SWCON104
Web & Python Programming

Methods (in Class)

Department of Software Convergence

경희대학교
KYUNG HEE UNIVERSITY

Today

- Methods

[Textbook]
Practical Programming
(An Introduction to Computer Science Using Python),
by Paul Gries, Jennifer Campbell, Jason Montojo,
The Pragmatic Bookshelf, 2017

Practice

- Practice_09_Methods.ipynb

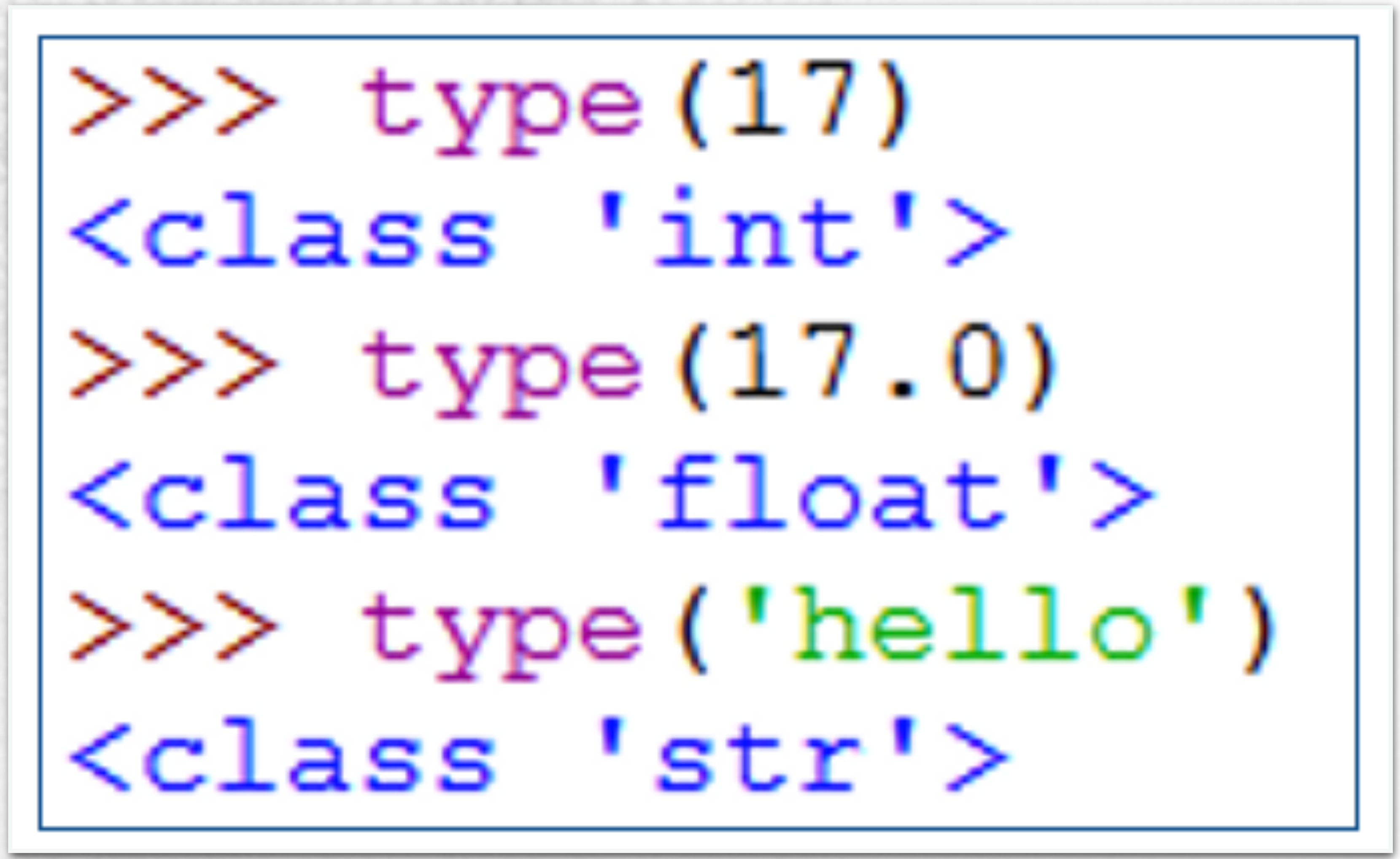
Methods

- Functions
  - Built-in functions
  - Functions inside modules
  - Functions that we've defined

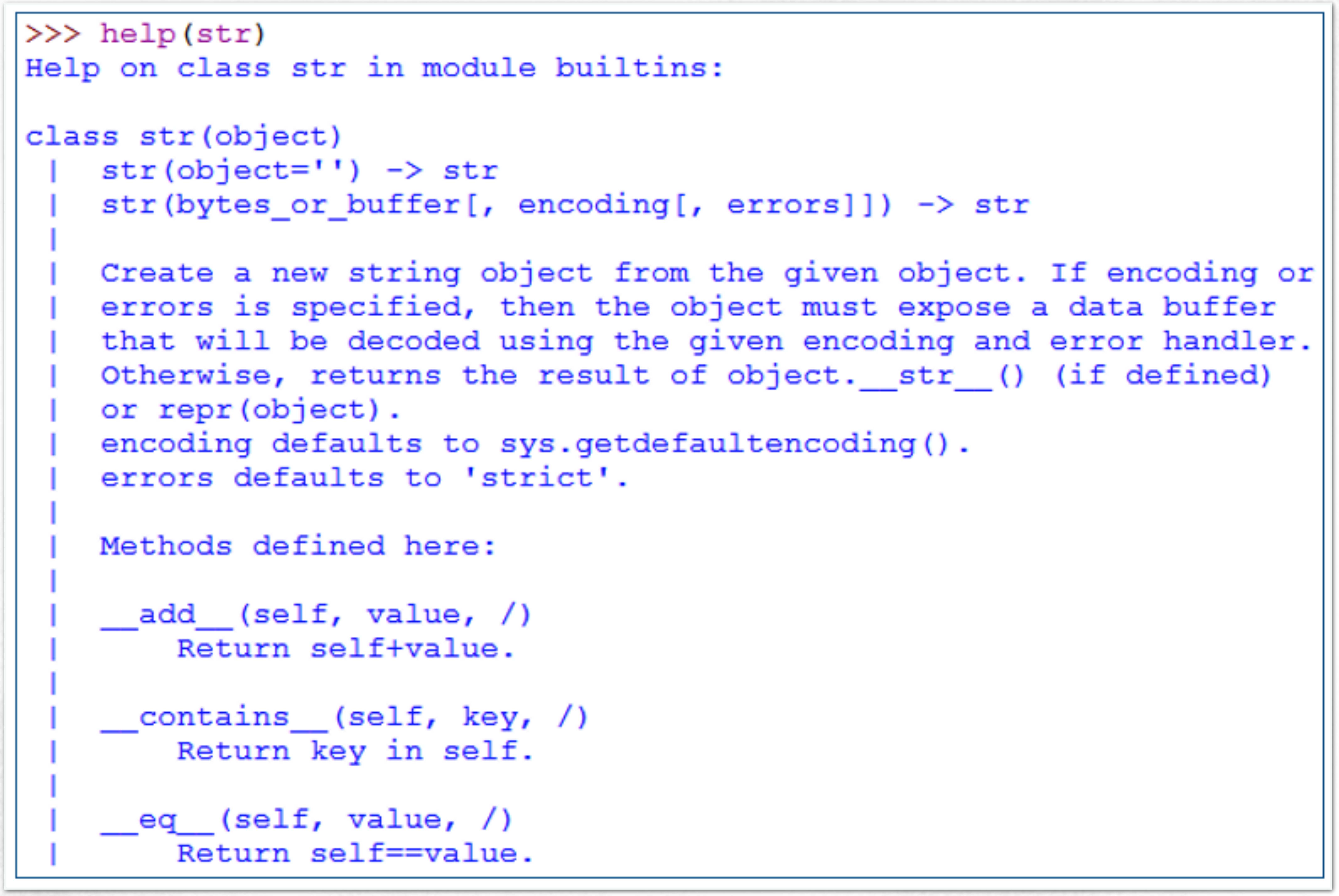
- A method is a kind of function that is attached to a particular type
  - `str` methods
  - `int` methods
  - `bool` methods
  - Every type has its own set of methods

Classes

- A class is another kind of object that is similar to a module.
- A class is how Python represents a type.
- A class has methods



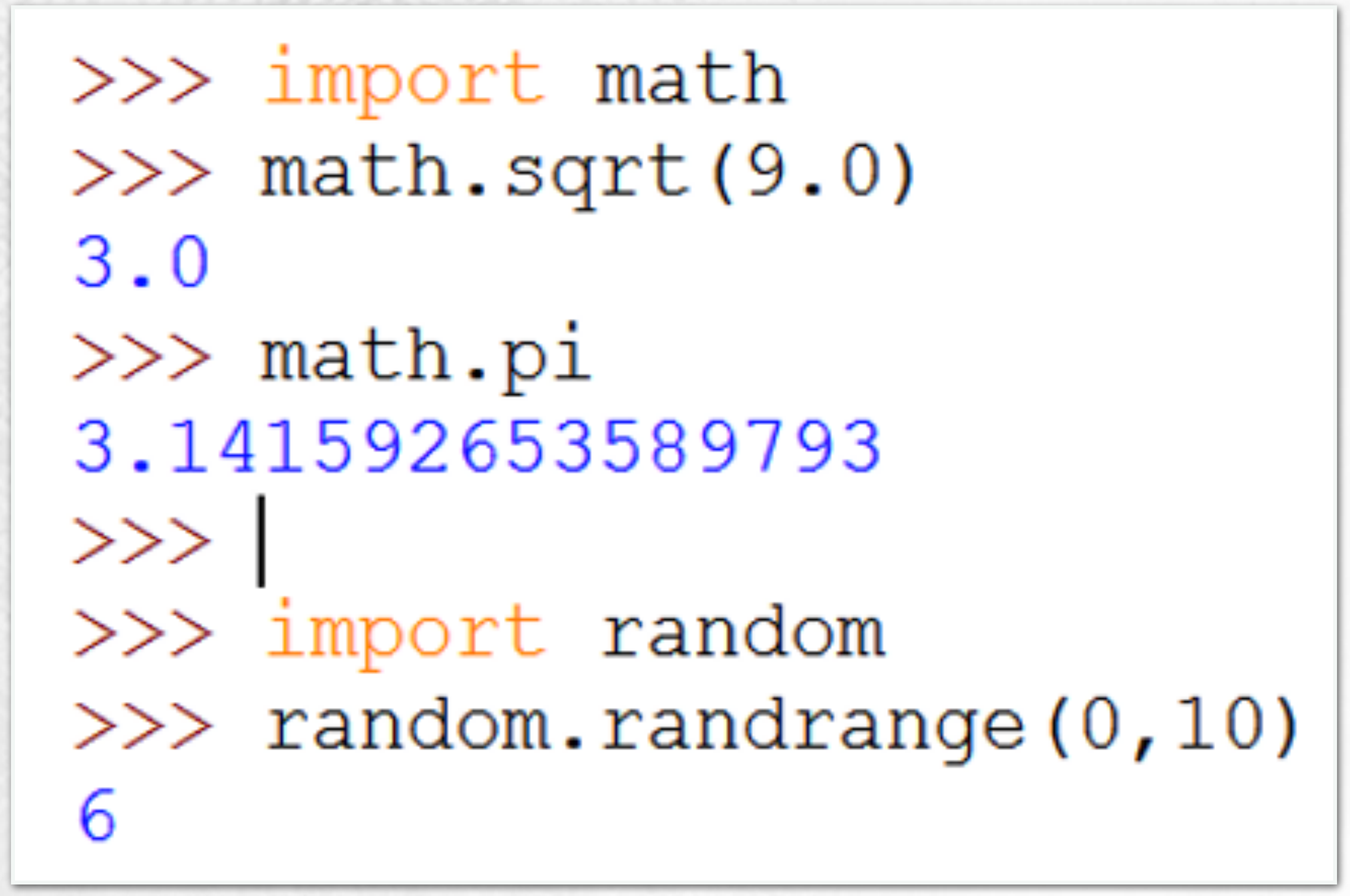
```
>>> type(17)
<class 'int'>
>>> type(17.0)
<class 'float'>
>>> type('hello')
<class 'str'>
```



```
>>> help(str)
Help on class str in module builtins:

class str(object)
 |  str(object='') -> str
 |  str(bytes_or_buffer[, encoding[, errors]]) -> str
 |
 |  Create a new string object from the given object. If encoding or
 |  errors is specified, then the object must expose a data buffer
 |  that will be decoded using the given encoding and error handler.
 |  Otherwise, returns the result of object.__str__() (if defined)
 |  or repr(object).
 |  encoding defaults to sys.getdefaultencoding().
 |  errors defaults to 'strict'.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return key in self.
 |
 |  __eq__(self, value, /)
 |      Return self==value.
```

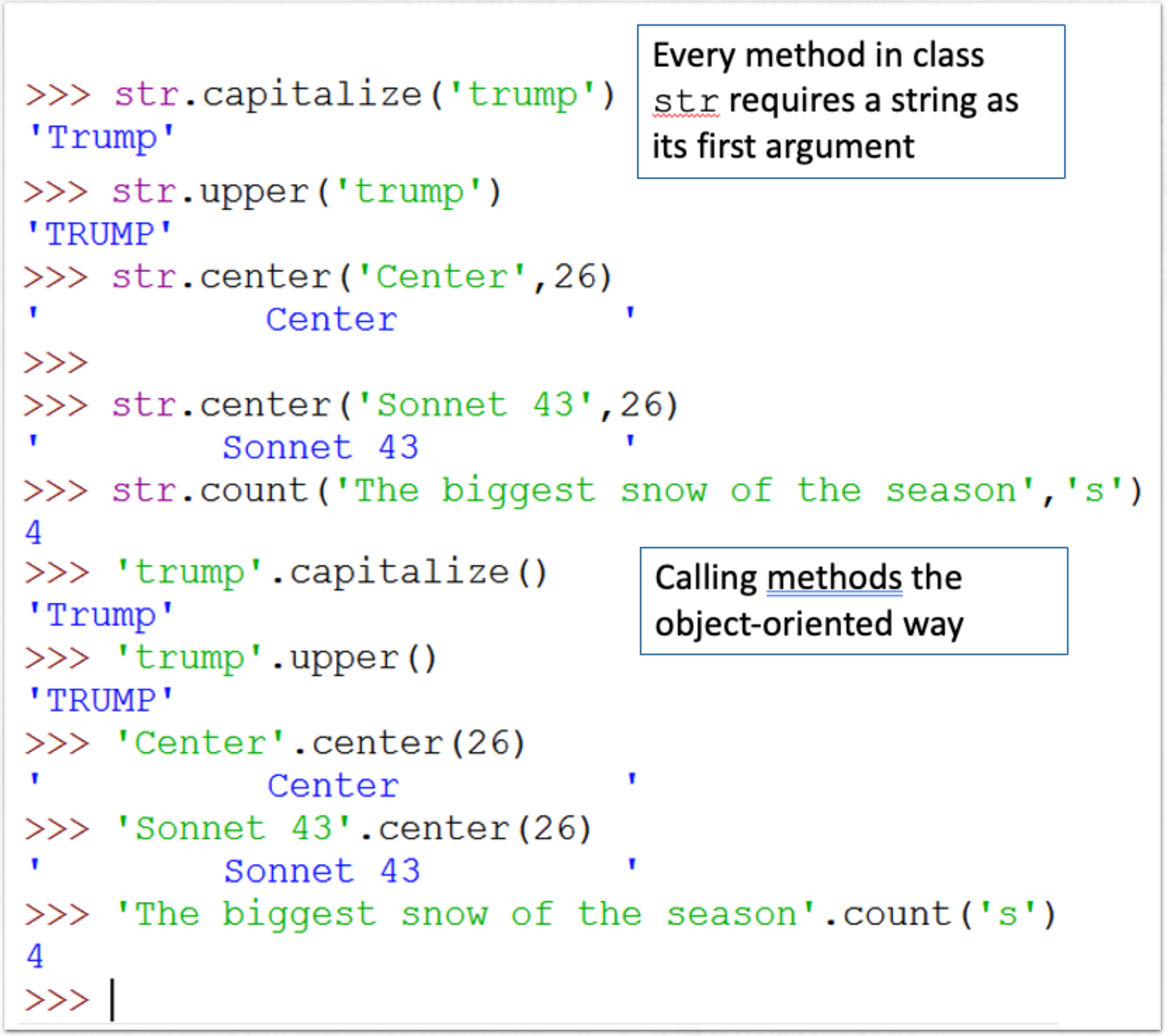
Module vs Class

- Functions in Module math
- Functions in Module random



```
>>> import math
>>> math.sqrt(9.0)
3.0
>>> math.pi
3.141592653589793
>>> |
>>> import random
>>> random.randrange(0,10)
6
```

- Methods in Class string

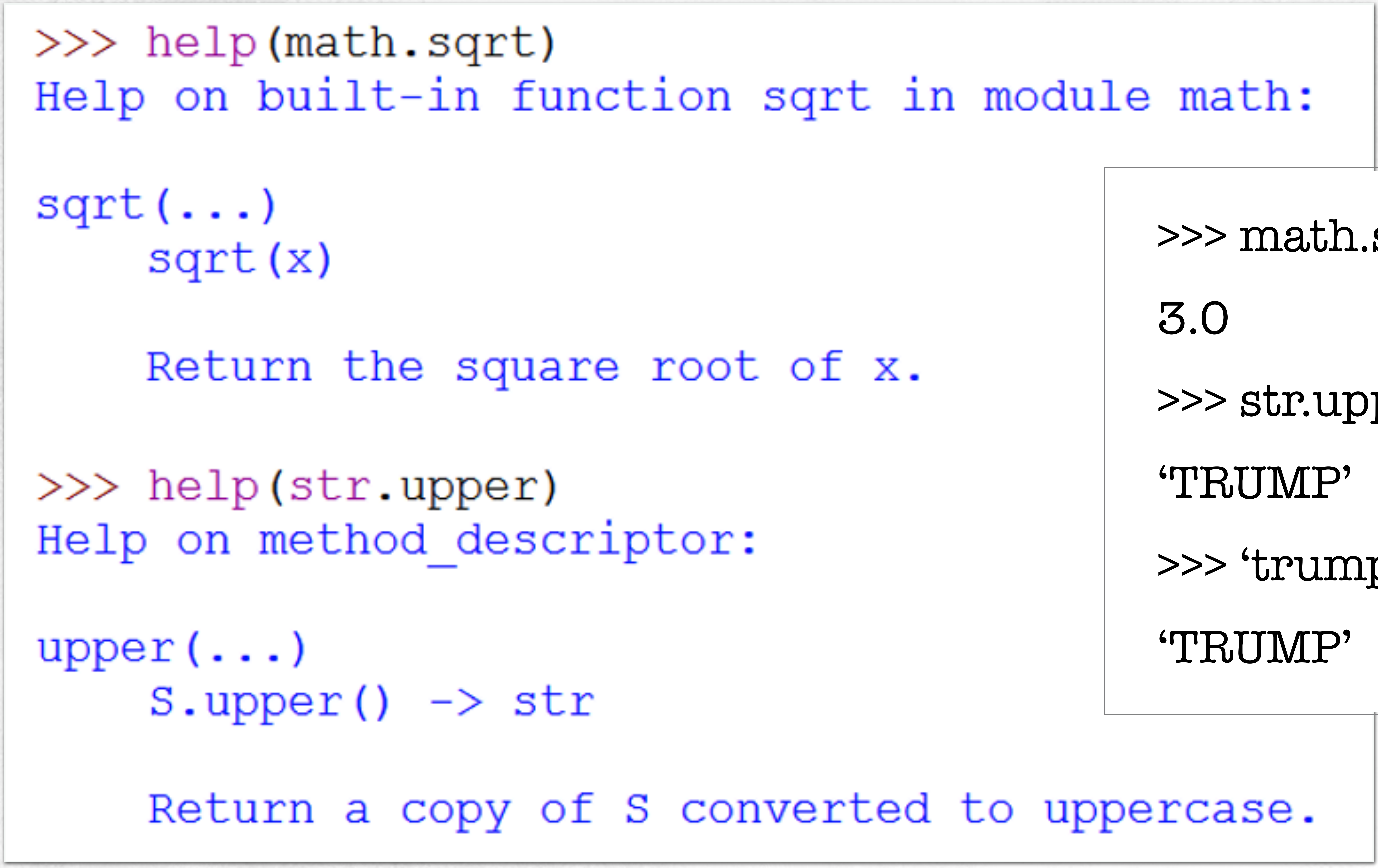


```
>>> str.capitalize('trump')
'Trump'
>>> str.upper('trump')
'TRUMP'
>>> str.center('Center',26)
'          Center          '
>>>
>>> str.center('Sonnet 43',26)
'        Sonnet 43         '
>>> str.count('The biggest snow of the season','s')
4
>>> 'trump'.capitalize()
'Trump'
>>> 'trump'.upper()
'TRUMP'
>>> 'Center'.center(26)
'          Center          '
>>> 'Sonnet 43'.center(26)
'        Sonnet 43         '
>>> 'The biggest snow of the season'.count('s')
4
>>> |
```

Every method in class `str` requires a string as its first argument

Calling methods the object-oriented way

Module vs Class



```
>>> help(math.sqrt)
Help on built-in function sqrt in module math:

sqrt(...)
    sqrt(x)

    Return the square root of x.

>>> help(str.upper)
Help on method_descriptor:

upper(...)
    S.upper() -> str

    Return a copy of S converted to uppercase.
```

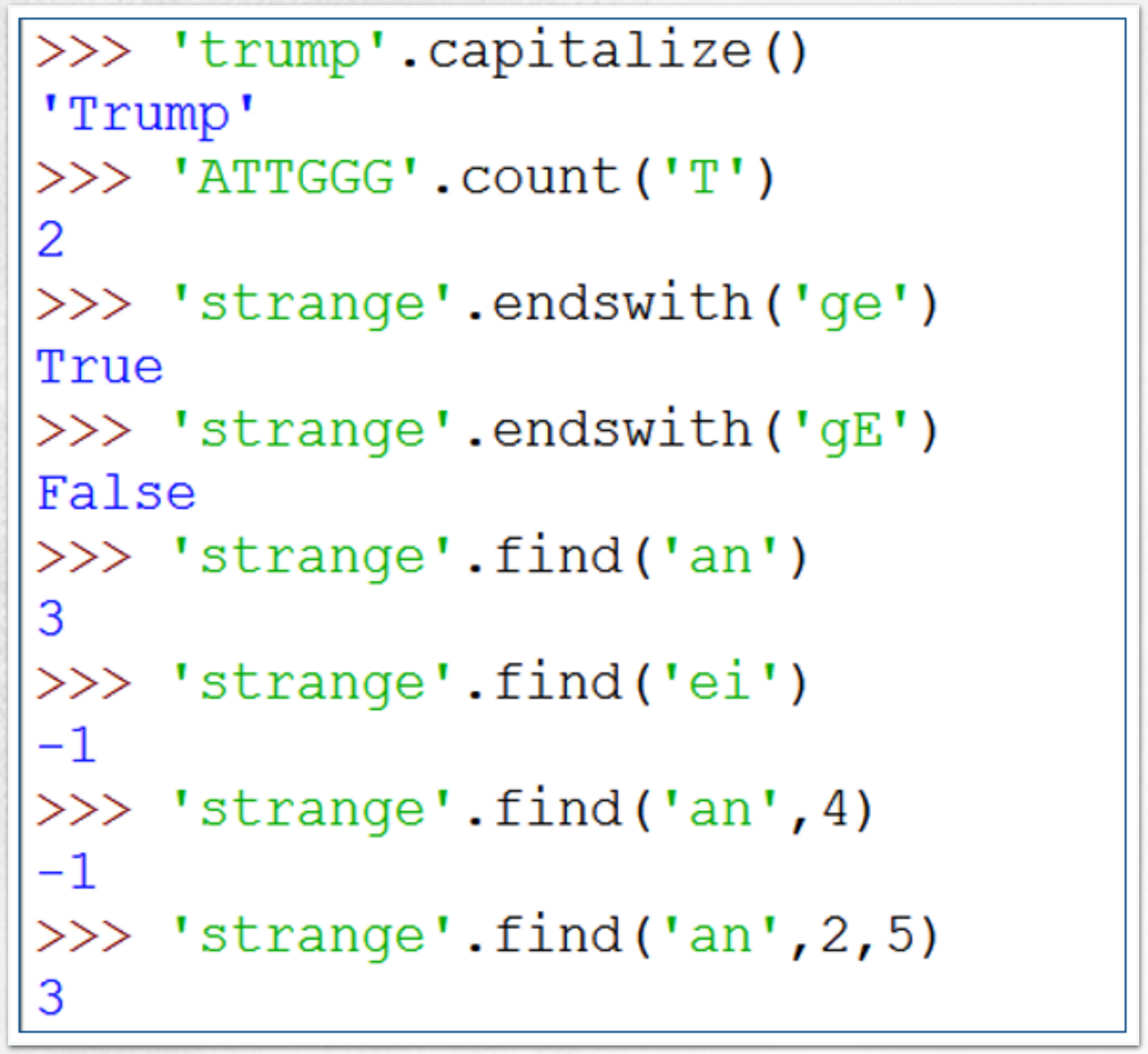
```
>>> math.sqrt(9.0)
3.0
>>> str.upper('trump')
'TRUMP'
>>> 'trump'.upper()
'TRUMP'
```

Methods

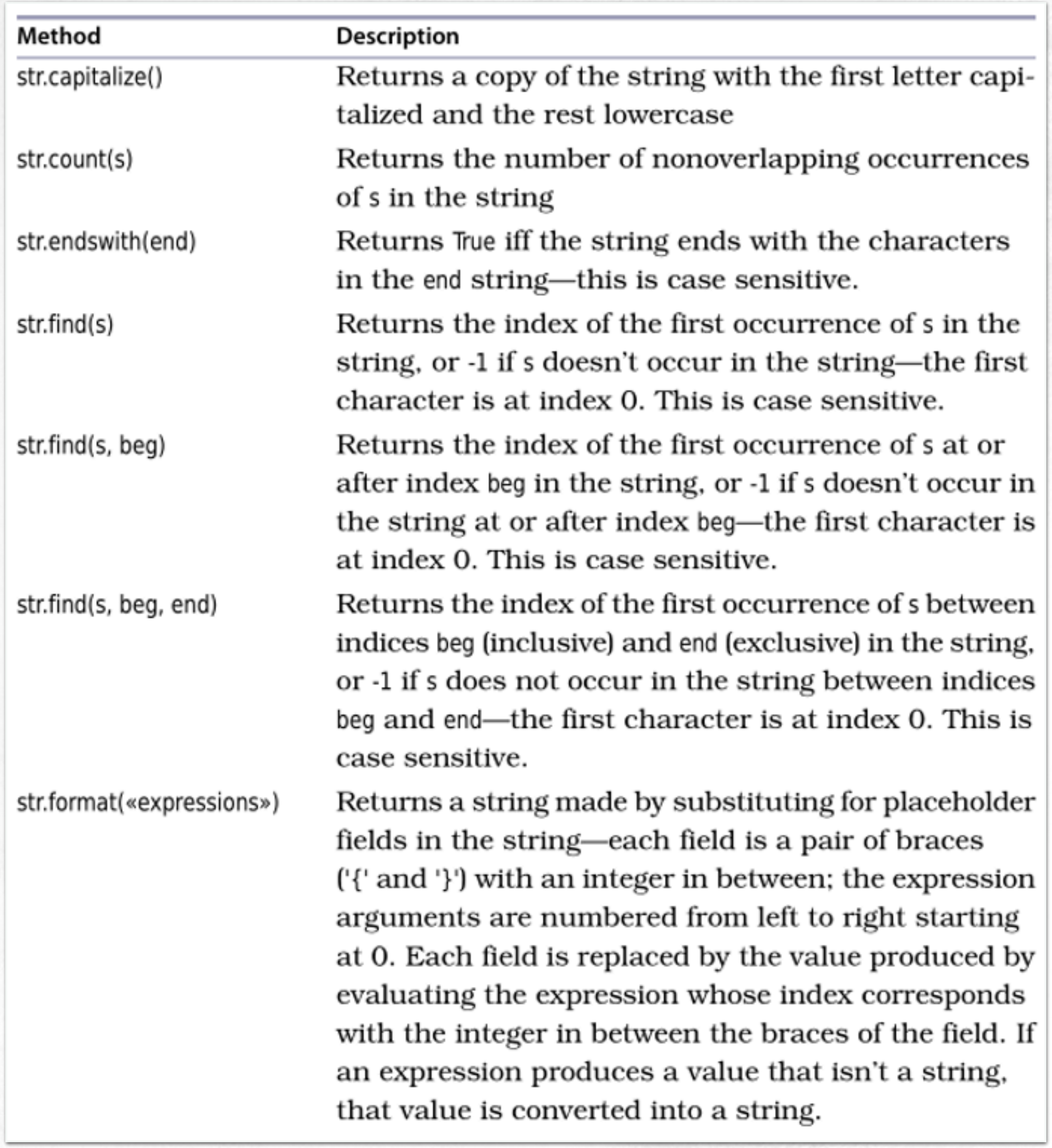
```
<<expression>>.<<method_name>>(<<arguments>>)
```

```
>>> ('TTA' + 'G' * 3).count('T')
2

>>> ('TTAGGG').count('T')
2

>>> str.count('TTAGGG', 'T')
2
```

String methods

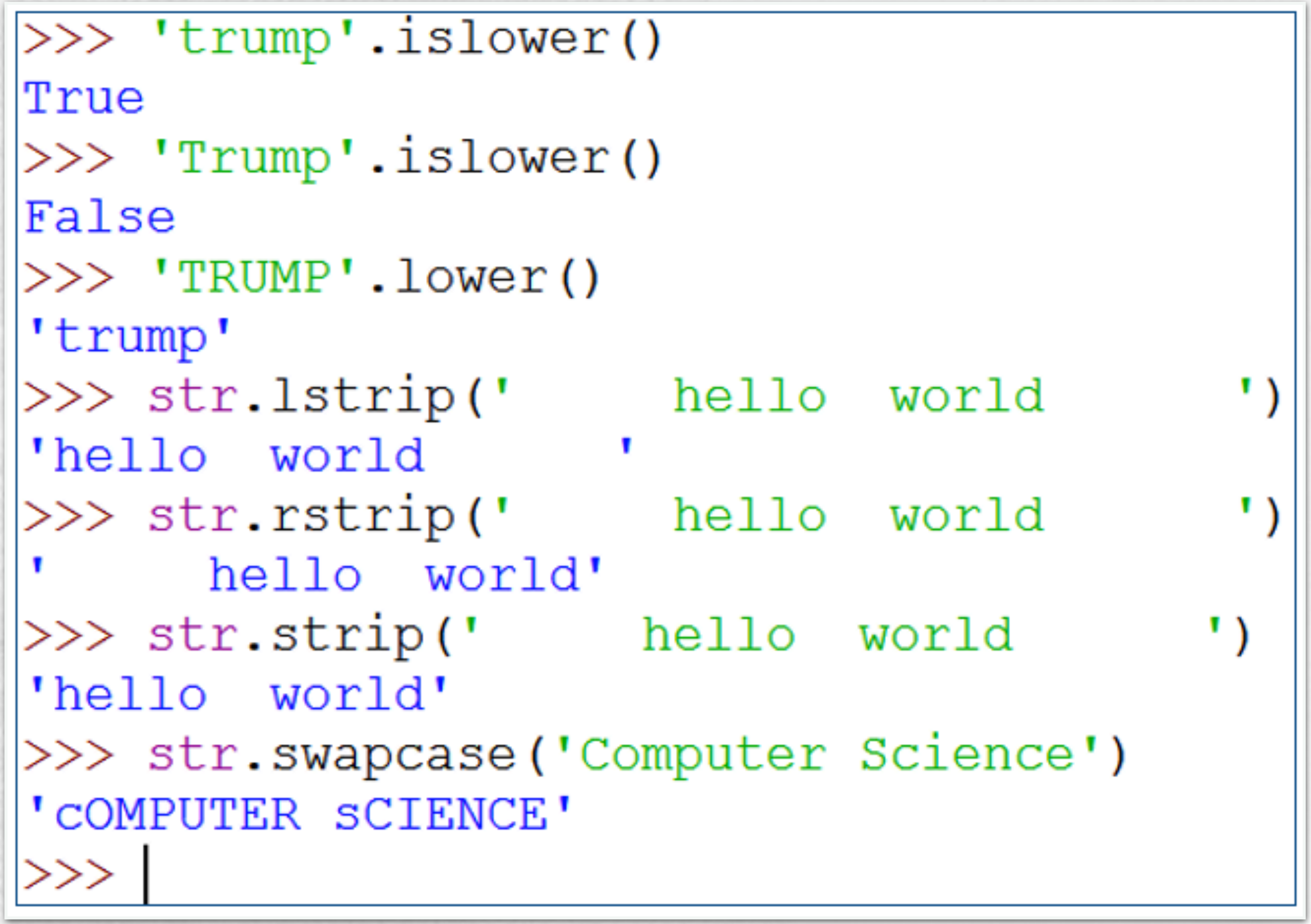


```
>>> 'trump'.capitalize()
'Trump'
>>> 'ATTGGG'.count('T')
2
>>> 'strange'.endswith('ge')
True
>>> 'strange'.endswith('gE')
False
>>> 'strange'.find('an')
3
>>> 'strange'.find('ei')
-1
>>> 'strange'.find('an',4)
-1
>>> 'strange'.find('an',2,5)
3
```

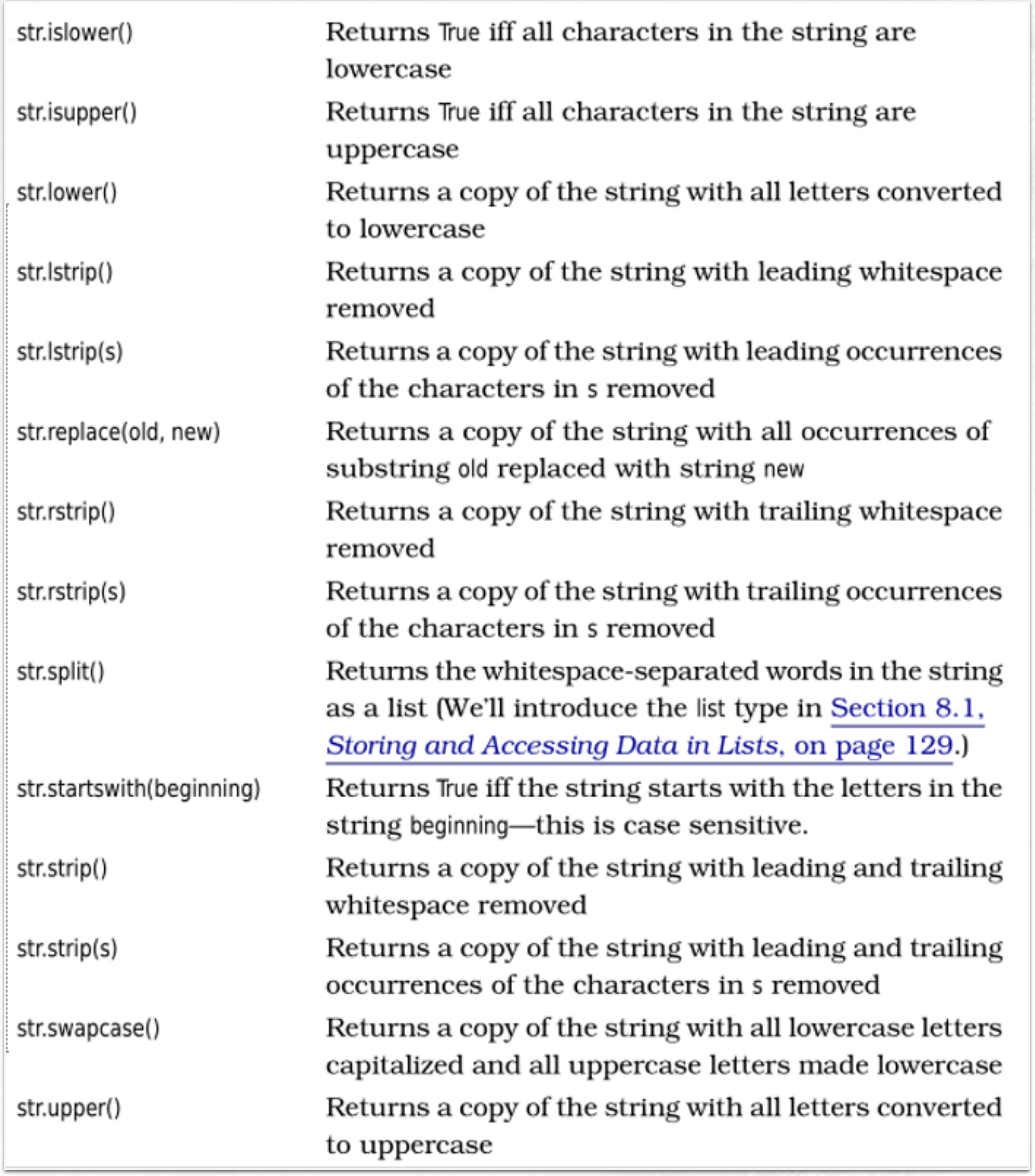


Method	Description
str.capitalize()	Returns a copy of the string with the first letter capitalized and the rest lowercase
str.count(s)	Returns the number of nonoverlapping occurrences of s in the string
str.endswith(end)	Returns True iff the string ends with the characters in the end string—this is case sensitive.
str.find(s)	Returns the index of the first occurrence of s in the string, or -1 if s doesn't occur in the string—the first character is at index 0. This is case sensitive.
str.find(s, beg)	Returns the index of the first occurrence of s at or after index beg in the string, or -1 if s doesn't occur in the string at or after index beg—the first character is at index 0. This is case sensitive.
str.find(s, beg, end)	Returns the index of the first occurrence of s between indices beg (inclusive) and end (exclusive) in the string, or -1 if s does not occur in the string between indices beg and end—the first character is at index 0. This is case sensitive.
str.format(«expressions»)	Returns a string made by substituting for placeholder fields in the string—each field is a pair of braces ('{' and '}') with an integer in between; the expression arguments are numbered from left to right starting at 0. Each field is replaced by the value produced by evaluating the expression whose index corresponds with the integer in between the braces of the field. If an expression produces a value that isn't a string, that value is converted into a string.

String methods



```
>>> 'trump'.islower()
True
>>> 'Trump'.islower()
False
>>> 'TRUMP'.lower()
'trump'
>>> str.lstrip('     hello  world       ')
'hello  world       '
>>> str.rstrip('     hello  world       ')
'     hello  world'
>>> str.strip('     hello  world       ')
'hello  world'
>>> str.swapcase('Computer Science')
'cOMPUTER sCIENCE'
>>> 
```



str.islower()	Returns True iff all characters in the string are lowercase
str.isupper()	Returns True iff all characters in the string are uppercase
str.lower()	Returns a copy of the string with all letters converted to lowercase
str.lstrip()	Returns a copy of the string with leading whitespace removed
str.lstrip(s)	Returns a copy of the string with leading occurrences of the characters in s removed
str.replace(old, new)	Returns a copy of the string with all occurrences of substring old replaced with string new
str.rstrip()	Returns a copy of the string with trailing whitespace removed
str.rstrip(s)	Returns a copy of the string with trailing occurrences of the characters in s removed
str.split()	Returns the whitespace-separated words in the string as a list (We'll introduce the list type in Section 8.1, *Storing and Accessing Data in Lists*, on page 129.)
str.startswith(beginning)	Returns True iff the string starts with the letters in the string beginning—this is case sensitive.
str.strip()	Returns a copy of the string with leading and trailing whitespace removed
str.strip(s)	Returns a copy of the string with leading and trailing occurrences of the characters in s removed
str.swapcase()	Returns a copy of the string with all lowercase letters capitalized and all uppercase letters made lowercase
str.upper()	Returns a copy of the string with all letters converted to uppercase

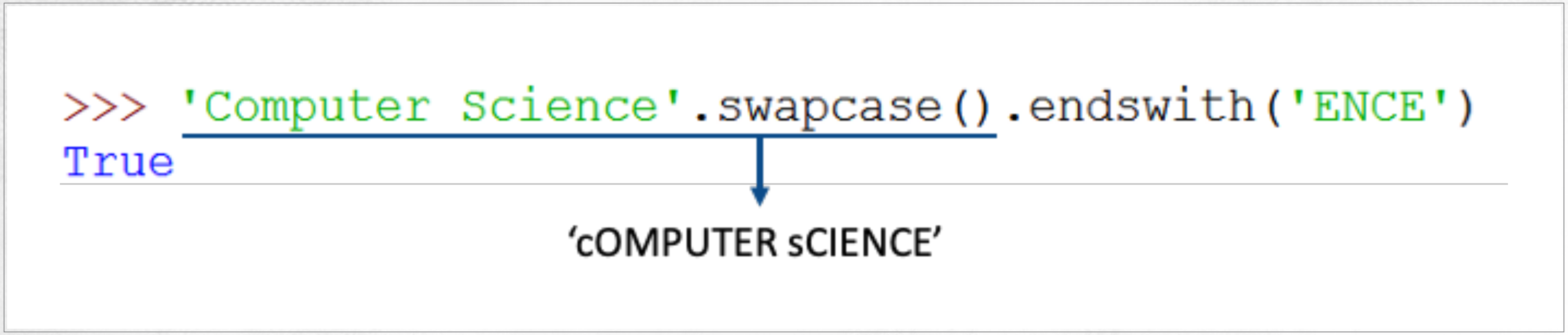
String methods: Practice

```
>>> ‘hello’.upper()
>>> ‘Happy Birthday!’.lower()
>>> ‘WeeeEEEeeeEEee’.swapcase()
>>> ‘ABC123’.isupper()
>>> ‘aeiusAEIOU’.count(‘a’)
>>> ‘hello’.endswith(‘o’)
>>> ‘hello’.startswith(‘H’)
>>> ‘Hello {0}’.format(‘Python’)
>>> ‘Hello {}! Hello {}!’.format(‘Python’,’World’)
```

str.format()

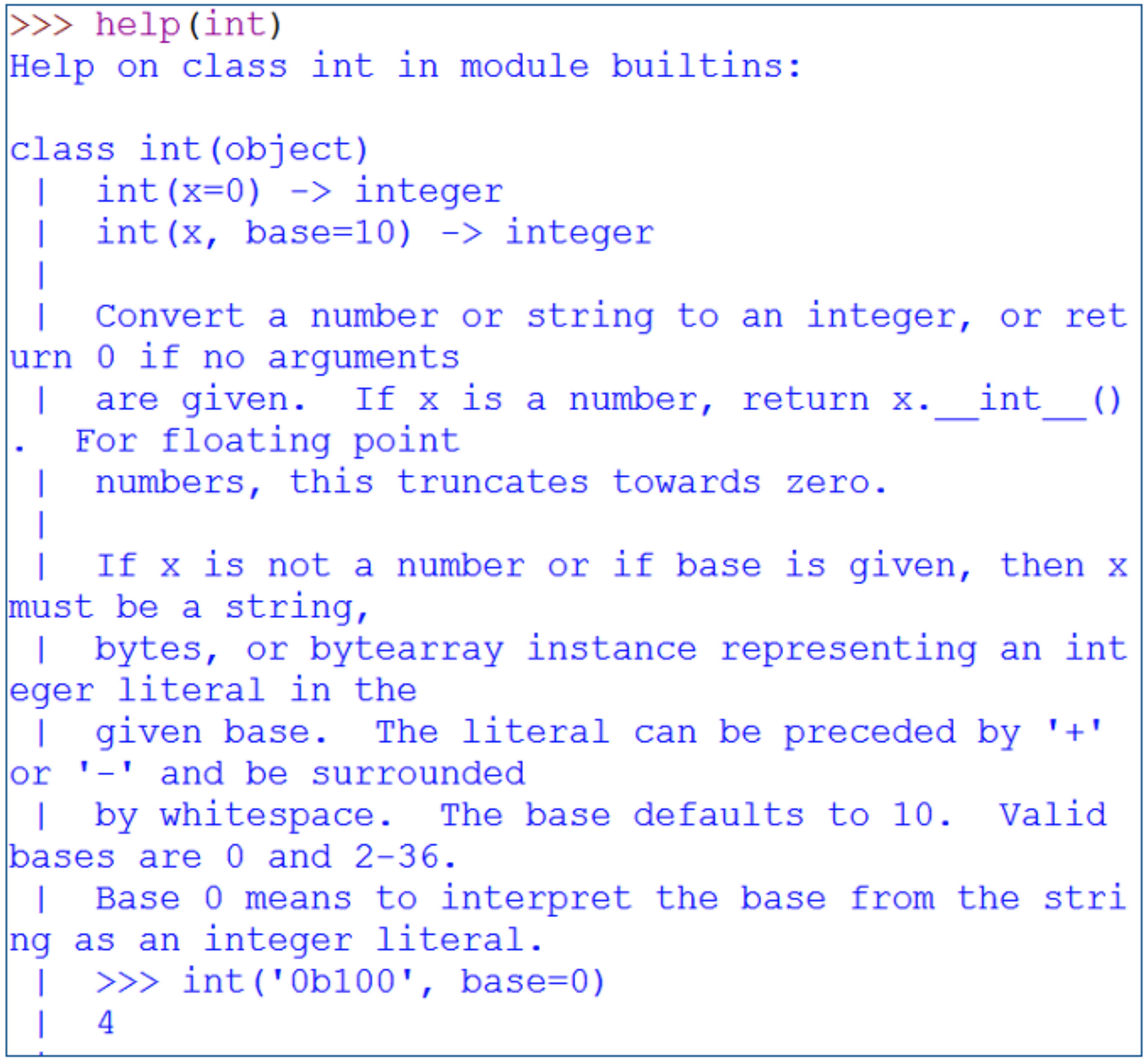
```
>>> '{0} ate {1} apples {2}'.format('I','3','yesterday')
'I ate 3 apples yesterday'
>>> '{0} ate {1} apples {2}'.format('You','five','at 2 pm')
'You ate five apples at 2 pm'
>>> '{1} ate {0} apples {2}'.format('two','He','on Monday')
'He ate two apples on Monday'
>>> '{} ate {} apples {}'.format('He','two','on Monday')
'He ate two apples on Monday'

>>> my_pi = 3.141592
>>> 'Pi rounded to {0} decimal places is {1:.2f}.'.format(2, my_pi)
'Pi rounded to 2 decimal places is 3.14.'
>>> 'Pi rounded to {0} decimal places is {1:.3f}.'.format(3, my_pi)
'Pi rounded to 3 decimal places is 3.142.'
>>> sentence = 'Pi rounded to {0} decimal places is {1:.3f}.'
>>> sentence.format(3, my_pi)
'Pi rounded to 3 decimal places is 3.142.'
>>> 'Pi rounded to {} decimal places is {:.2f}.'.format(2, my_pi)
'Pi rounded to 2 decimal places is 3.14.'
```

Nesting

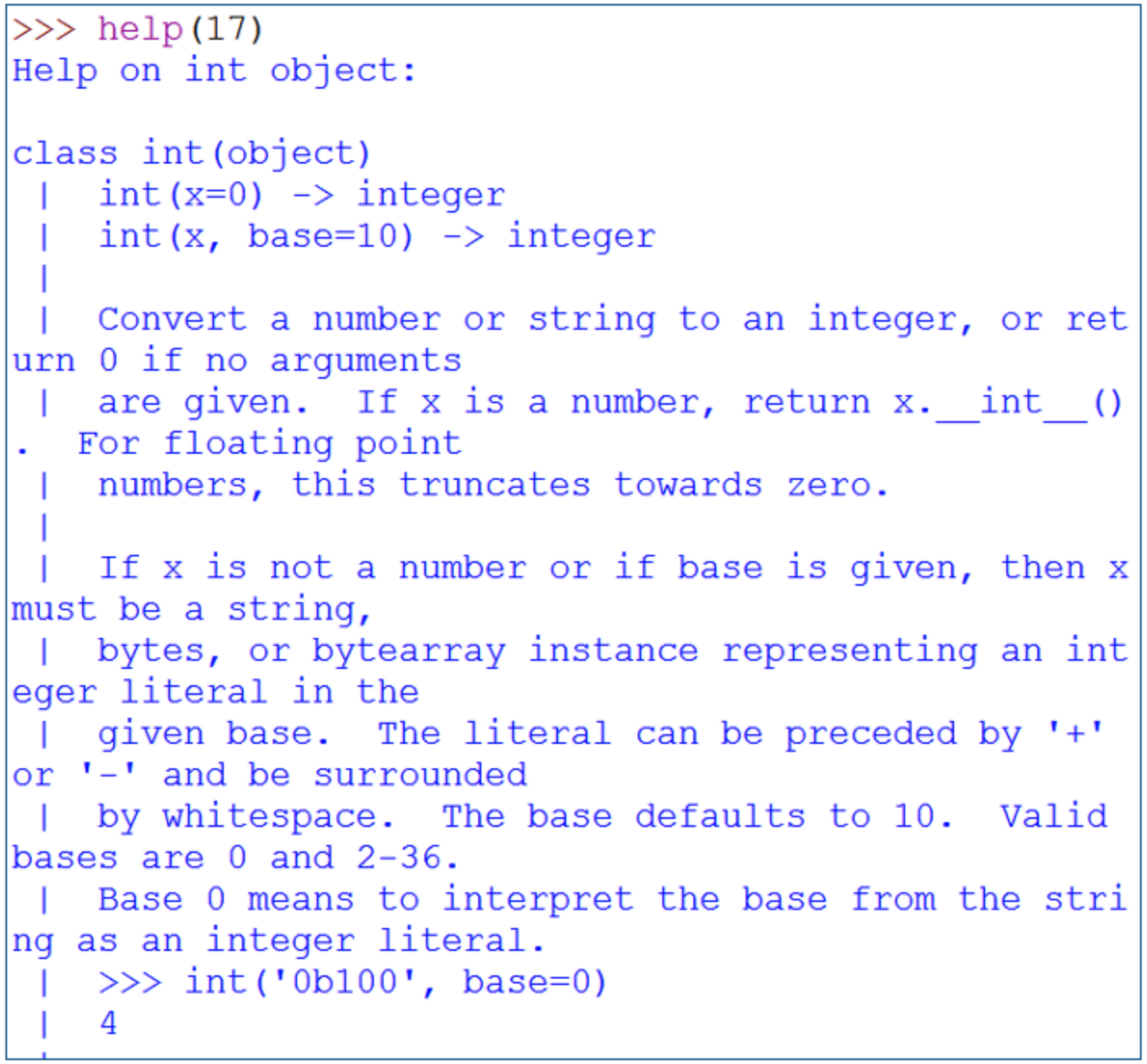


```
>>> 'Computer Science'.swapcase().endswith('ENCE')
True
```

Class and object



```
>>> help(int)
Help on class int in module builtins:

class int(object)
 |  int(x=0) -> integer
 |  int(x, base=10) -> integer
 |
 |  Convert a number or string to an integer, or ret
urn 0 if no arguments
 |  are given.  If x is a number, return x.__int__()
.  For floating point
 |  numbers, this truncates towards zero.
 |
 |  If x is not a number or if base is given, then x
must be a string,
 |  bytes, or bytearray instance representing an int
eger literal in the
 |  given base.  The literal can be preceded by '+'
or '-' and be surrounded
 |  by whitespace.  The base defaults to 10.  Valid
bases are 0 and 2-36.
 |  Base 0 means to interpret the base from the stri
ng as an integer literal.
 |  >>> int('0b100', base=0)
 |  4
```



```
>>> help(17)
Help on int object:

class int(object)
 |  int(x=0) -> integer
 |  int(x, base=10) -> integer
 |
 |  Convert a number or string to an integer, or ret
urn 0 if no arguments
 |  are given.  If x is a number, return x.__int__()
.  For floating point
 |  numbers, this truncates towards zero.
 |
 |  If x is not a number or if base is given, then x
must be a string,
 |  bytes, or bytearray instance representing an int
eger literal in the
 |  given base.  The literal can be preceded by '+'
or '-' and be surrounded
 |  by whitespace.  The base defaults to 10.  Valid
bases are 0 and 2-36.
 |  Base 0 means to interpret the base from the stri
ng as an integer literal.
 |  >>> int('0b100', base=0)
 |  4
```

Object-oriented programming

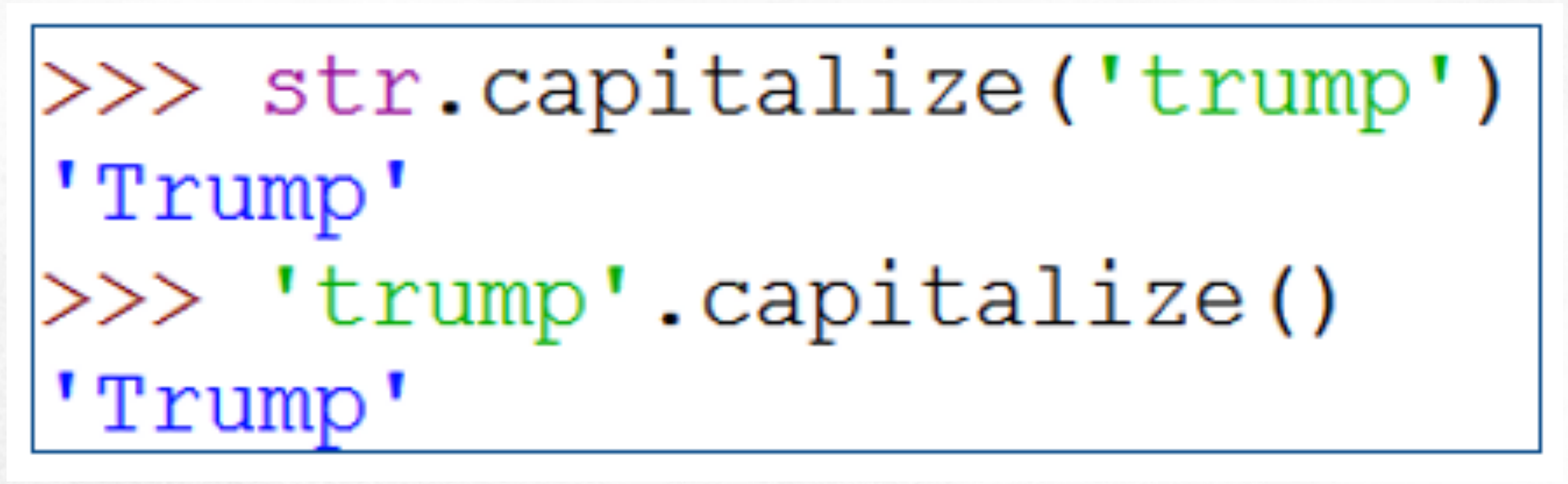
- Python is an object-oriented programming language
- "Object-oriented" is a style of programming
- The objects are the main focus

- Imperative programming set the primary focus on functions, and pass the objects to the functions.
- Python allows a mixture of both styles.

- Later we will learn how to create new kinds of objects

Summary

- Classes are like modules, except that class contain methods and modules contain functions
- Methods are like functions, except that the first argument must be an object of the class in which the method is defined.
- Method calls:



```
>>> str.capitalize('trump')
'Trump'
>>> 'trump'.capitalize()
'Trump'
```

Thank you

경희대학교
KYUNG HEE UNIVERSITY