.x and 3.x. These are subdivided into minor versions, such as 2.7 and 3.3.
Code written for Python 3.x is guaranteed to work in all future versions.
Both Python Version 2.x and 3.x are used currently.
This course covers **Python 3.x**, but it isn't hard to change from one version to another.

Python has several different implementations, written in various languages.
The version used in this course, **CPython**, is the most popular by far.

> An **interpreter** is a program that runs scripts written in an interpreted language such as Python.

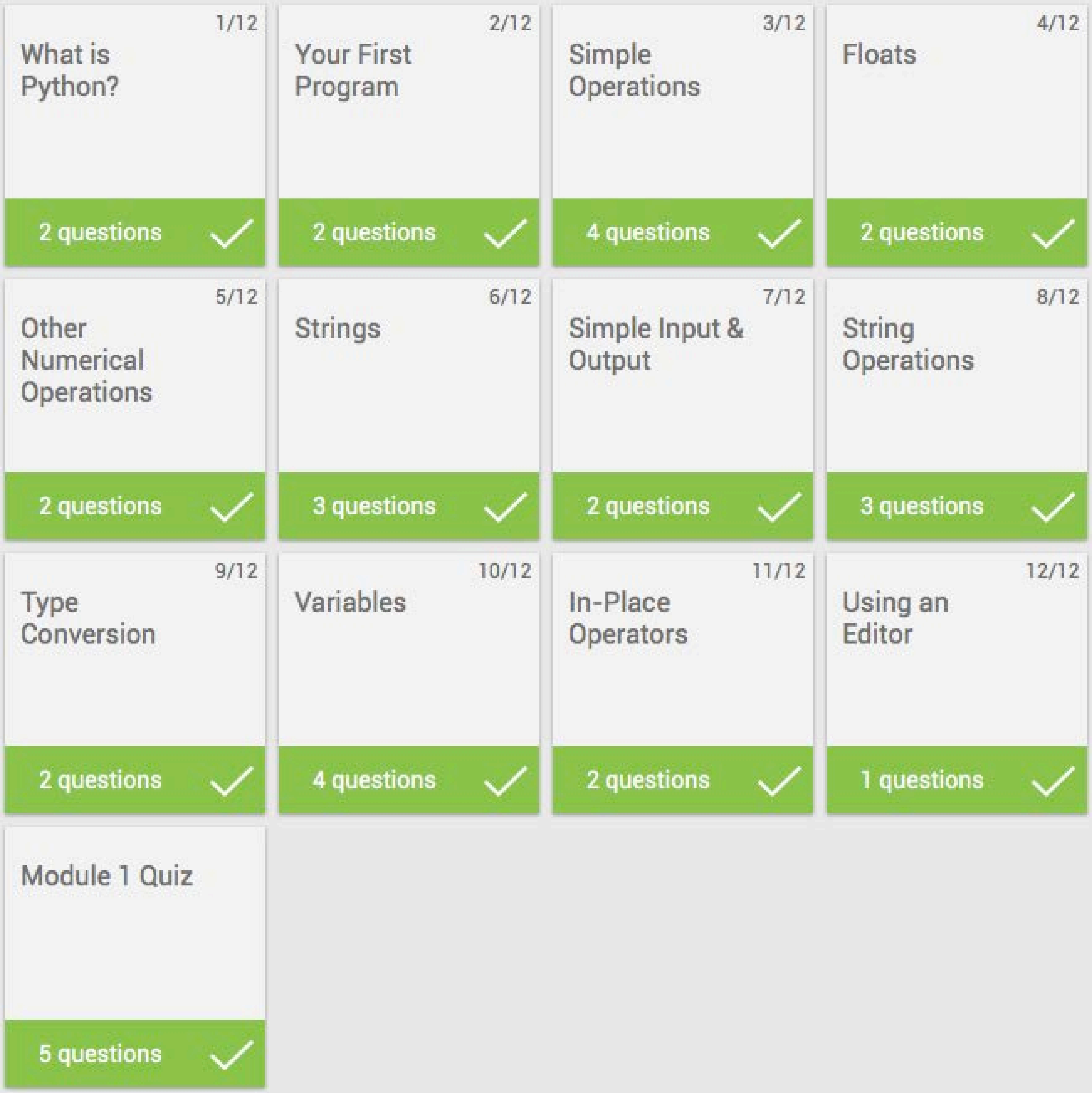
153 COMMENTS

Q&A

Which of these statements is true?

- ○ Python code must be always compiled
- ○ CPython is an implementation of Python
- ○ Python 1.7 is the most widely used version

Q&A Unlock



1/12
What is Python?
2 questions ✓

2/12
Your First Program
2 questions ✓

3/12
Simple Operations
4 questions ✓

4/12
Floats
2 questions ✓

5/12
Other Numerical Operations
2 questions ✓

6/12
Strings
3 questions ✓

7/12
Simple Input & Output
2 questions ✓

8/12
String Operations
3 questions ✓

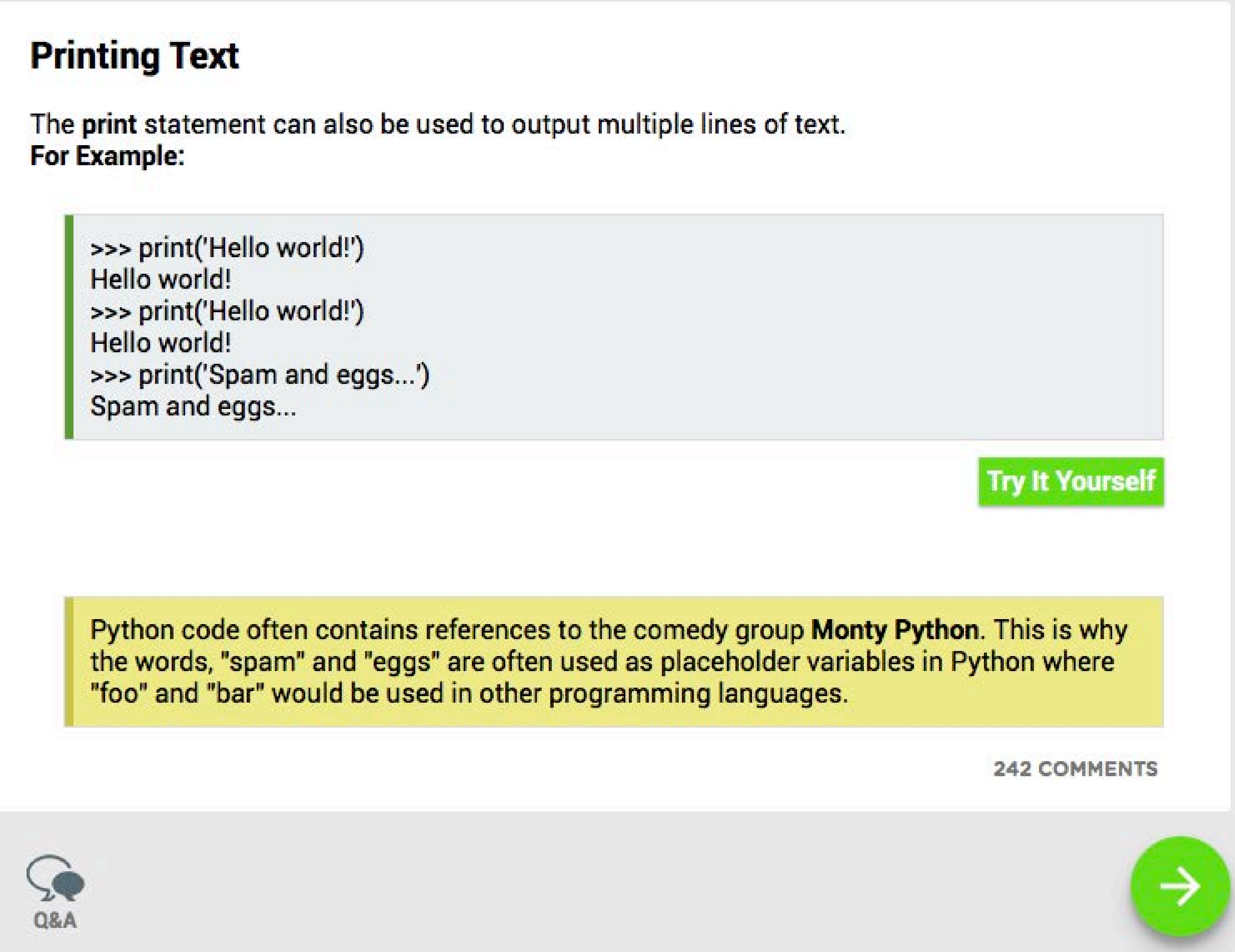
9/12
Type Conversion
2 questions ✓

10/12
Variables
4 questions ✓

11/12
In-Place Operators
2 questions ✓

12/12
Using an Editor
1 questions ✓

Module 1 Quiz
5 questions ✓

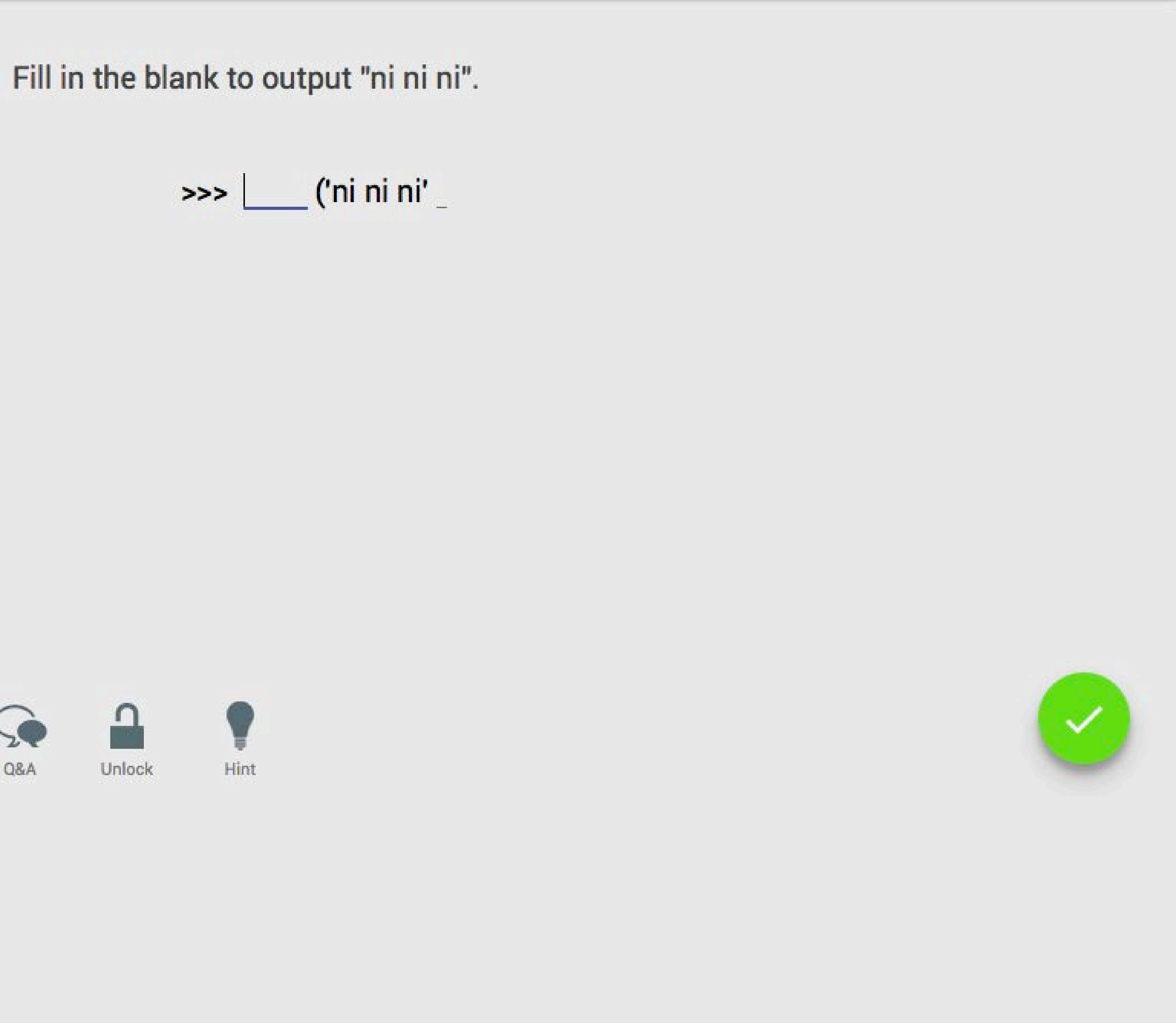


Printing Text

The **print** statement can also be used to output multiple lines of text.
For Example:

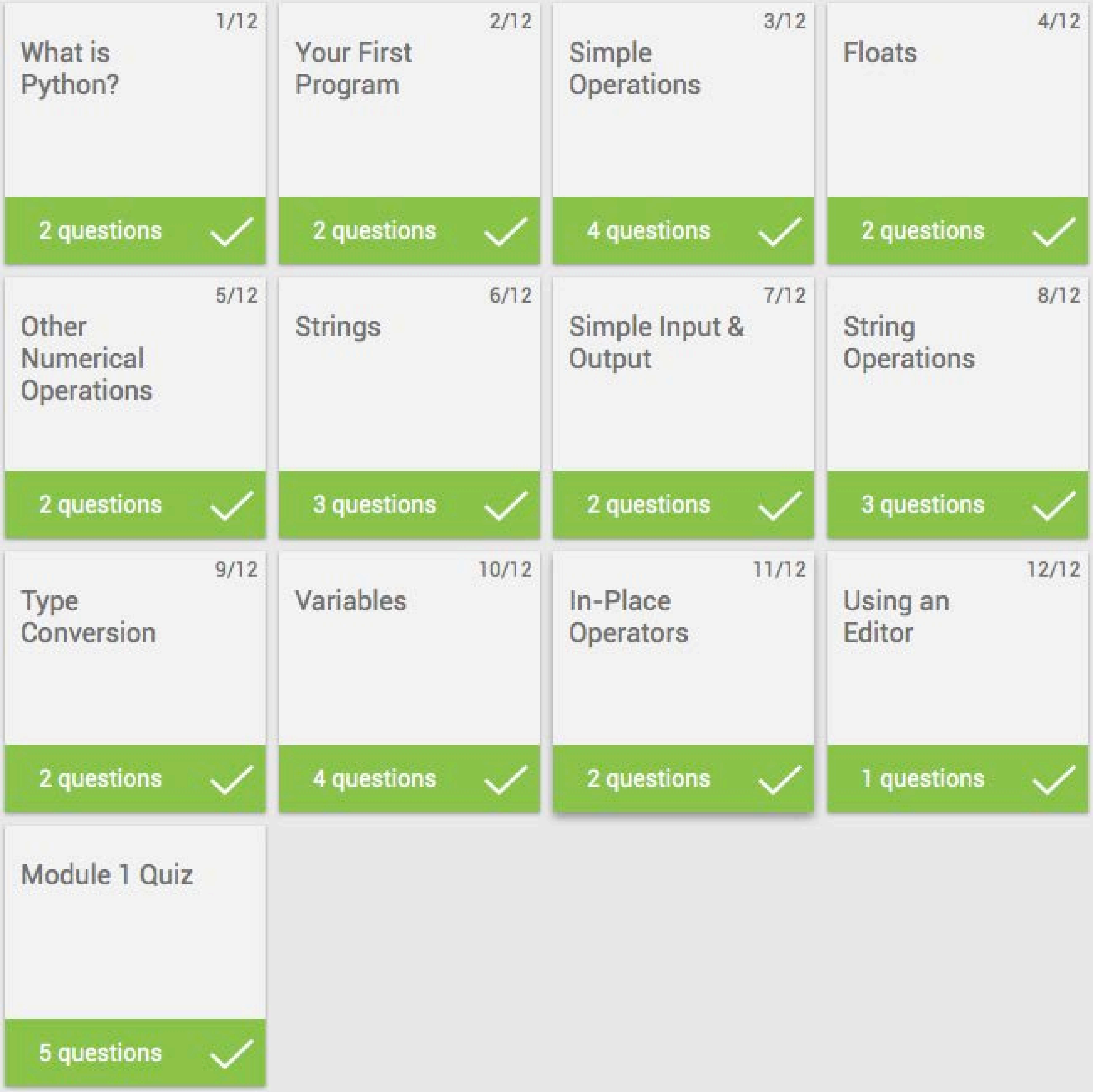
```
>>> print('Hello world!')
Hello world!
>>> print('Hello world!')
Hello world!
>>> print('Spam and eggs...')
Spam and eggs...
```

Python code often contains references to the comedy group **Monty Python**. This is why the words, "spam" and "eggs" are often used as placeholder variables in Python where "foo" and "bar" would be used in other programming languages.



Fill in the blank to output "ni ni ni".

```
>>> ____('ni ni ni' _
```



1/12
What is Python?
2 questions ✓

2/12
Your First Program
2 questions ✓

3/12
Simple Operations
4 questions ✓

4/12
Floats
2 questions ✓

5/12
Other Numerical Operations
2 questions ✓

6/12
Strings
3 questions ✓

7/12
Simple Input & Output
2 questions ✓

8/12
String Operations
3 questions ✓

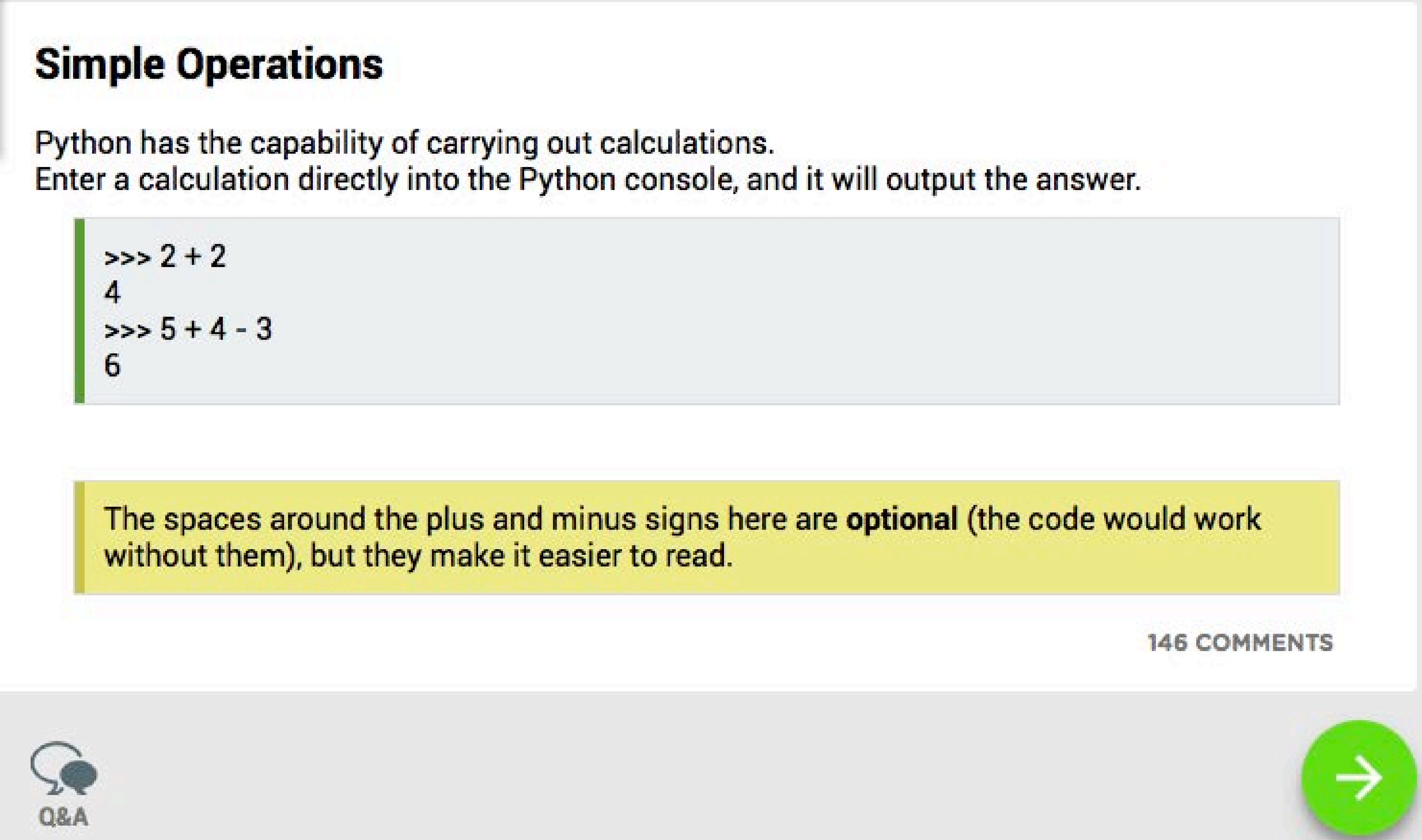
9/12
Type Conversion
2 questions ✓

10/12
Variables
4 questions ✓

11/12
In-Place Operators
2 questions ✓

12/12
Using an Editor
1 questions ✓

Module 1 Quiz
5 questions ✓

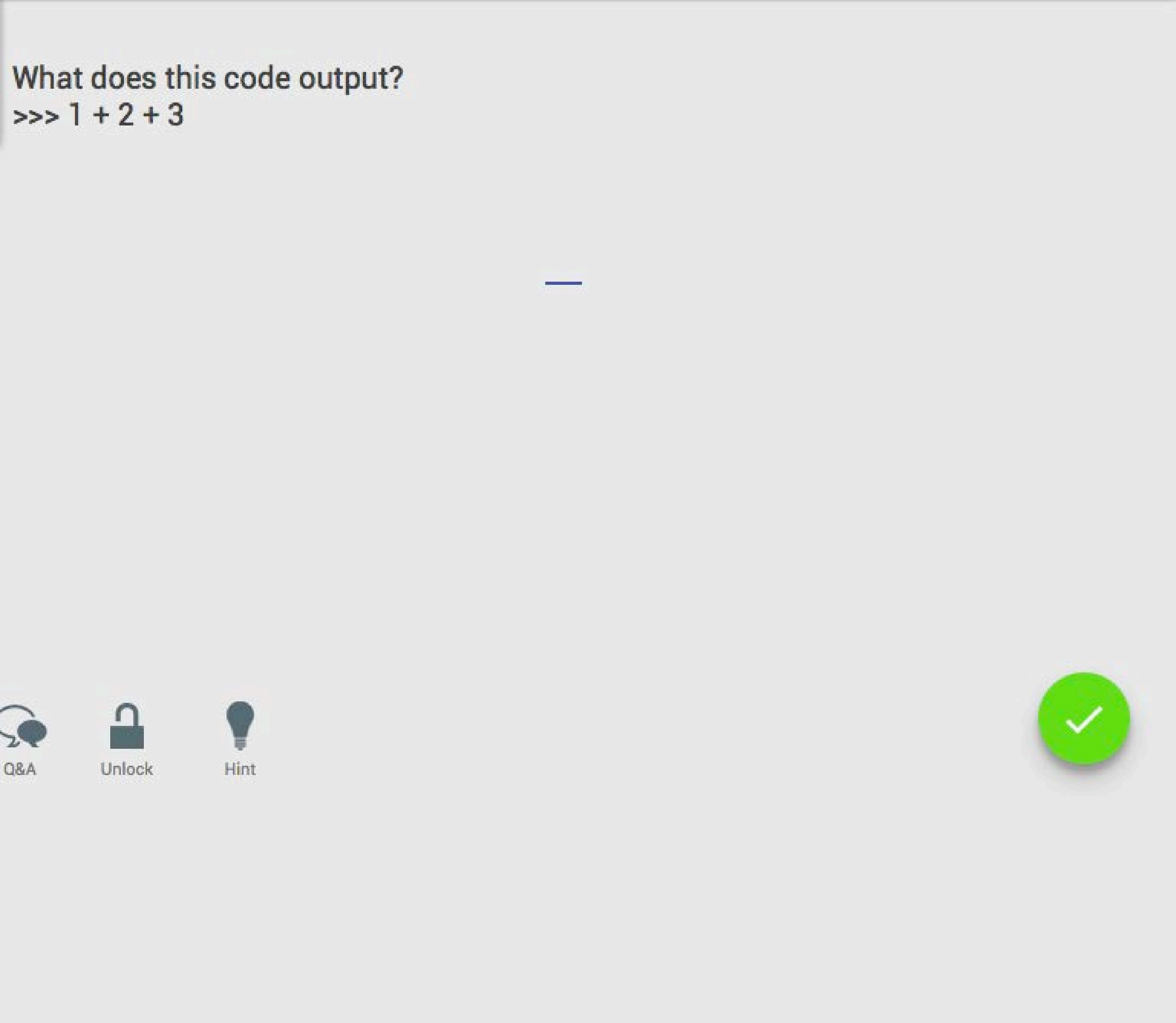


Simple Operations

Python has the capability of carrying out calculations.
Enter a calculation directly into the Python console, and it will output the answer.

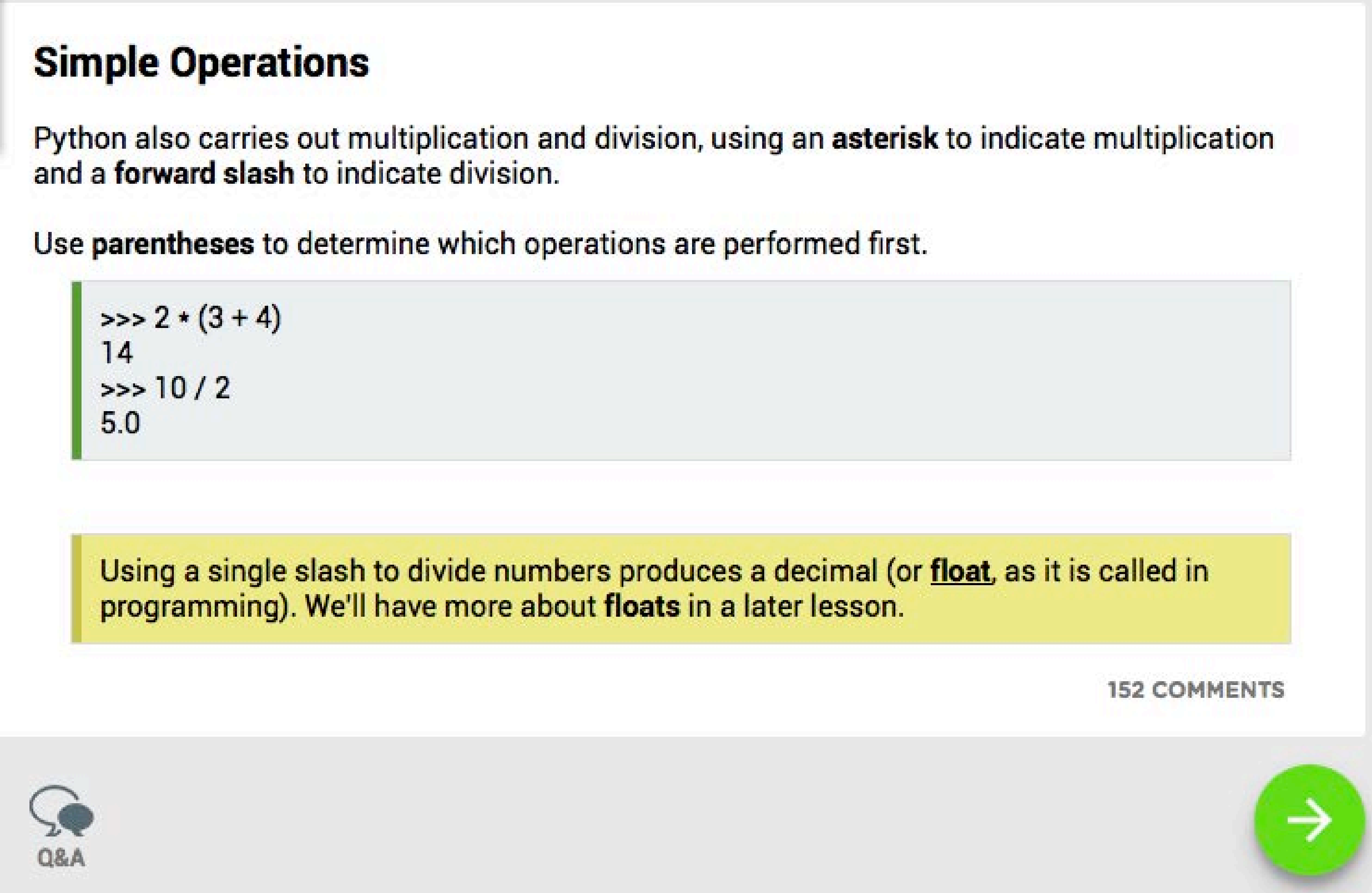
```
>>> 2 + 2
4
>>> 5 + 4 - 3
6
```

The spaces around the plus and minus signs here are **optional** (the code would work without them), but they make it easier to read.



What does this code output?

```
>>> 1 + 2 + 3
```



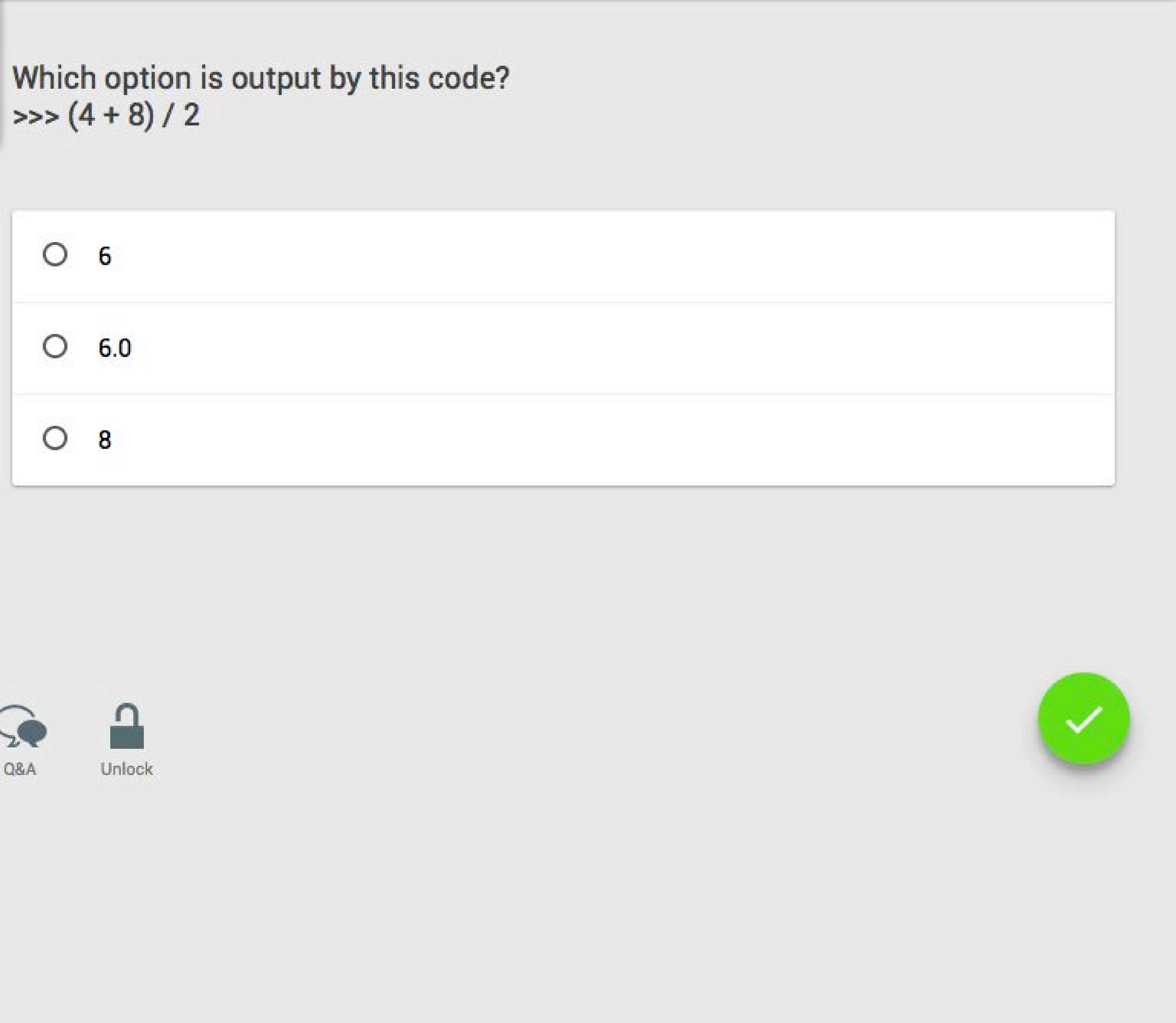
Simple Operations

Python also carries out multiplication and division, using an **asterisk** to indicate multiplication and a **forward slash** to indicate division.

Use **parentheses** to determine which operations are performed first.

```
>>> 2 * (3 + 4)
14
>>> 10 / 2
5.0
```

Using a single slash to divide numbers produces a decimal (or **float**, as it is called in programming). We'll have more about **floats** in a later lesson.



Which option is output by this code?

```
>>> (4 + 8) / 2
```

- ○ 6
- ○ 6.0
- ○ 8

Simple Operations

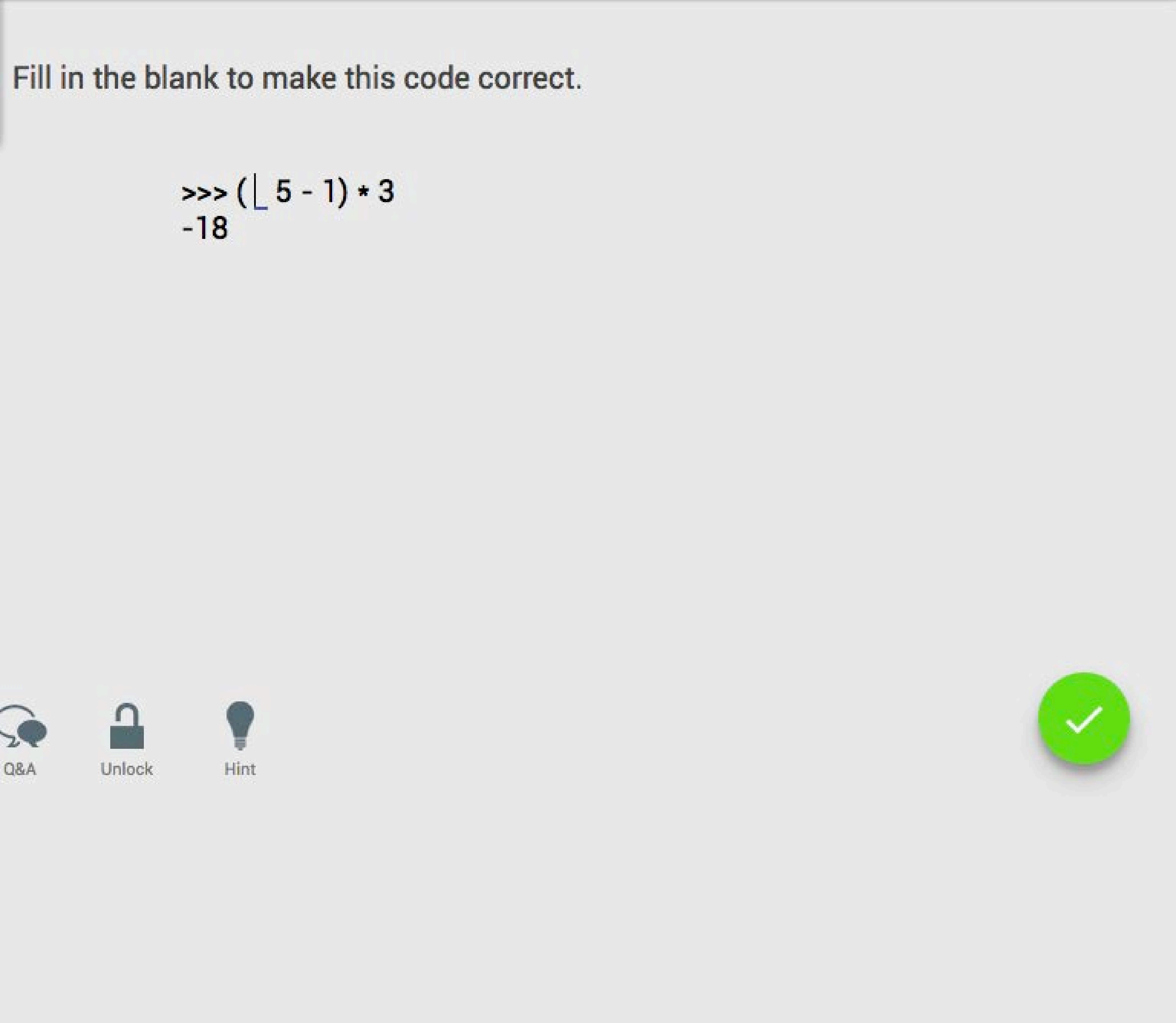
The minus sign indicates a **negative** number.
Operations are performed on negative numbers, just as they are on positive ones.

```
>>> -7
-7
>>> (-7 + 2) * (-4)
20
```

The plus signs can also be put in front of numbers, but this has no effect, and is mostly used to emphasize that a number is positive to increase readability of code.

98 COMMENTS

Q&A



Fill in the blank to make this code correct.

```
>>> ( 5 - 1) * 3
-18
```

Simple Operations

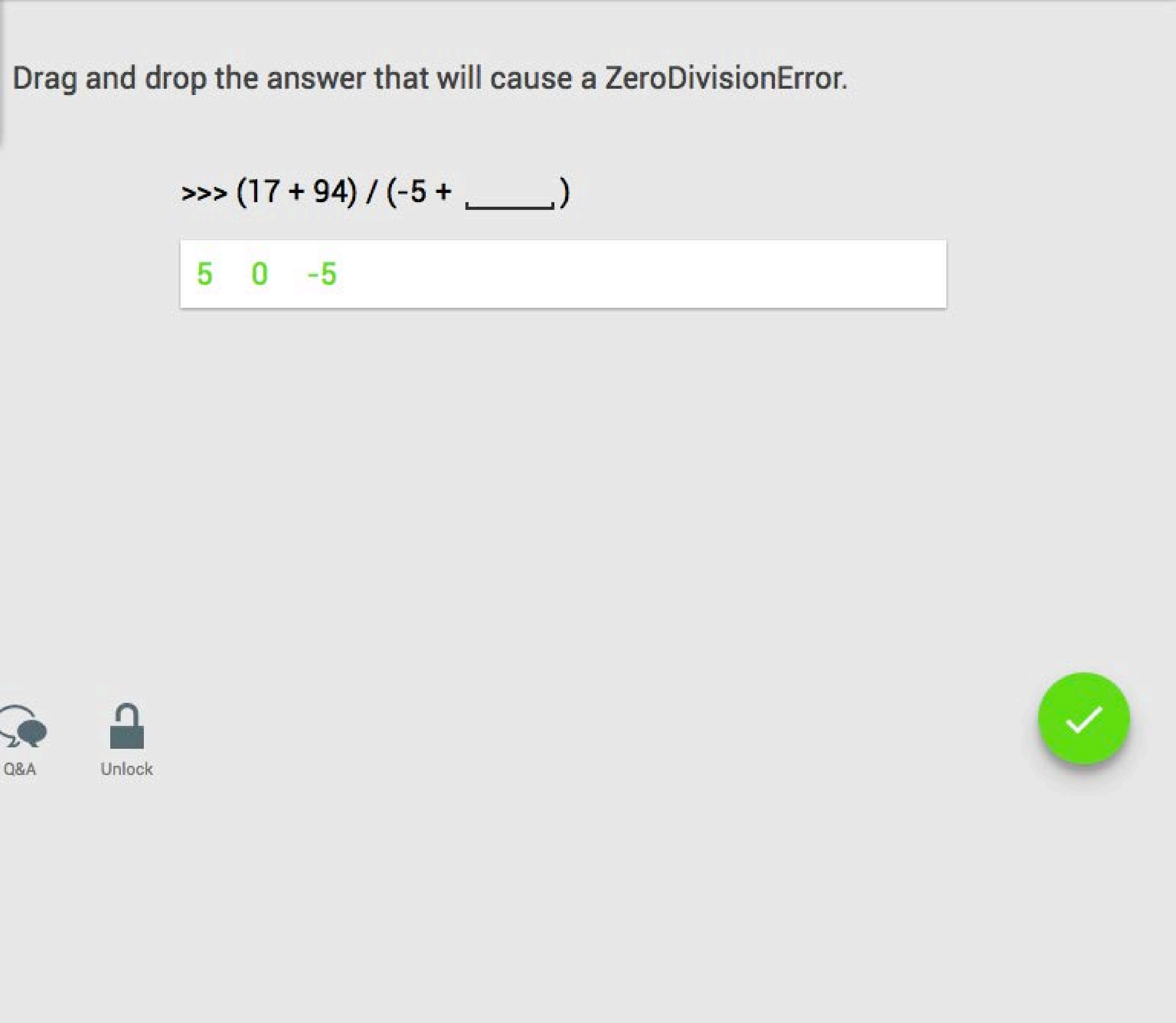
Dividing by zero in Python produces an **error**, as no answer can be calculated.

```
>>> 11 / 0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
```

In Python, the last line of an error message indicates the error's type.
Read error messages carefully, as they often tell you how to fix a program!

89 COMMENTS

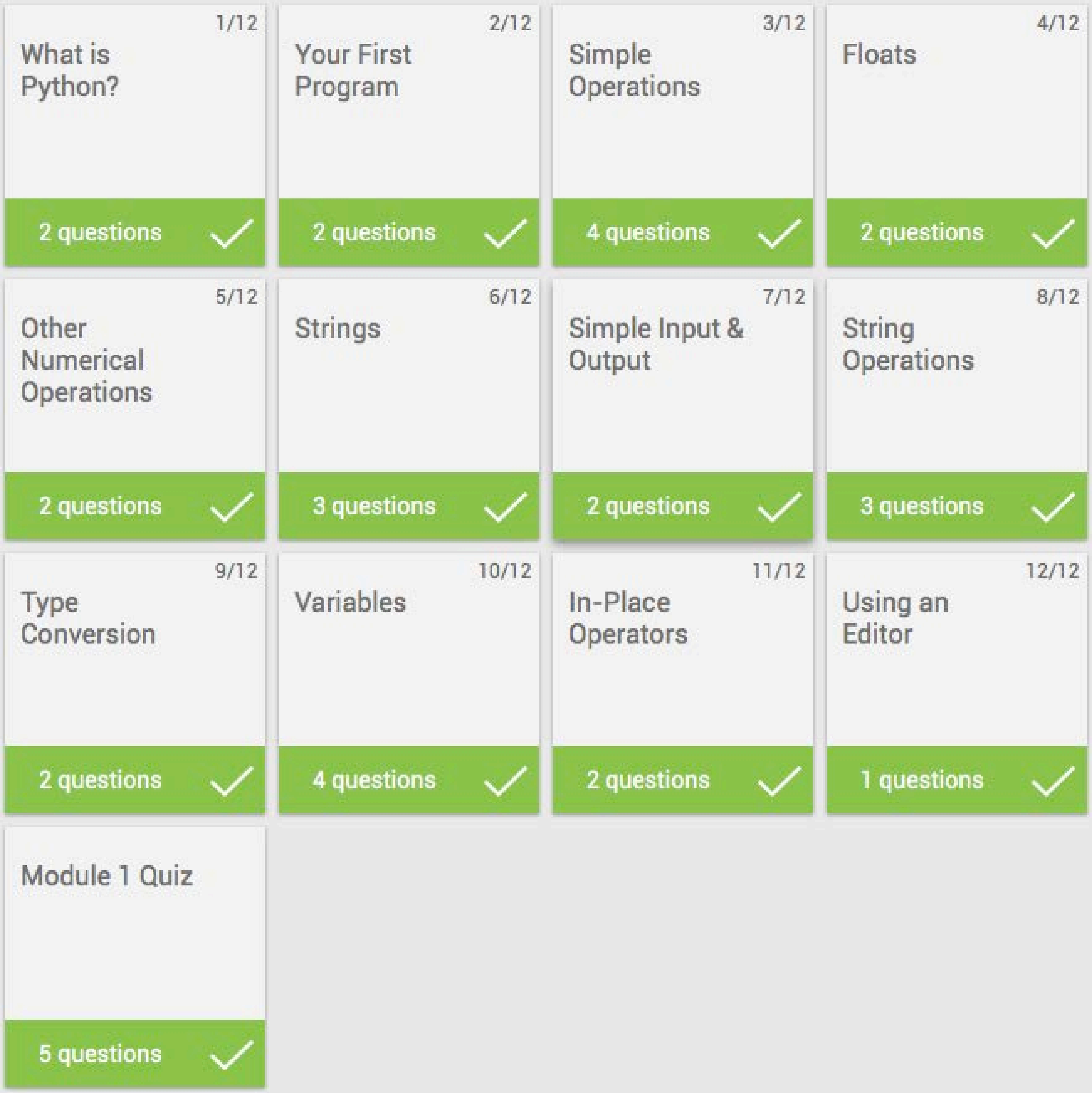
Q&A



Drag and drop the answer that will cause a ZeroDivisionError.

```
>>> (17 + 94) / (-5 + _____)
```

5   0   -5



1/12
What is Python?
2 questions ✓

2/12
Your First Program
2 questions ✓

3/12
Simple Operations
4 questions ✓

4/12
Floats
2 questions ✓

5/12
Other Numerical Operations
2 questions ✓

6/12
Strings
3 questions ✓

7/12
Simple Input & Output
2 questions ✓

8/12
String Operations
3 questions ✓

9/12
Type Conversion
2 questions ✓

10/12
Variables
4 questions ✓

11/12
In-Place Operators
2 questions ✓

12/12
Using an Editor
1 questions ✓

Module 1 Quiz
5 questions ✓

Floats

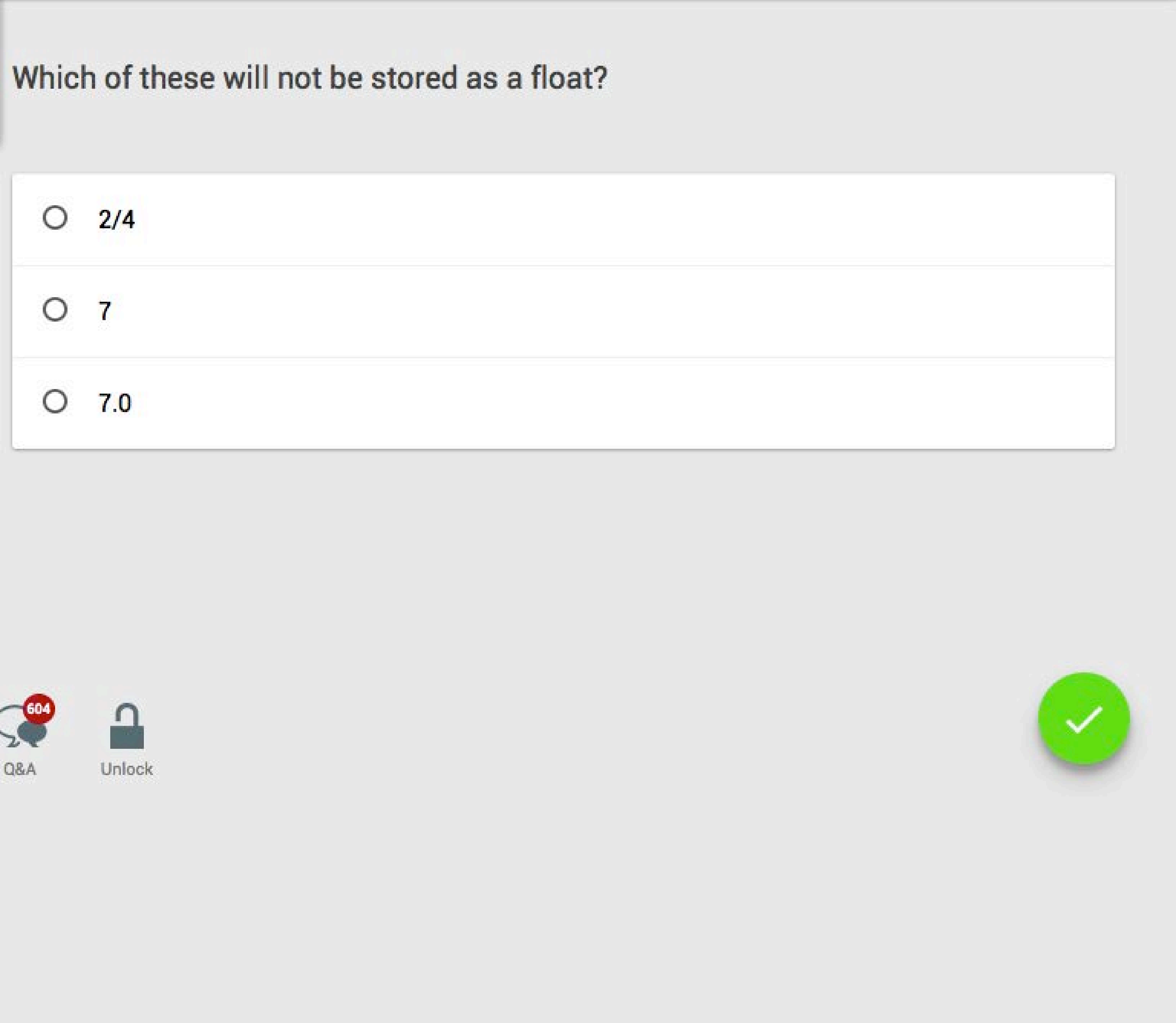
Floats are used in Python to represent numbers that aren't integers.
Some examples of numbers that are represented as floats are 0.5 and -7.8237591.
They can be created directly by entering a number with a decimal point, or by using operations such as division on integers. Extra zeros at the number's end are ignored.

```
>>> 3/4
0.75
>>> 9.8765000
9.8765
```

Computers can't store floats perfectly accurately, in the same way that we can't write down the complete decimal expansion of 1/3 (0.3333333333333333...). Keep this in mind, because it often leads to infuriating bugs!

152 COMMENTS

604
Q&A



Which of these will not be stored as a float?

- ○ 2/4
- ○ 7
- ○ 7.0

Floats

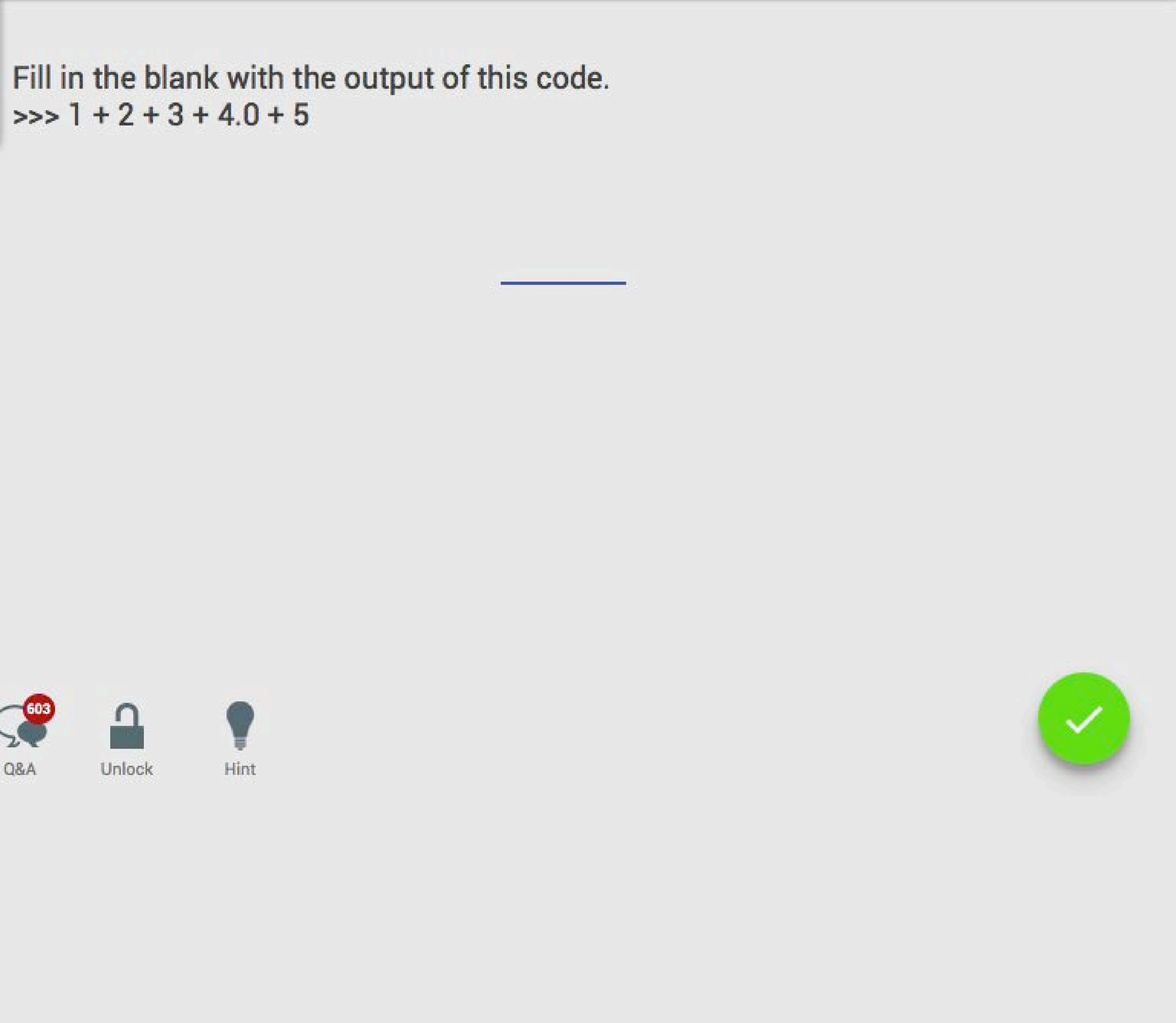
As you saw previously, dividing any two integers produces a float.
A float is also produced by running an operation on two floats, or on a float and an integer.

```
>>> 8 / 2
4.0
>>> 6 * 7.0
42.0
>>> 4 + 1.65
5.65
```

A float can be added to an integer, because Python silently converts the integer to a float. However, this implicit conversion is the exception rather the rule in Python - usually you have to convert values manually if you want to operate on them.

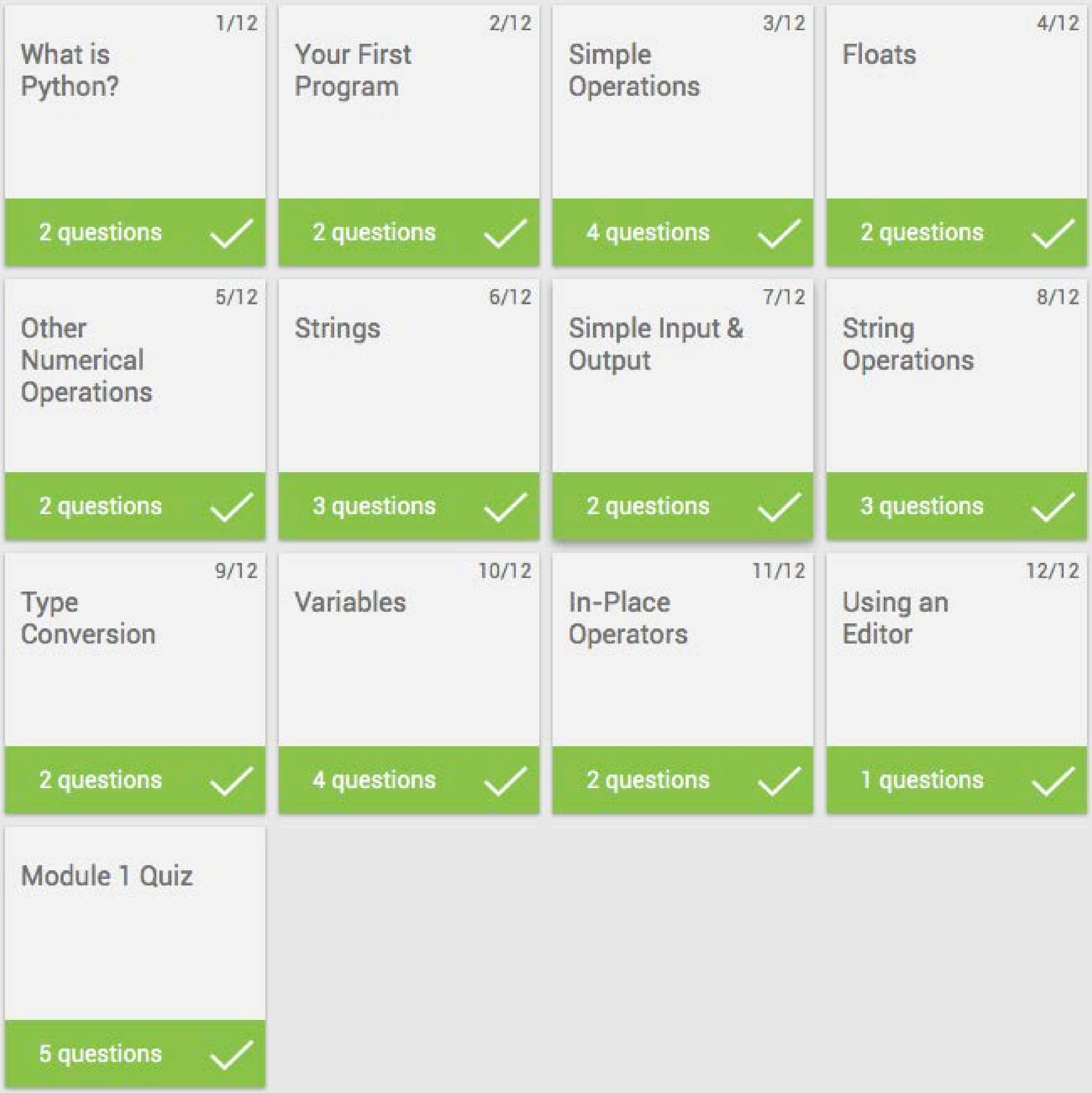
159 COMMENTS

604
Q&A



Fill in the blank with the output of this code.

```
>>> 1 + 2 + 3 + 4.0 + 5
```



1/12
What is Python?
2 questions ✓

2/12
Your First Program
2 questions ✓

3/12
Simple Operations
4 questions ✓

4/12
Floats
2 questions ✓

5/12
Other Numerical Operations
2 questions ✓

6/12
Strings
3 questions ✓

7/12
Simple Input & Output
2 questions ✓

8/12
String Operations
3 questions ✓

9/12
Type Conversion
2 questions ✓

10/12
Variables
4 questions ✓

11/12
In-Place Operators
2 questions ✓

12/12
Using an Editor
1 questions ✓

Module 1 Quiz
5 questions ✓

Exponentiation

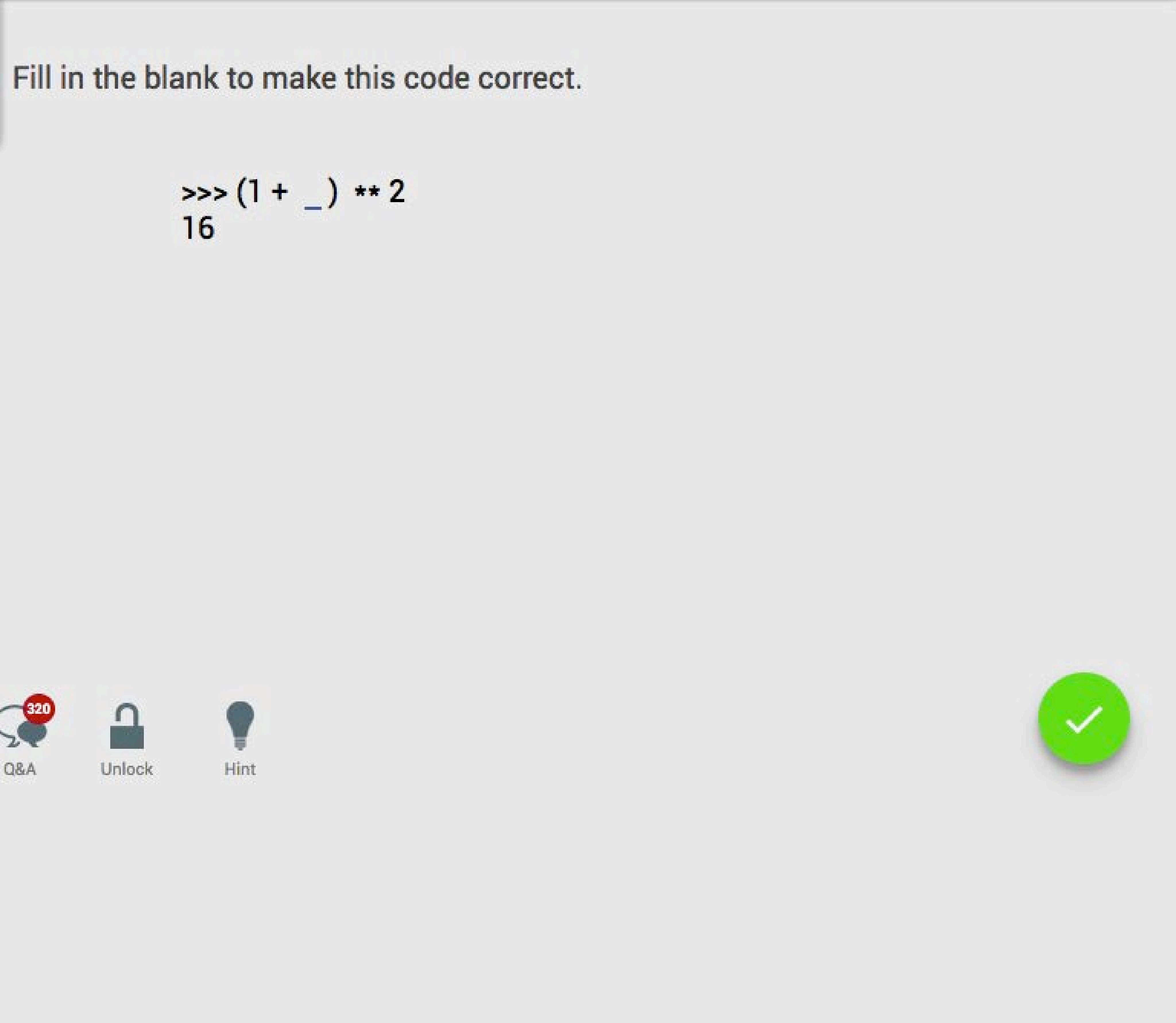
Besides addition, subtraction, multiplication, and division, Python also supports exponentiation, which is the raising of one number to the power of another. This operation is performed using two asterisks.

```
>>> 2**5
32
>>> 9 ** (1/2)
3.0
```

434 COMMENTS

320

Q&A



Fill in the blank to make this code correct.

```
>>> (1 + _ ) ** 2
16
```

Quotient & Remainder

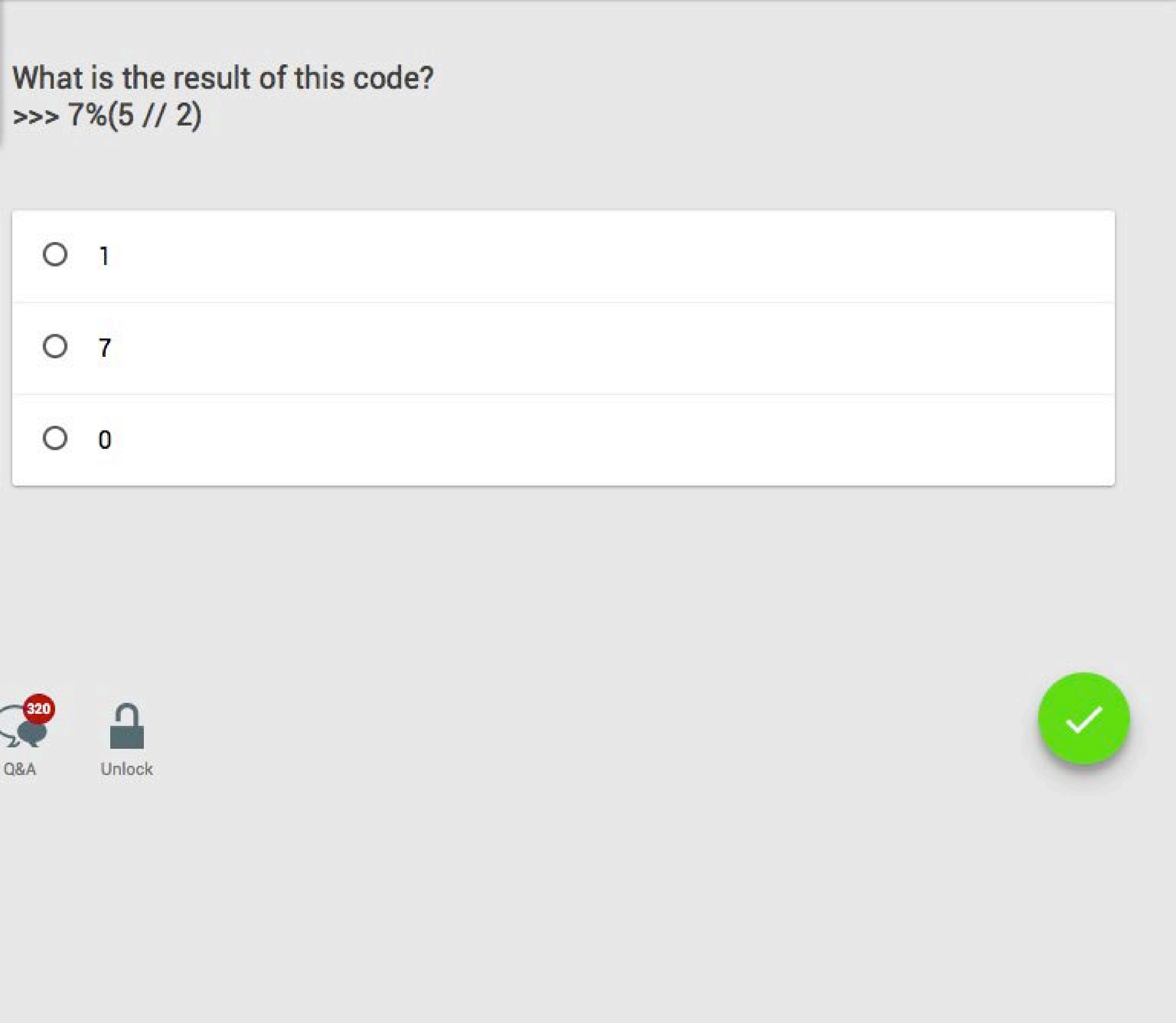
To determine the **quotient** and **remainder** of a division, use the **floor division** and **modulo** operators, respectively.
Floor division is done using two forward slashes.
The modulo operator is carried out with a percent symbol (%).
These operators can be used with both floats and integers.

This code shows that 6 goes into 20 three times, and the remainder when 1.25 is divided by 0.5 is 0.25.

```
>>> 20 // 6
3
>>> 1.25 % 0.5
0.25
```

459 COMMENTS

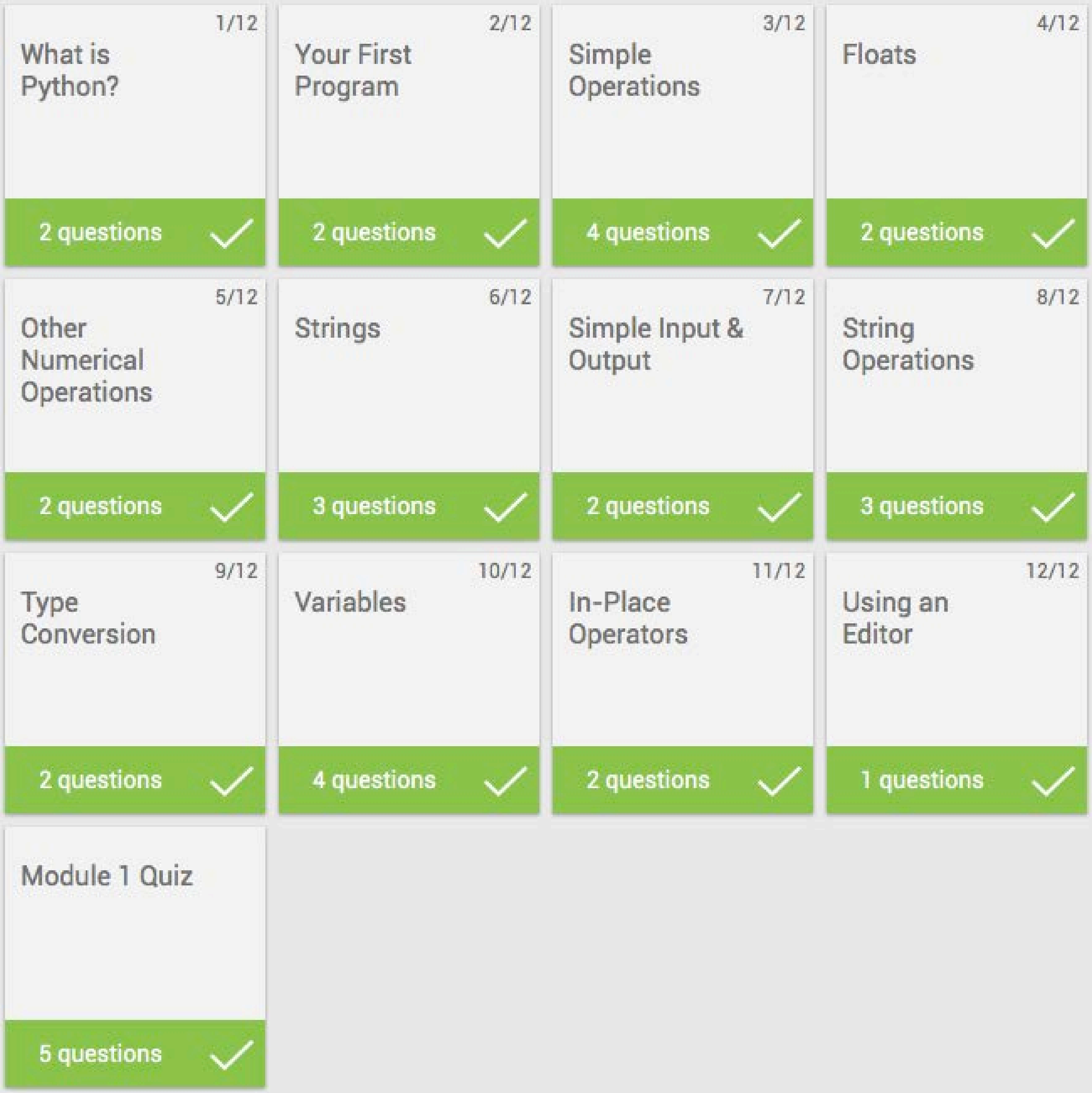
320
Q&A



What is the result of this code?

```
>>> 7%(5 // 2)
```

- 1
- 7
- 0



1/12
What is Python?
2 questions ✓

2/12
Your First Program
2 questions ✓

3/12
Simple Operations
4 questions ✓

4/12
Floats
2 questions ✓

5/12
Other Numerical Operations
2 questions ✓

6/12
Strings
3 questions ✓

7/12
Simple Input & Output
2 questions ✓

8/12
String Operations
3 questions ✓

9/12
Type Conversion
2 questions ✓

10/12
Variables
4 questions ✓

11/12
In-Place Operators
2 questions ✓

12/12
Using an Editor
1 questions ✓

Module 1 Quiz
5 questions ✓

Strings

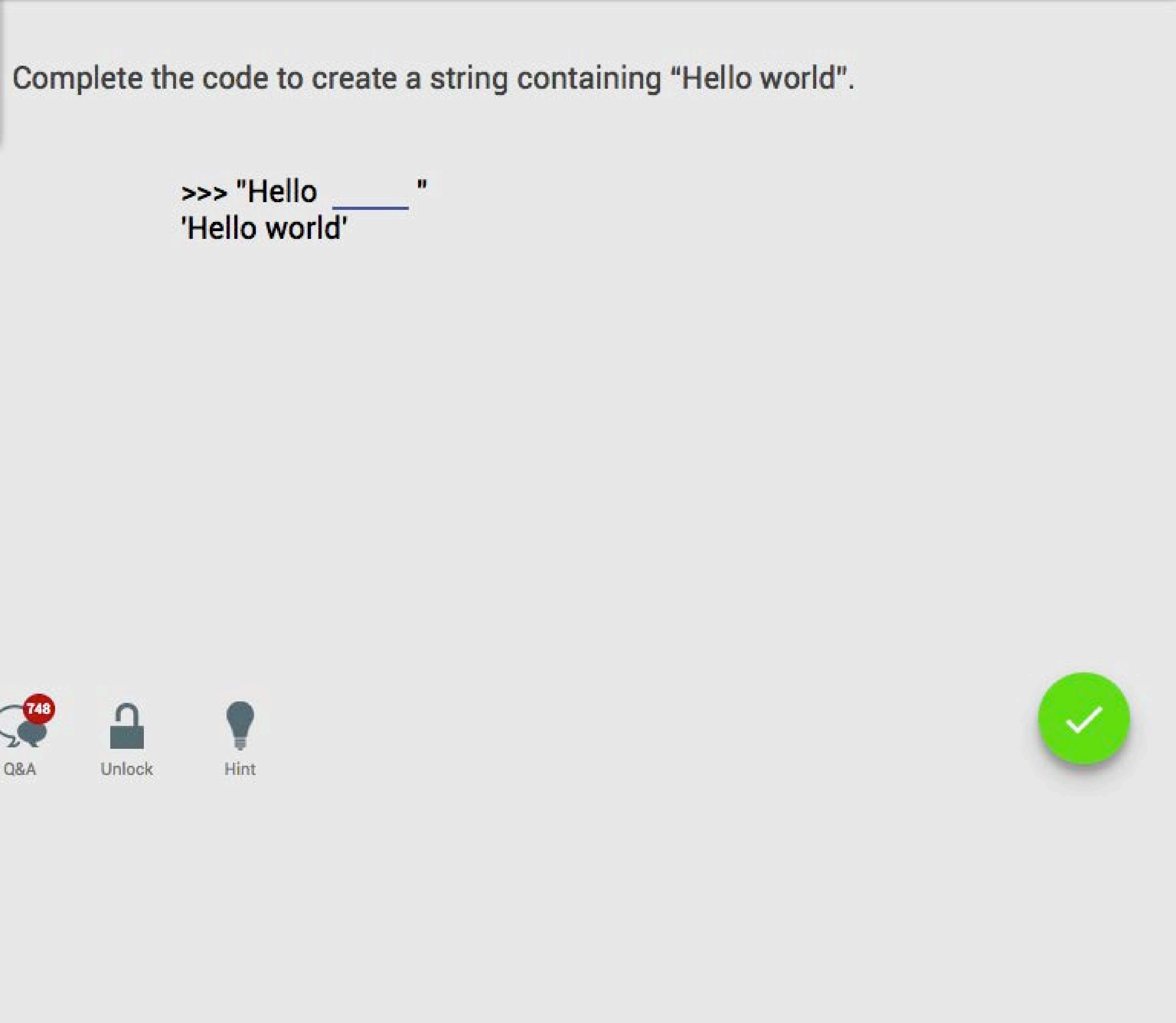
If you want to use text in Python, you have to use a string.
A string is created by entering text between **two single or double quotation marks**.

When the Python console displays a string, it generally uses single quotes. The delimiter used for a string doesn't affect how it behaves in any way.

```
>>> "Python is fun!"
'Python is fun!'
>>> 'Always look on the bright side of life'
'Always look on the bright side of life'
```

207 COMMENTS

748
Q&A



Complete the code to create a string containing “Hello world".

```
>>> "Hello ______ "
'Hello world'
```

Strings

Some characters can't be directly included in a string. For instance, double quotes can't be directly included in a double quote string; this would cause it to end prematurely.

Characters like these must be escaped by placing a **backslash** before them.
Other common characters that must be escaped are newlines and backslashes.
Double quotes only need to be escaped in double quote strings, and the same is true for single quote strings.

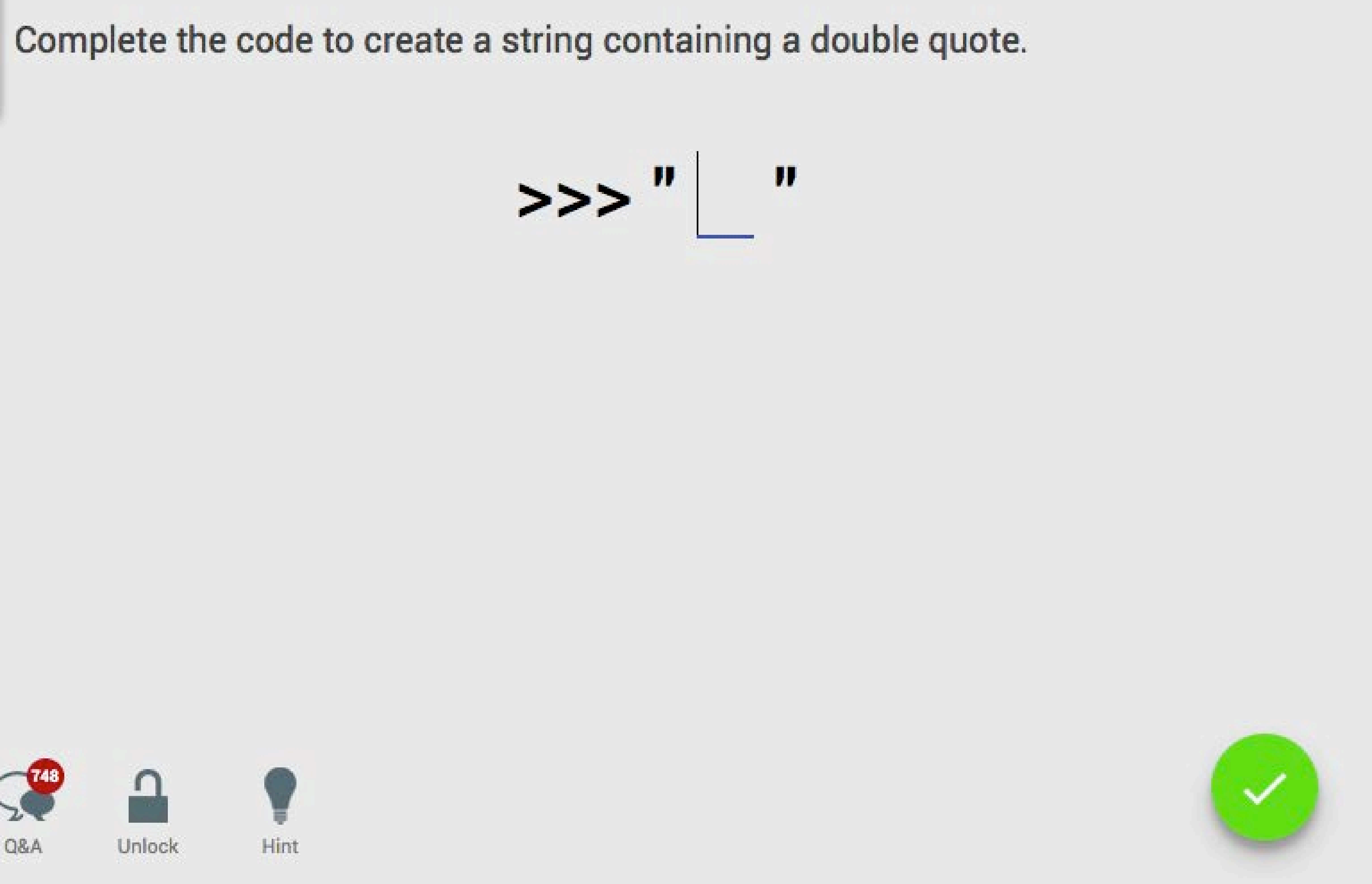
```
>>> 'Brian\'s mother: He\'s not the Messiah. He\'s a very naughty boy!'
'Brian's mother: He's not the Messiah. He's a very naughty boy!'
```

\n represents a new line.

> **Backslashes** can also be used to escape tabs, arbitrary Unicode characters, and various other things that can't be reliably printed. These characters are known as escape characters.

251 COMMENTS

748
Q&A



Complete the code to create a string containing a double quote.

```
>>> "___"
```

Newlines

Python provides an easy way to avoid manually writing "\n" to escape newlines in a string. Create a string with **three sets of quotes**, and newlines that are created by pressing Enter are automatically escaped for you.

```
>>> """Customer: Good morning.
Owner: Good morning, Sir. Welcome to the National Cheese Emporium."""

'Customer: Good morning.\nOwner: Good morning, Sir. Welcome to the National Cheese Emporium.'
```

As you can see, the **\n** was automatically put in the output, where we pressed Enter.

187 COMMENTS

748

Q&A

Fill in the missing part of the output.

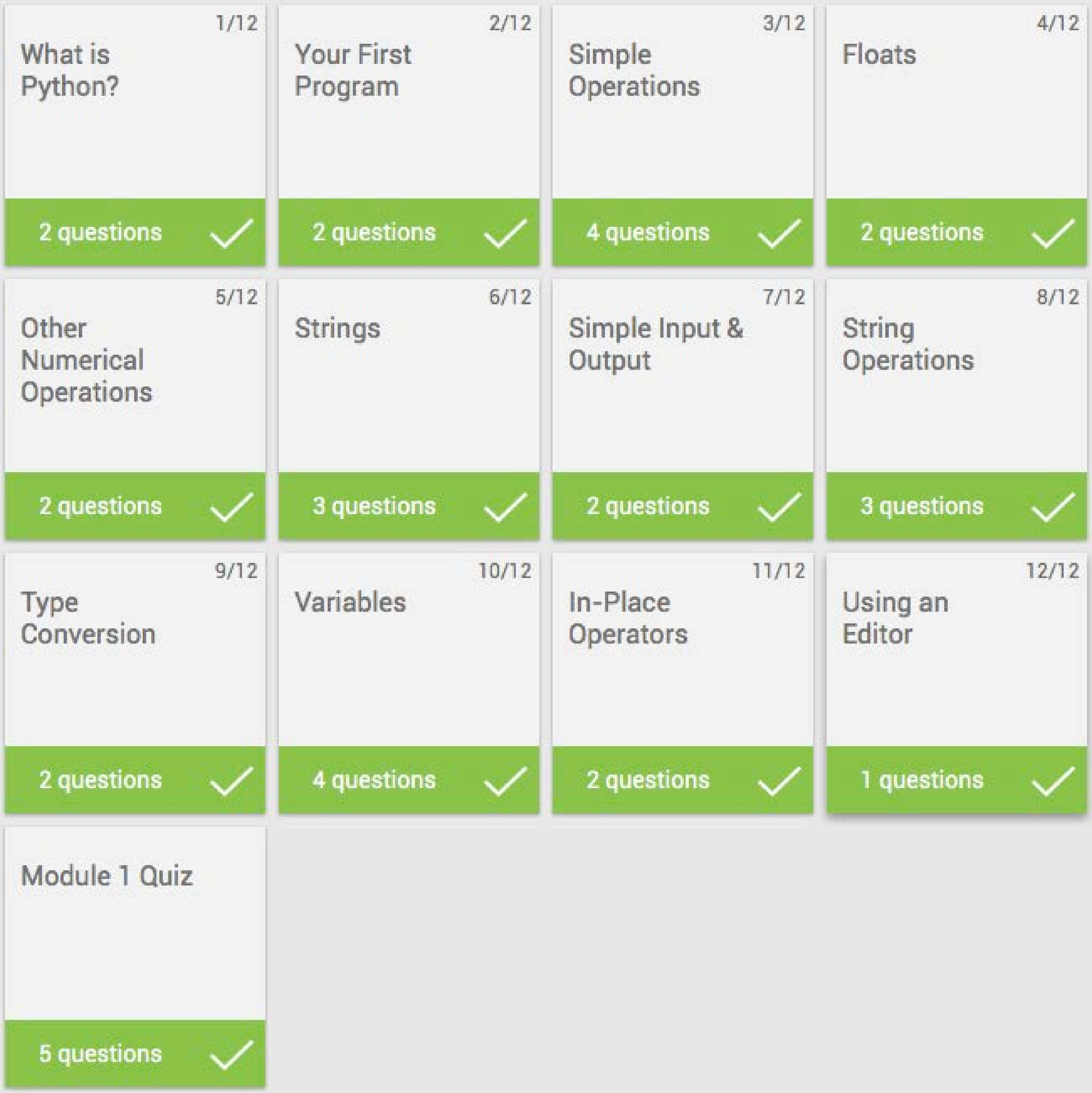
```
>>> """First line
second line"""
'First line  __  second line'
```

748

Q&A

Unlock

Hint



1/12
What is Python?
2 questions ✓

2/12
Your First Program
2 questions ✓

3/12
Simple Operations
4 questions ✓

4/12
Floats
2 questions ✓

5/12
Other Numerical Operations
2 questions ✓

6/12
Strings
3 questions ✓

7/12
Simple Input & Output
2 questions ✓

8/12
String Operations
3 questions ✓

9/12
Type Conversion
2 questions ✓

10/12
Variables
4 questions ✓

11/12
In-Place Operators
2 questions ✓

12/12
Using an Editor
1 questions ✓

Module 1 Quiz
5 questions ✓

Output

Usually, programs take **input** and process it to produce **output**.
In Python, you can use the **print** function to produce output. This displays a textual representation of something to the screen.

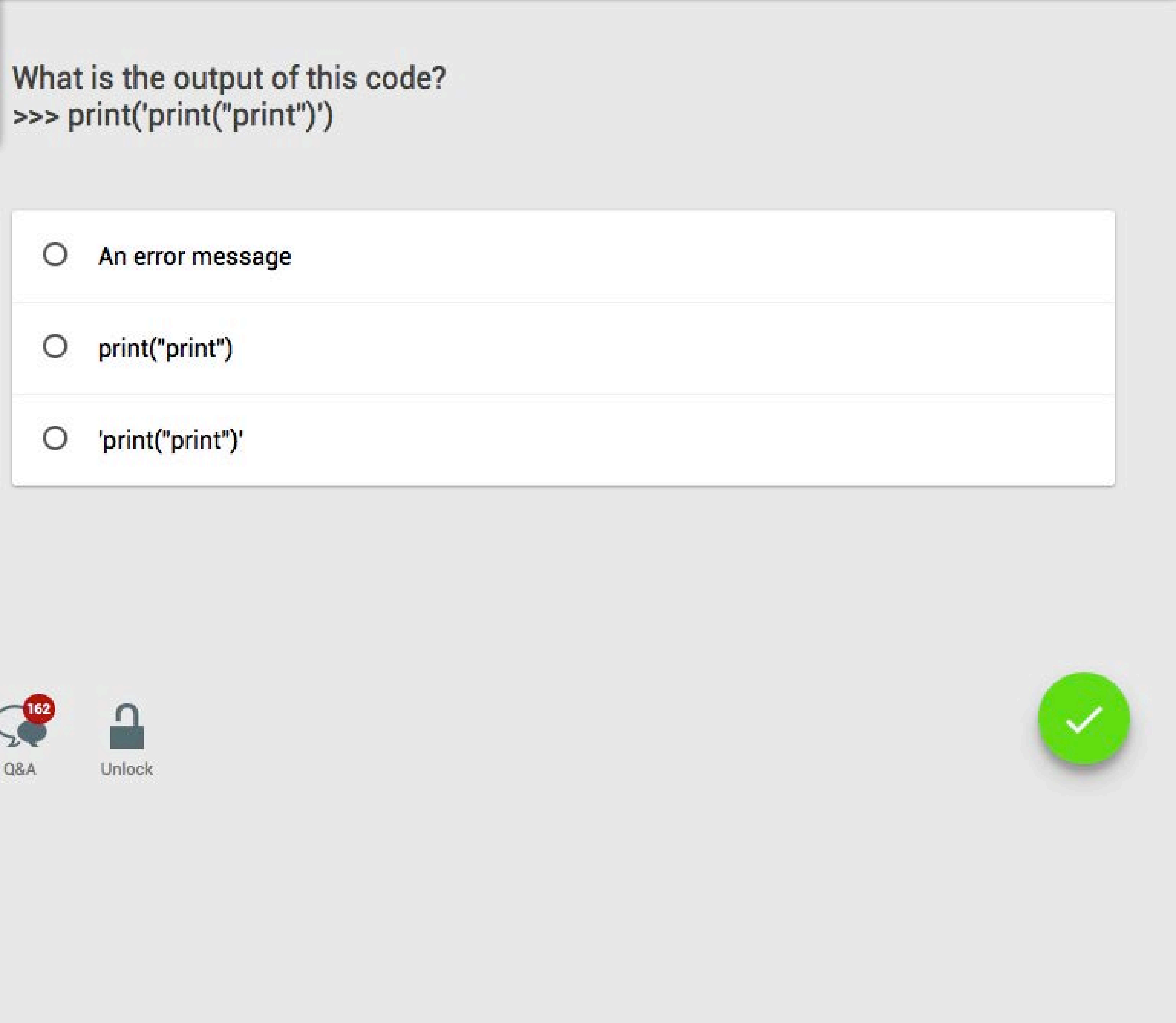
```
>>> print(1 + 1)
2
>>> print("Hello\nWorld!")
Hello
World!
```

Try It Yourself

When a string is printed, the quotes around it are not displayed.

117 COMMENTS

162
Q&A



What is the output of this code?

```
>>> print('print("print")')
```

- ○ An error message
- ○ print("print")
- ○ 'print("print")'

Input

To get input from the user in Python, you can use the intuitively named **input** function.
The function prompts the user for input, and returns what they enter as a string (with the contents automatically escaped).

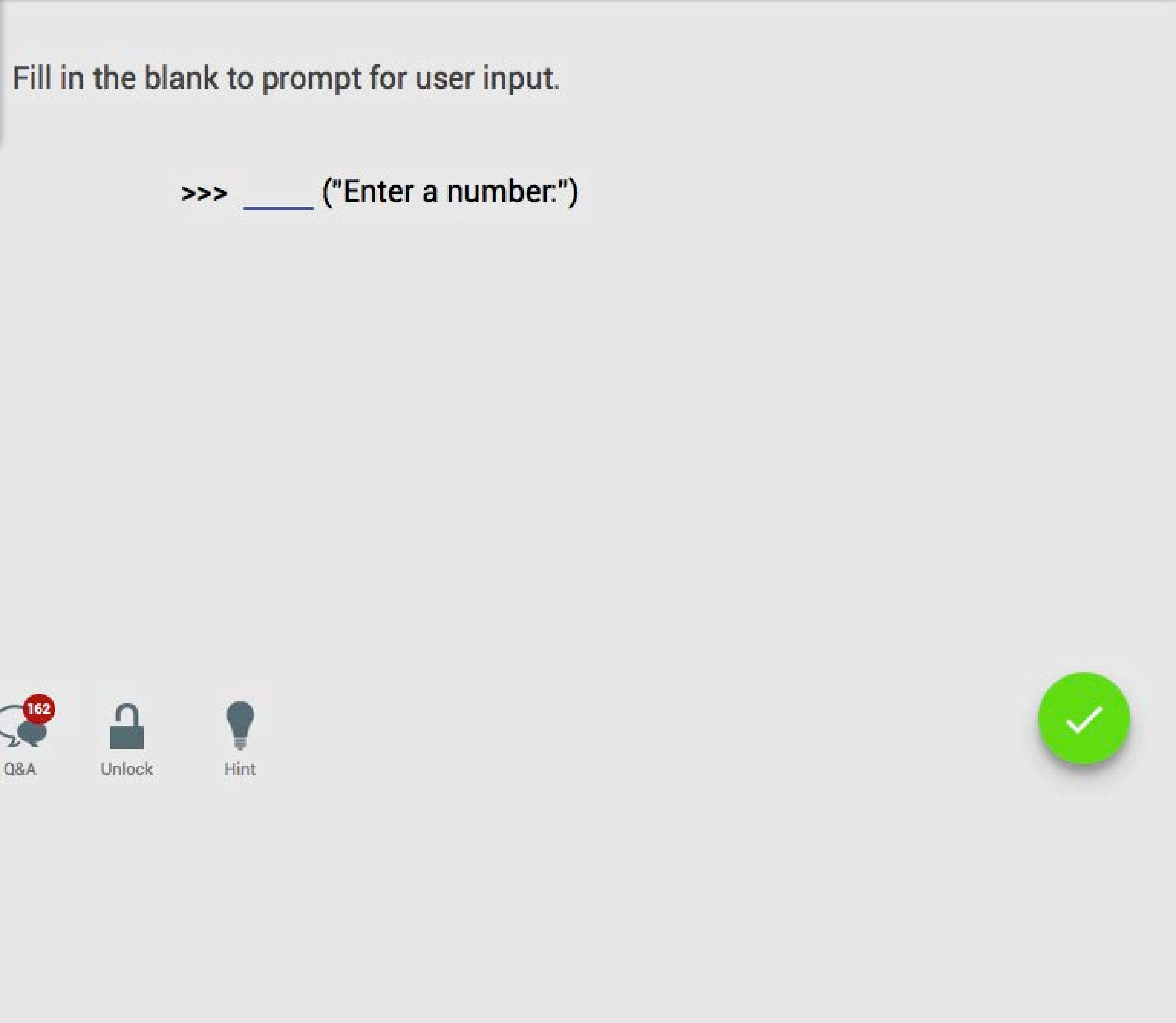
```
>>> input("Enter something please: ")
Enter something please: This is what\nthe user enters!

'This is what\\nthe user enters!'
```

Try It Yourself

The **print** and **input** functions aren't very useful at the Python console, which automatically does input and output. However, they are very useful in actual programs.

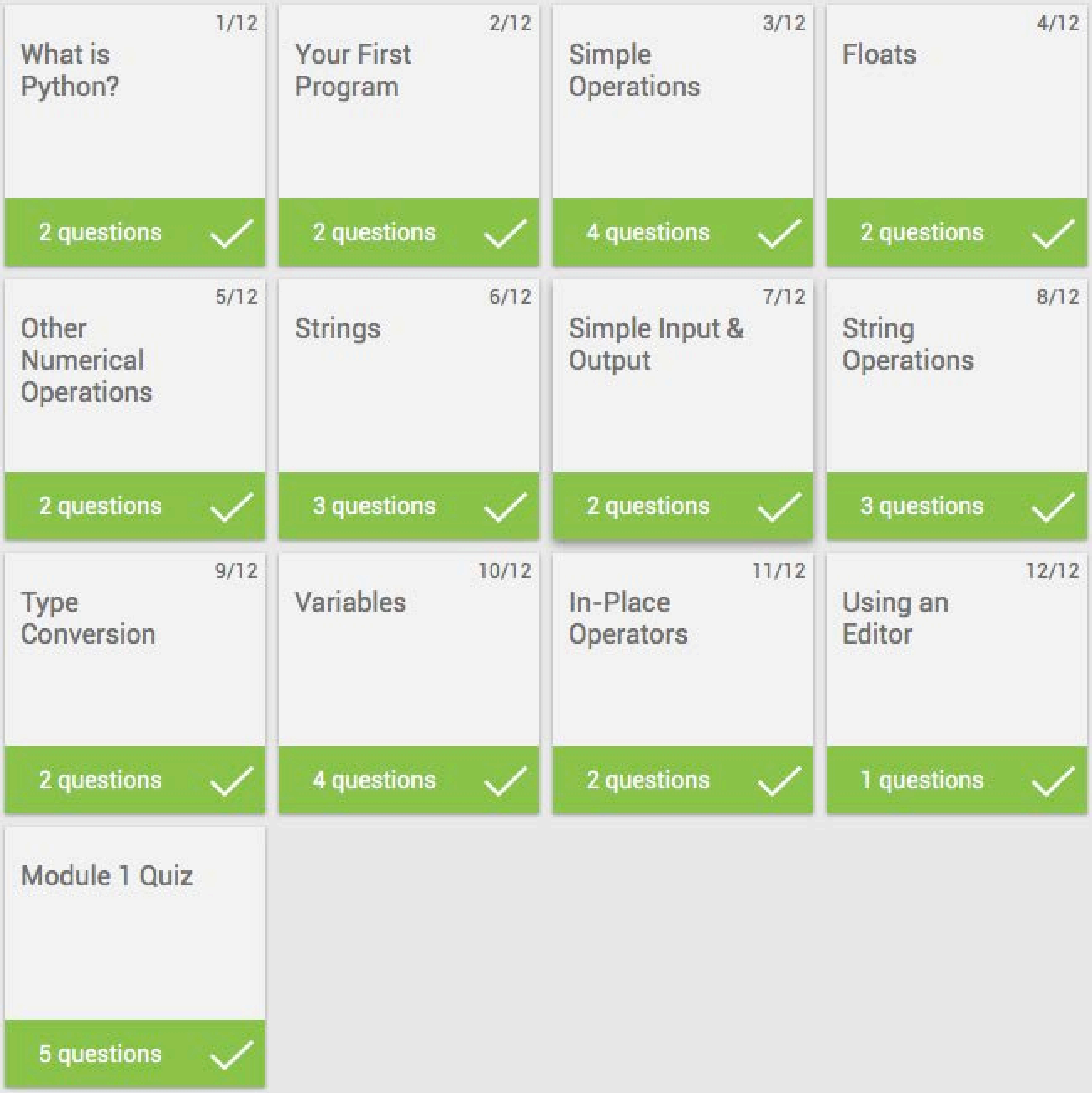
310 COMMENTS

162
Q&A



Fill in the blank to prompt for user input.

```
>>> _____ ("Enter a number:")
```



1/12
What is Python?
2 questions ✓

2/12
Your First Program
2 questions ✓

3/12
Simple Operations
4 questions ✓

4/12
Floats
2 questions ✓

5/12
Other Numerical Operations
2 questions ✓

6/12
Strings
3 questions ✓

7/12
Simple Input & Output
2 questions ✓

8/12
String Operations
3 questions ✓

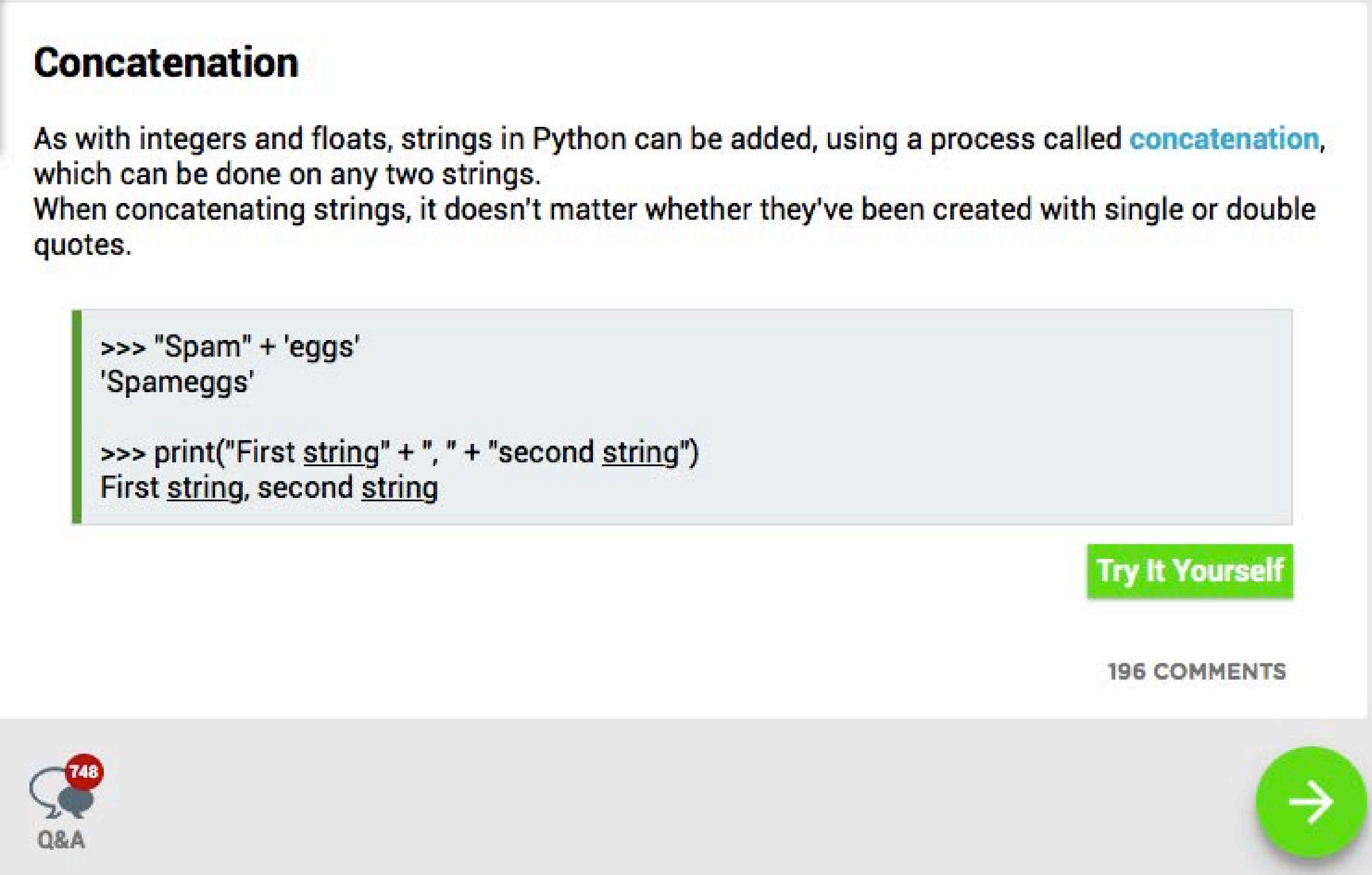
9/12
Type Conversion
2 questions ✓

10/12
Variables
4 questions ✓

11/12
In-Place Operators
2 questions ✓

12/12
Using an Editor
1 questions ✓

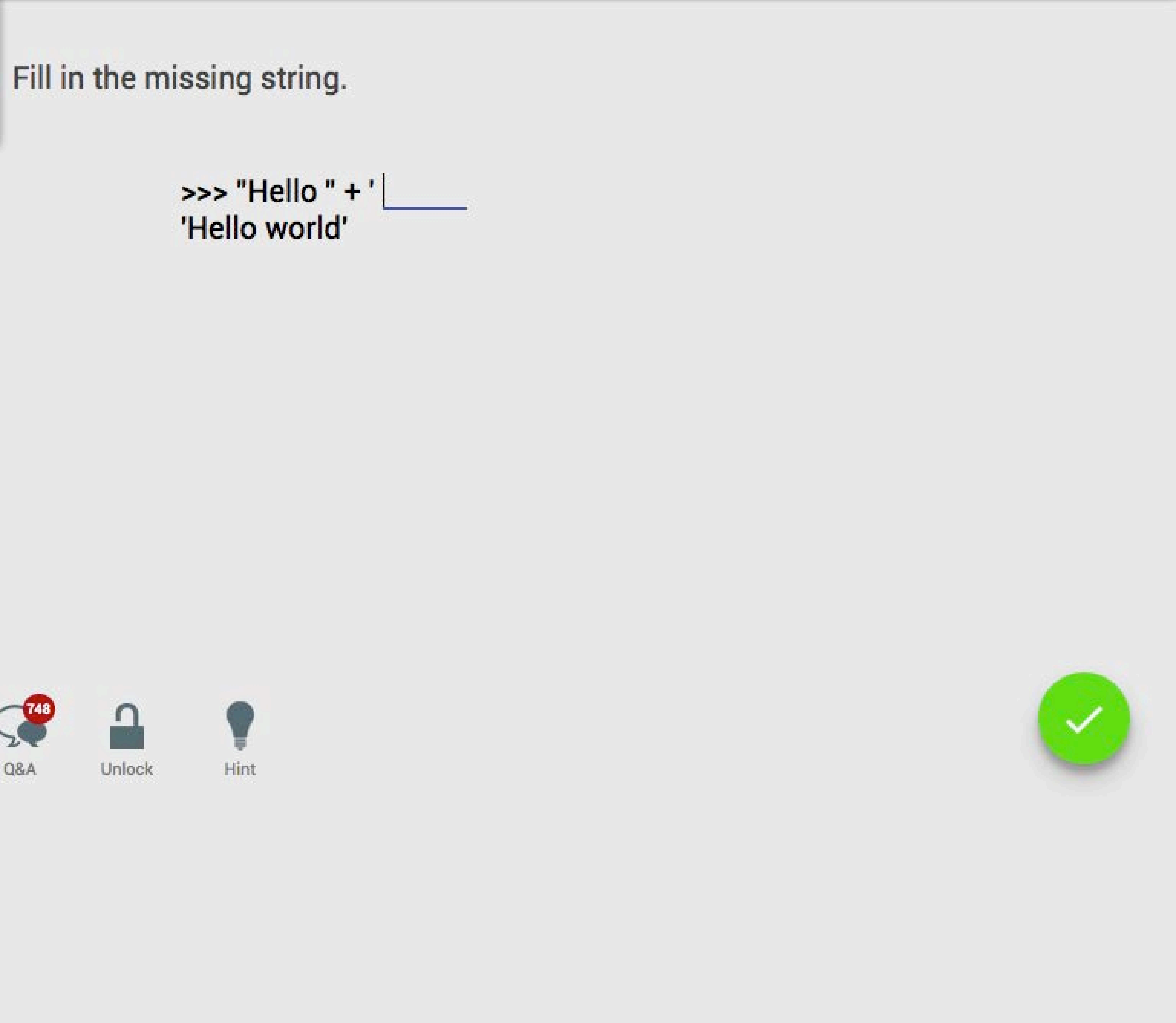
Module 1 Quiz
5 questions ✓



Concatenation

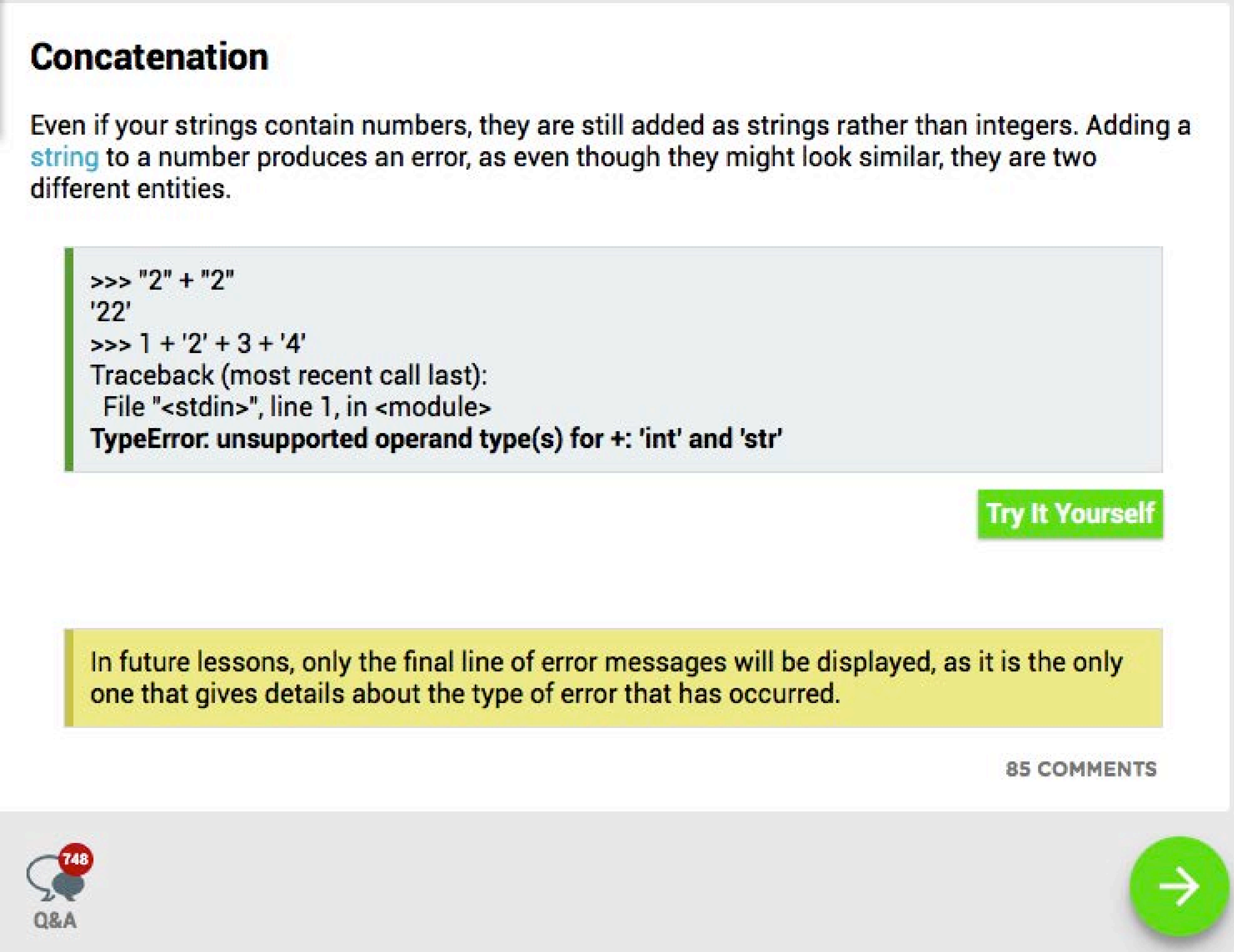
As with integers and floats, strings in Python can be added, using a process called concatenation, which can be done on any two strings.
When concatenating strings, it doesn't matter whether they've been created with single or double quotes.

```
>>> "Spam" + 'eggs'
'Spameggs'

>>> print("First string" + ", " + "second string")
First string, second string
```



Fill in the missing string.

```
>>> "Hello " + '
'Hello world'
```

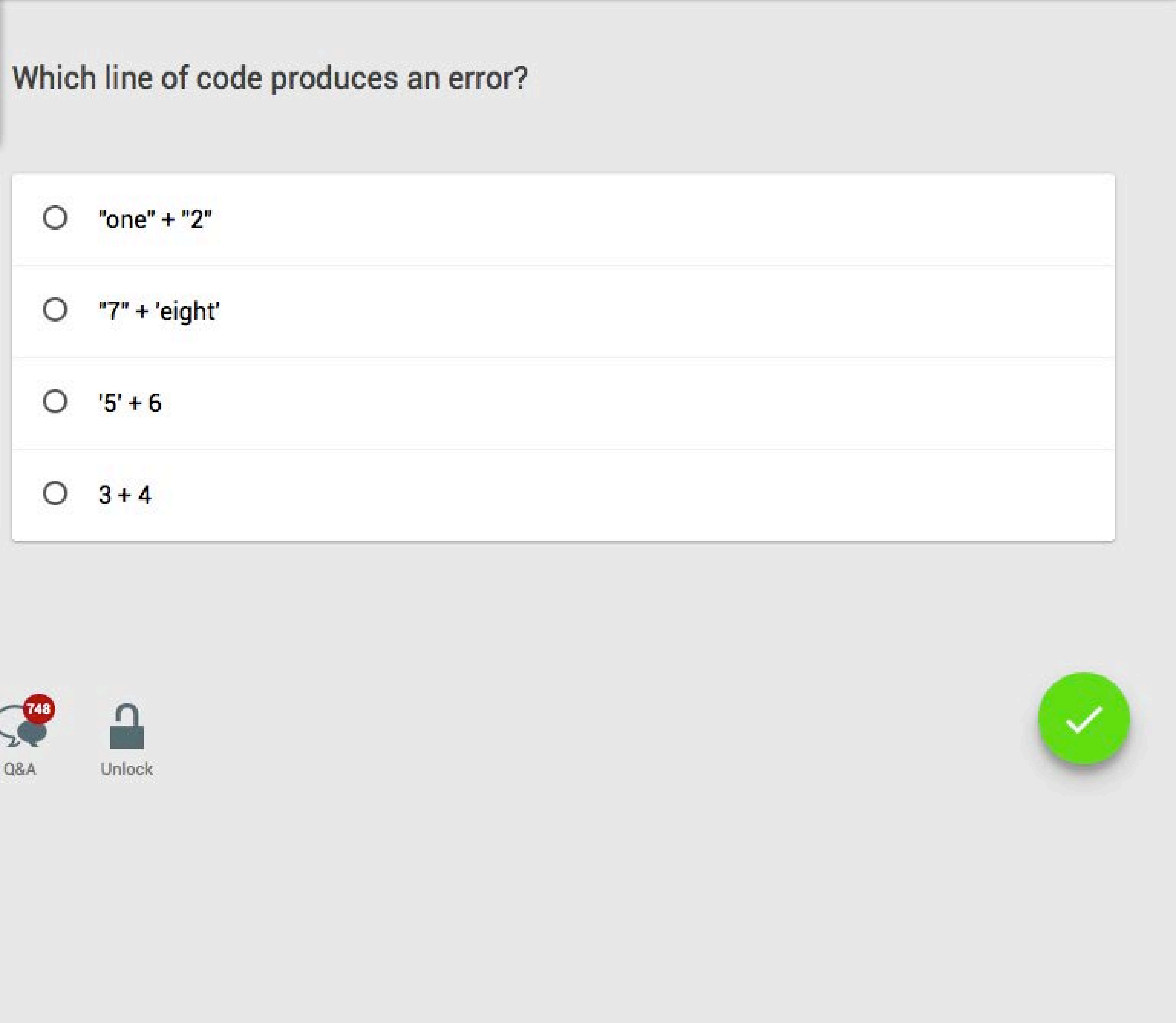


Concatenation

Even if your strings contain numbers, they are still added as strings rather than integers. Adding a string to a number produces an error, as even though they might look similar, they are two different entities.

```
>>> "2" + "2"
'22'
>>> 1 + '2' + 3 + '4'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In future lessons, only the final line of error messages will be displayed, as it is the only one that gives details about the type of error that has occurred.



Which line of code produces an error?

- ○ "one" + "2"
- ○ "7" + 'eight'
- ○ '5' + 6
- ○ 3 + 4

String Operations

Strings can also be **multiplied** by integers. This produces a repeated version of the original string. The order of the string and the integer doesn't matter, but the string usually comes first.

Strings can't be multiplied by other strings. Strings also can't be multiplied by floats, even if the floats are whole numbers.

```
>>> print("spam" * 3)
spamspamspam

>>> 4 * '2'
'2222'

>>> '17' * '87'
TypeError: can't multiply sequence by non-int of type 'str'

>>> 'pythonisfun' * 7.0
TypeError: can't multiply sequence by non-int of type 'float'
```

Try It Yourself

122 COMMENTS

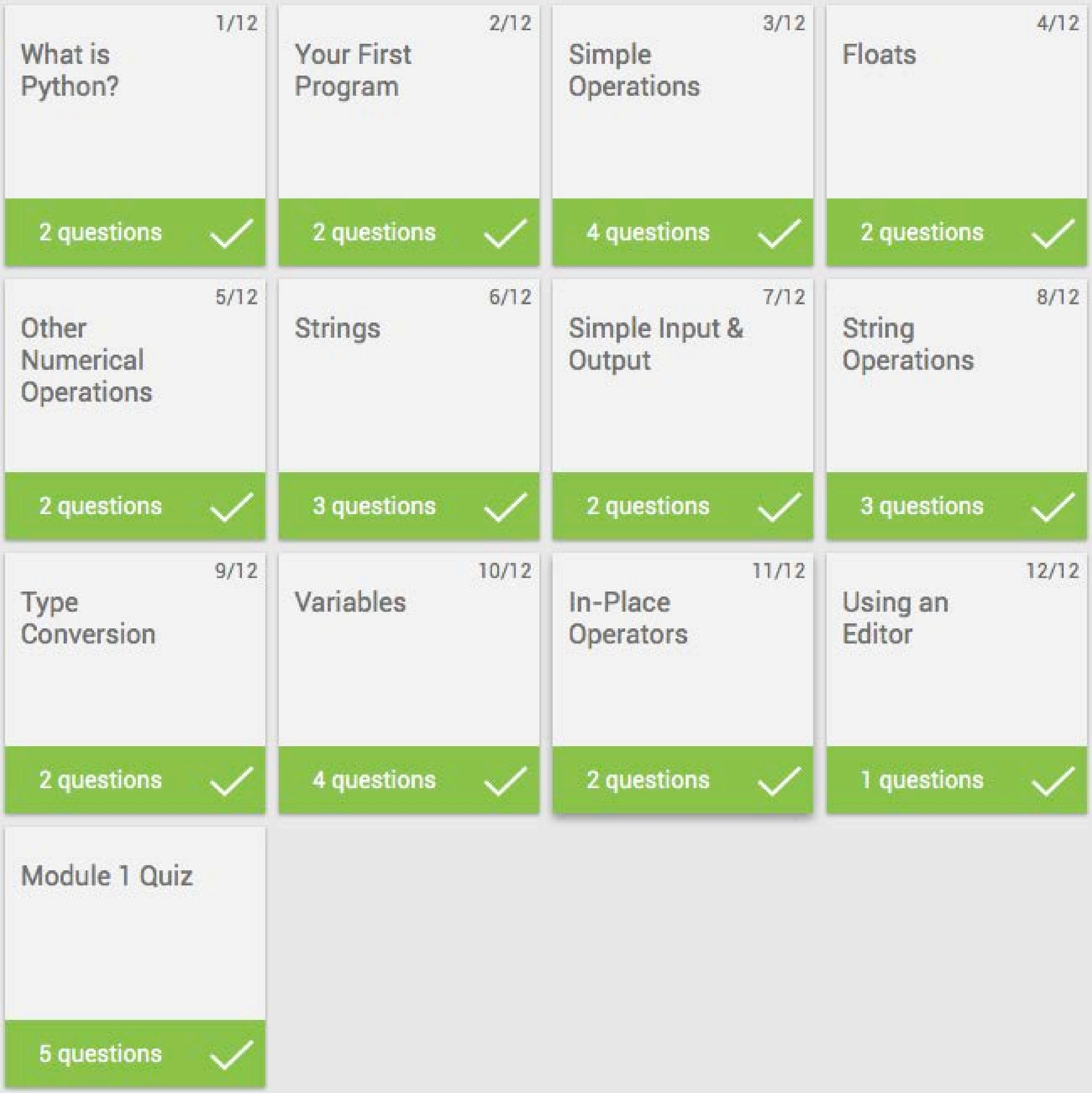
748
Q&A

What is the output of this code?

```
>>> print(3 * '7')
```

748

Q&A Unlock Hint



1/12
What is Python?
2 questions ✓

2/12
Your First Program
2 questions ✓

3/12
Simple Operations
4 questions ✓

4/12
Floats
2 questions ✓

5/12
Other Numerical Operations
2 questions ✓

6/12
Strings
3 questions ✓

7/12
Simple Input & Output
2 questions ✓

8/12
String Operations
3 questions ✓

9/12
Type Conversion
2 questions ✓

10/12
Variables
4 questions ✓

11/12
In-Place Operators
2 questions ✓

12/12
Using an Editor
1 questions ✓

Module 1 Quiz
5 questions ✓

Type Conversion

In Python, it's impossible to complete certain operations due to the types involved. For instance, you can't add two strings containing the numbers 2 and 3 together to produce the integer 5, as the operation will be performed on strings, making the result '23'.
The solution to this is **type conversion.**
In that example, you would use the **int** function.

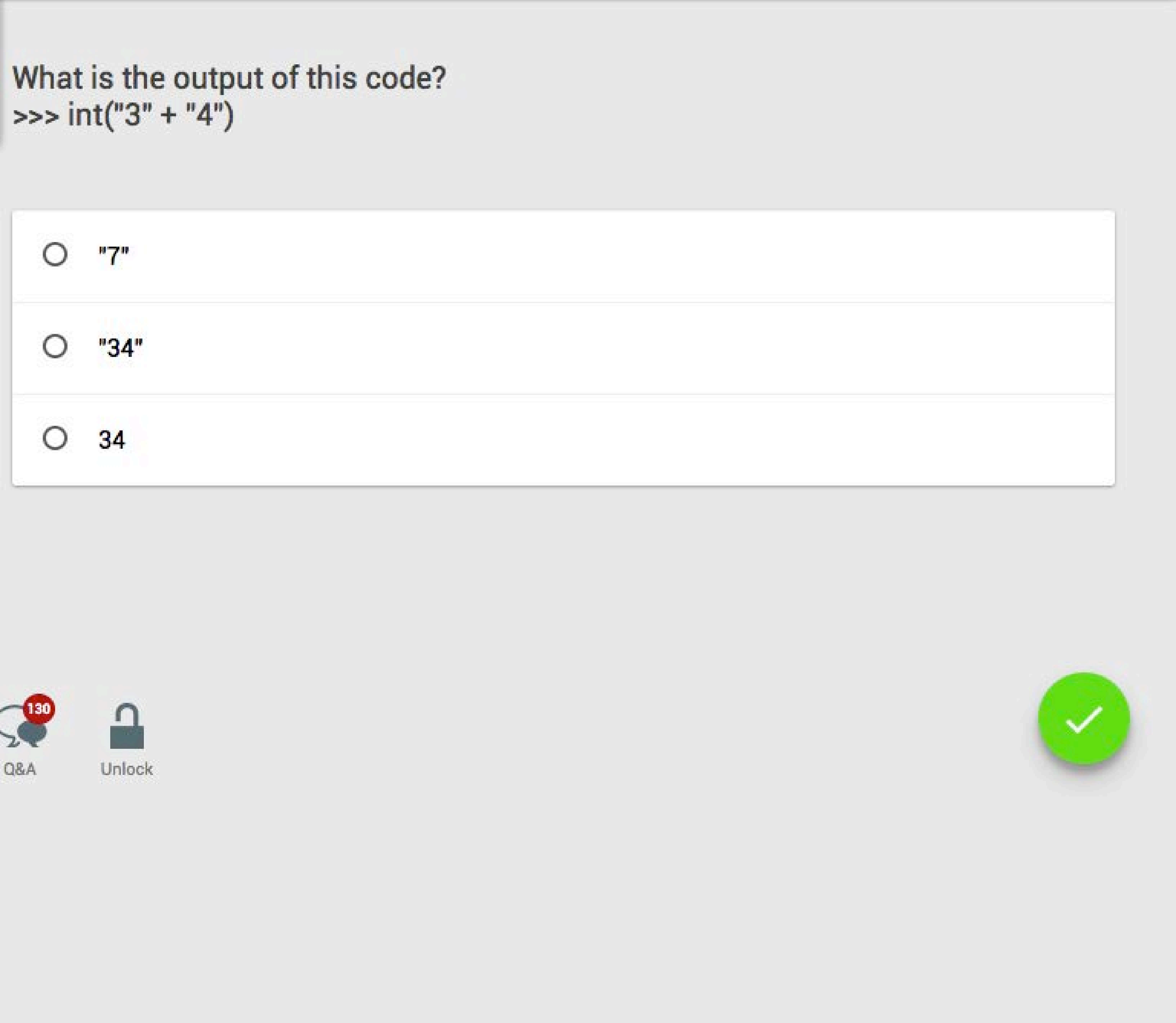
```
>>> "2" + "3"
'23'
>>> int("2") + int("3")
5
```

Try It Yourself

In Python, the types we have used so far have been **integers**, **floats**, and **strings**. The functions used to convert to these are **int**, **float** and **str**, respectively.

135 COMMENTS

130
Q&A



What is the output of this code?

```
>>> int("3" + "4")
```

- ○ "7"
- ○ "34"
- ○ 34

Type Conversion

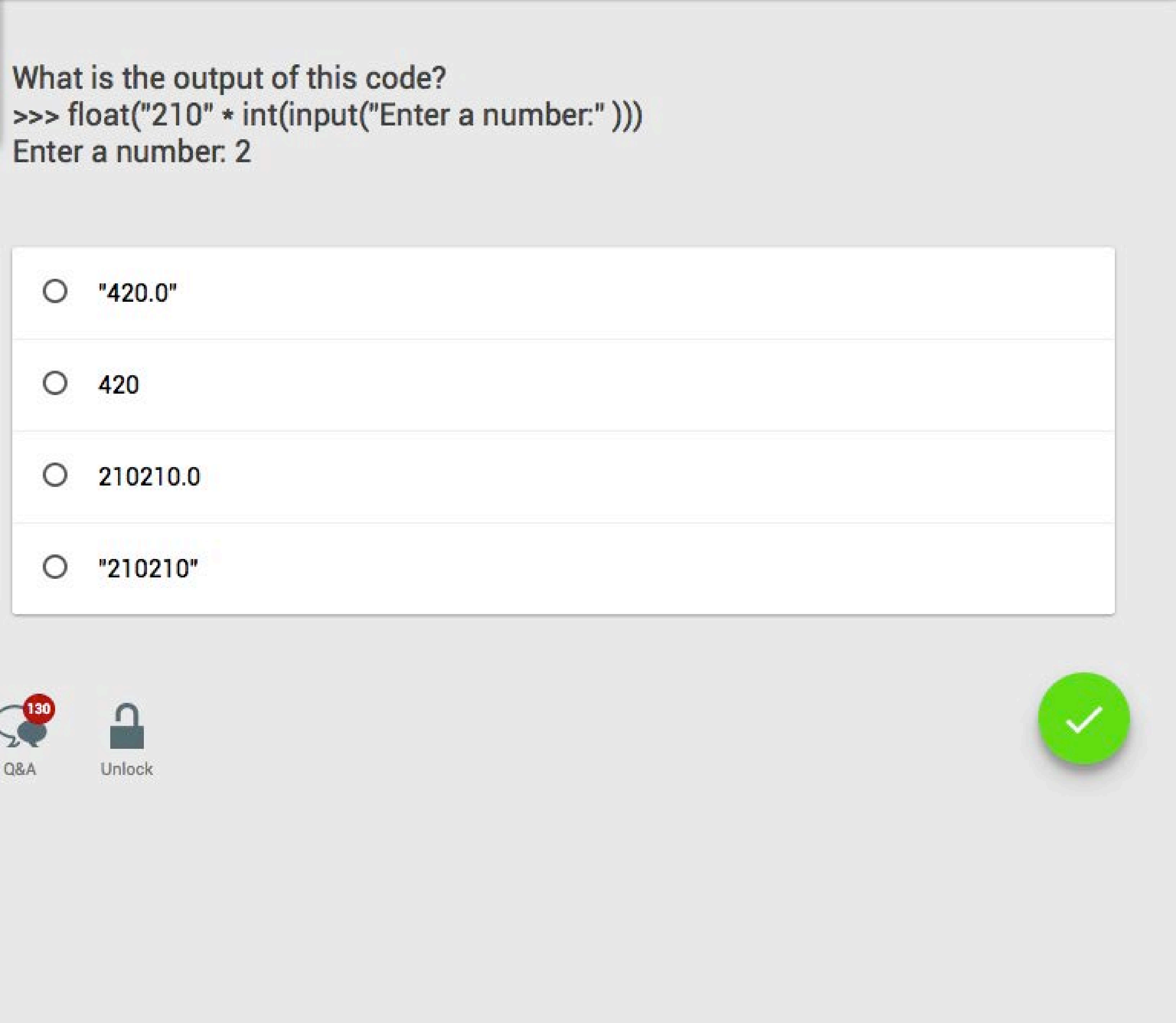
Another example of type conversion is turning user input (which is a string) to numbers (**integers** or **floats**), to allow for the performance of calculations.

```
>>> float(input("Enter a number: ")) + float(input("Enter another number: "))
Enter a number: 40
Enter another number: 2
42.0
```

Try It Yourself

368 COMMENTS

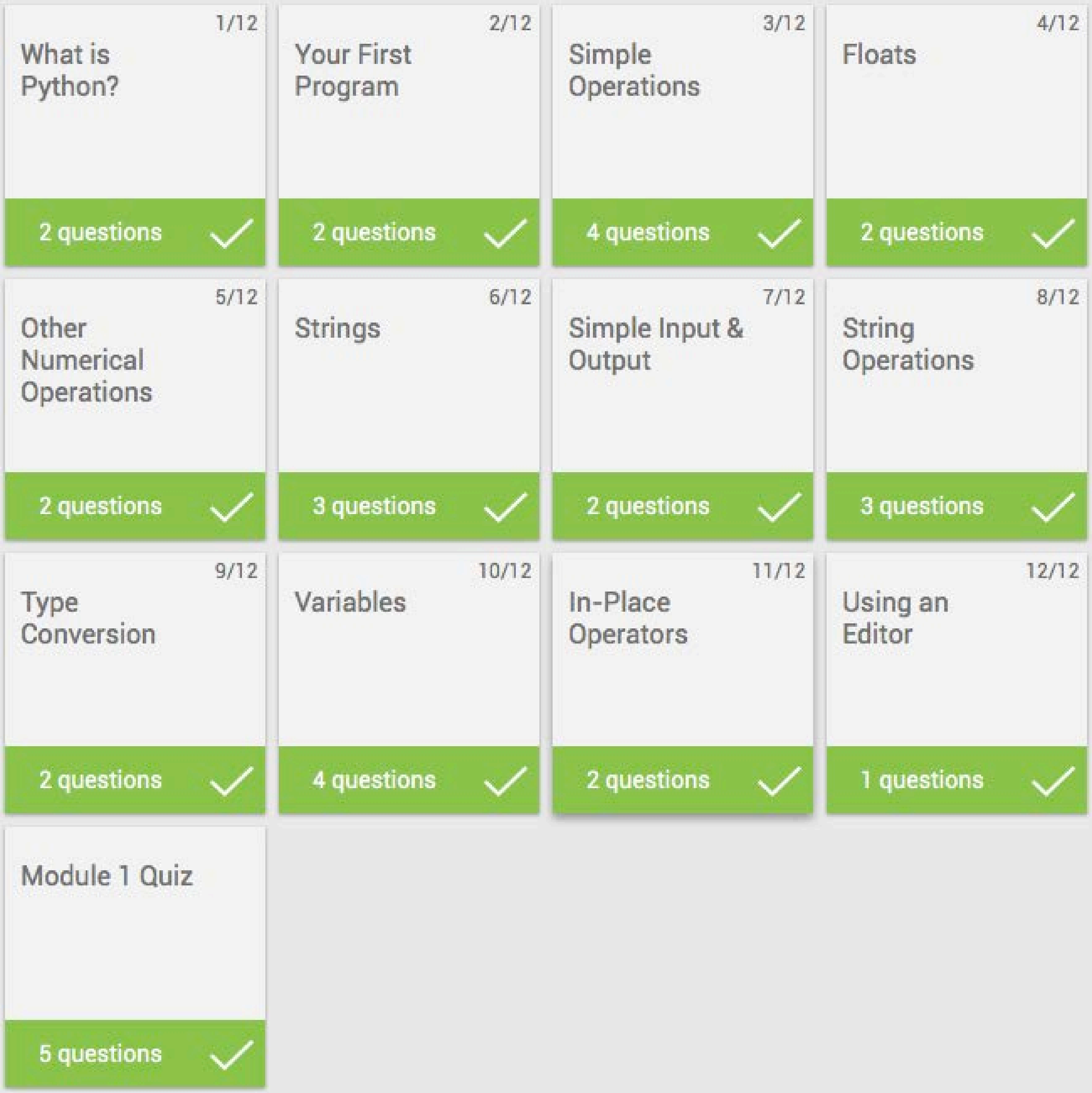
130
Q&A



What is the output of this code?

```
>>> float("210" * int(input("Enter a number:" )))
Enter a number: 2
```

- ○ "420.0"
- ○ 420
- ○ 210210.0
- ○ "210210"


1/12
What is Python?
2 questions ✓

2/12
Your First Program
2 questions ✓

3/12
Simple Operations
4 questions ✓

4/12
Floats
2 questions ✓

5/12
Other Numerical Operations
2 questions ✓

6/12
Strings
3 questions ✓

7/12
Simple Input & Output
2 questions ✓

8/12
String Operations
3 questions ✓

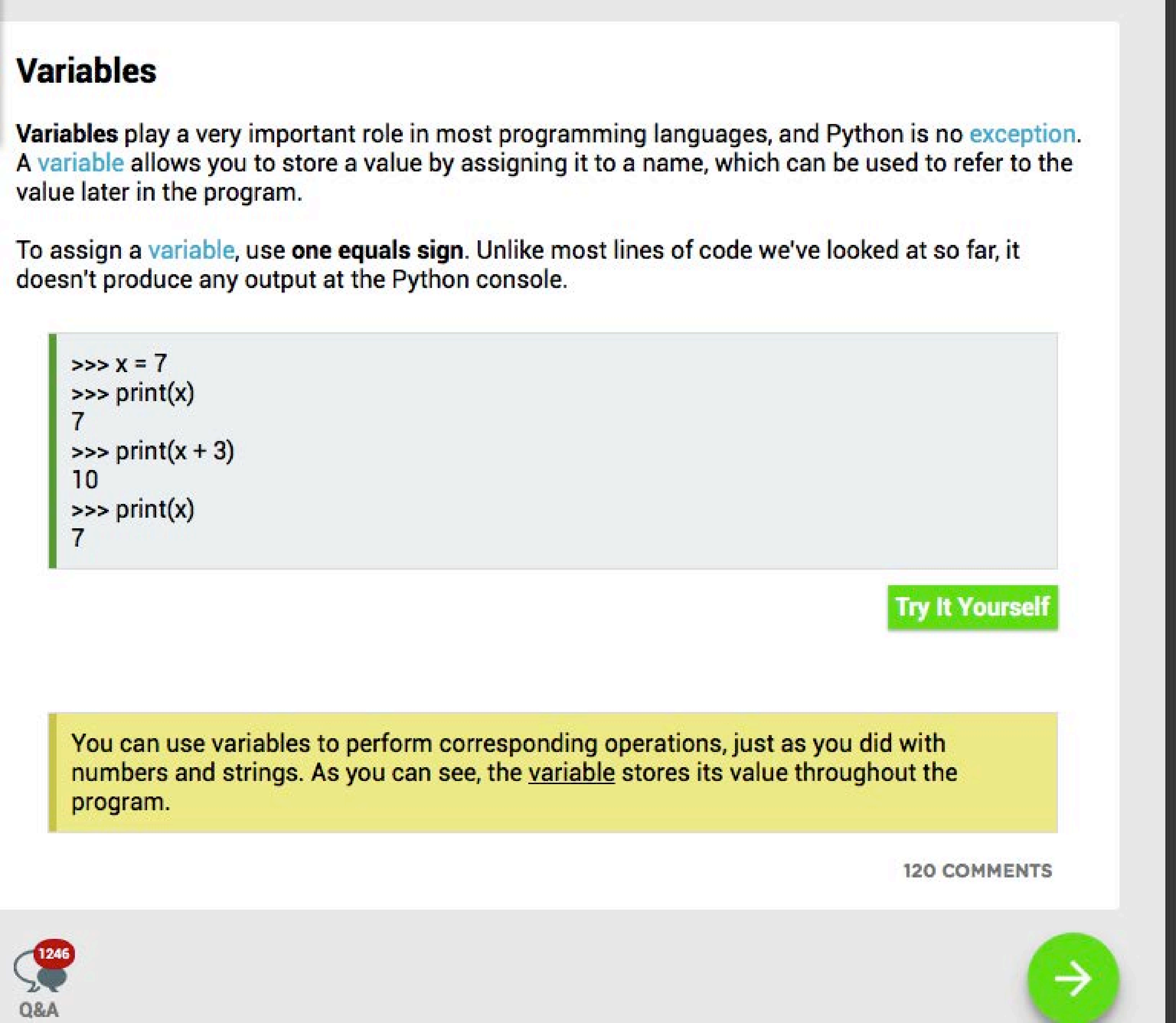
9/12
Type Conversion
2 questions ✓

10/12
Variables
4 questions ✓

11/12
In-Place Operators
2 questions ✓

12/12
Using an Editor
1 questions ✓

Module 1 Quiz
5 questions ✓



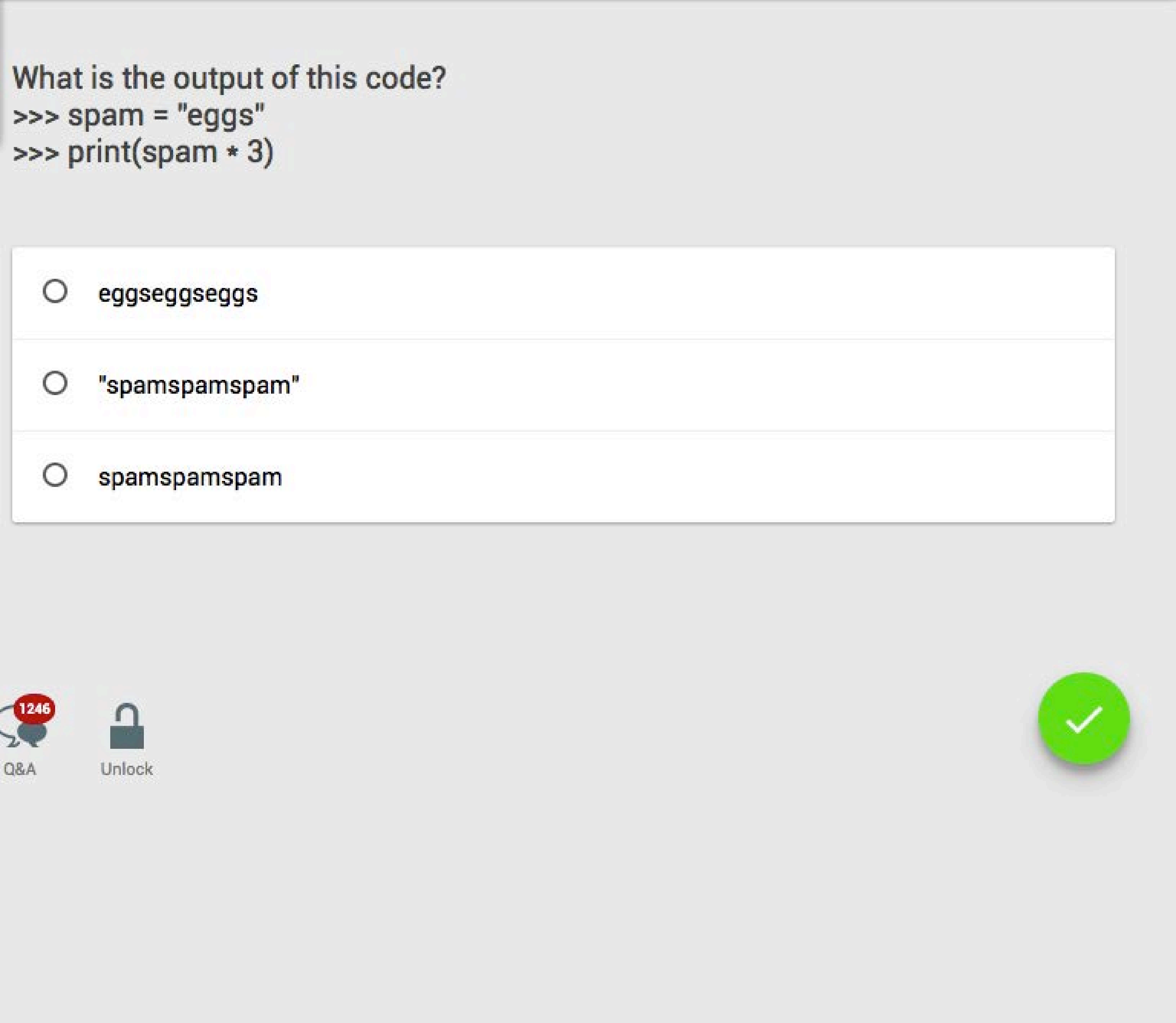
Variables

Variables play a very important role in most programming languages, and Python is no exception. A variable allows you to store a value by assigning it to a name, which can be used to refer to the value later in the program.

To assign a variable, use **one equals sign**. Unlike most lines of code we've looked at so far, it doesn't produce any output at the Python console.

```
>>> x = 7
>>> print(x)
7
>>> print(x + 3)
10
>>> print(x)
7
```

You can use variables to perform corresponding operations, just as you did with numbers and strings. As you can see, the variable stores its value throughout the program.



What is the output of this code?

```
>>> spam = "eggs"
>>> print(spam * 3)
```

- ○ eggseggseggs
- ○ "spamspamspam"
- ○ spamspamspam

Variables

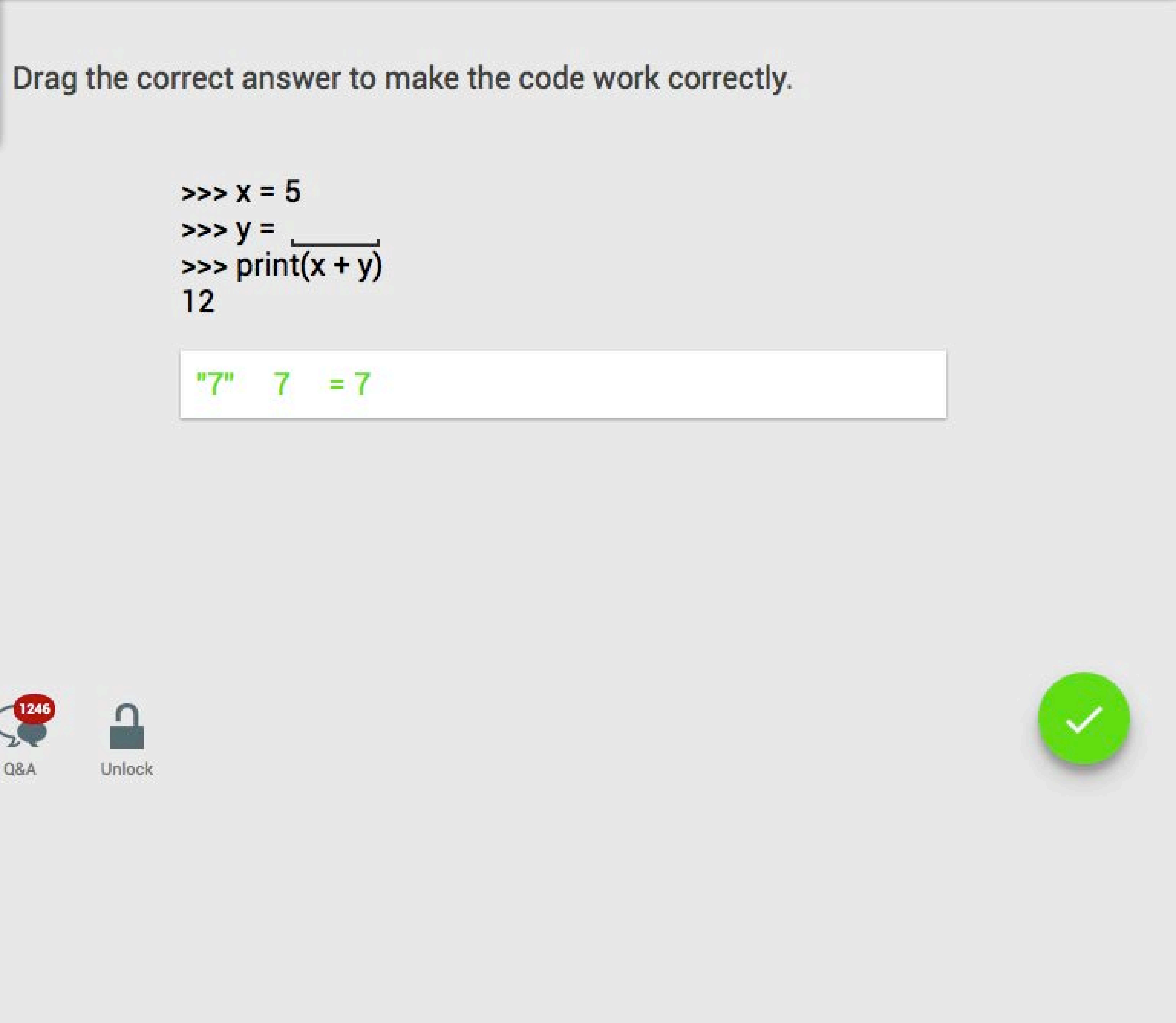
Variables can be reassigned as many times as you want, in order to change their value.
In Python, variables don't have specific types, so you can assign a string to a variable, and later assign an integer to the same variable.

```
>>> x = 123.456
>>> print(x)
123.456
>>> x = "This is a string"
>>> print(x + "!")
This is a string!
```

Try It Yourself

120 COMMENTS

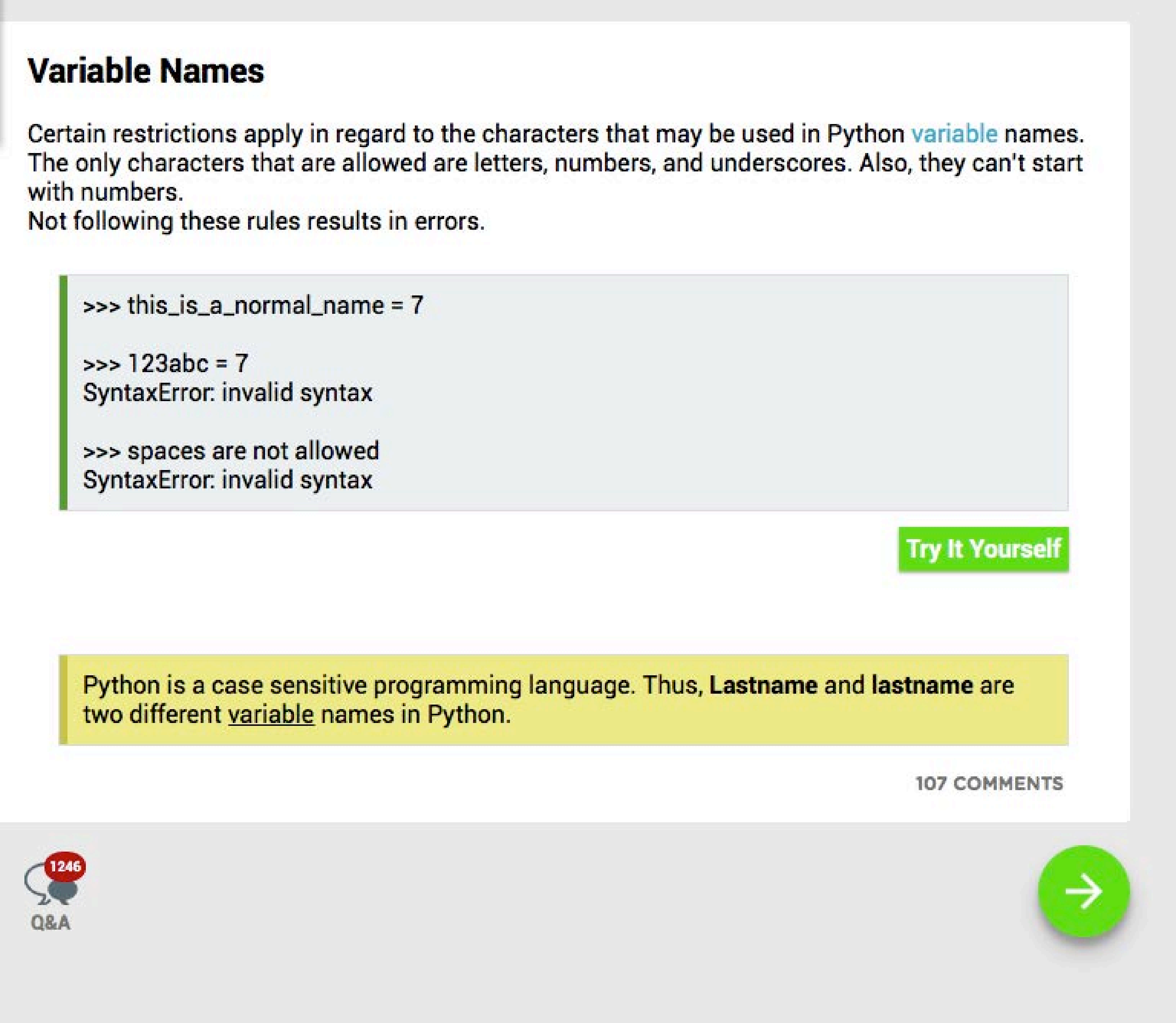
1246
Q&A



Drag the correct answer to make the code work correctly.

```
>>> x = 5
>>> y = ______
>>> print(x + y)
12
```

"7"   7   = 7



Variable Names

Certain restrictions apply in regard to the characters that may be used in Python variable names. The only characters that are allowed are letters, numbers, and underscores. Also, they can't start with numbers.
Not following these rules results in errors.

```
>>> this_is_a_normal_name = 7

>>> 123abc = 7
SyntaxError: invalid syntax

>>> spaces are not allowed
SyntaxError: invalid syntax
```

Python is a case sensitive programming language. Thus, **Lastname** and **lastname** are two different variable names in Python.

Which of these is a valid variable name in Python?

- ○ a variable name
- ○ A_VARIABLE_NAME
- ○ a-variable-name

1246 Q&A

Unlock

Variables

Trying to reference a variable you haven't assigned to causes an **error**.
You can use the **del** statement to remove a variable, which means the reference from the name to the value is deleted, and trying to use the variable causes an error. Deleted variables can be reassigned to later as normal.

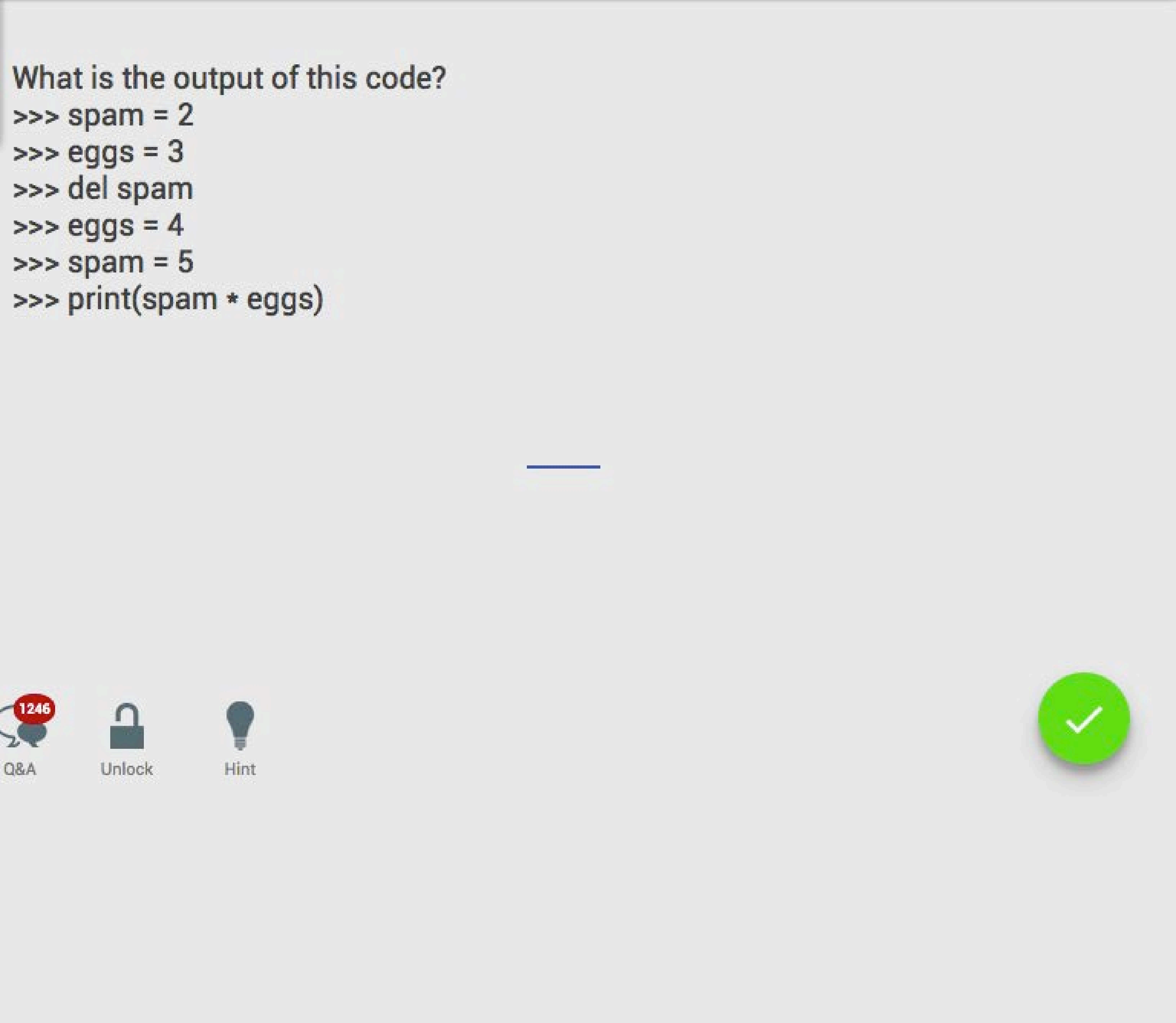
```
>>> foo = "a string"
>>> foo
'a string'
>>> bar
NameError: name 'bar' is not defined
>>> del foo
>>> foo
NameError: name 'foo' is not defined
```

You can also take the value of the variable from the user input.

```
>>> foo = input("Enter a number: ")
Enter a number: 7
>>> print(foo)
7
```

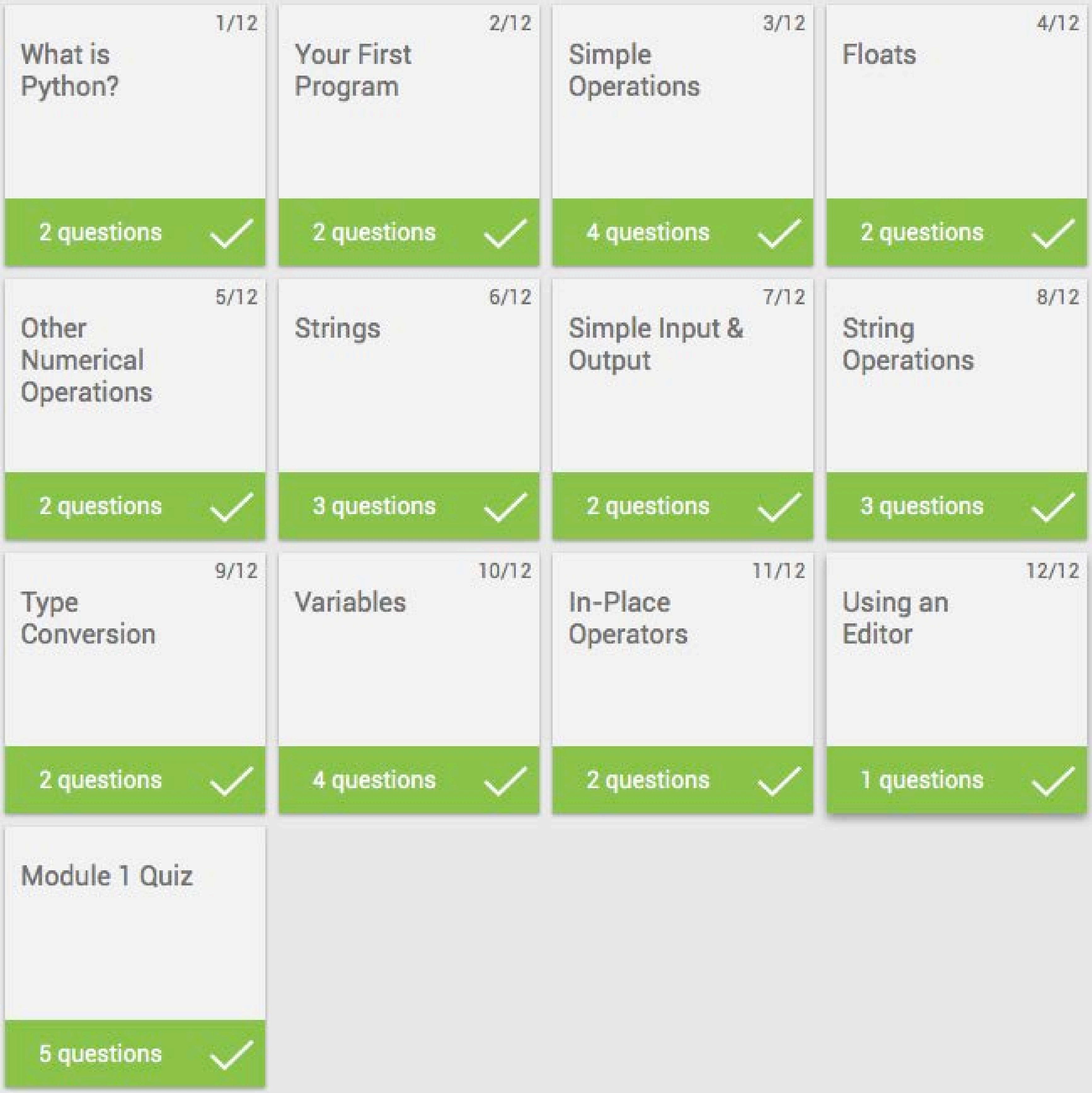
Try It Yourself

The variables **foo** and **bar** are called **metasyntactic** variables, meaning that they are used as placeholder names in example code to demonstrate something.



What is the output of this code?

```
>>> spam = 2
>>> eggs = 3
>>> del spam
>>> eggs = 4
>>> spam = 5
>>> print(spam * eggs)
```



1/12
What is Python?
2 questions ✓

2/12
Your First Program
2 questions ✓

3/12
Simple Operations
4 questions ✓

4/12
Floats
2 questions ✓

5/12
Other Numerical Operations
2 questions ✓

6/12
Strings
3 questions ✓

7/12
Simple Input & Output
2 questions ✓

8/12
String Operations
3 questions ✓

9/12
Type Conversion
2 questions ✓

10/12
Variables
4 questions ✓

11/12
In-Place Operators
2 questions ✓

12/12
Using an Editor
1 questions ✓

Module 1 Quiz
5 questions ✓

In-Place Operators

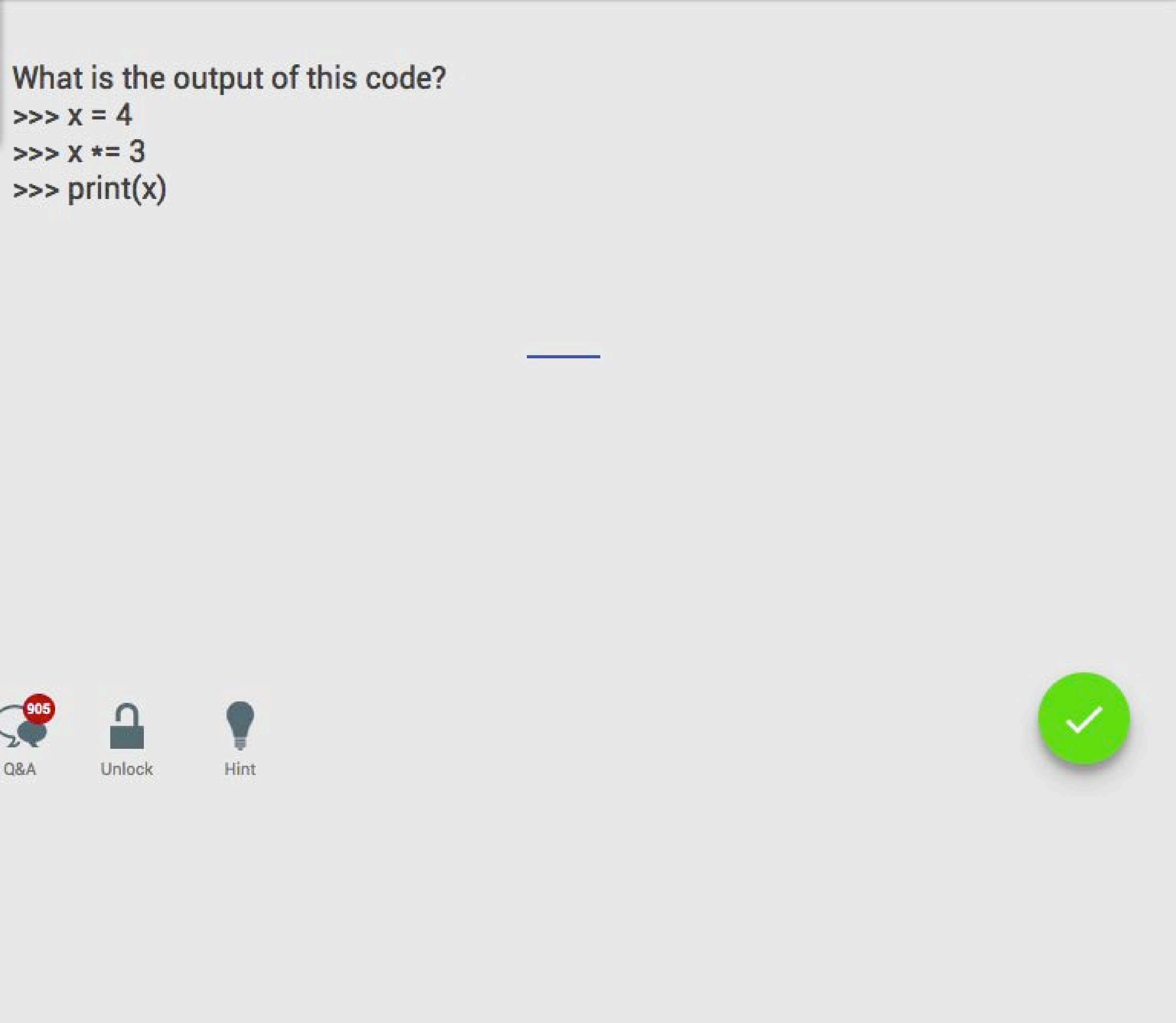
In-place operators allow you to write code like 'x = x + 3' more concisely, as 'x += 3'.
The same thing is possible with other operators such as **-**, *****, **/** and **%** as well.

```
>>> x = 2
>>> print(x)
2
>>> x += 3
>>> print(x)
5
```

Try It Yourself

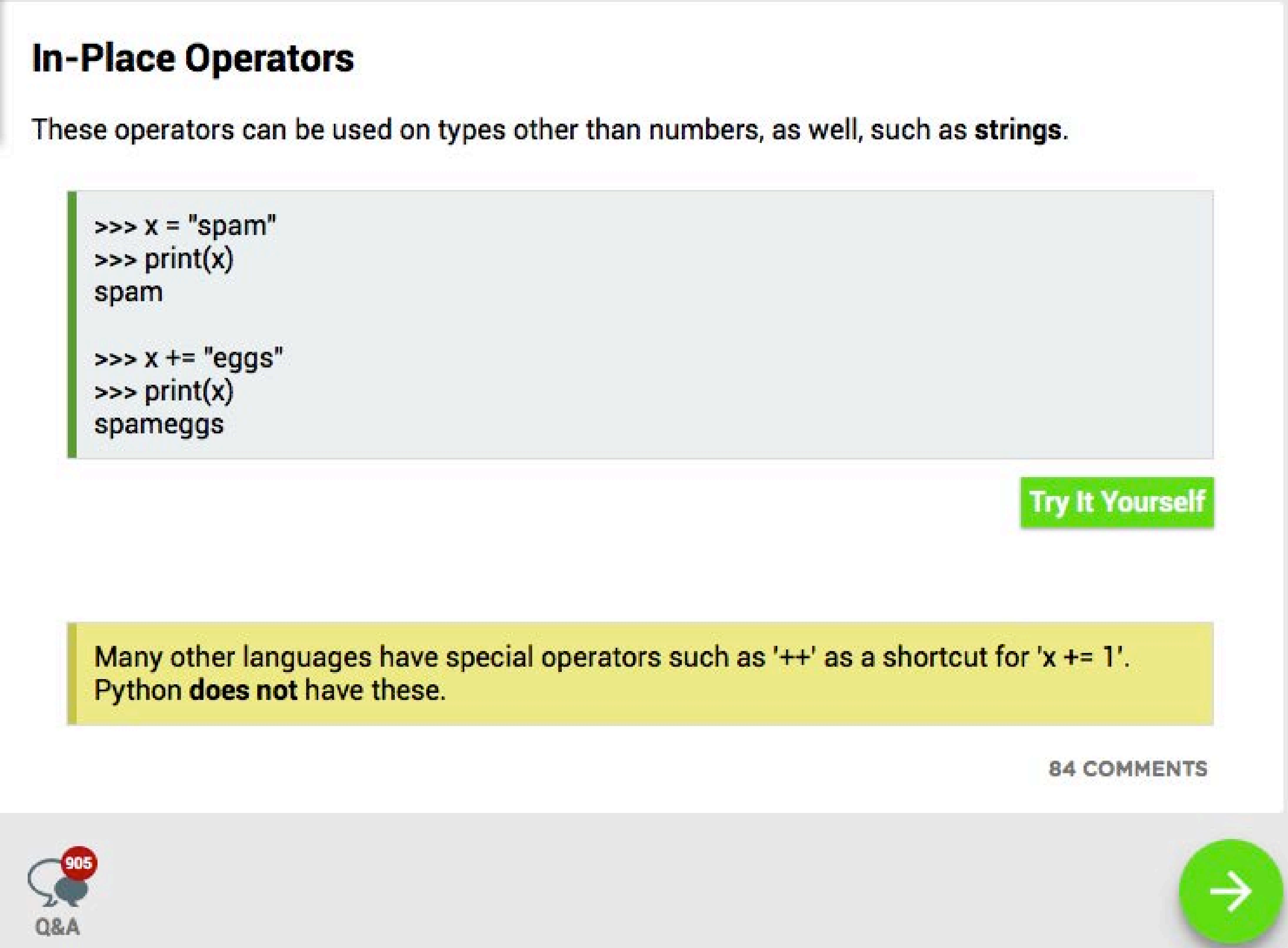
222 COMMENTS

905
Q&A



What is the output of this code?

```
>>> x = 4
>>> x *= 3
>>> print(x)
```

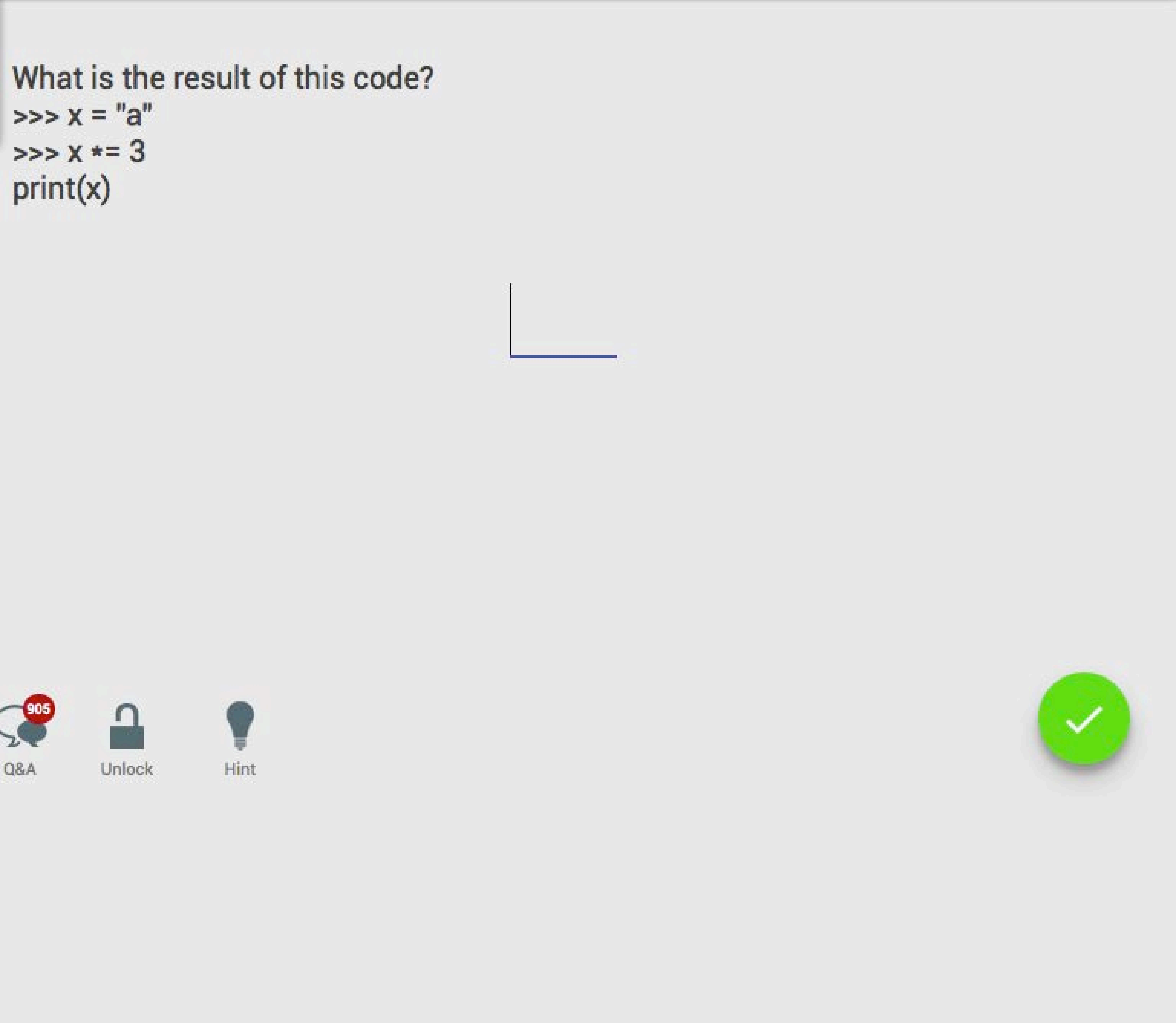


In-Place Operators

These operators can be used on types other than numbers, as well, such as **strings**.

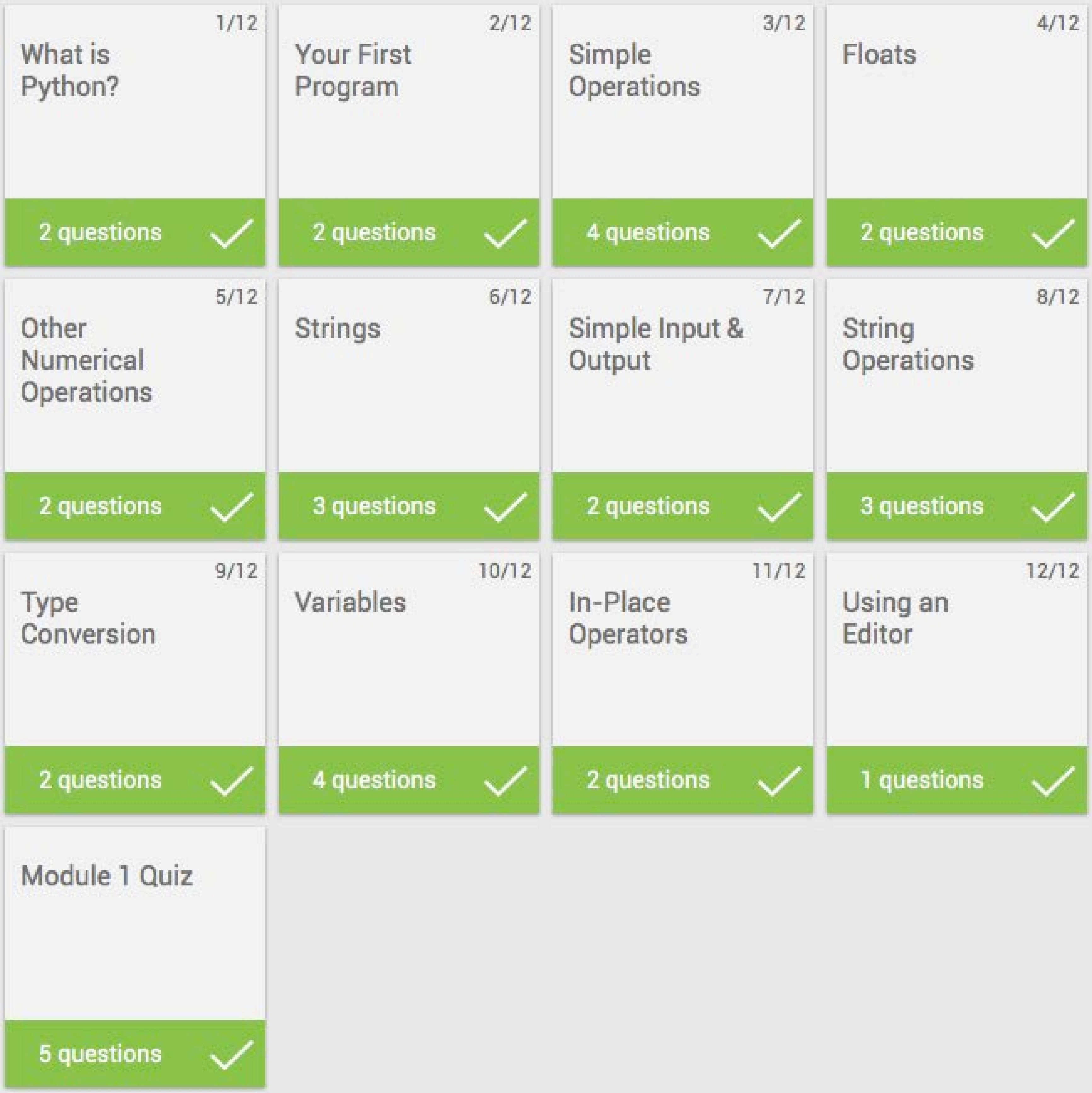
```
>>> x = "spam"
>>> print(x)
spam

>>> x += "eggs"
>>> print(x)
spameggs
```

Many other languages have special operators such as '++' as a shortcut for 'x += 1'. Python **does not** have these.



What is the result of this code?

```
>>> x = "a"
>>> x *= 3
print(x)
```



1/12
What is Python?
2 questions ✓

2/12
Your First Program
2 questions ✓

3/12
Simple Operations
4 questions ✓

4/12
Floats
2 questions ✓

5/12
Other Numerical Operations
2 questions ✓

6/12
Strings
3 questions ✓

7/12
Simple Input & Output
2 questions ✓

8/12
String Operations
3 questions ✓

9/12
Type Conversion
2 questions ✓

10/12
Variables
4 questions ✓

11/12
In-Place Operators
2 questions ✓

12/12
Using an Editor
1 questions ✓

Module 1 Quiz
5 questions ✓

Using an Editor

So far, we've only used Python with the console, entering and running one line of code at a time. Actual programs are created differently; many lines of code are written in a file, and then executed with the Python interpreter.

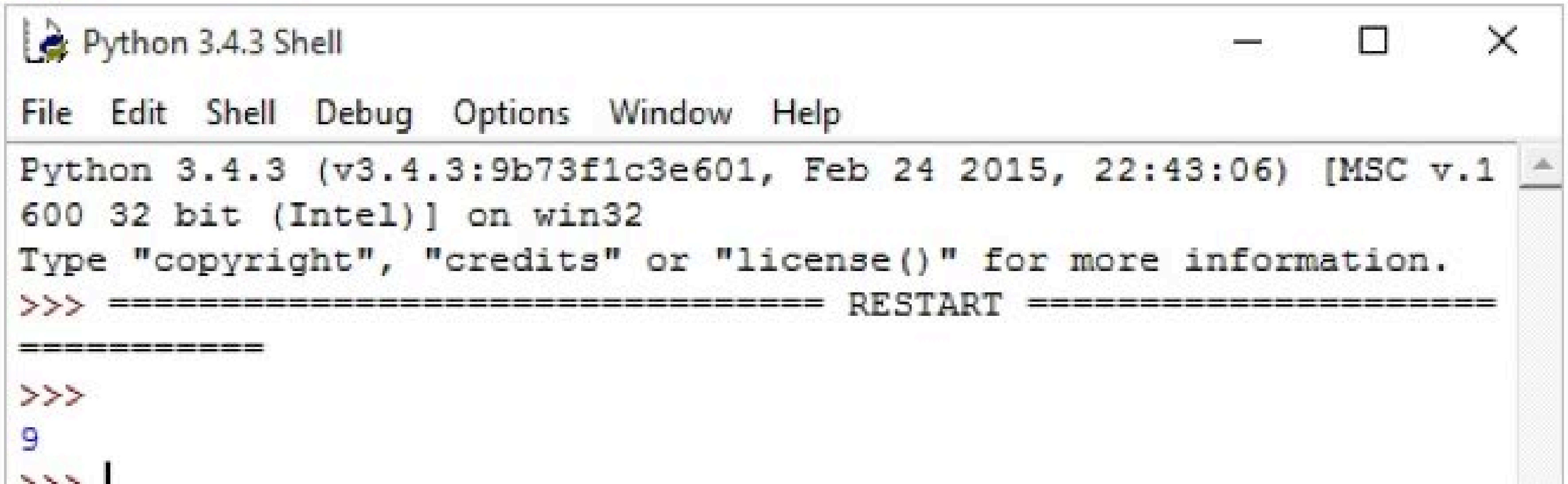
In IDLE, this can be done by creating a new file, entering some code, saving the file, and running it. This can be done either with the menus or with the keyboard shortcuts Ctrl-N, Ctrl-S and F5.

Each line of code in the file is interpreted as though you entered it one line at a time at the console.

```
x = 7
x = x + 2
print(x)
```

Try It Yourself

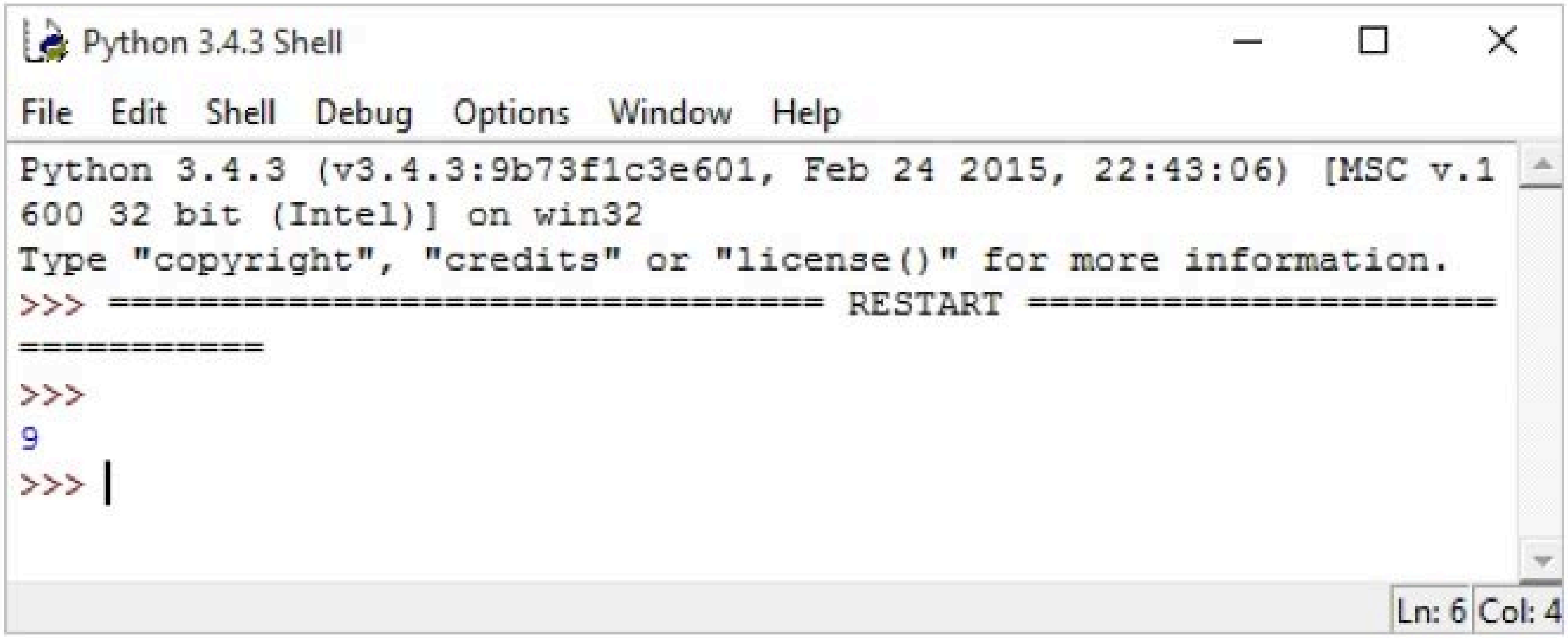
Result:



```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1
600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ======================
===========
>>>
9
>>>
```

Try It Yourself

Result:



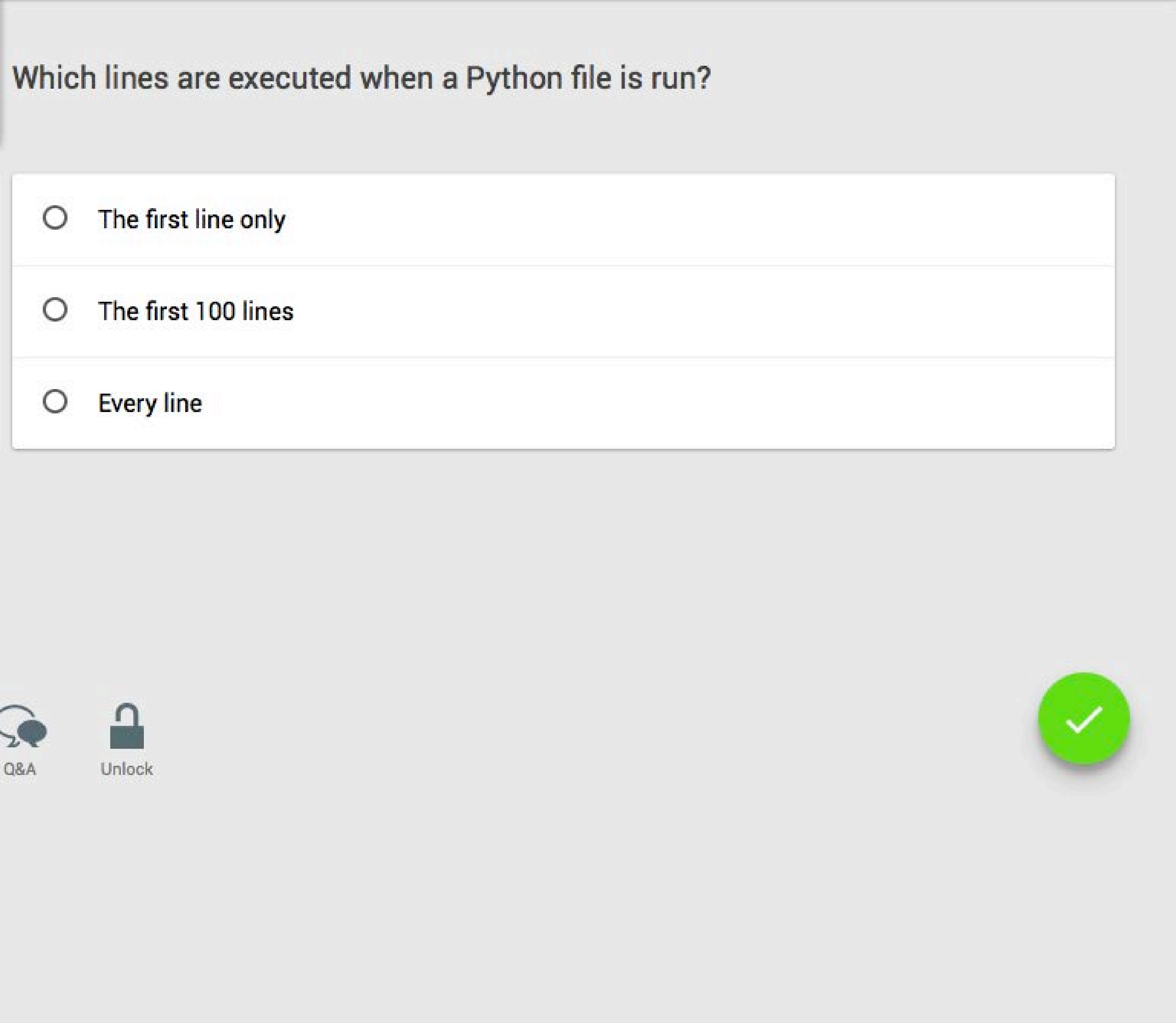
```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1
600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART =====================
===========
>>> 
9
>>> 
```

Python source files have an extension of **.py**

You can run, save, and share your Python codes on our **Code Playground**, without installing any additional software.
Reference this lesson if you need to install the software on your computer.

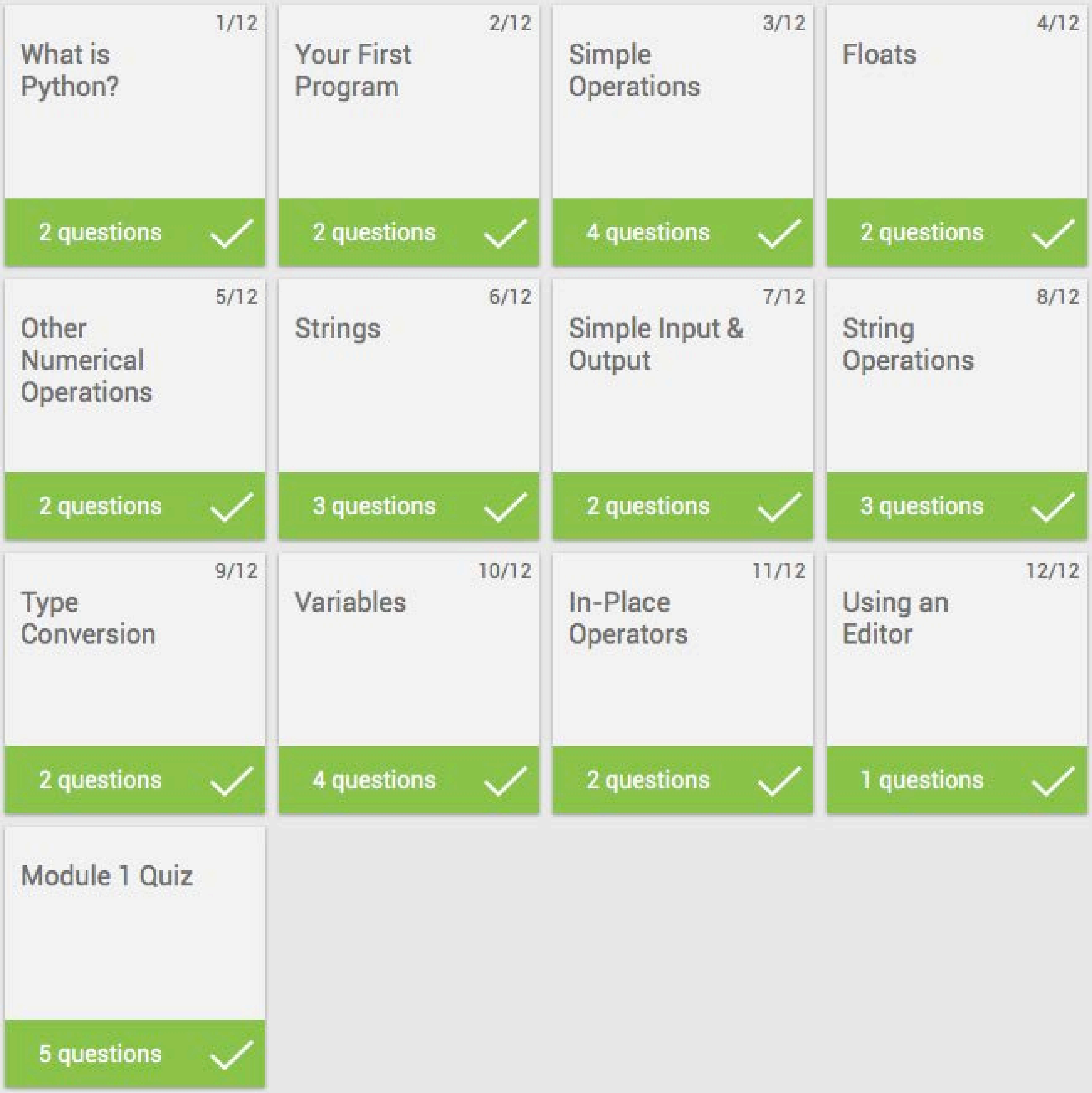
160 COMMENTS

Q&A



Which lines are executed when a Python file is run?

- ○ The first line only
- ○ The first 100 lines
- ○ Every line


1/12
What is Python?
2 questions ✓

2/12
Your First Program
2 questions ✓

3/12
Simple Operations
4 questions ✓

4/12
Floats
2 questions ✓

5/12
Other Numerical Operations
2 questions ✓

6/12
Strings
3 questions ✓

7/12
Simple Input & Output
2 questions ✓

8/12
String Operations
3 questions ✓

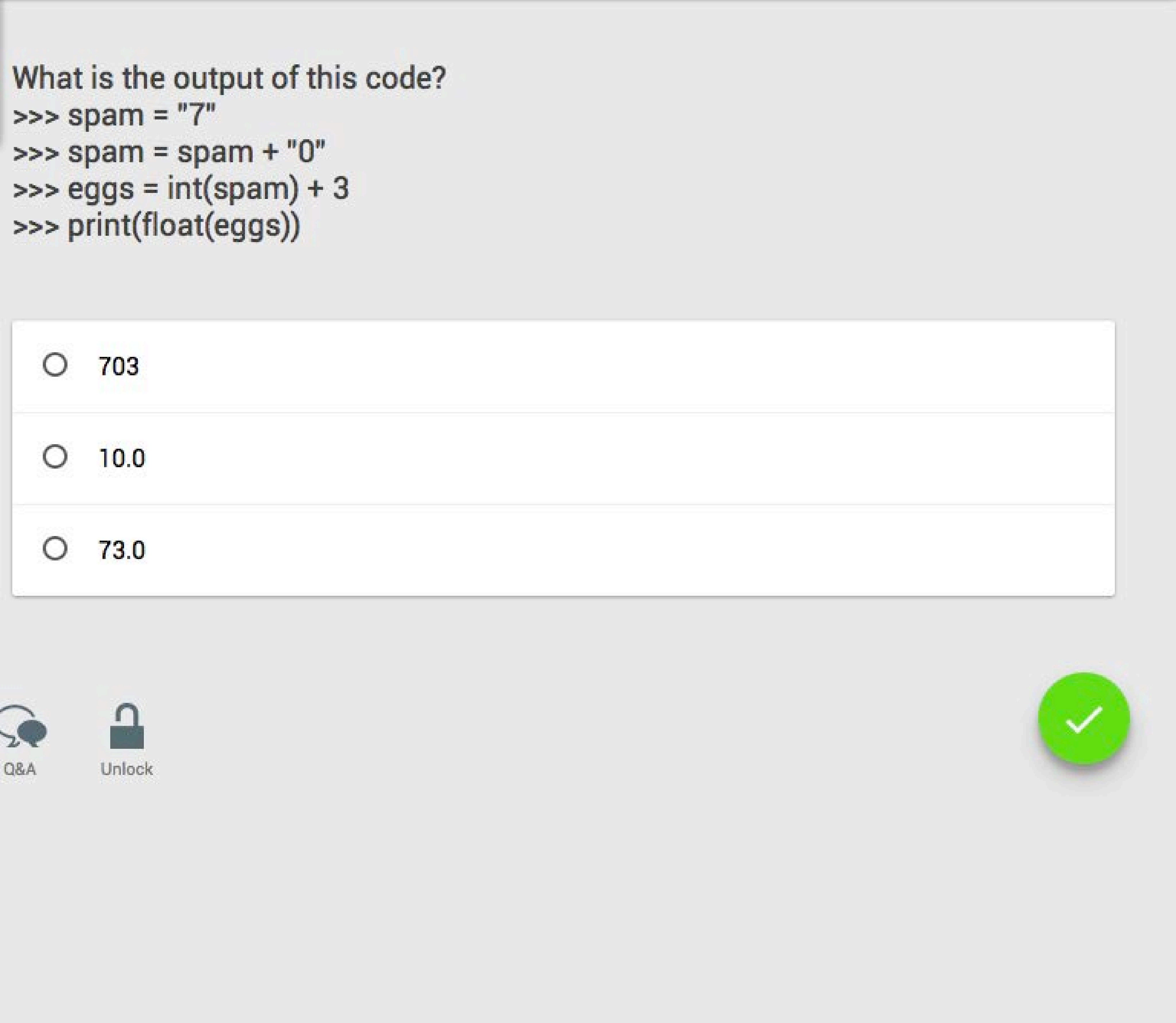
9/12
Type Conversion
2 questions ✓

10/12
Variables
4 questions ✓

11/12
In-Place Operators
2 questions ✓

12/12
Using an Editor
1 questions ✓

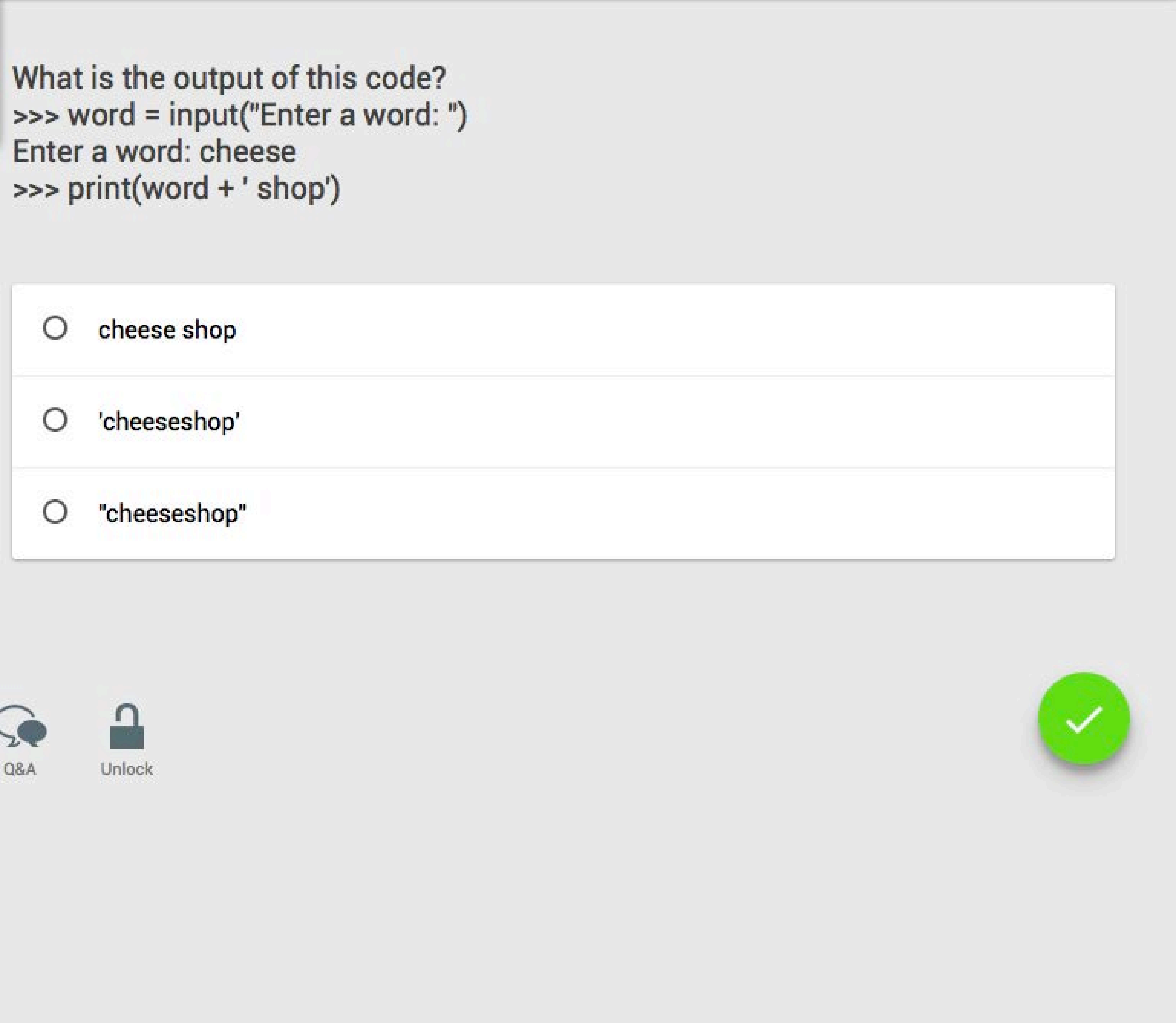
Module 1 Quiz
5 questions ✓



What is the output of this code?

```
>>> spam = "7"
>>> spam = spam + "0"
>>> eggs = int(spam) + 3
>>> print(float(eggs))
```

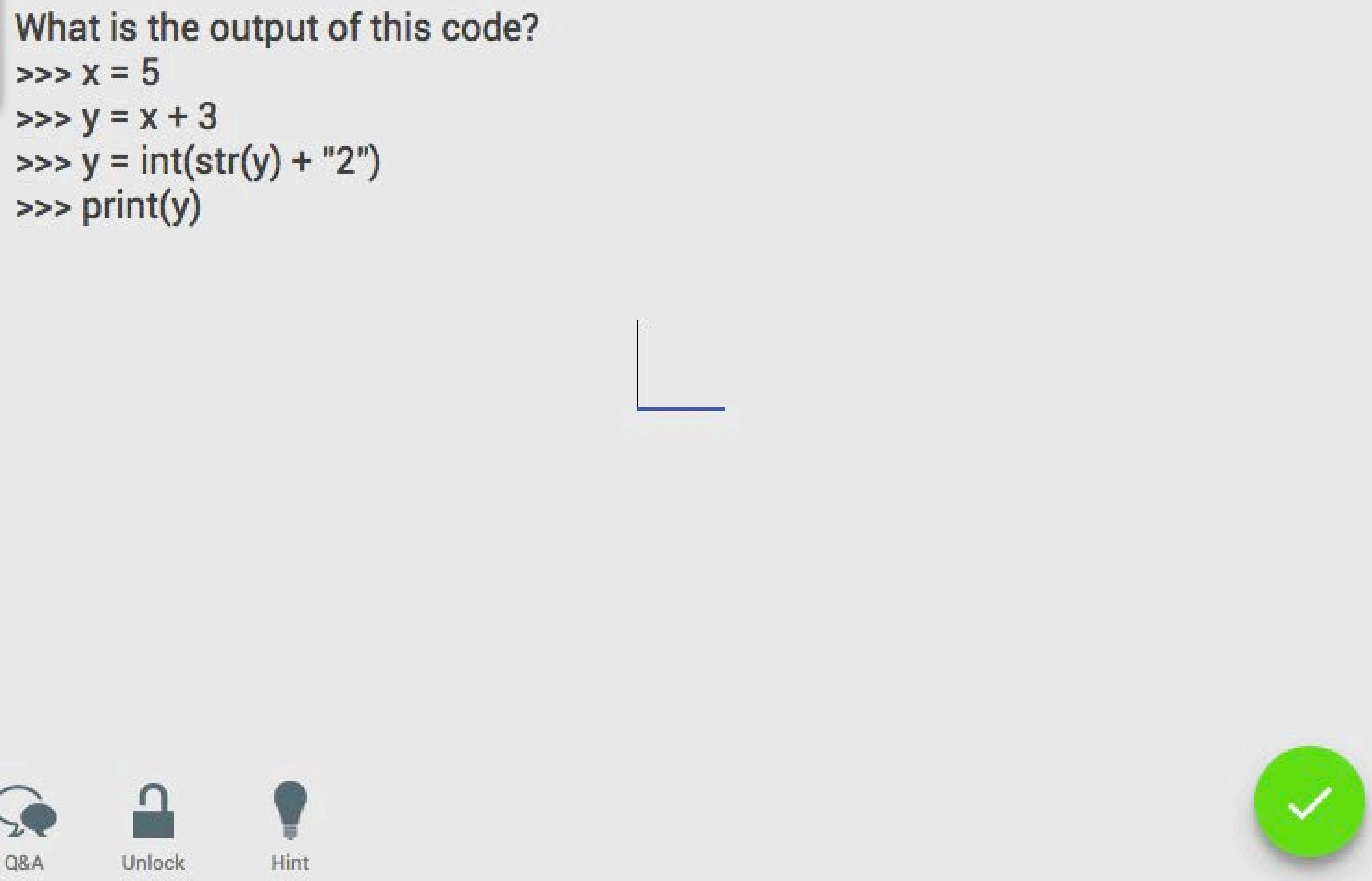
- ○ 703
- ○ 10.0
- ○ 73.0



What is the output of this code?

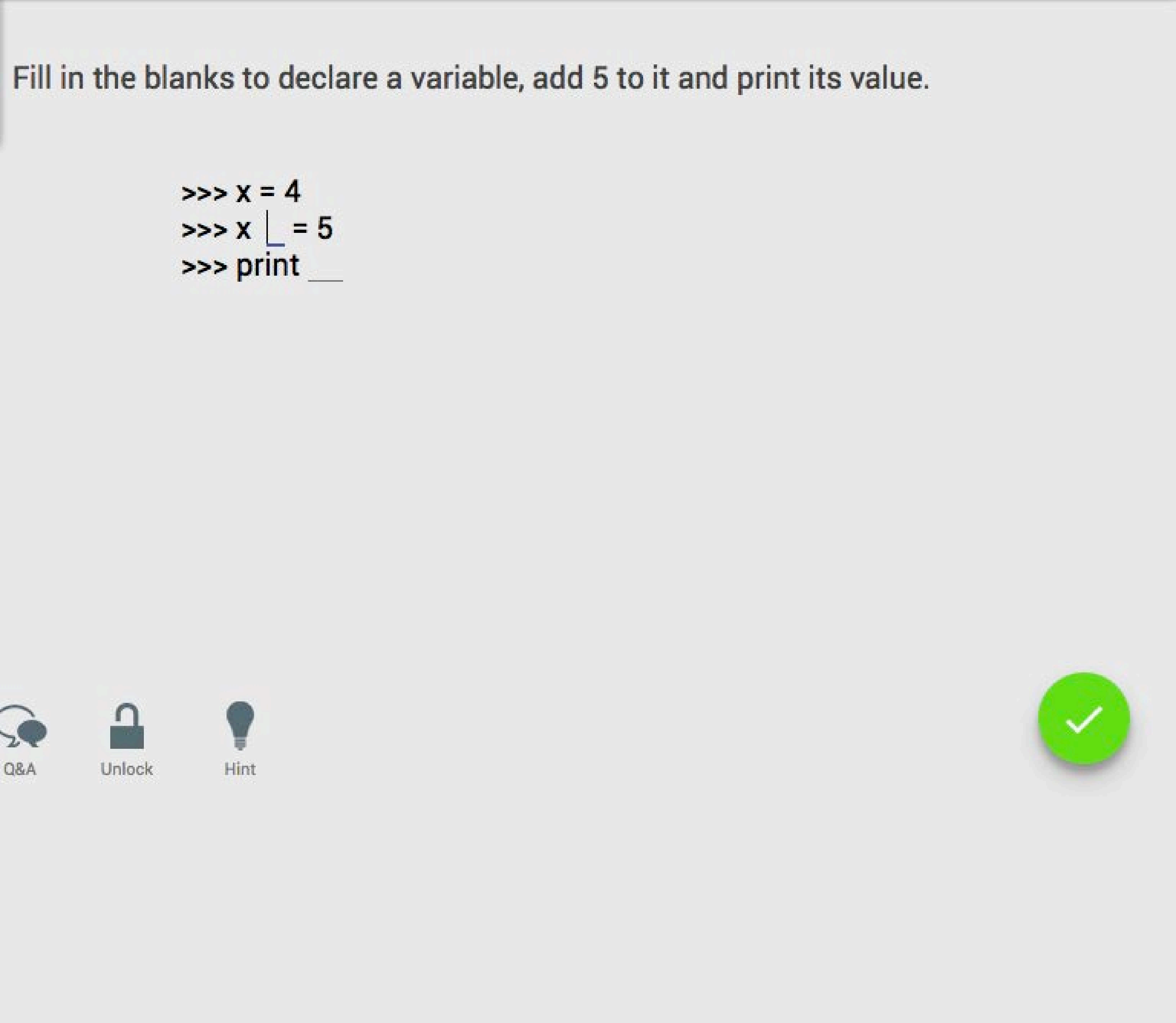
```
>>> word = input("Enter a word: ")
Enter a word: cheese
>>> print(word + ' shop')
```

- ○ cheese shop
- ○ 'cheeseshop'
- ○ "cheeseshop"



What is the output of this code?

```
>>> x = 5
>>> y = x + 3
>>> y = int(str(y) + "2")
>>> print(y)
```

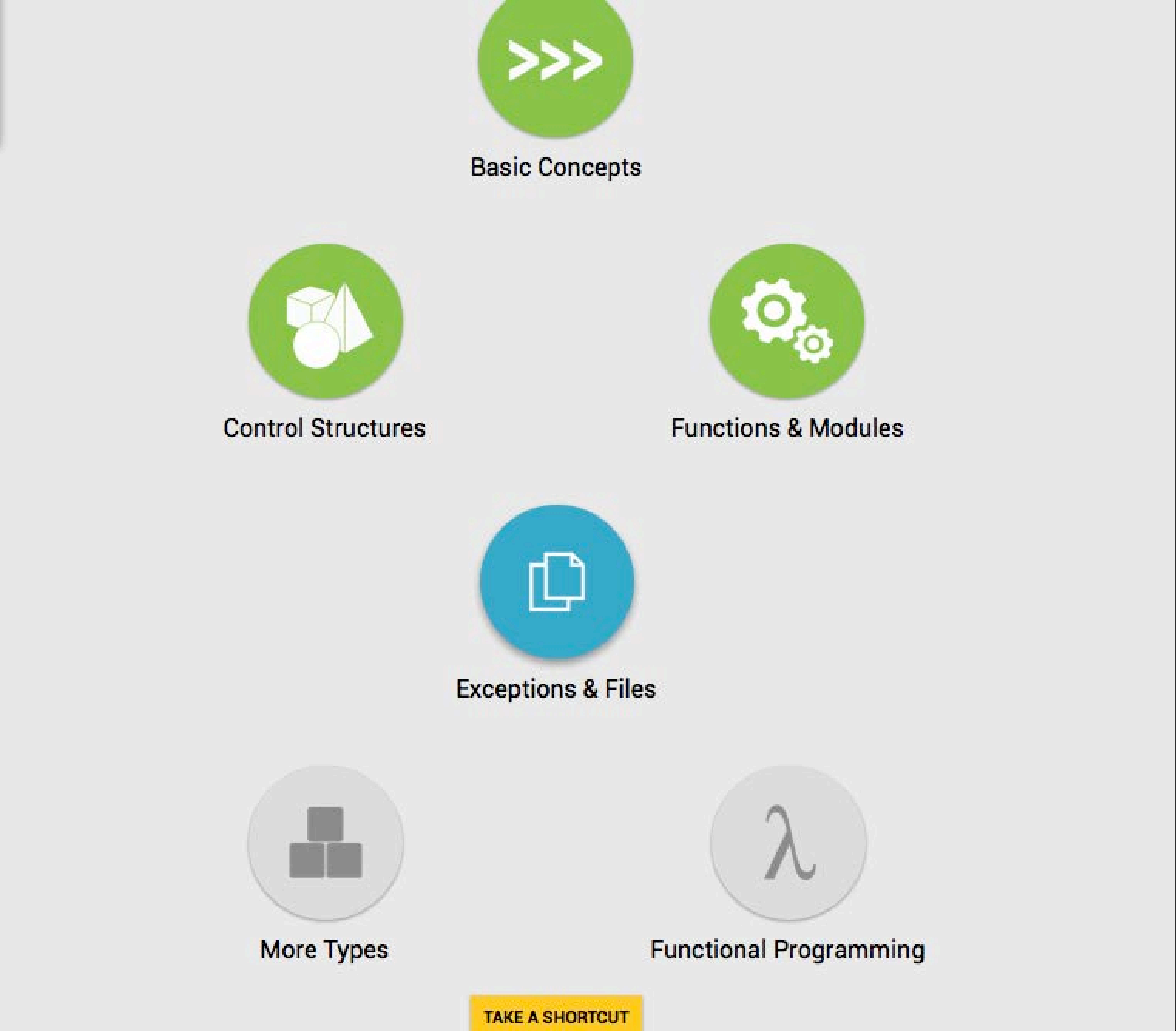

Fill in the blanks to declare a variable, add 5 to it and print its value.

```
>>> x = 4
>>> x _ = 5
>>> print __
```

What is the output of this code?

```
>>> x = 3
>>> num = 17
>>> print(num % x)
```

Q&A Unlock Hint


Basic Concepts
Control Structures
Functions & Modules
Exceptions & Files
More Types
Functional Programming
TAKE A SHORTCUT