Lists

ESS 116 | Fall 2024

Prof. Henri Drake, Prof. Jane Baldwin, and Prof. Michael Pritchard

(Modified from Ethan Campbell and Katy Christensen's materials for UW's Ocean 215)

What we’ll cover in this lesson

1. What is a list?

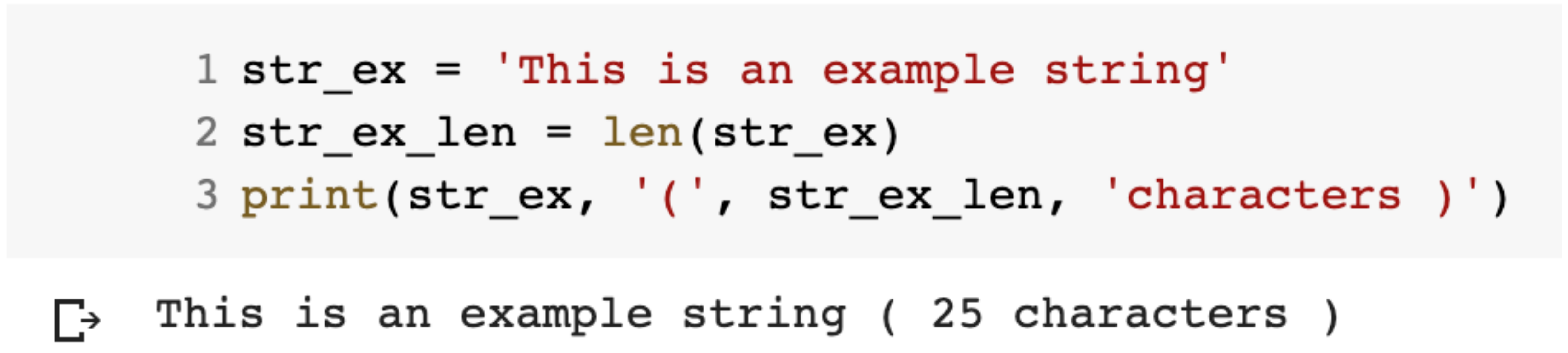
2. List functions

What we'll cover in this lesson

1. **What is a list?**
2. List functions

Lists are objects with length

Remember: strings have a length that includes each character



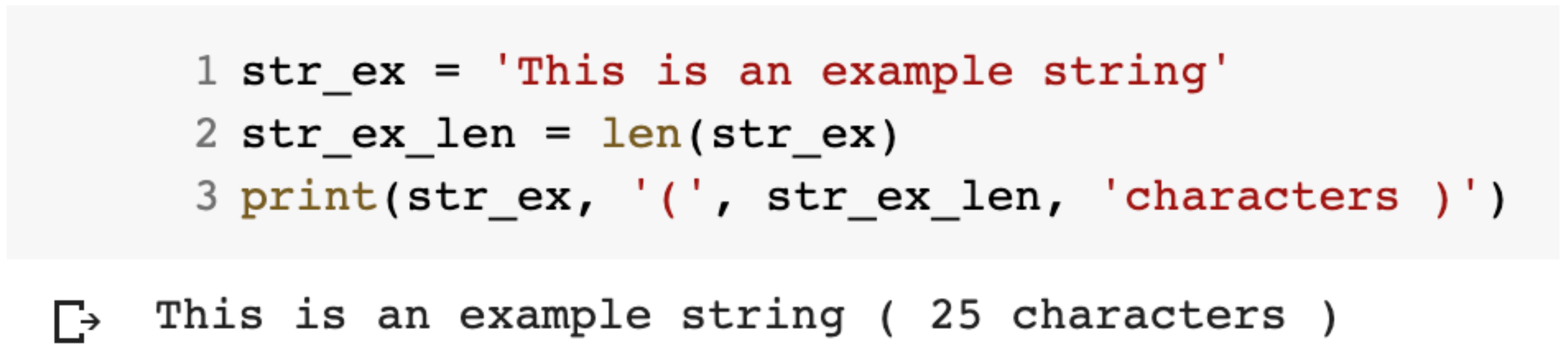
```
str_ex = 'This is an example string'
str_ex_len = len(str_ex)
print(str_ex, '(', str_ex_len, 'characters )')
```

```
This is an example string ( 25 characters )
```

Lists also have length! The length of a list includes all of the items within...

A list can have any object type as an item

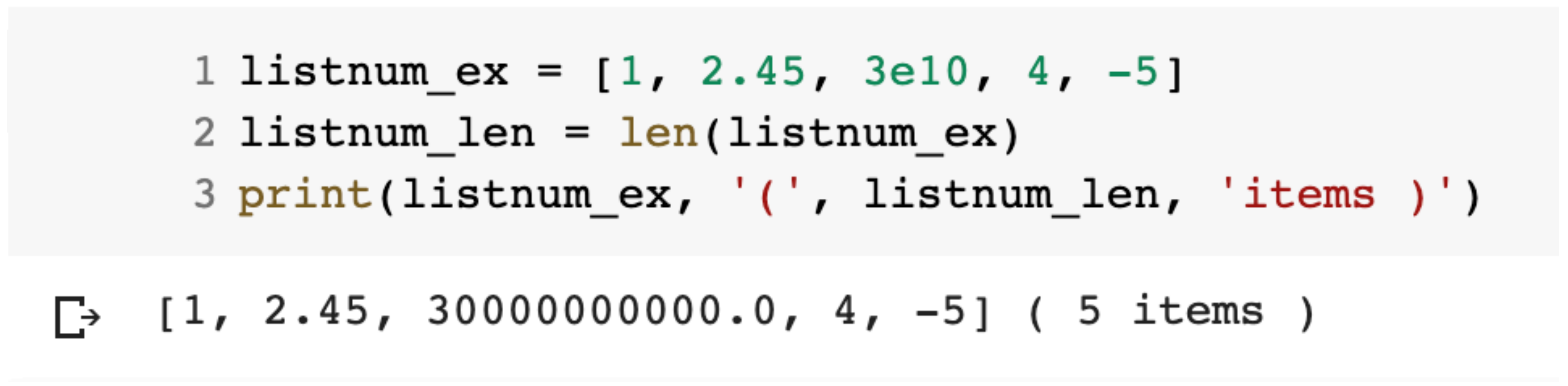
Remember: strings have a length that includes each character



```
str_ex = 'This is an example string'
str_ex_len = len(str_ex)
print(str_ex, '(', str_ex_len, 'characters )')
```

```
This is an example string ( 25 characters )
```

A list can have numbers.



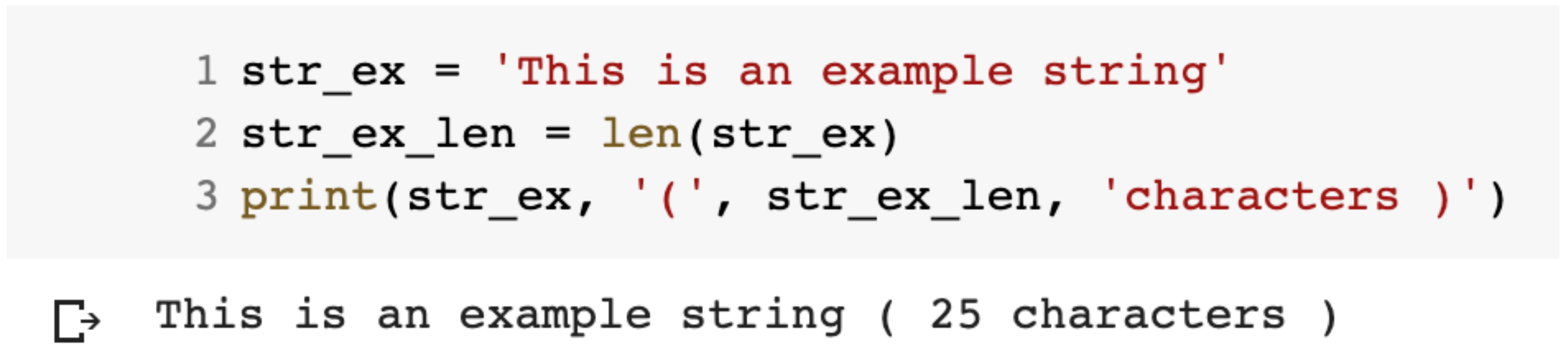
```
listnum_ex = [1, 2.45, 3e10, 4, -5]
listnum_len = len(listnum_ex)
print(listnum_ex, '(', listnum_len, 'items )')
```

```
[1, 2.45, 30000000000.0, 4, -5] ( 5 items )
```

Lists also have length! The length of a list includes all of the items within...

A list can have any object type as an item

Remember: strings have a length that includes each character

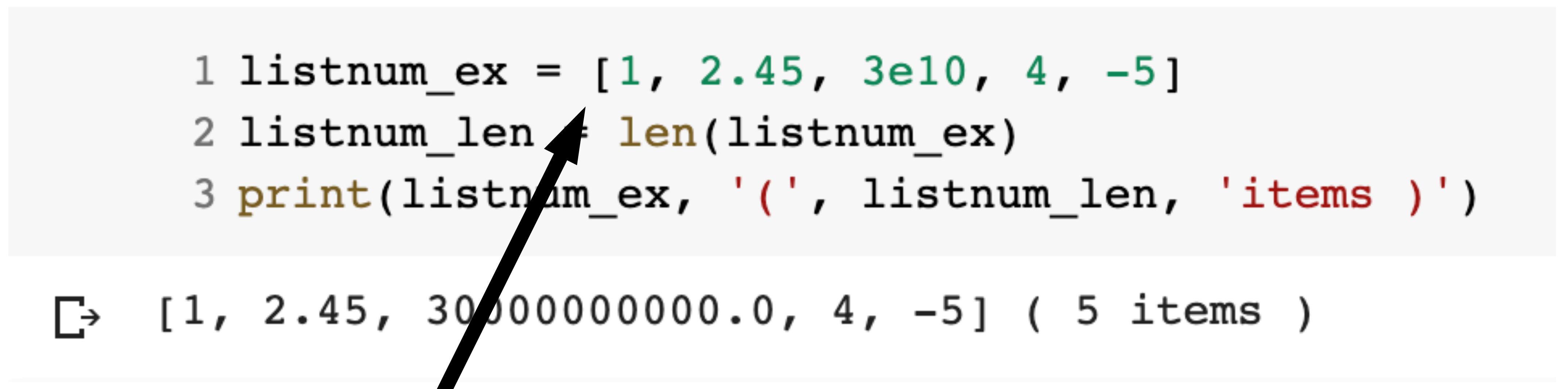


```python
str_ex = 'This is an example string'
str_ex_len = len(str_ex)
print(str_ex, '(', str_ex_len, 'characters )')
```

```
This is an example string ( 25 characters )
```

Lists also have length! The length of a list includes all of the items within...

A list can have numbers.



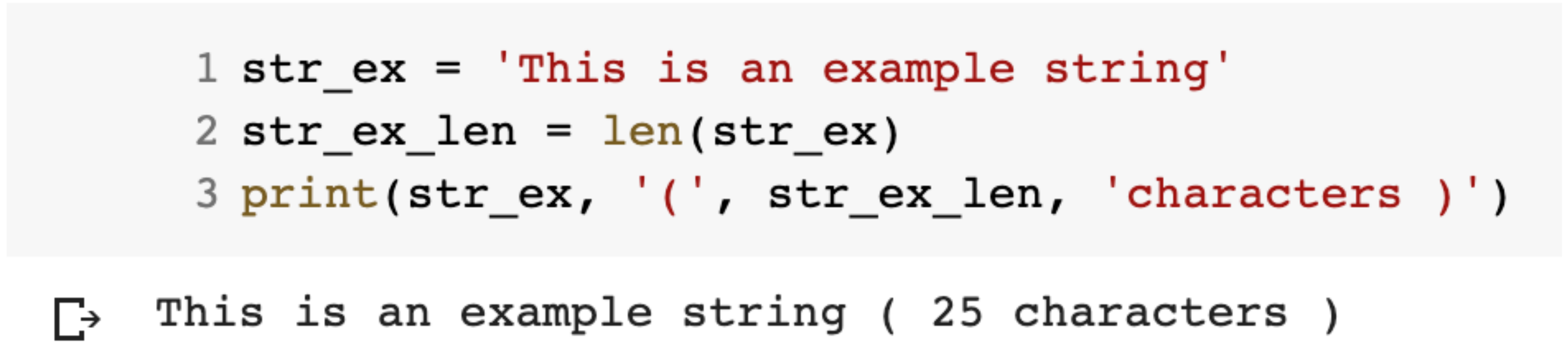
```python
listnum_ex = [1, 2.45, 3e10, 4, -5]
listnum_len = len(listnum_ex)
print(listnum_ex, '(', listnum_len, 'items )')
```

```
[1, 2.45, 30000000000.0, 4, -5] ( 5 items )
```

To put objects into a list, use square brackets [ ]

A list can have any object type as an item

Remember: strings have a length that includes each character

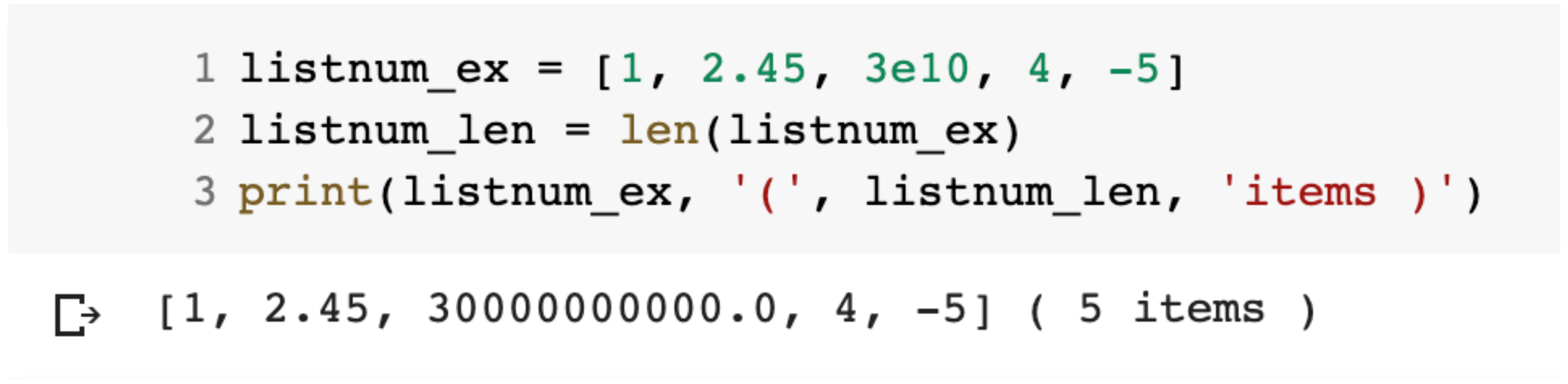


```python
str_ex = 'This is an example string'
str_ex_len = len(str_ex)
print(str_ex, '(', str_ex_len, 'characters )')
```

```
This is an example string ( 25 characters )
```

Lists also have length! The length of a list includes all of the items within...

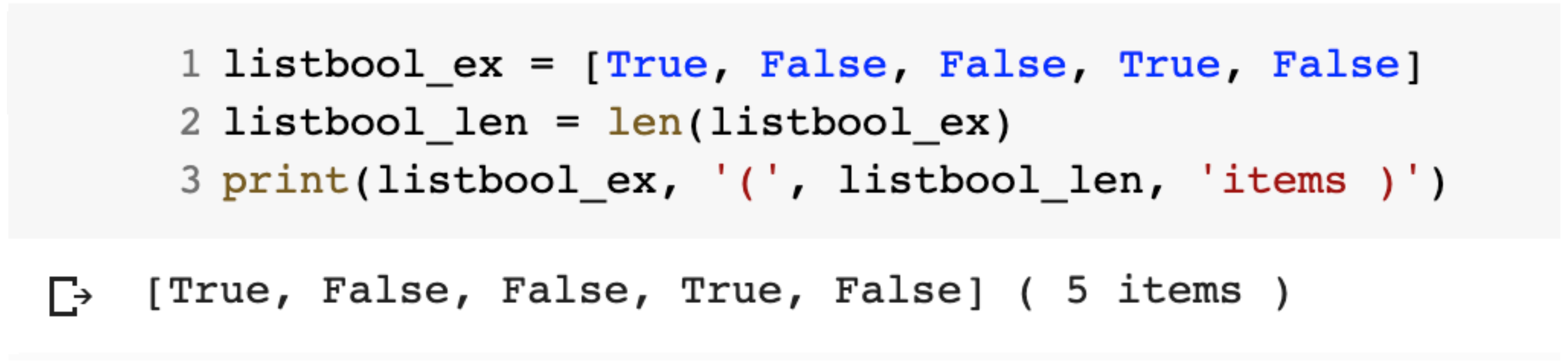
A list can have numbers.



```python
listnum_ex = [1, 2.45, 3e10, 4, -5]
listnum_len = len(listnum_ex)
print(listnum_ex, '(', listnum_len, 'items )')
```

```
[1, 2.45, 30000000000.0, 4, -5] ( 5 items )
```

A list can have Booleans.

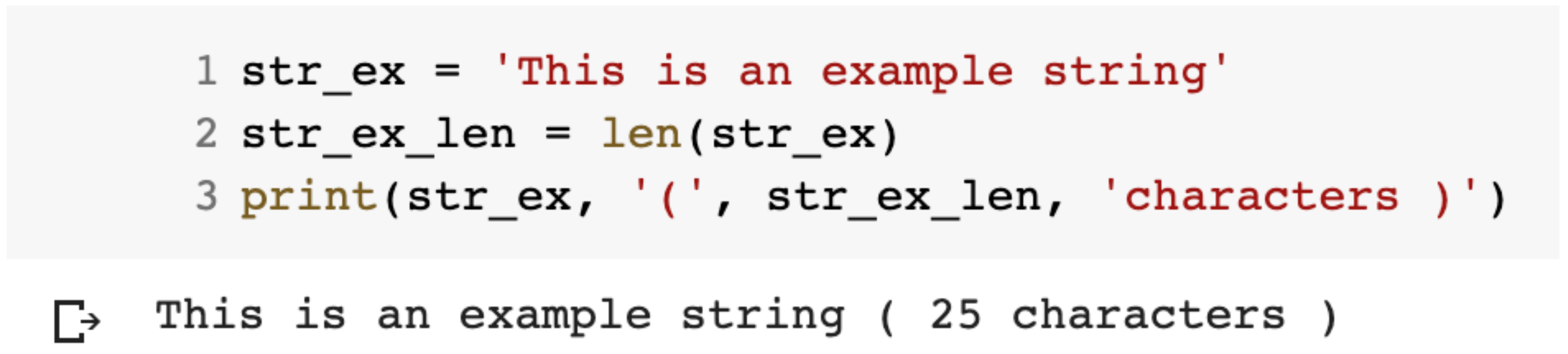


```python
listbool_ex = [True, False, False, True, False]
listbool_len = len(listbool_ex)
print(listbool_ex, '(', listbool_len, 'items )')
```

```
[True, False, False, True, False] ( 5 items )
```

A list can have any object type as an item

Remember: strings have a length that includes each character

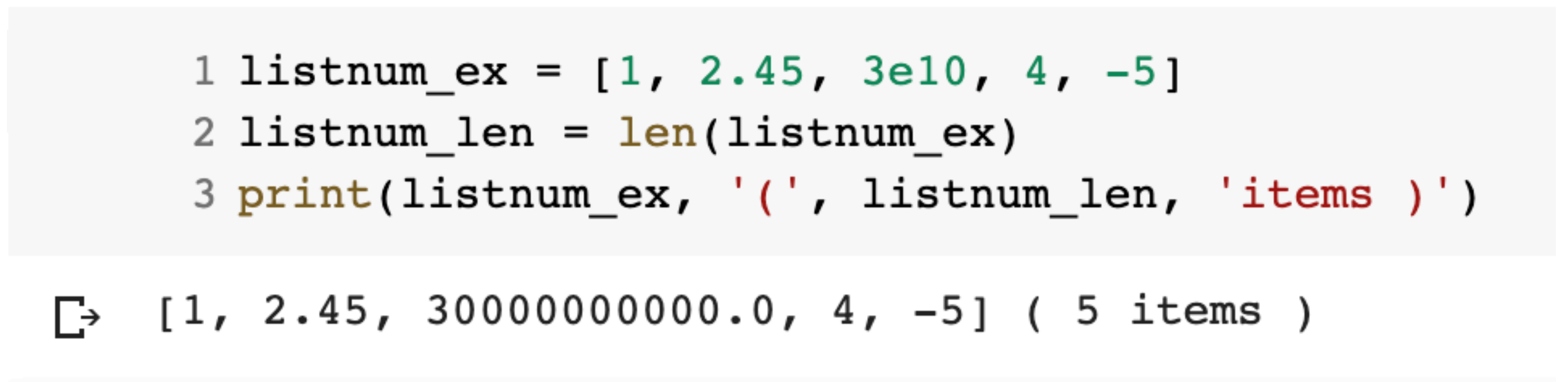


```python
str_ex = 'This is an example string'
str_ex_len = len(str_ex)
print(str_ex, '(', str_ex_len, 'characters )')
```

```
This is an example string ( 25 characters )
```

Lists also have length! The length of a list includes all of the items within...

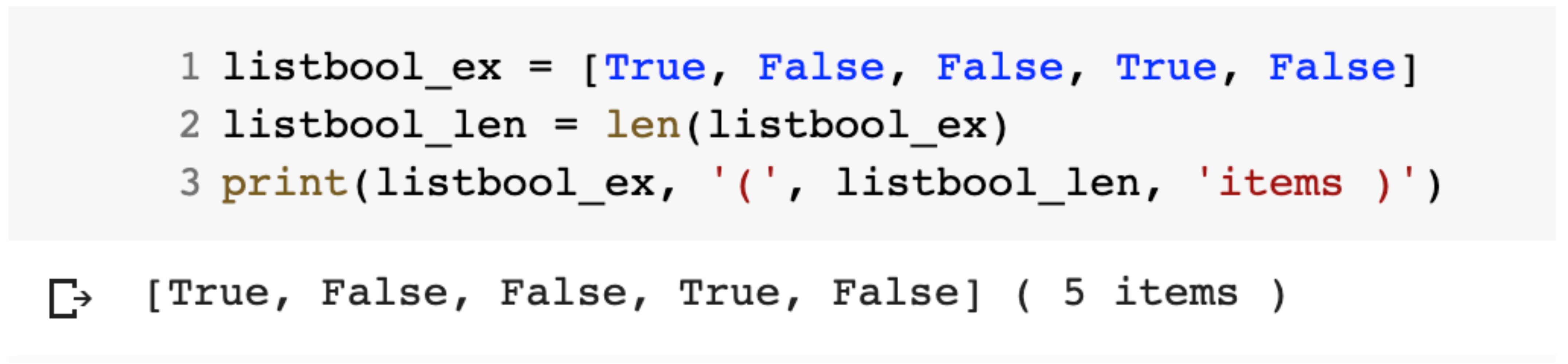
A list can have numbers.



```python
listnum_ex = [1, 2.45, 3e10, 4, -5]
listnum_len = len(listnum_ex)
print(listnum_ex, '(', listnum_len, 'items )')
```

```
[1, 2.45, 30000000000.0, 4, -5] ( 5 items )
```

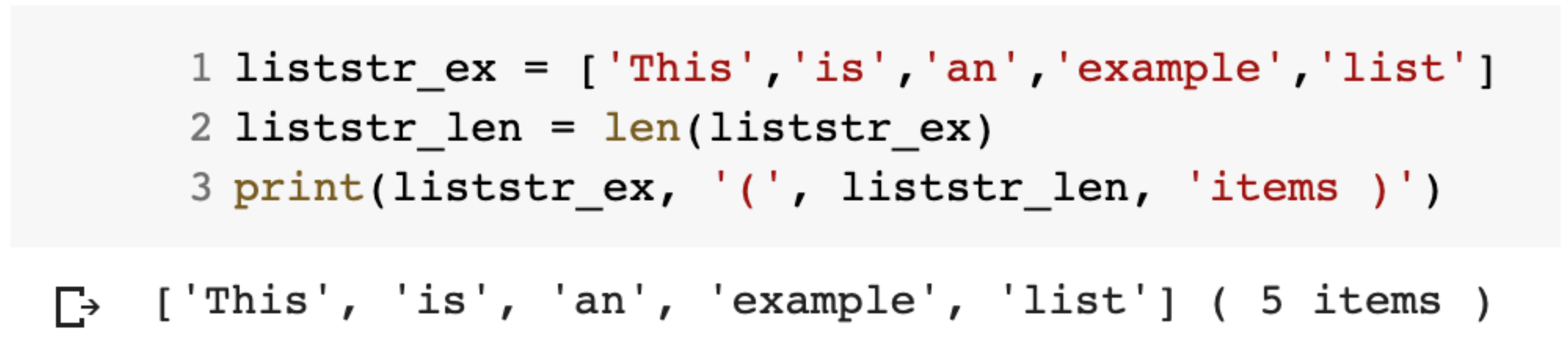
A list can have Booleans.



```python
listbool_ex = [True, False, False, True, False]
listbool_len = len(listbool_ex)
print(listbool_ex, '(', listbool_len, 'items )')
```

```
[True, False, False, True, False] ( 5 items )
```

A list can have strings.

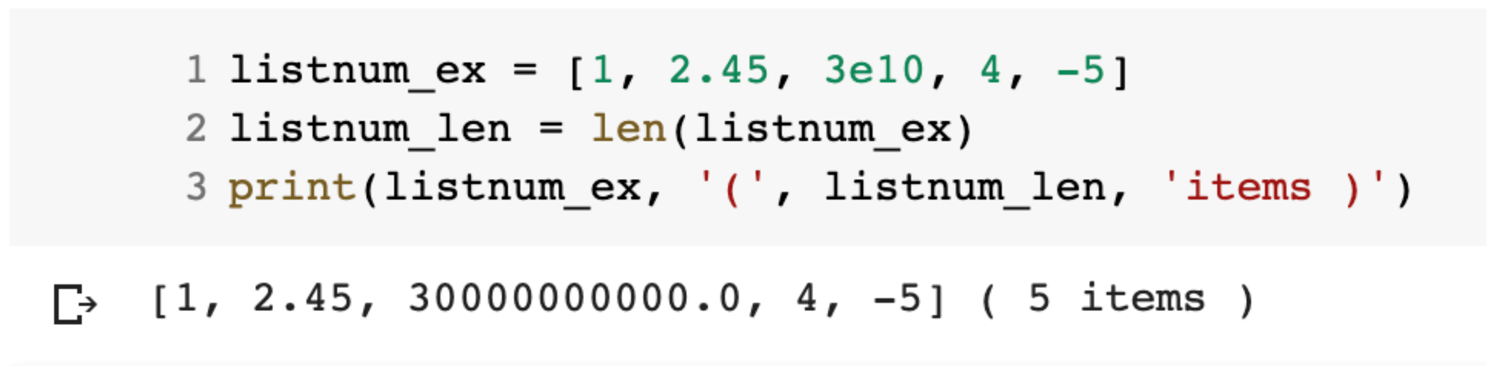


```python
liststr_ex = ['This','is','an','example','list']
liststr_len = len(liststr_ex)
print(liststr_ex, '(', liststr_len, 'items )')
```

```
['This', 'is', 'an', 'example', 'list'] ( 5 items )
```

A list can have any object type as an item

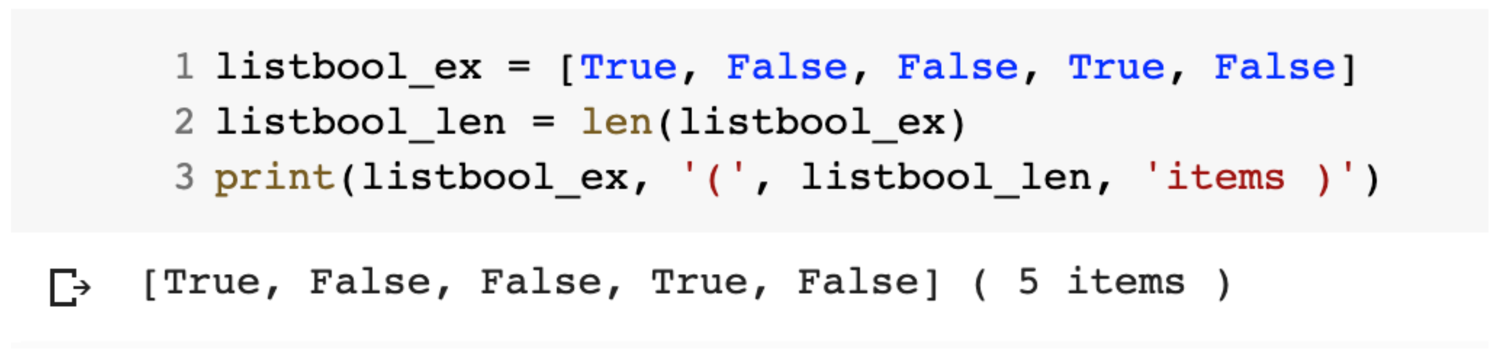
A list can have numbers.



```
listnum_ex = [1, 2.45, 3e10, 4, -5]
listnum_len = len(listnum_ex)
print(listnum_ex, '(', listnum_len, 'items )')
```

```
[1, 2.45, 30000000000.0, 4, -5] ( 5 items )
```

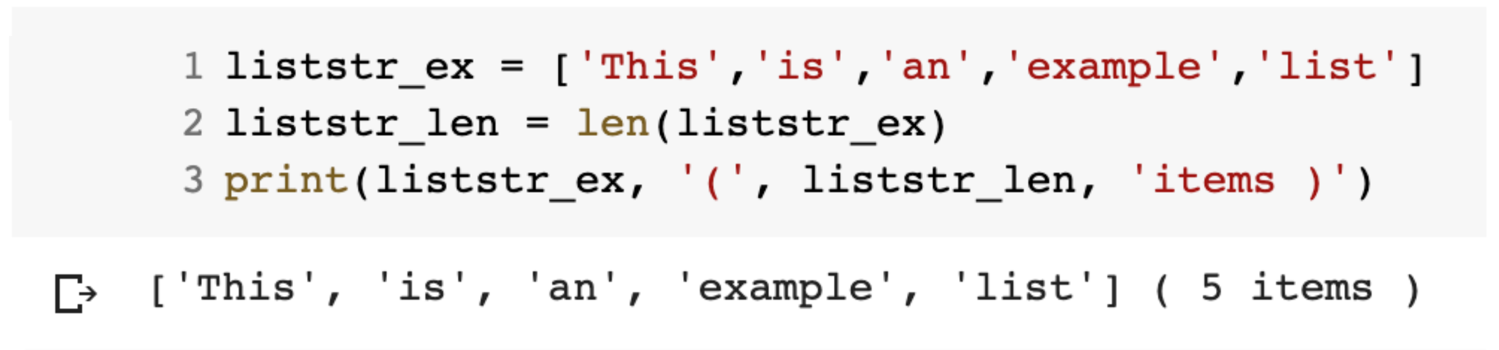
A list can have Booleans.



```
listbool_ex = [True, False, False, True, False]
listbool_len = len(listbool_ex)
print(listbool_ex, '(', listbool_len, 'items )')
```

```
[True, False, False, True, False] ( 5 items )
```

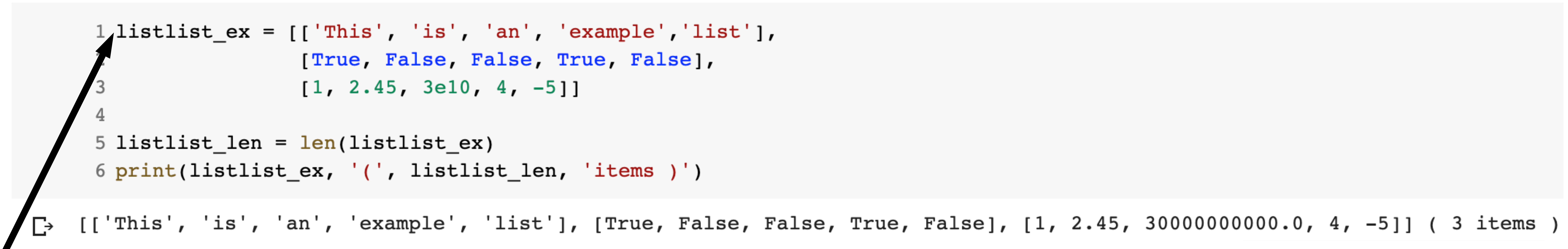
A list can have strings.



```
liststr_ex = ['This','is','an','example','list']
liststr_len = len(liststr_ex)
print(liststr_ex, '(', liststr_len, 'items )')
```

```
['This', 'is', 'an', 'example', 'list'] ( 5 items )
```

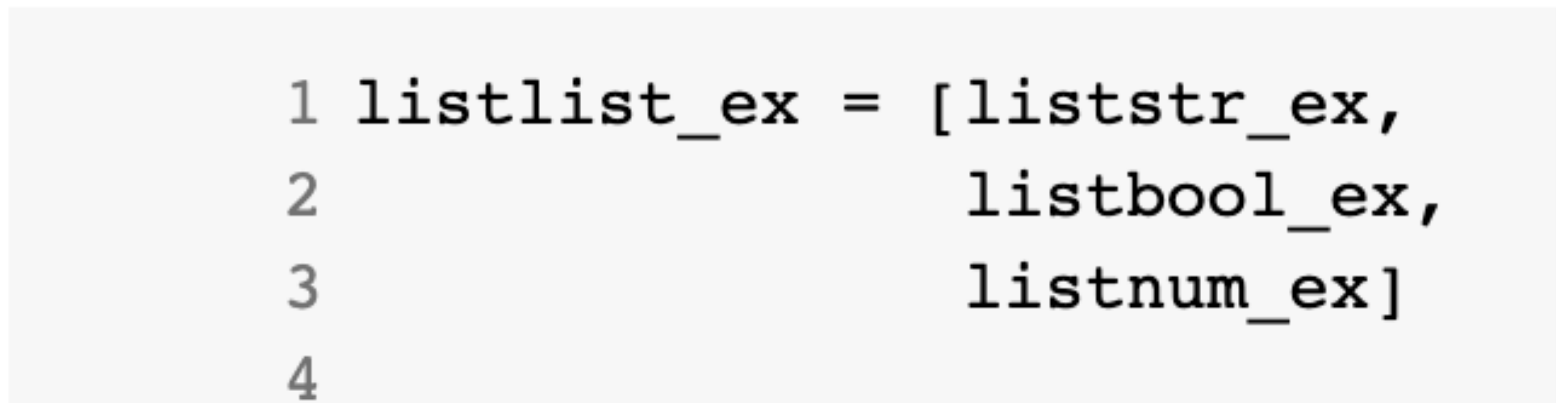
A list can have lists.



```
listlist_ex = [['This', 'is', 'an', 'example','list'],
               [True, False, False, True, False],
               [1, 2.45, 3e10, 4, -5]]

listlist_len = len(listlist_ex)
print(listlist_ex, '(', listlist_len, 'items )')
```

```
[['This', 'is', 'an', 'example', 'list'], [True, False, False, True, False], [1, 2.45, 30000000000.0, 4, -5]] ( 3 items )
```

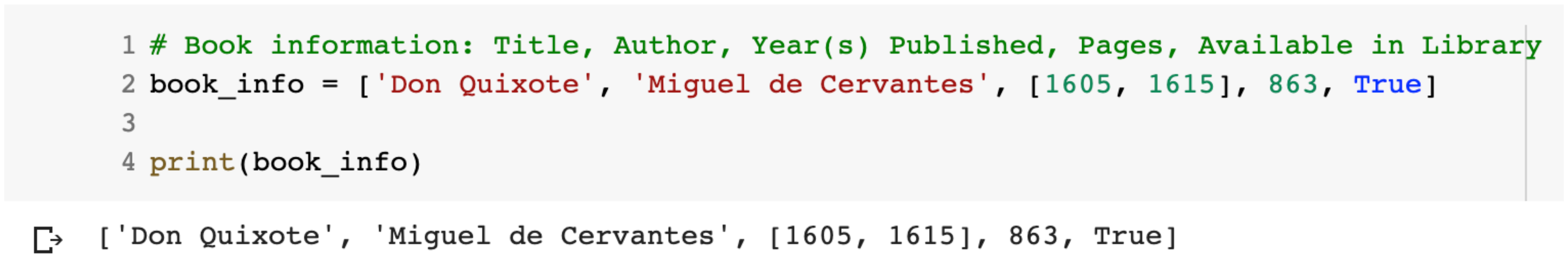
Another way to create this list would be to use the variable names for each of the previously created lists!



```
listlist_ex = [liststr_ex,
               listbool_ex,
               listnum_ex]

```

A list can have any object type as an item

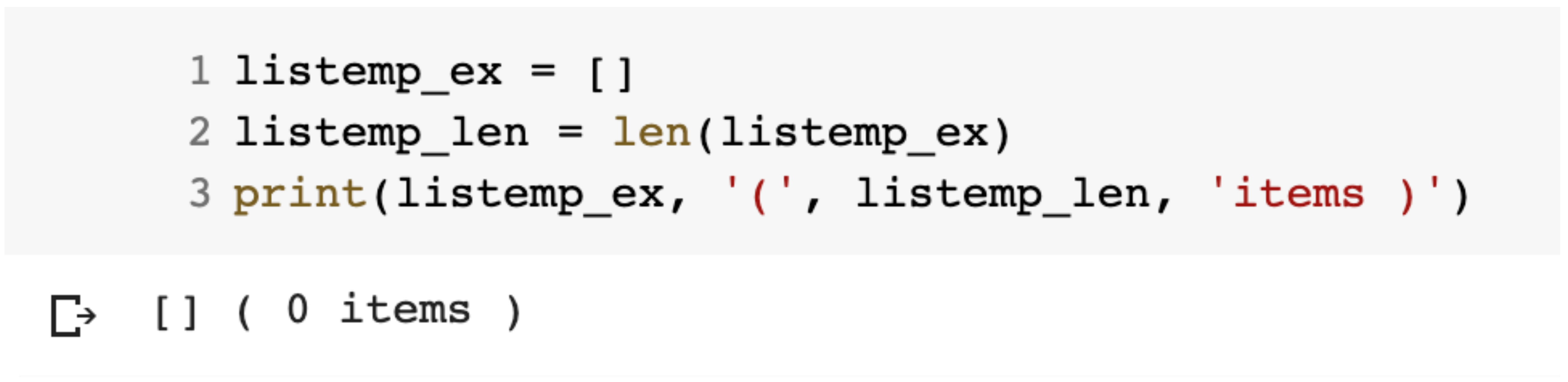
A list can have a mix of object types.



```python
# Book information: Title, Author, Year(s) Published, Pages, Available in Library
book_info = ['Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True]

print(book_info)
```

```
['Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True]
```

A list can also be empty.



```python
listemp_ex = []
listemp_len = len(listemp_ex)
print(listemp_ex, '(', listemp_len, 'items )')
```

```
[] ( 0 items )
```

List indexing and slicing

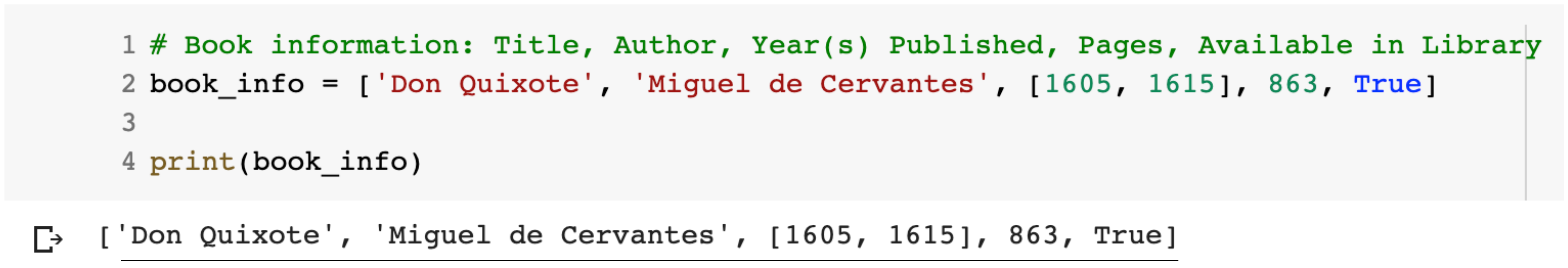
Remember:

String indexing

P	y	t	h	o	n		i	s		f	u	n	!
0	1	2	3	4	5	6	7	8	9	10	11	12	13

Indexing and slicing is the same for lists and strings

How python counts list items (indexing):



```python
# Book information: Title, Author, Year(s) Published, Pages, Available in Library
book_info = ['Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True]

print(book_info)
```

```
['Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True]
```

0 1 2 3 4

Single items	Slices
`book_info[1]`	`book_info[0:2]`
`‘Miguel de Cervantes’`	`[‘Don Quixote’, ‘Miguel de Cervantes’]`

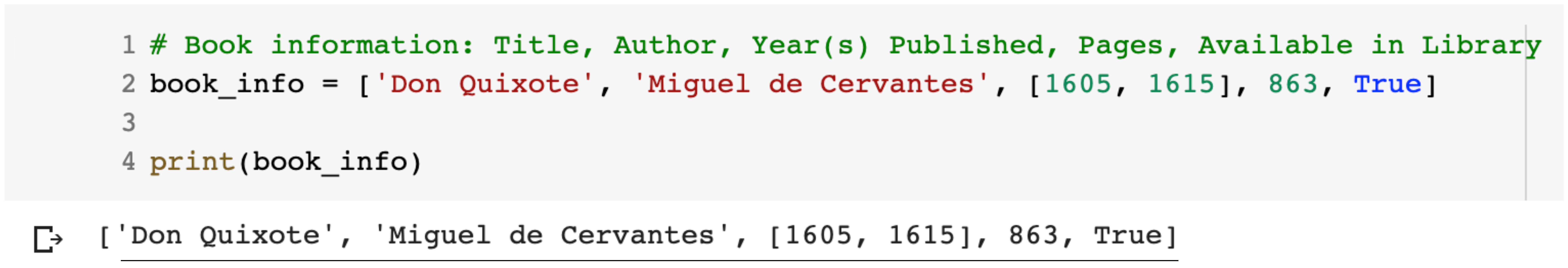
List indexing and slicing

Remember:

String indexing

P	y	t	h	o	n		i	s		f	u	n	!
0	1	2	3	4	5	6	7	8	9	10	11	12	13

How python counts list items (indexing):



```python
# Book information: Title, Author, Year(s) Published, Pages, Available in Library
book_info = ['Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True]

print(book_info)
```

```
['Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True]
```

0 1 2 3 4

Multi-level indexing	
`book_info[1]`	‘Miguel de Cervantes’
`book_info[1][0:6]`	‘Miguel’

List items that are objects with length can be sliced

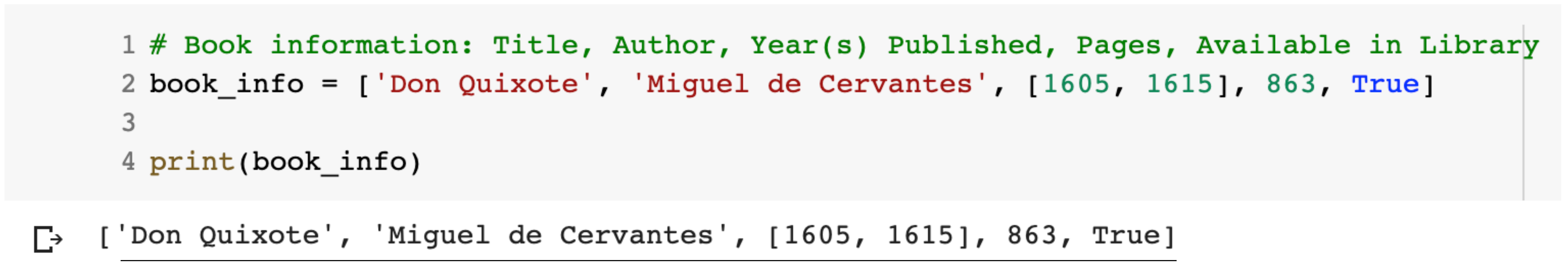
List indexing and slicing

Remember:

String indexing

P	y	t	h	o	n		i	s		f	u	n	!
0	1	2	3	4	5	6	7	8	9	10	11	12	13

How python counts list items (indexing):



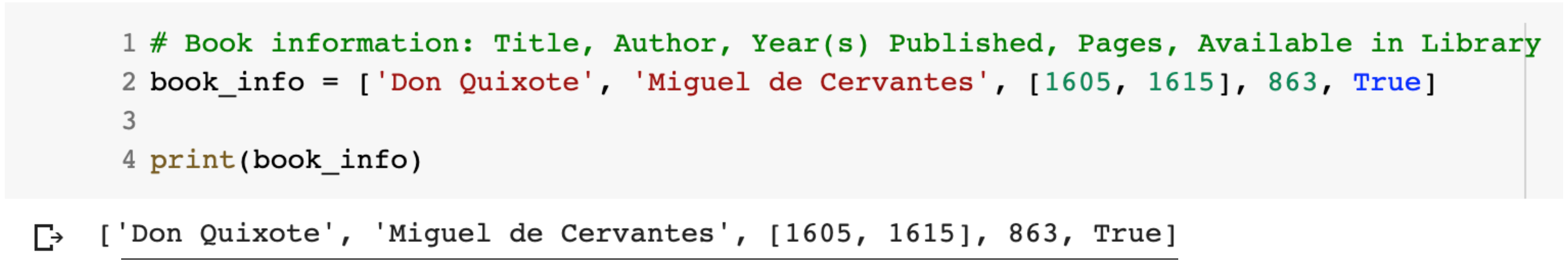
```python
# Book information: Title, Author, Year(s) Published, Pages, Available in Library
book_info = ['Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True]

print(book_info)
```

```
['Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True]
```

'Don Quixote'	'Miguel de Cervantes'	[1605, 1615]	863	True
0	1	2	3	4
-5	-4	-3	-2	-1

Negative indexing		
book_info[-1]	book_info[4]	True
book_info[-3]	book_info[2]	[1605, 1615]

List indexing and slicing

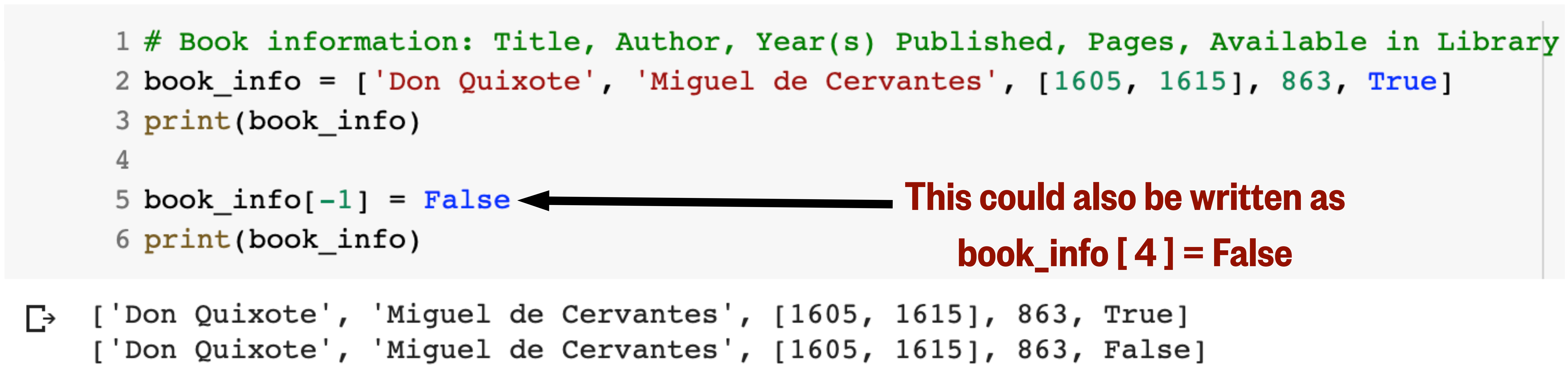


```python
# Book information: Title, Author, Year(s) Published, Pages, Available in Library
book_info = ['Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True]

print(book_info)
```

```
['Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True]
```

```
      0                  1                   2          3      4
     -5                 -4                  -3         -2     -1
```

Items in a list can be replaced using their index values



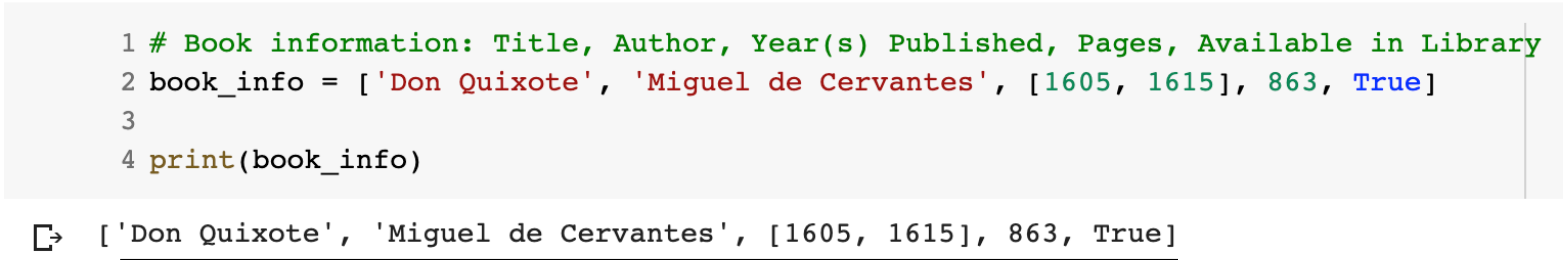
```python
# Book information: Title, Author, Year(s) Published, Pages, Available in Library
book_info = ['Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True]
print(book_info)

book_info[-1] = False
print(book_info)
```

This could also be written as

book_info [4] = False

```
['Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True]
['Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, False]
```

List indexing and slicing



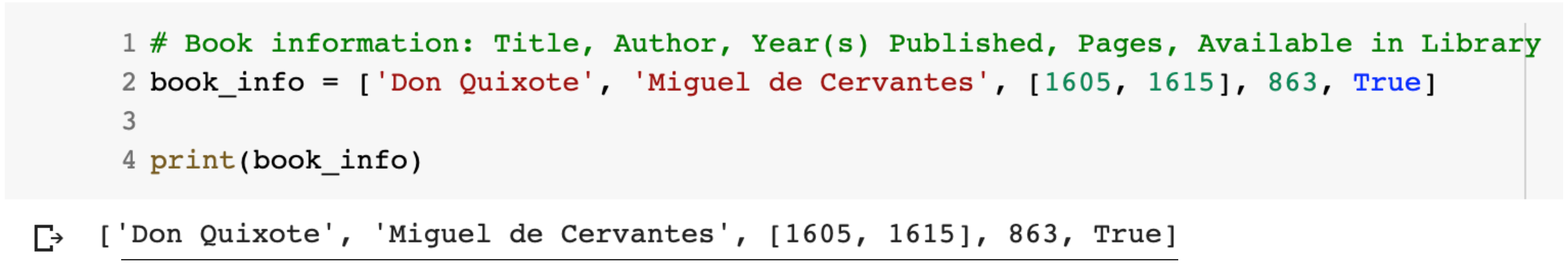
```python
# Book information: Title, Author, Year(s) Published, Pages, Available in Library
book_info = ['Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True]

print(book_info)
```

```
['Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True]
     0               1                     2          3     4
    -5              -4                    -3         -2    -1
```

If you are starting at the beginning or stopping at the end of a list, you can omit the index value in your slice

More Slicing		
`book_info[:3]`	`book_info[0:3]`	`[‘Don Quixote’, ‘Miguel de Cervantes’, [1605,1615]]`
`book_info[2:]`	`book_info[2:5]`	`[[1605, 1615], 863, True]`

List indexing and slicing



```python
# Book information: Title, Author, Year(s) Published, Pages, Available in Library
book_info = ['Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True]

print(book_info)
```

```
['Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True]
```

```
      0              1                2        3     4
     -5             -4               -3       -2    -1
```

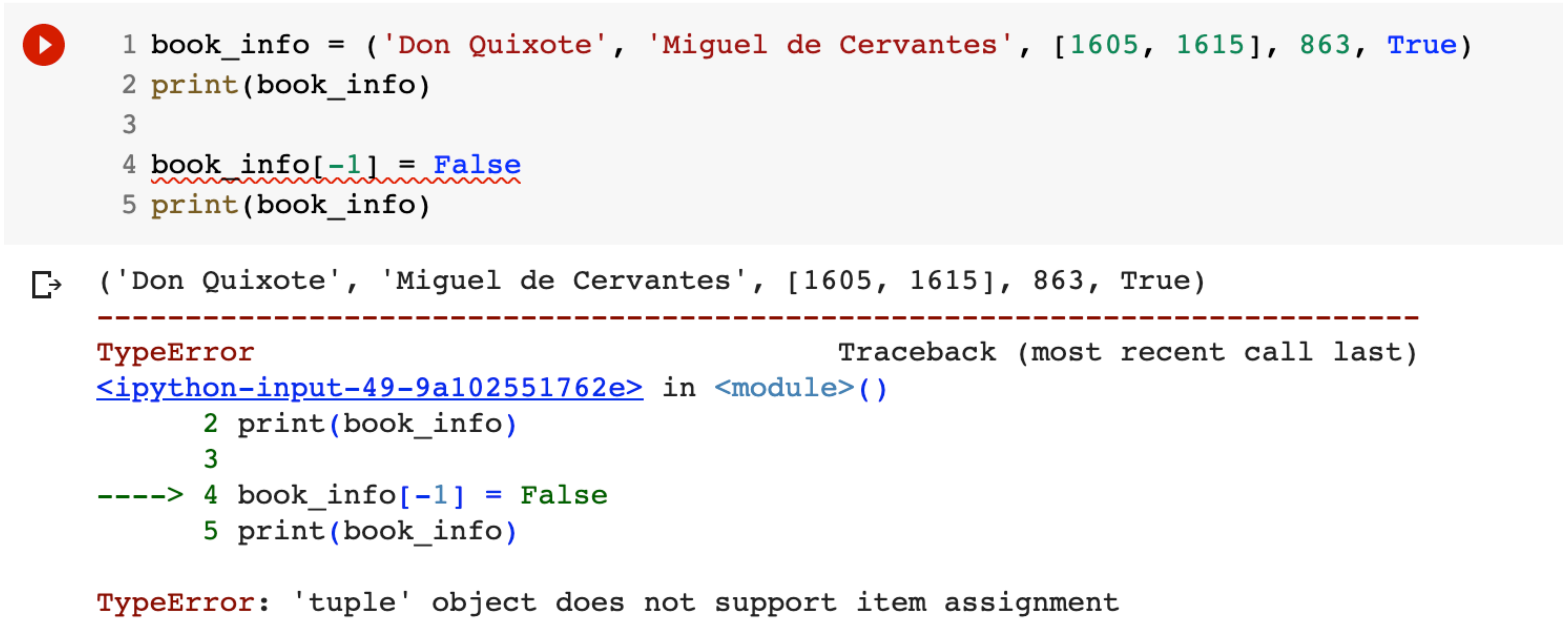
You can slice using a step argument to get every nth (1st, 2nd, 3rd, 4th, etc.) items in a list

Extended Slicing		
`book_info[0:5:1]`	`book_info[::1]`	`[‘Don Quixote’, ‘Miguel de Cervantes’, [1605,1615],863, True]`
`book_info[0:5:2]`	`book_info[::2]`	`[‘Don Quixote’, [1605,1615], True]`
`book_info[0:5:3]`	`book_info[::3]`	`[‘Don Quixote’, 863]`
`book_info[0:5:-1]`	`book_info[::-1]`	`[True, 863, [1605,1615], ‘Miguel de Cervantes’, ‘Don Quixote’]`

Objects like lists...

Tuple

A tuple is the same as a list except they are immutable - we cannot change the items inside once they are assigned. Create a tuple using parentheses ( ) around the objects you want inside.



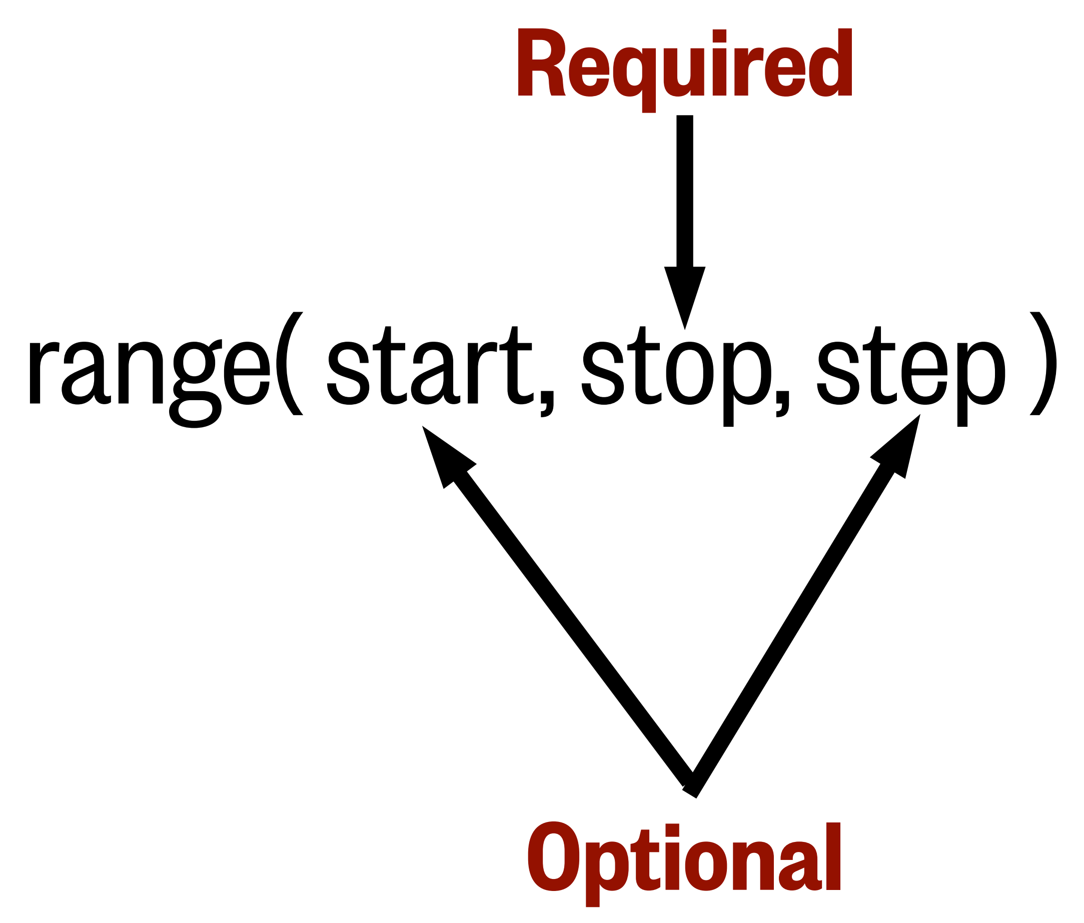
```
book_info = ('Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True)
print(book_info)

book_info[-1] = False
print(book_info)
```

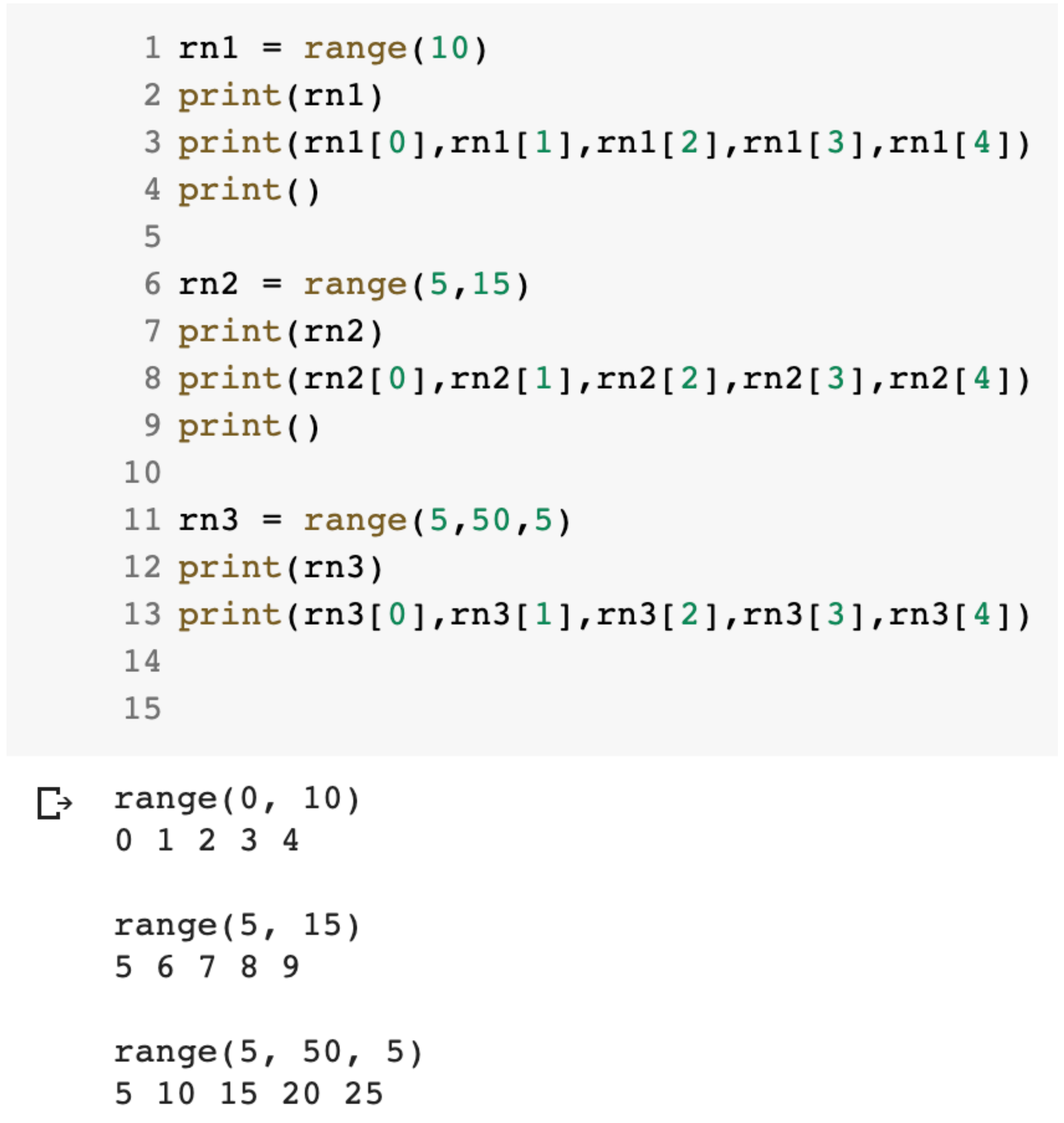
```
('Don Quixote', 'Miguel de Cervantes', [1605, 1615], 863, True)
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-49-9a102551762e> in <module>()
      2 print(book_info)
      3
----> 4 book_info[-1] = False
      5 print(book_info)

TypeError: 'tuple' object does not support item assignment
```

Objects like lists...

Range

A range object creates a sequence of numbers. The sequence starts from zero and increases by ones by default. The object created is a range object, but specific numbers in the sequence can be called using the same indexing as lists.





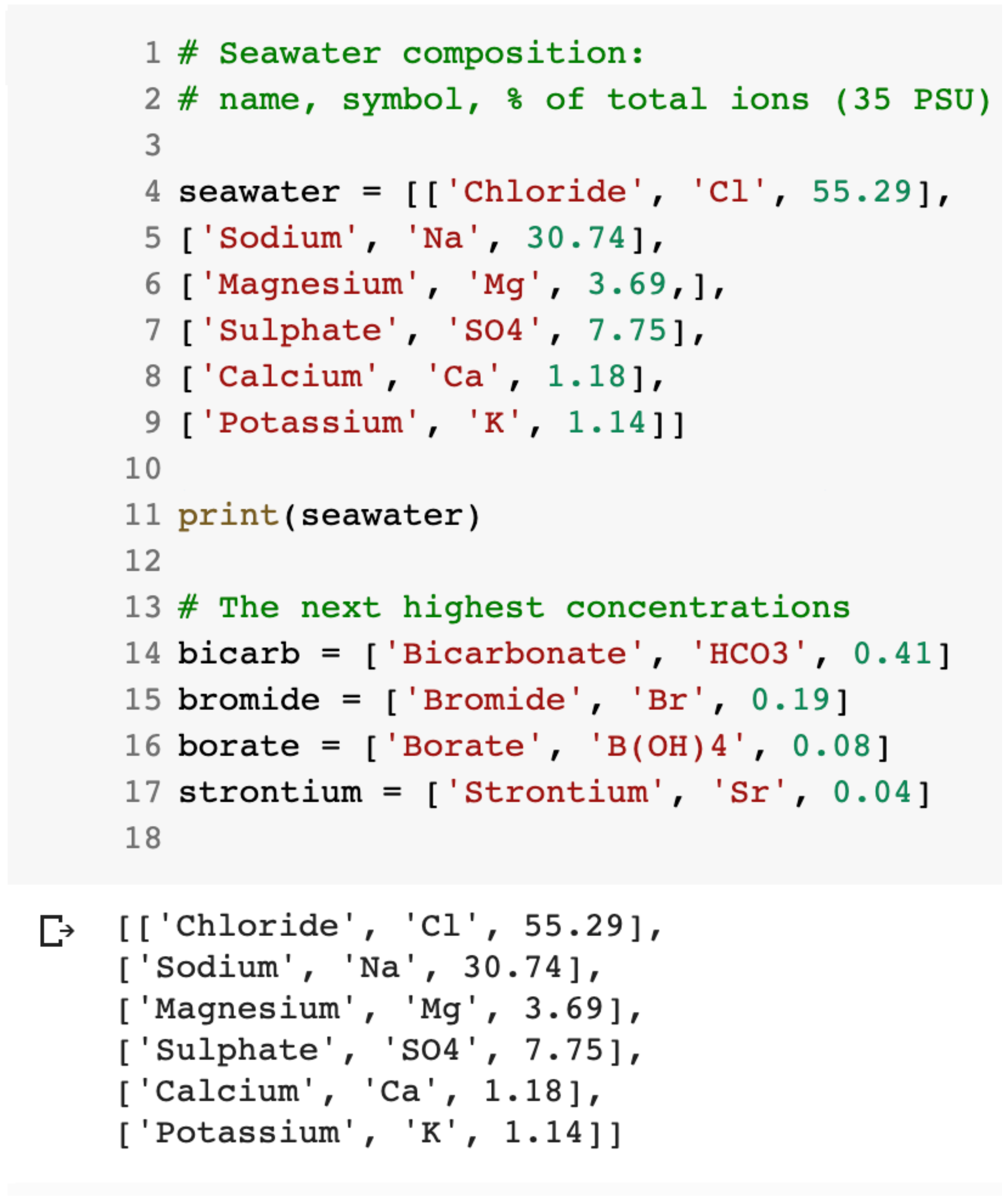
```python
rn1 = range(10)
print(rn1)
print(rn1[0],rn1[1],rn1[2],rn1[3],rn1[4])
print()

rn2 = range(5,15)
print(rn2)
print(rn2[0],rn2[1],rn2[2],rn2[3],rn2[4])
print()

rn3 = range(5,50,5)
print(rn3)
print(rn3[0],rn3[1],rn3[2],rn3[3],rn3[4])


```

```
range(0, 10)
0 1 2 3 4

range(5, 15)
5 6 7 8 9

range(5, 50, 5)
5 10 15 20 25
```

What we'll cover in this lesson

1. What is a list?
2. **List functions**
3. Object types
4. Logical Operations

Our sample list



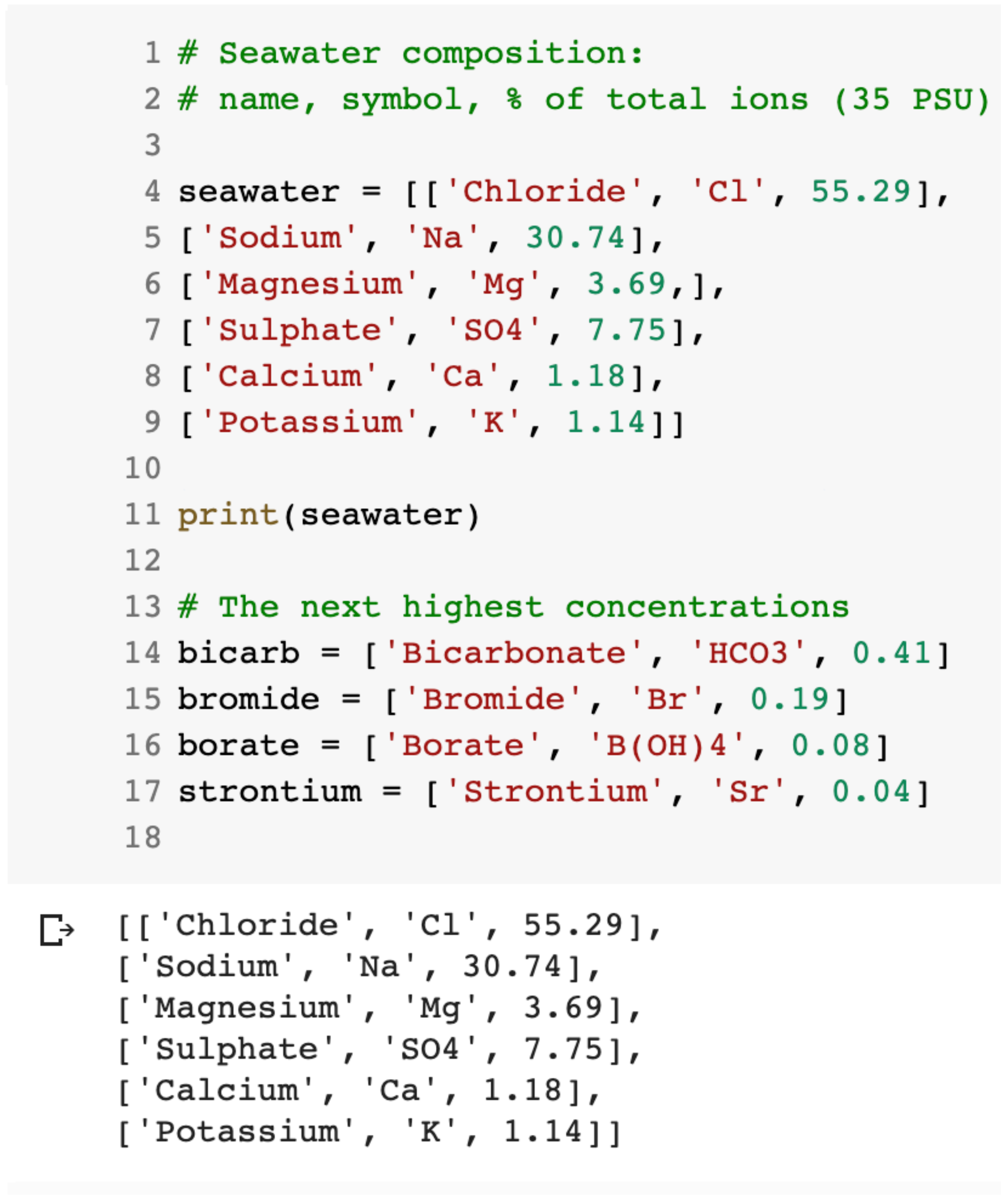
```python
# Seawater composition:
# name, symbol, % of total ions (35 PSU)

seawater = [['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69,],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]

print(seawater)

# The next highest concentrations
bicarb = ['Bicarbonate', 'HCO3', 0.41]
bromide = ['Bromide', 'Br', 0.19]
borate = ['Borate', 'B(OH)4', 0.08]
strontium = ['Strontium', 'Sr', 0.04]

```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]
```

Here we have a sample list (seawater) containing the 6 ions in seawater with the highest concentrations (Chloride, Sodium, Magnesium, Sulphate, Calcium, Potassium). Each item in the list has the ion name, the chemical symbol, and the percent of total ions (%). Below the sample list, lists containing the next 4 highest concentration ions are shown as well.

We will be using all of these lists to showcase common list functions.

List functions

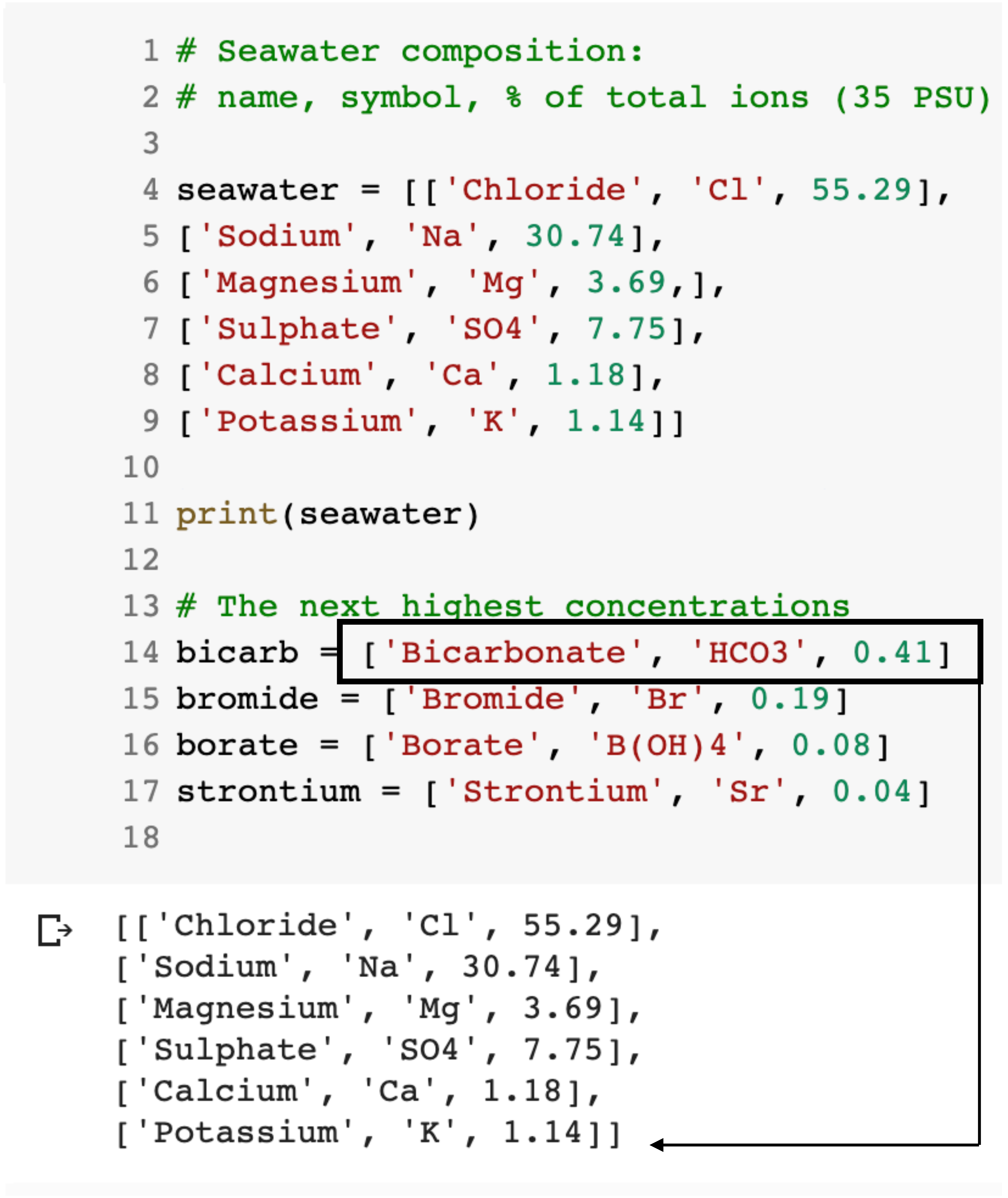


```
# Seawater composition:
# name, symbol, % of total ions (35 PSU)

seawater = [['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69,],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]

print(seawater)

# The next highest concentrations
bicarb = ['Bicarbonate', 'HCO3', 0.41]
bromide = ['Bromide', 'Br', 0.19]
borate = ['Borate', 'B(OH)4', 0.08]
strontium = ['Strontium', 'Sr', 0.04]

```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]
```

Adding items to a list:

List functions



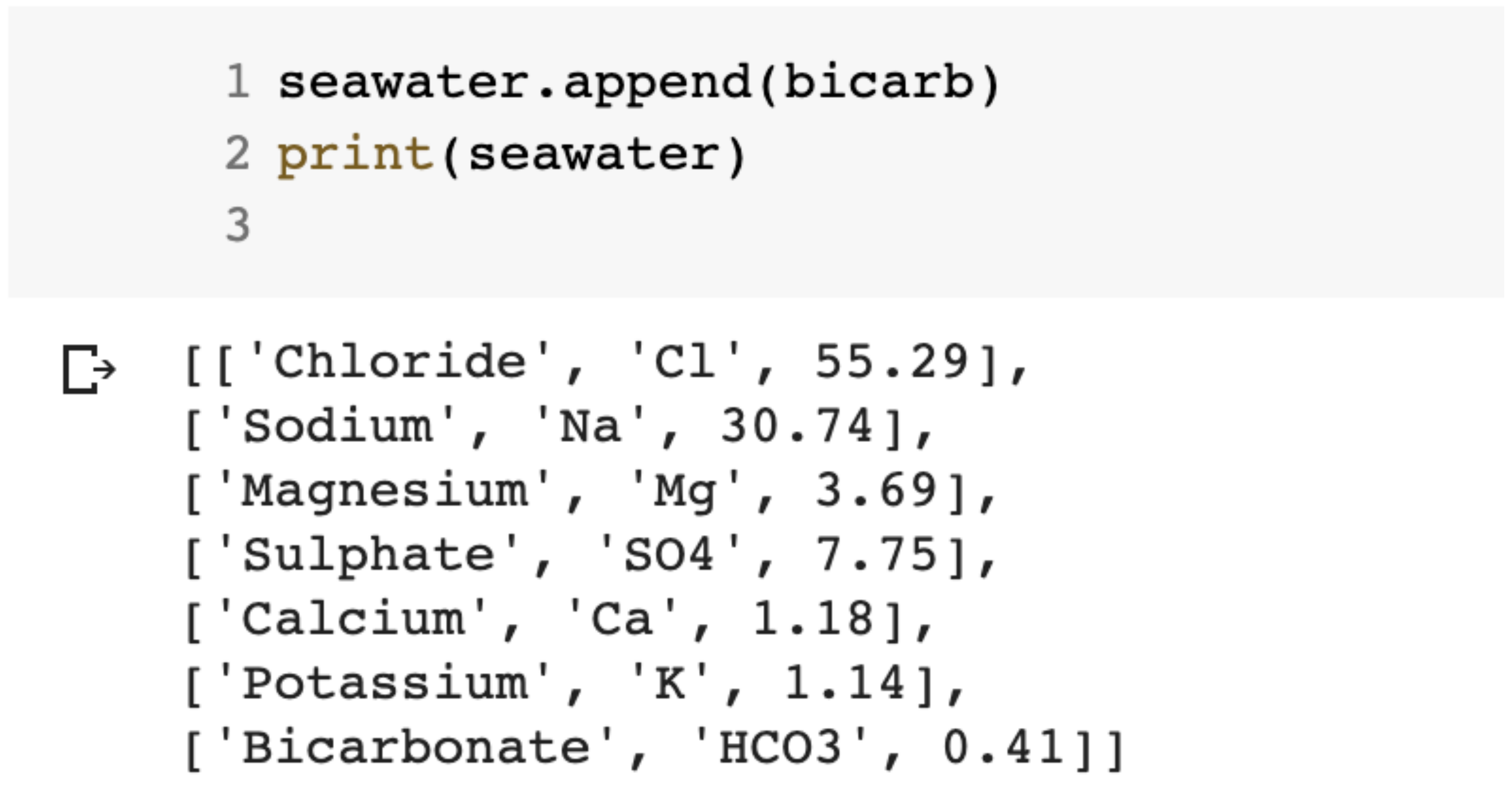
```python
# Seawater composition:
# name, symbol, % of total ions (35 PSU)

seawater = [['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69,],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]

print(seawater)

# The next highest concentrations
bicarb = ['Bicarbonate', 'HCO3', 0.41]
bromide = ['Bromide', 'Br', 0.19]
borate = ['Borate', 'B(OH)4', 0.08]
strontium = ['Strontium', 'Sr', 0.04]
```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]
```

Adding items to a list:

append( )

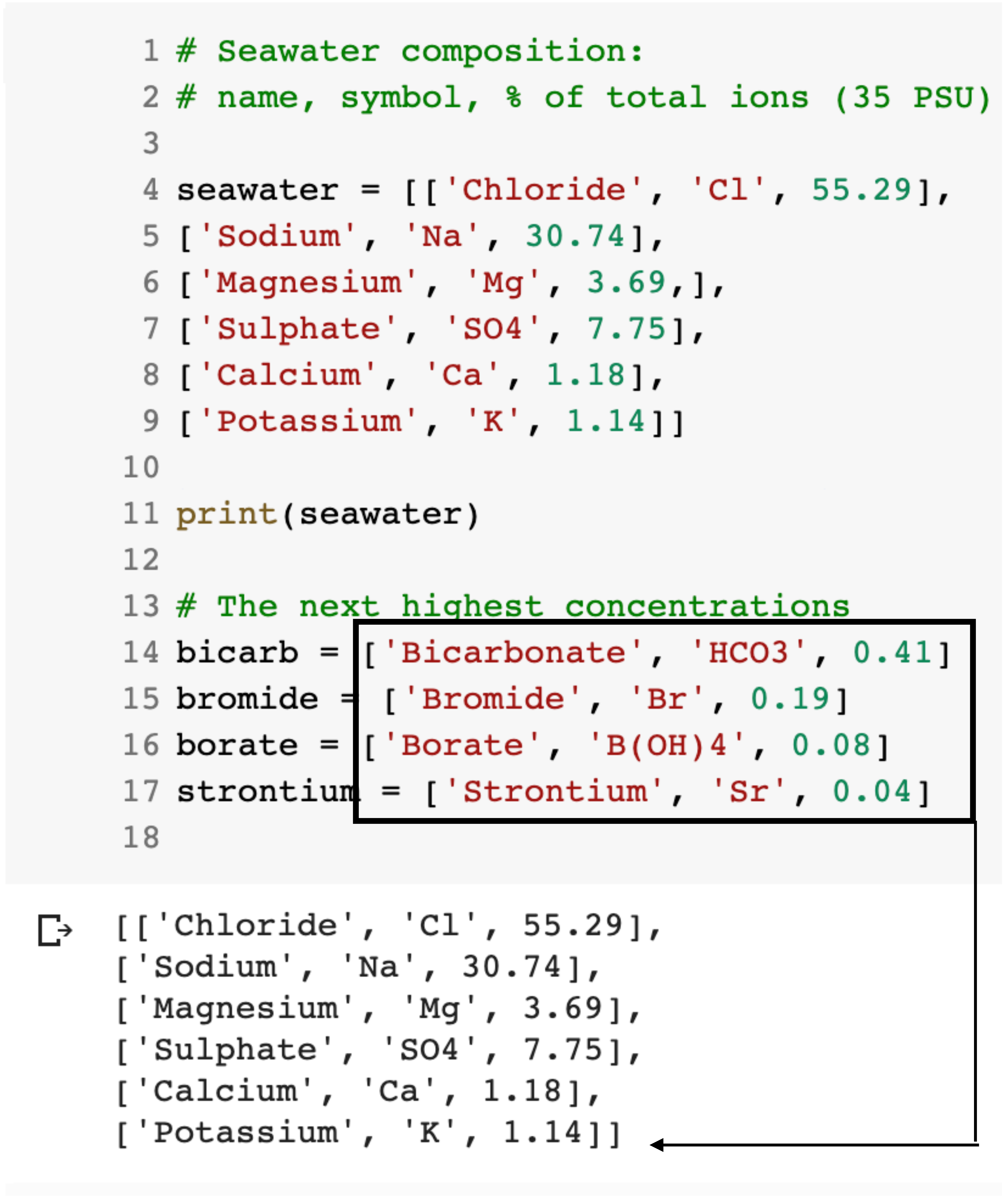
Adds a single item to the end of the list



```python
seawater.append(bicarb)
print(seawater)
```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14],
['Bicarbonate', 'HCO3', 0.41]]
```

List functions



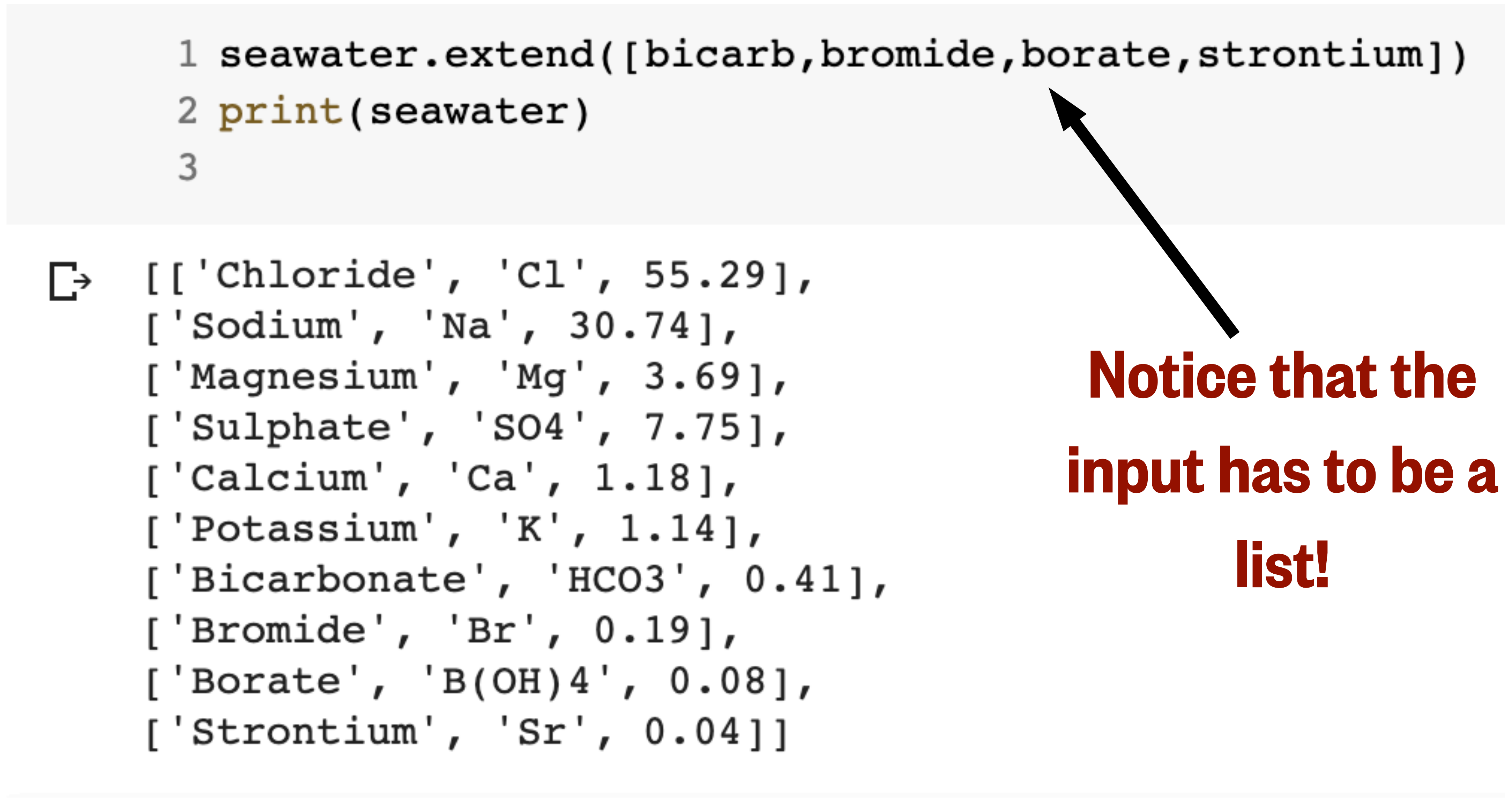
```python
# Seawater composition:
# name, symbol, % of total ions (35 PSU)

seawater = [['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69,],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]

print(seawater)

# The next highest concentrations
bicarb = ['Bicarbonate', 'HCO3', 0.41]
bromide = ['Bromide', 'Br', 0.19]
borate = ['Borate', 'B(OH)4', 0.08]
strontium = ['Strontium', 'Sr', 0.04]

```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]
```

Adding items to a list:

extend( )

Adds multiple items to the end of the list

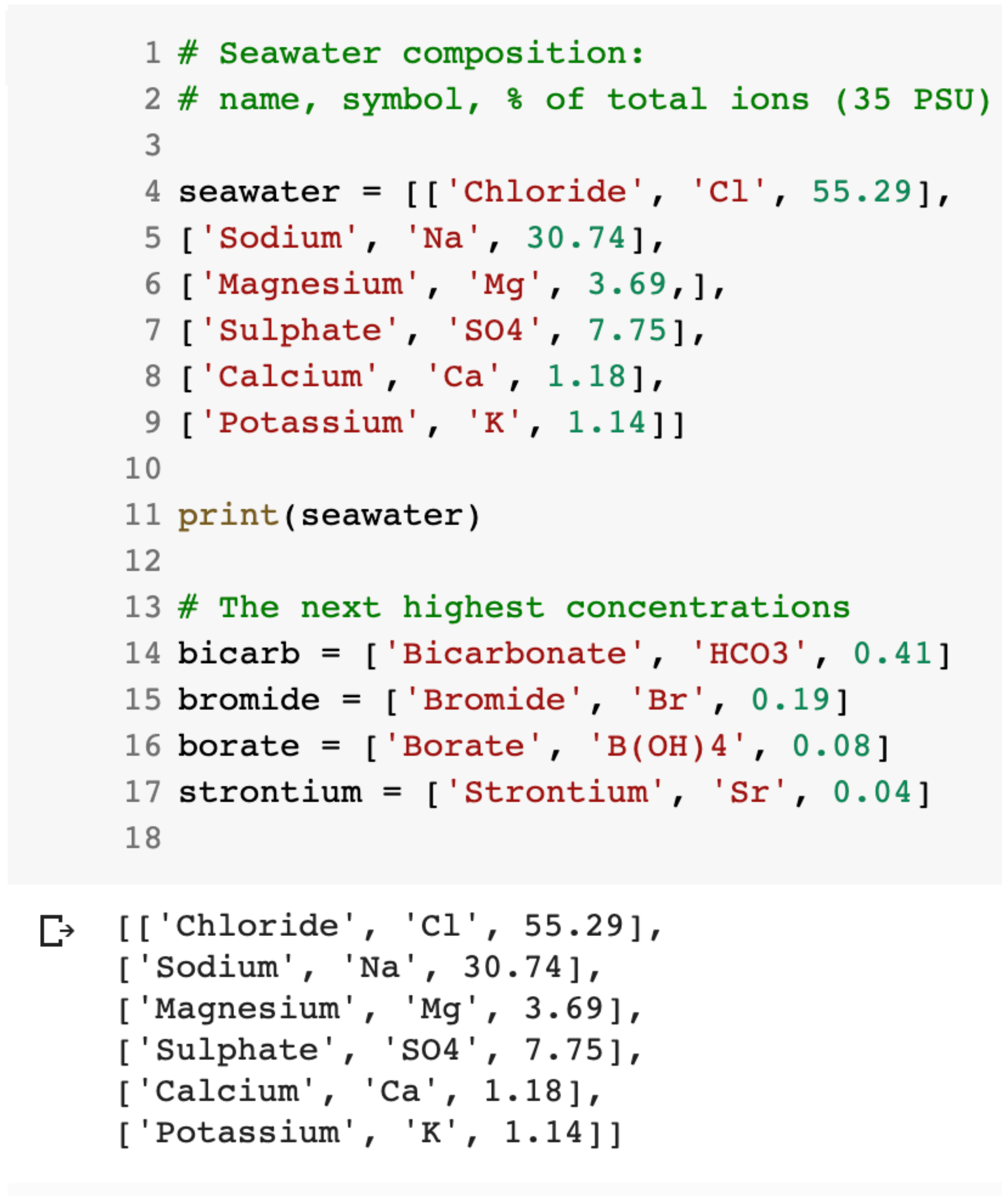


```python
seawater.extend([bicarb,bromide,borate,strontium])
print(seawater)

```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14],
['Bicarbonate', 'HCO3', 0.41],
['Bromide', 'Br', 0.19],
['Borate', 'B(OH)4', 0.08],
['Strontium', 'Sr', 0.04]]
```

Notice that the input has to be a list!

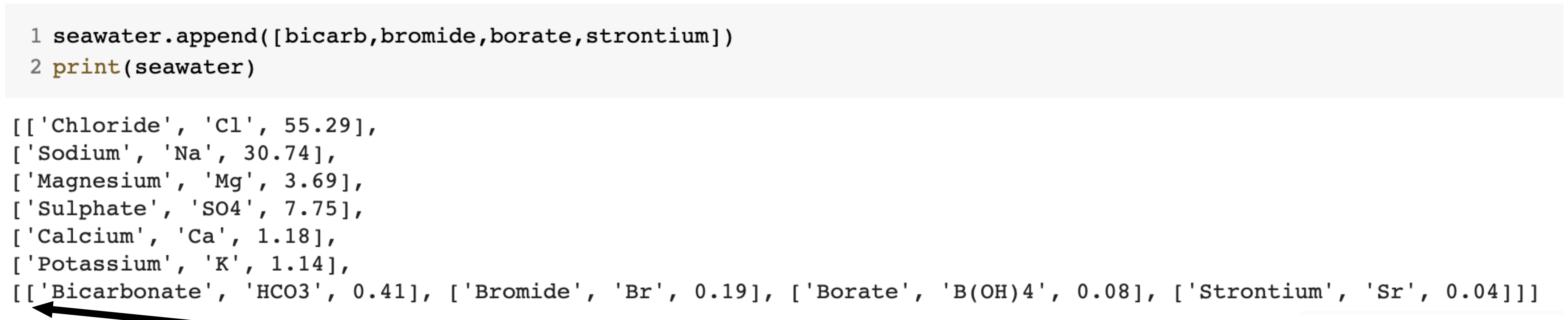
List functions



```
# Seawater composition:
# name, symbol, % of total ions (35 PSU)

seawater = [['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69,],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]

print(seawater)

# The next highest concentrations
bicarb = ['Bicarbonate', 'HCO3', 0.41]
bromide = ['Bromide', 'Br', 0.19]
borate = ['Borate', 'B(OH)4', 0.08]
strontium = ['Strontium', 'Sr', 0.04]

```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]
```

append( )

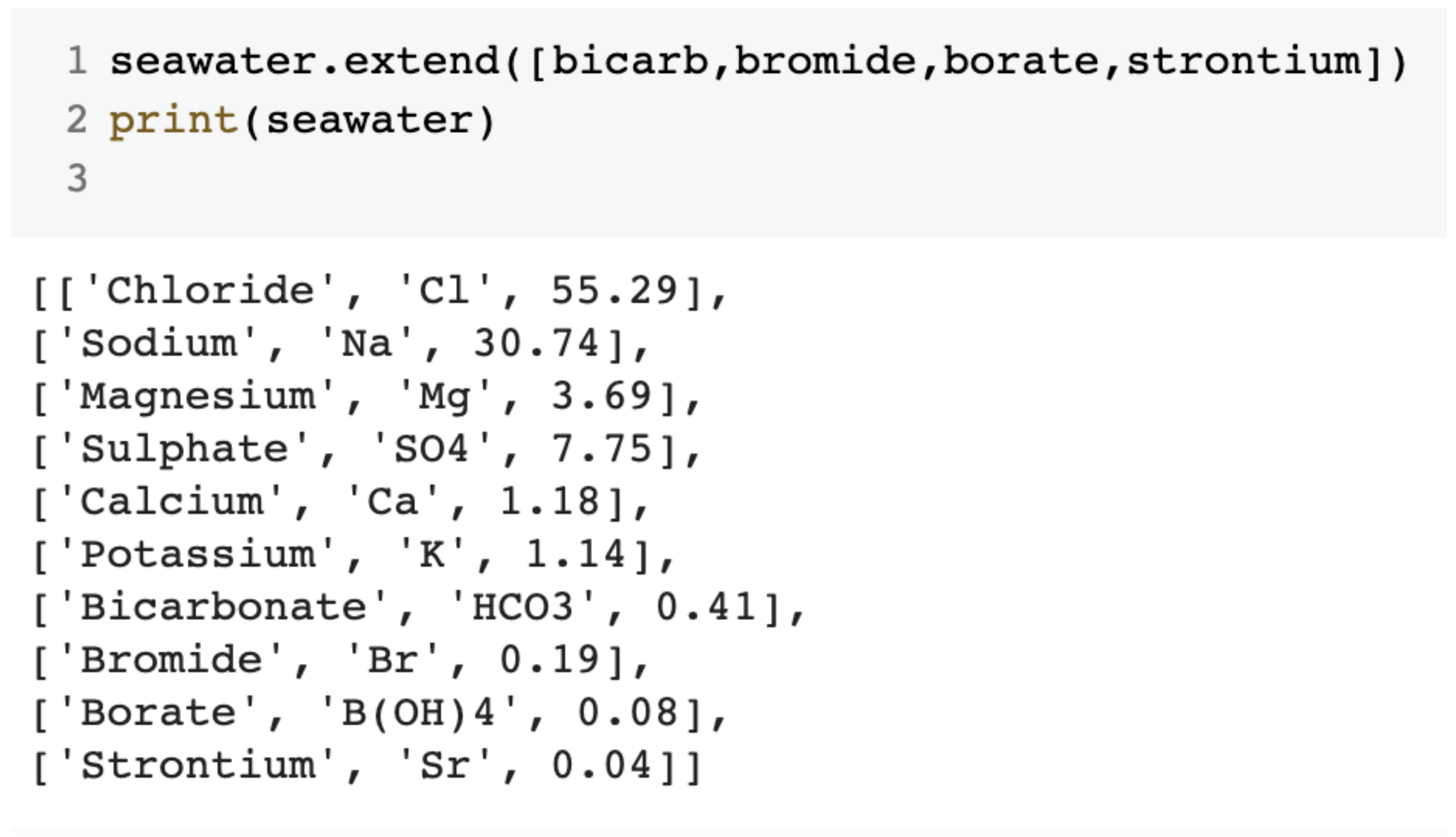


```
seawater.append([bicarb,bromide,borate,strontium])
print(seawater)
```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14],
[['Bicarbonate', 'HCO3', 0.41], ['Bromide', 'Br', 0.19], ['Borate', 'B(OH)4', 0.08], ['Strontium', 'Sr', 0.04]]]
```

Double brackets here mean that the appended list of concentrations was added as a single item. This is not what we want!

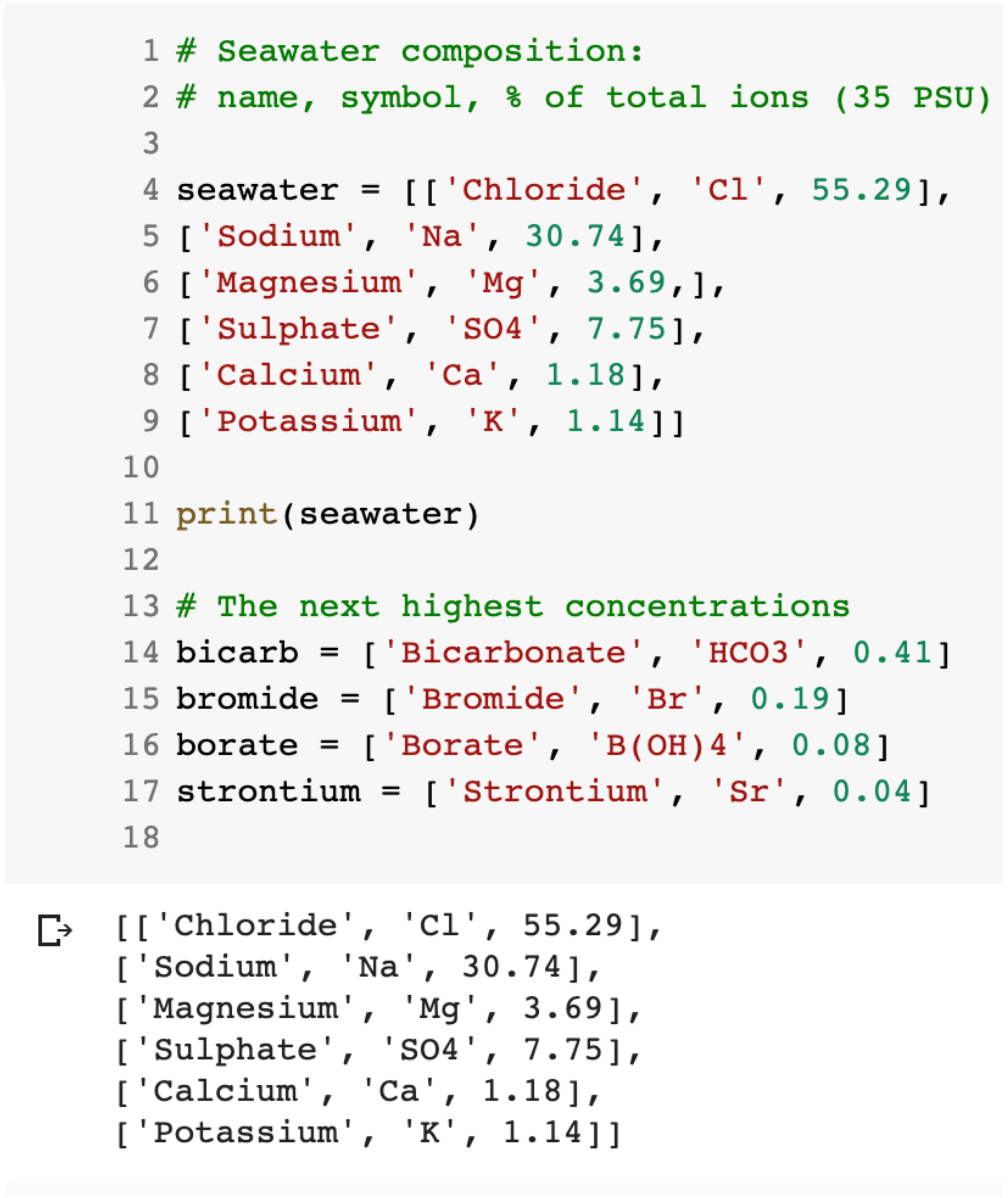
extend( )



```
seawater.extend([bicarb,bromide,borate,strontium])
print(seawater)

```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14],
['Bicarbonate', 'HCO3', 0.41],
['Bromide', 'Br', 0.19],
['Borate', 'B(OH)4', 0.08],
['Strontium', 'Sr', 0.04]]
```

List functions



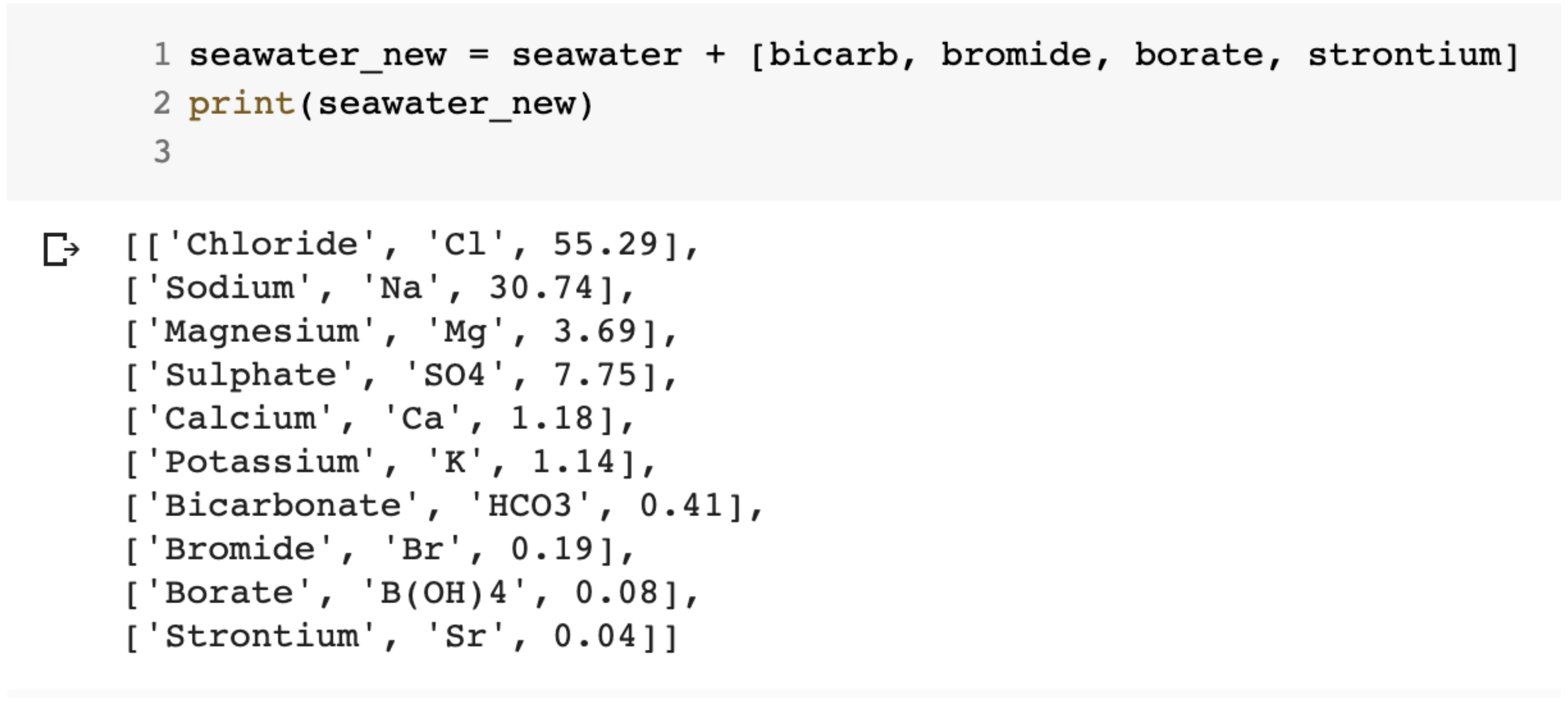
```python
# Seawater composition:
# name, symbol, % of total ions (35 PSU)

seawater = [['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69,],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]

print(seawater)

# The next highest concentrations
bicarb = ['Bicarbonate', 'HCO3', 0.41]
bromide = ['Bromide', 'Br', 0.19]
borate = ['Borate', 'B(OH)4', 0.08]
strontium = ['Strontium', 'Sr', 0.04]

```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]
```

Adding items to a list:

+

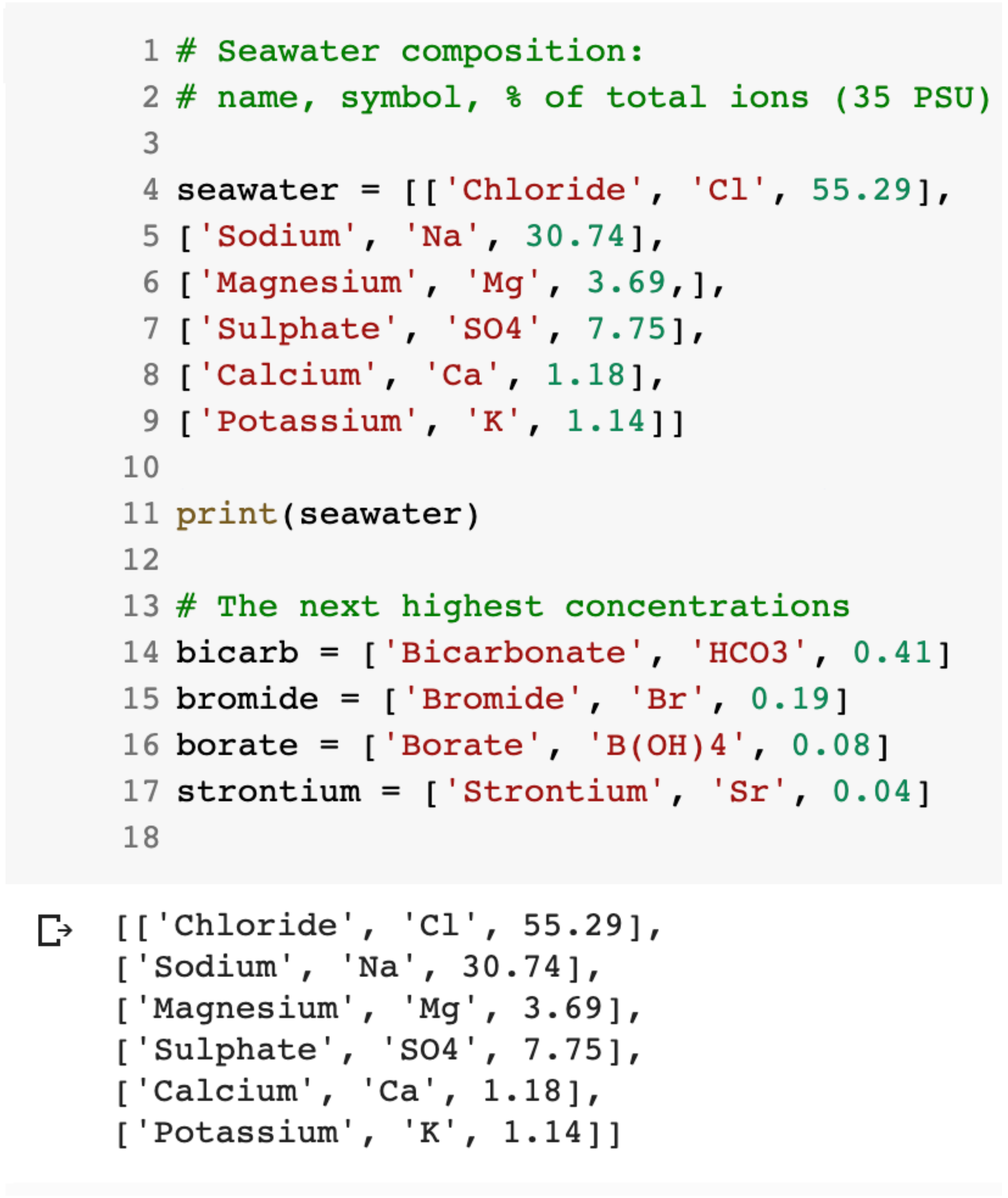
Concatenates the items from two lists



```python
seawater_new = seawater + [bicarb, bromide, borate, strontium]
print(seawater_new)

```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14],
['Bicarbonate', 'HCO3', 0.41],
['Bromide', 'Br', 0.19],
['Borate', 'B(OH)4', 0.08],
['Strontium', 'Sr', 0.04]]
```

List functions



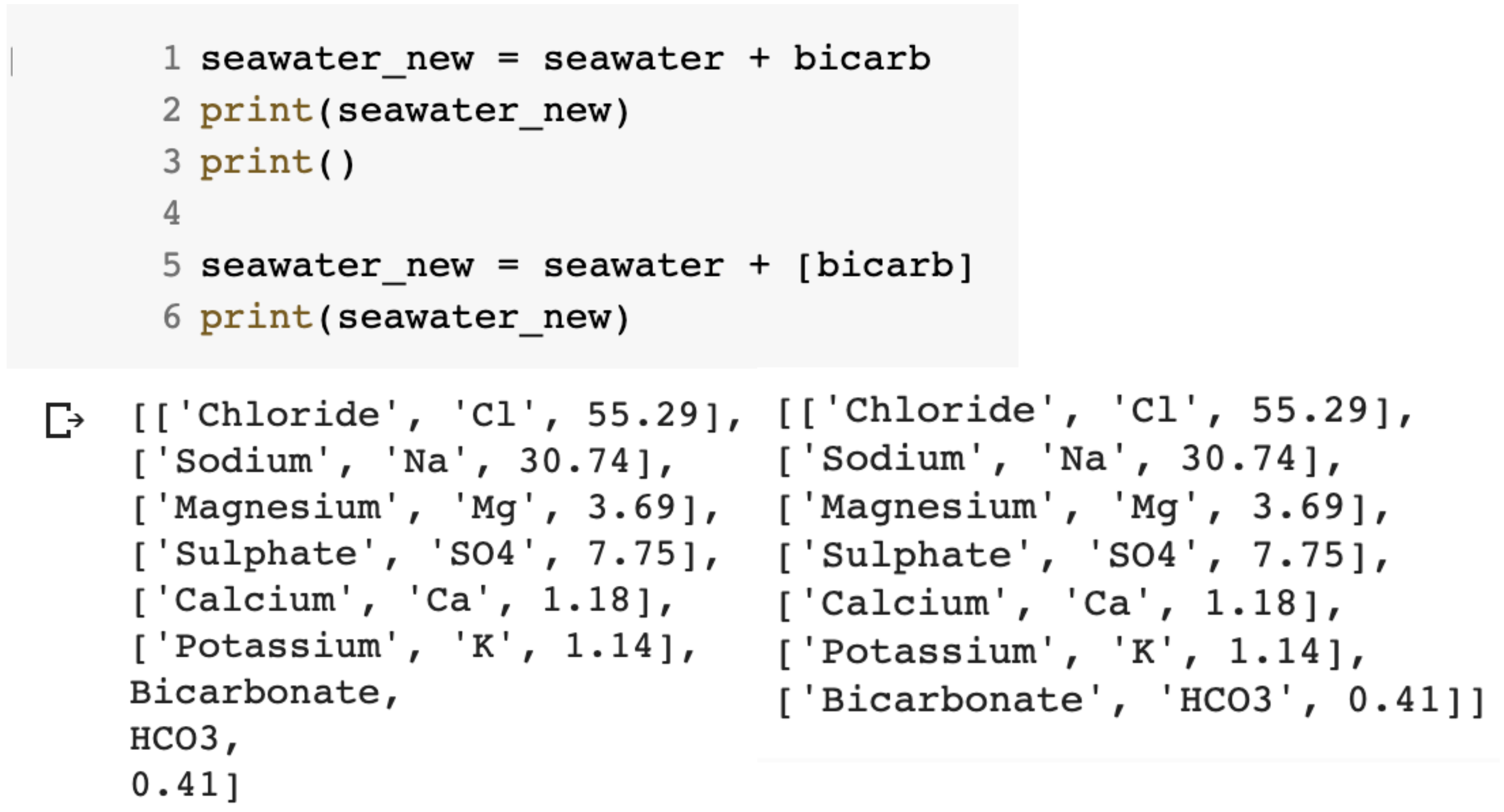
```python
# Seawater composition:
# name, symbol, % of total ions (35 PSU)

seawater = [['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69,],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]

print(seawater)

# The next highest concentrations
bicarb = ['Bicarbonate', 'HCO3', 0.41]
bromide = ['Bromide', 'Br', 0.19]
borate = ['Borate', 'B(OH)4', 0.08]
strontium = ['Strontium', 'Sr', 0.04]

```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]
```

Adding items to a list:

+

Concatenates the items from two lists

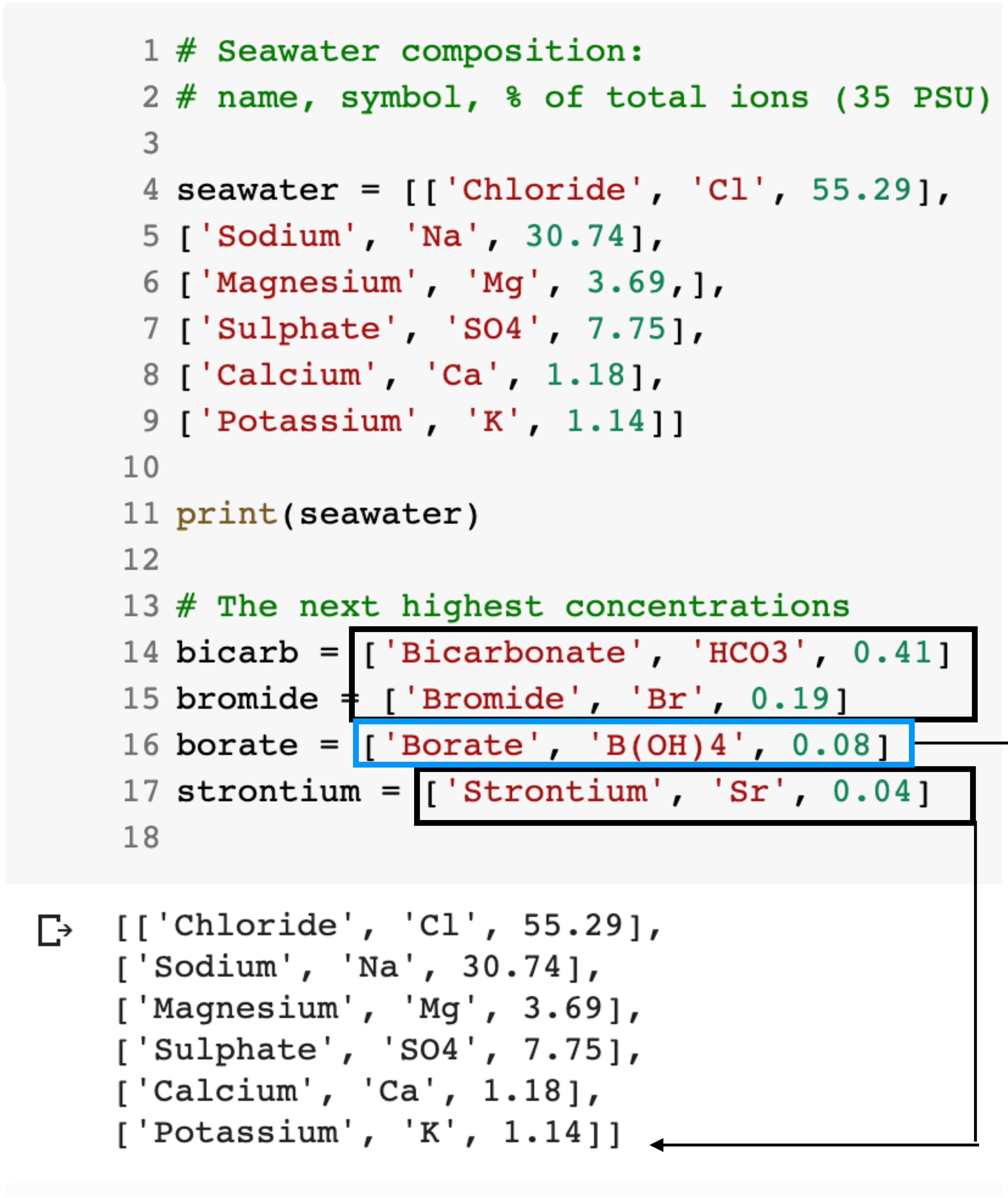


```python
seawater_new = seawater + bicarb
print(seawater_new)
print()

seawater_new = seawater + [bicarb]
print(seawater_new)
```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14],
Bicarbonate,
HCO3,
0.41]
```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14],
['Bicarbonate', 'HCO3', 0.41]]
```

List functions



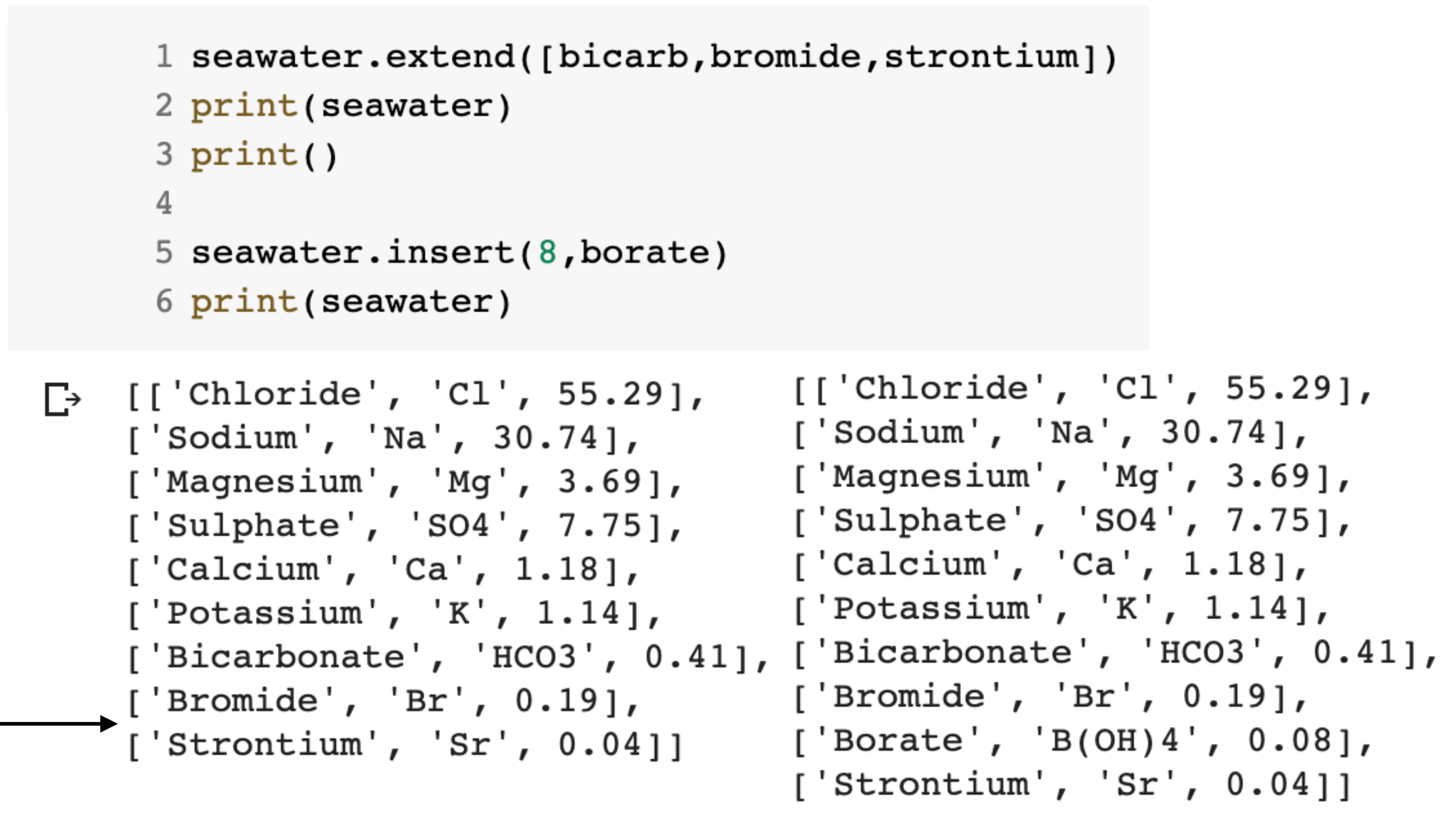
```python
# Seawater composition:
# name, symbol, % of total ions (35 PSU)

seawater = [['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69,],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]

print(seawater)

# The next highest concentrations
bicarb = ['Bicarbonate', 'HCO3', 0.41]
bromide = ['Bromide', 'Br', 0.19]
borate = ['Borate', 'B(OH)4', 0.08]
strontium = ['Strontium', 'Sr', 0.04]

```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]
```

Adding items to a list:

insert( )

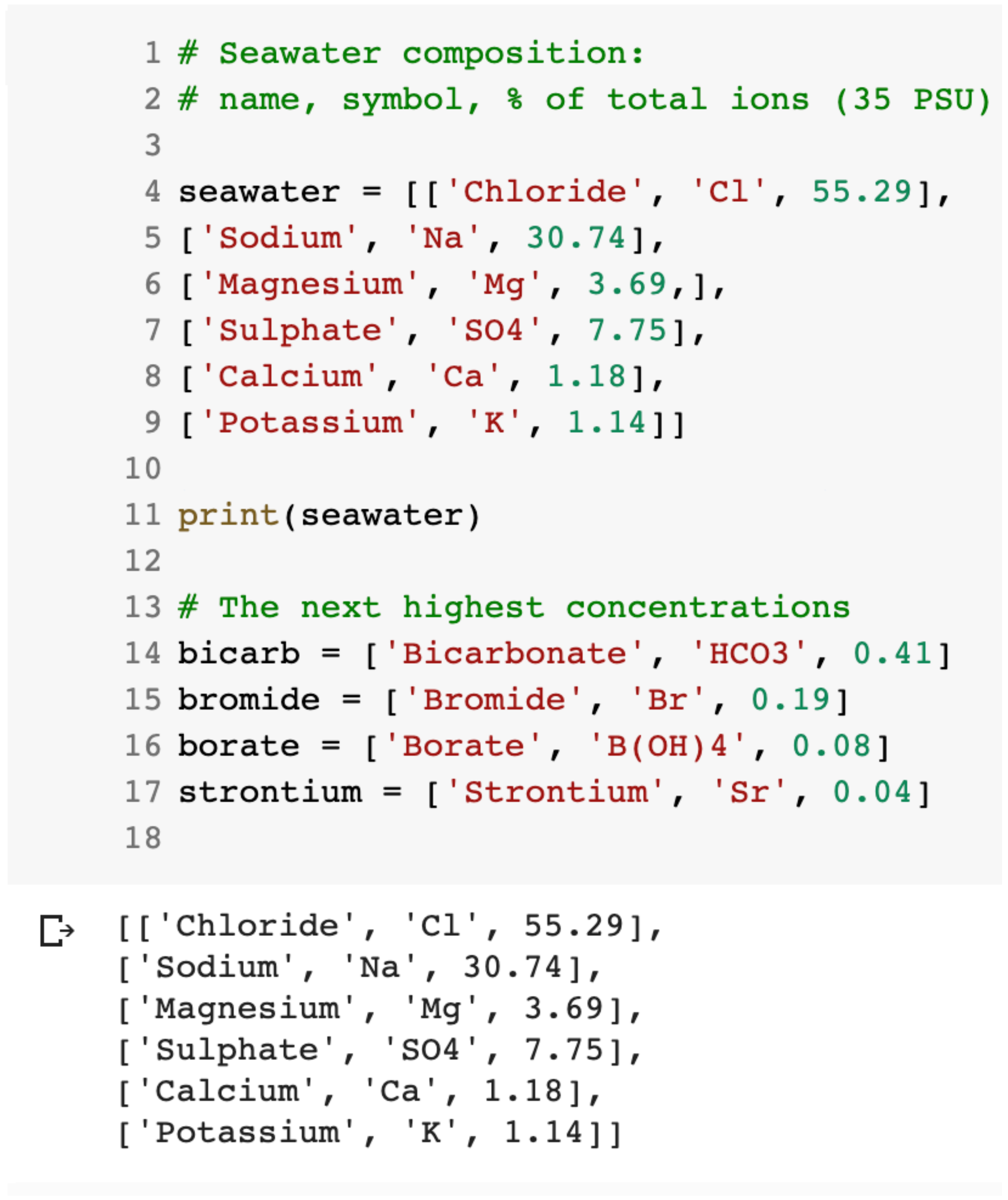
Adds a single item to a given index in the list



```python
seawater.extend([bicarb,bromide,strontium])
print(seawater)
print()

seawater.insert(8,borate)
print(seawater)
```

```
[['Chloride', 'Cl', 55.29],        [['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],           ['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],         ['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],         ['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],           ['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14],          ['Potassium', 'K', 1.14],
['Bicarbonate', 'HCO3', 0.41],     ['Bicarbonate', 'HCO3', 0.41],
['Bromide', 'Br', 0.19],           ['Bromide', 'Br', 0.19],
['Strontium', 'Sr', 0.04]]         ['Borate', 'B(OH)4', 0.08],
                                   ['Strontium', 'Sr', 0.04]]
```

List functions

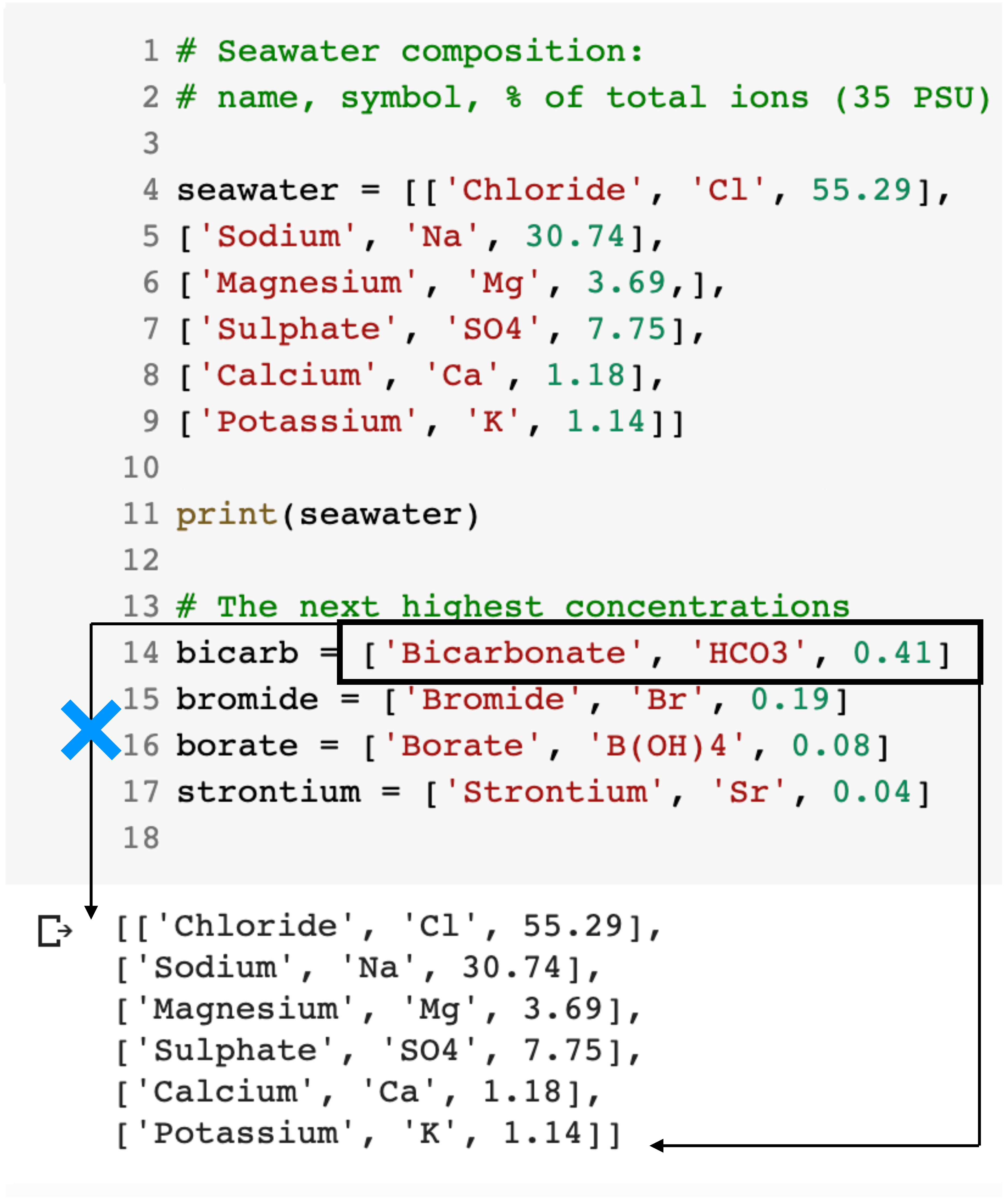


```python
# Seawater composition:
# name, symbol, % of total ions (35 PSU)

seawater = [['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69,],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]

print(seawater)

# The next highest concentrations
bicarb = ['Bicarbonate', 'HCO3', 0.41]
bromide = ['Bromide', 'Br', 0.19]
borate = ['Borate', 'B(OH)4', 0.08]
strontium = ['Strontium', 'Sr', 0.04]

```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]
```

Removing items from a list:

List functions



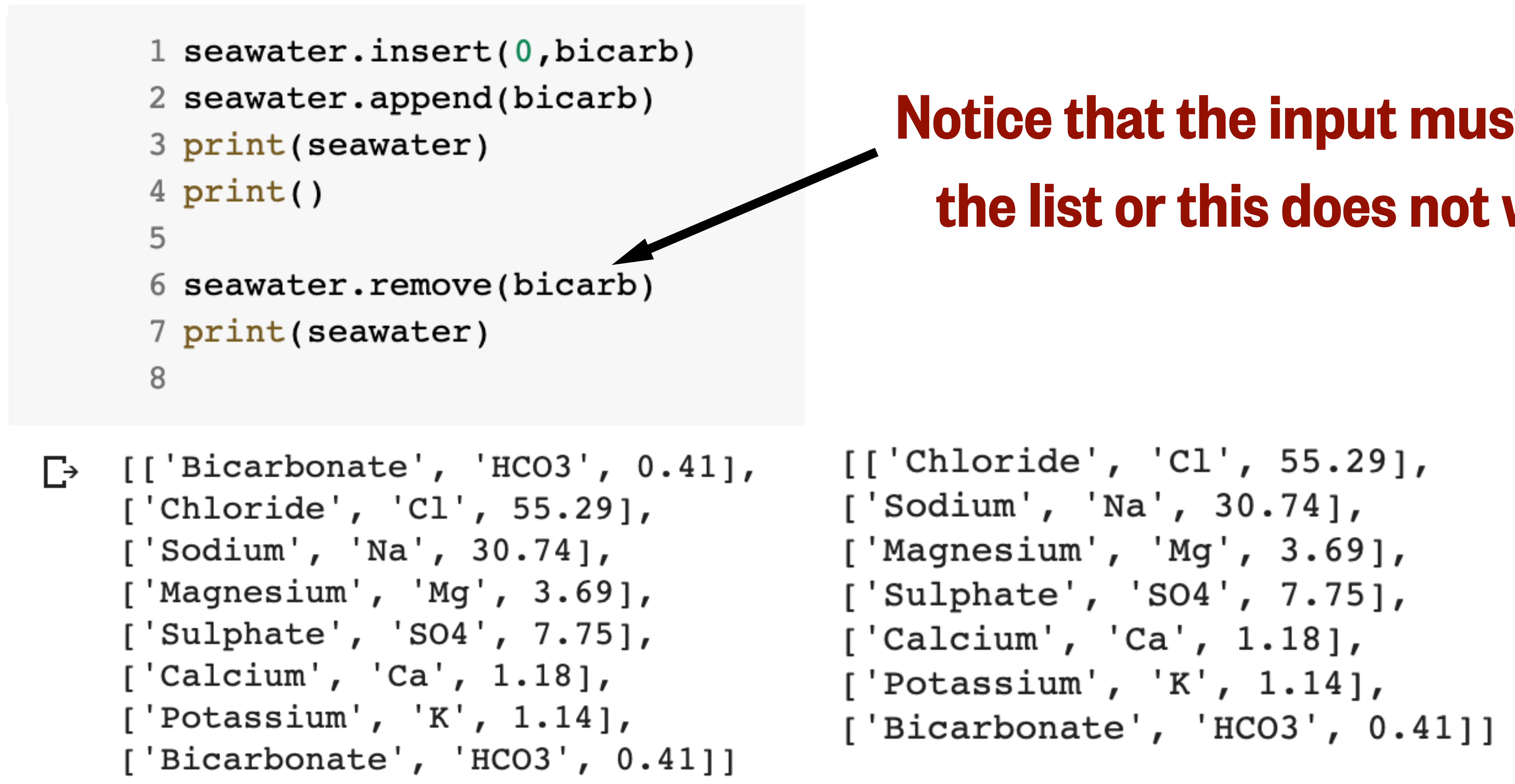
```python
# Seawater composition:
# name, symbol, % of total ions (35 PSU)

seawater = [['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69,],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]

print(seawater)

# The next highest concentrations
bicarb = ['Bicarbonate', 'HCO3', 0.41]
bromide = ['Bromide', 'Br', 0.19]
borate = ['Borate', 'B(OH)4', 0.08]
strontium = ['Strontium', 'Sr', 0.04]

```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]
```

Removing items from a list:

remove( )

Deletes the first occurrence of a given item



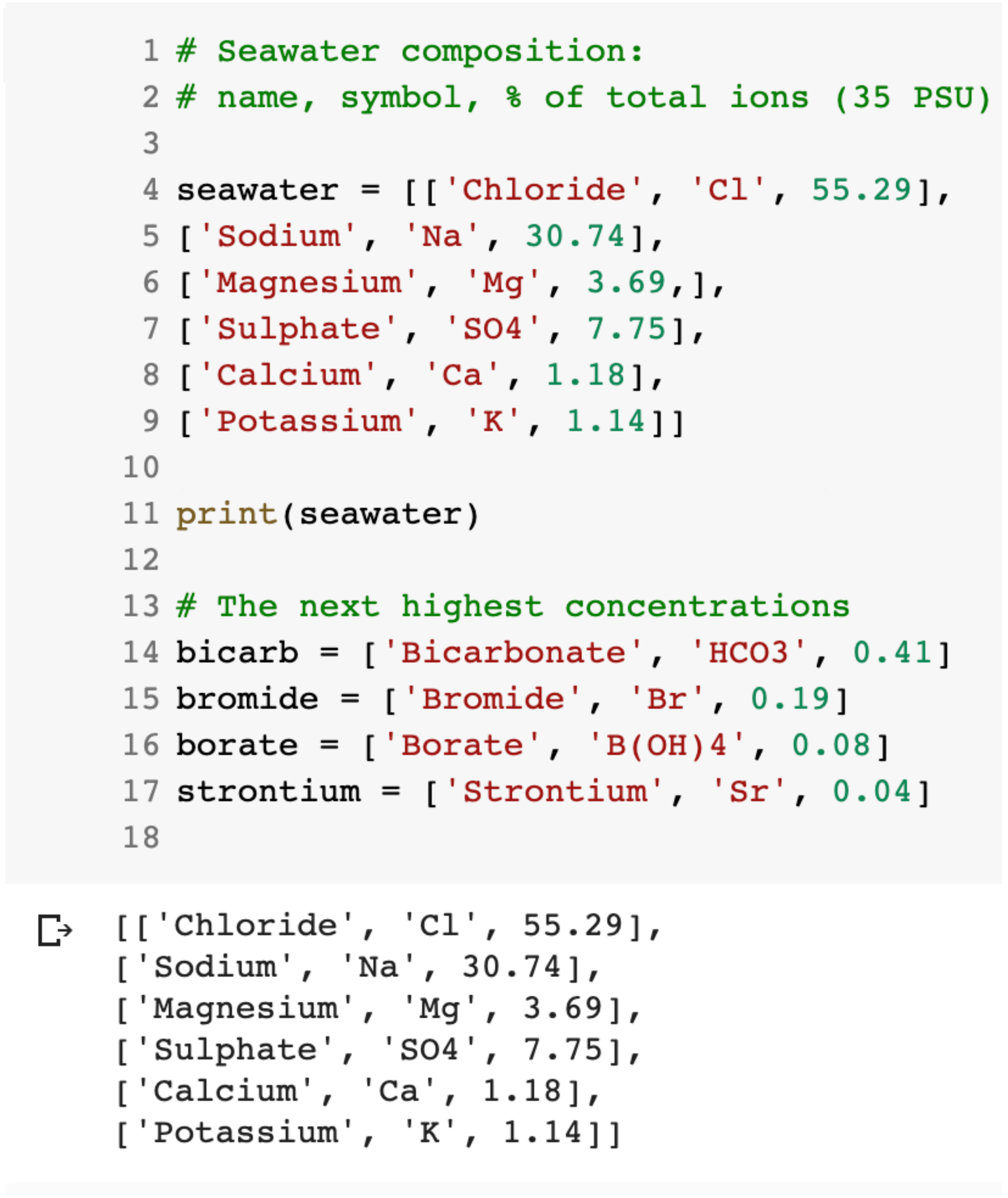
```python
seawater.insert(0,bicarb)
seawater.append(bicarb)
print(seawater)
print()

seawater.remove(bicarb)
print(seawater)

```

```
[['Bicarbonate', 'HCO3', 0.41],
['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14],
['Bicarbonate', 'HCO3', 0.41]]
```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14],
['Bicarbonate', 'HCO3', 0.41]]
```

Notice that the input must be in the list or this does not work

List functions



```
# Seawater composition:
# name, symbol, % of total ions (35 PSU)

seawater = [['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69,],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]

print(seawater)

# The next highest concentrations
bicarb = ['Bicarbonate', 'HCO3', 0.41]
bromide = ['Bromide', 'Br', 0.19]
borate = ['Borate', 'B(OH)4', 0.08]
strontium = ['Strontium', 'Sr', 0.04]

```

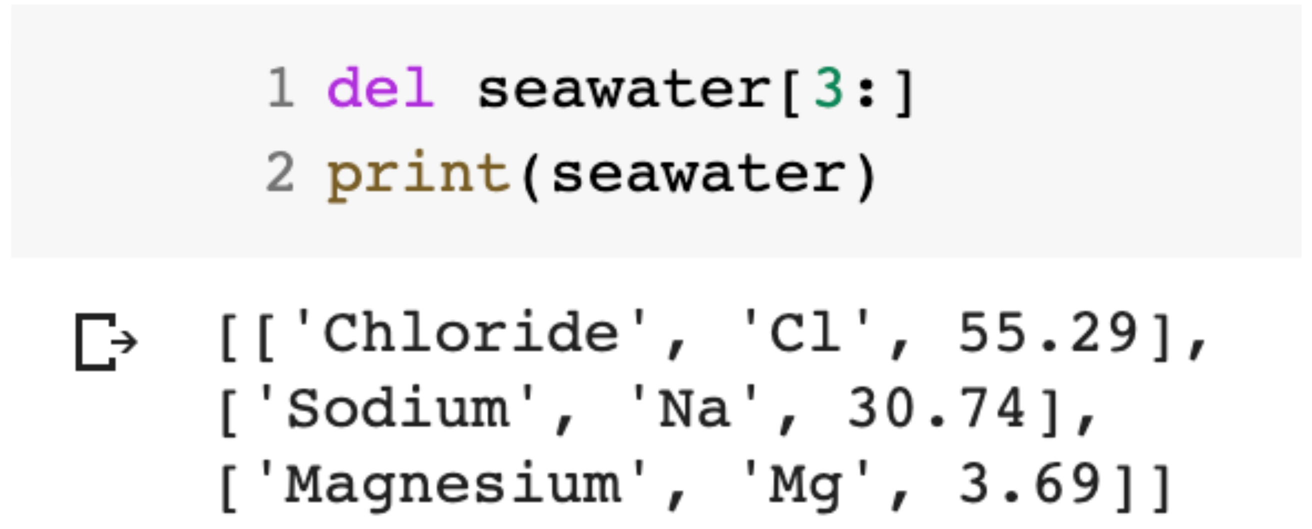
```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]
```

Removing items from a list:

del

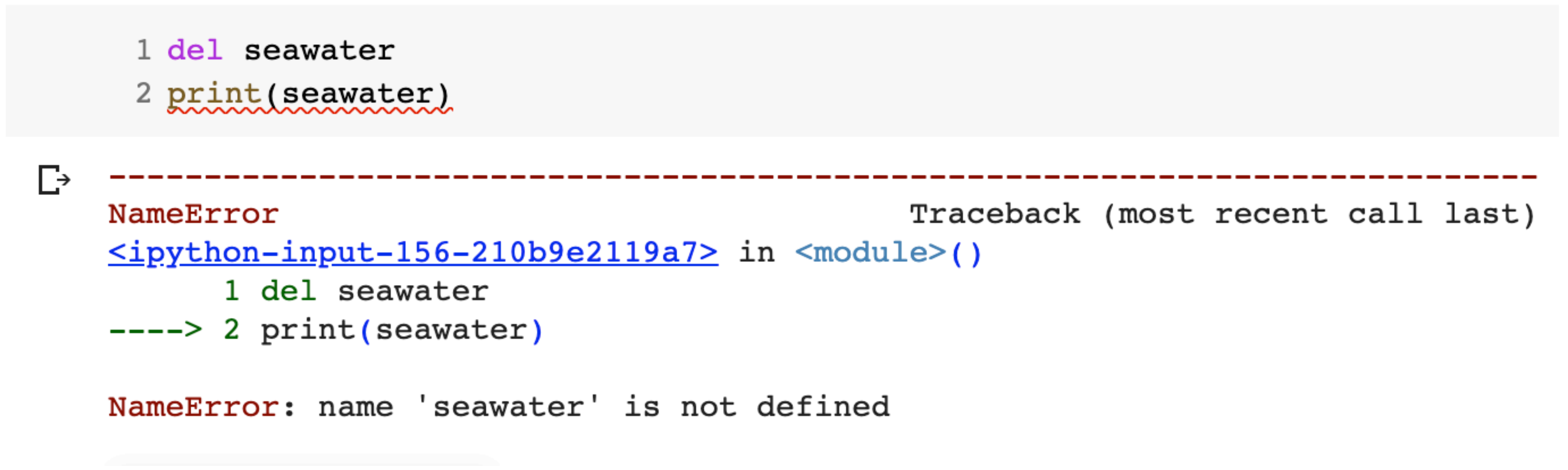
Deletes the items in a given index.

Can also delete the whole list (and any other objects)!



```
del seawater[3:]
print(seawater)
```

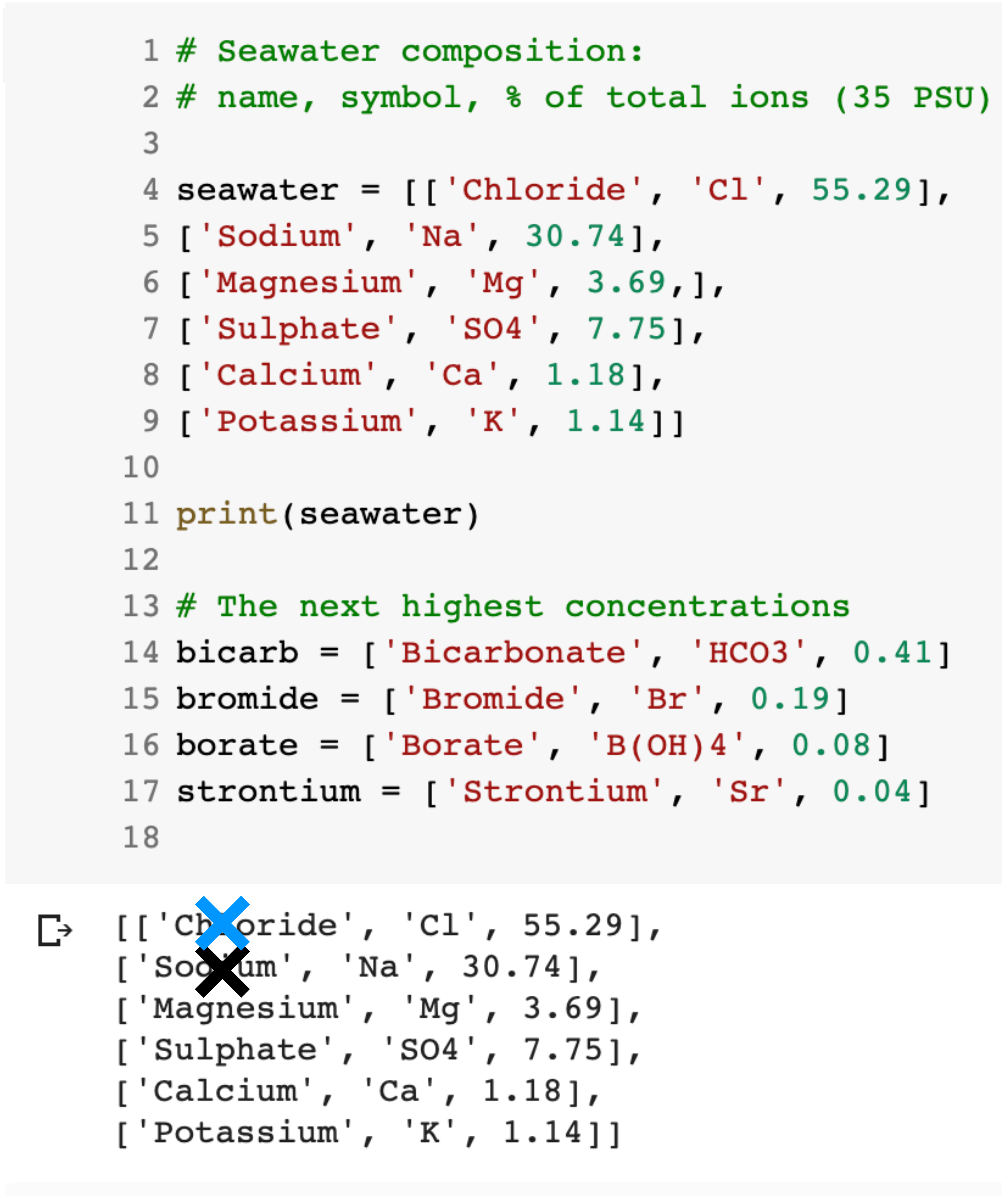
```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69]]
```



```
del seawater
print(seawater)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-156-210b9e2119a7> in <module>()
      1 del seawater
----> 2 print(seawater)

NameError: name 'seawater' is not defined
```

List functions



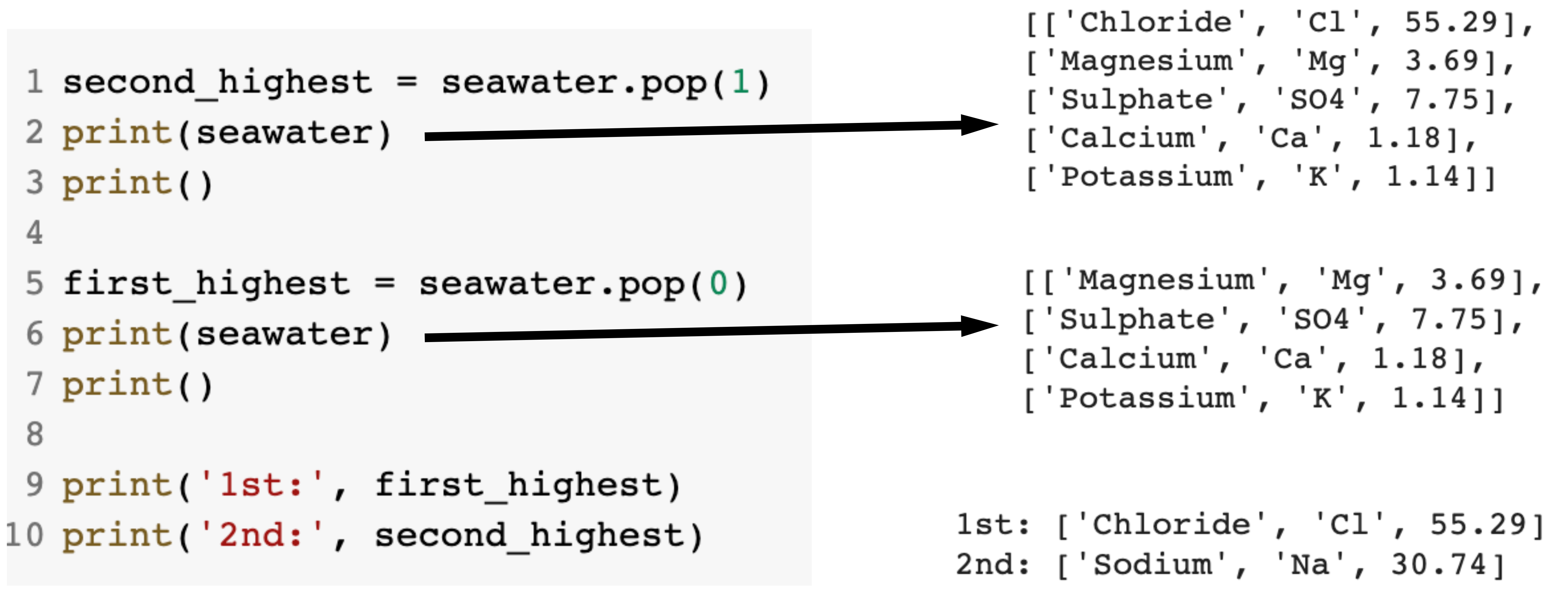
```python
# Seawater composition:
# name, symbol, % of total ions (35 PSU)

seawater = [['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69,],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]

print(seawater)

# The next highest concentrations
bicarb = ['Bicarbonate', 'HCO3', 0.41]
bromide = ['Bromide', 'Br', 0.19]
borate = ['Borate', 'B(OH)4', 0.08]
strontium = ['Strontium', 'Sr', 0.04]

```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]
```

Removing items from a list:

pop( )

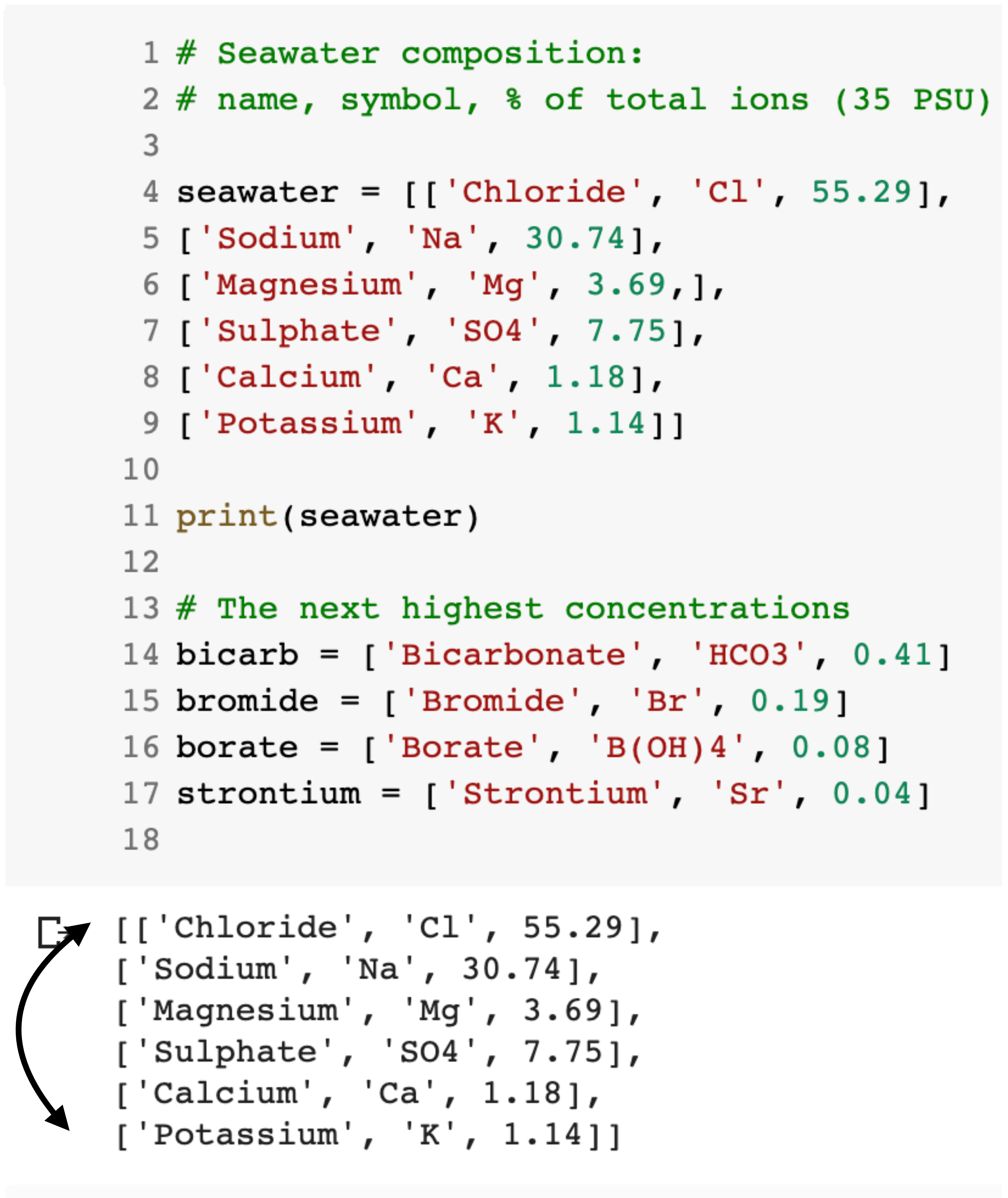
Removes and outputs an indexed item (default= -1)



```python
second_highest = seawater.pop(1)
print(seawater)
print()

first_highest = seawater.pop(0)
print(seawater)
print()

print('1st:', first_highest)
print('2nd:', second_highest)
```

```
[['Chloride', 'Cl', 55.29],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]

[['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]

1st: ['Chloride', 'Cl', 55.29]
2nd: ['Sodium', 'Na', 30.74]
```

List functions



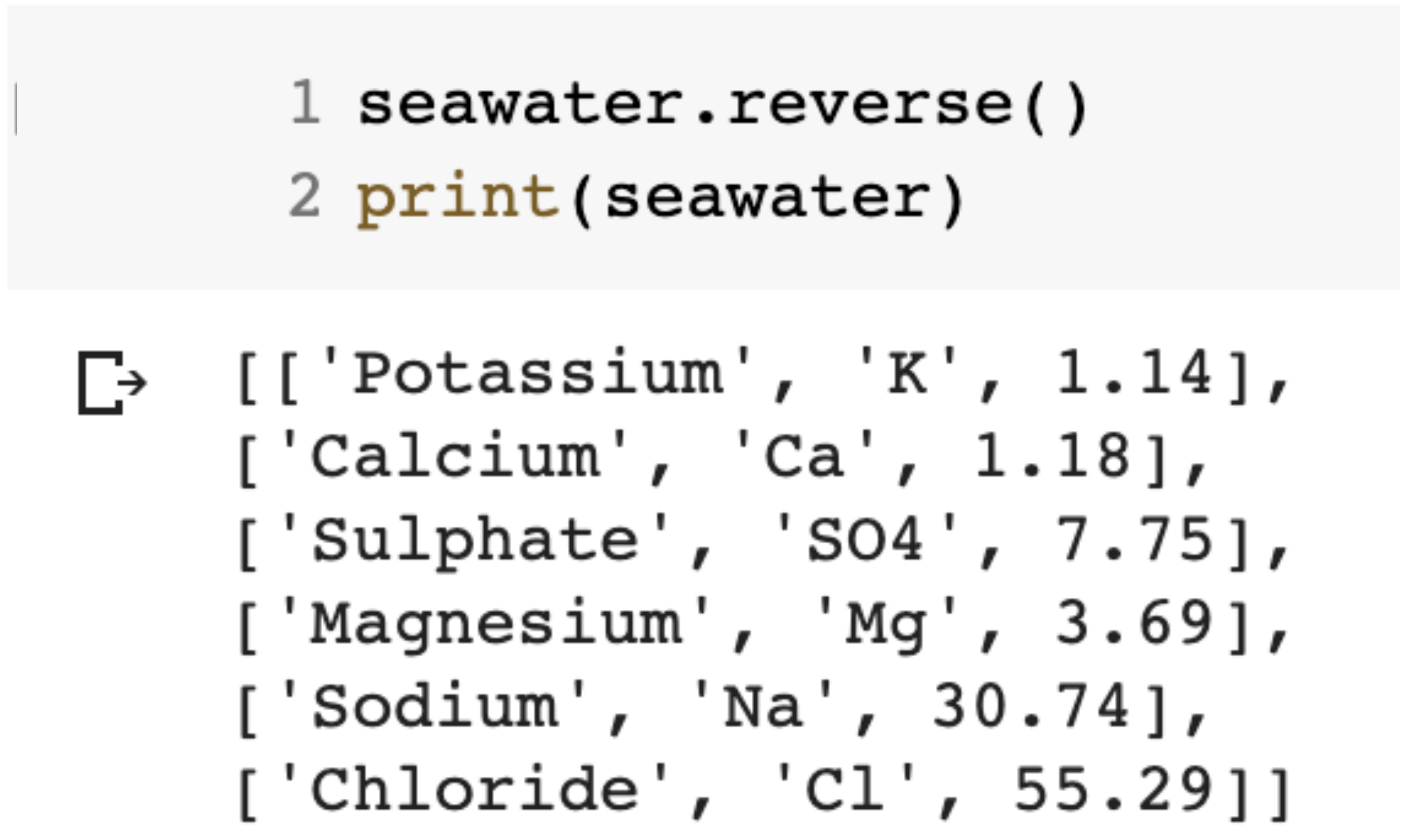
```python
# Seawater composition:
# name, symbol, % of total ions (35 PSU)

seawater = [['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69,],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]

print(seawater)

# The next highest concentrations
bicarb = ['Bicarbonate', 'HCO3', 0.41]
bromide = ['Bromide', 'Br', 0.19]
borate = ['Borate', 'B(OH)4', 0.08]
strontium = ['Strontium', 'Sr', 0.04]
```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]
```

Reversing a list:

reverse( )

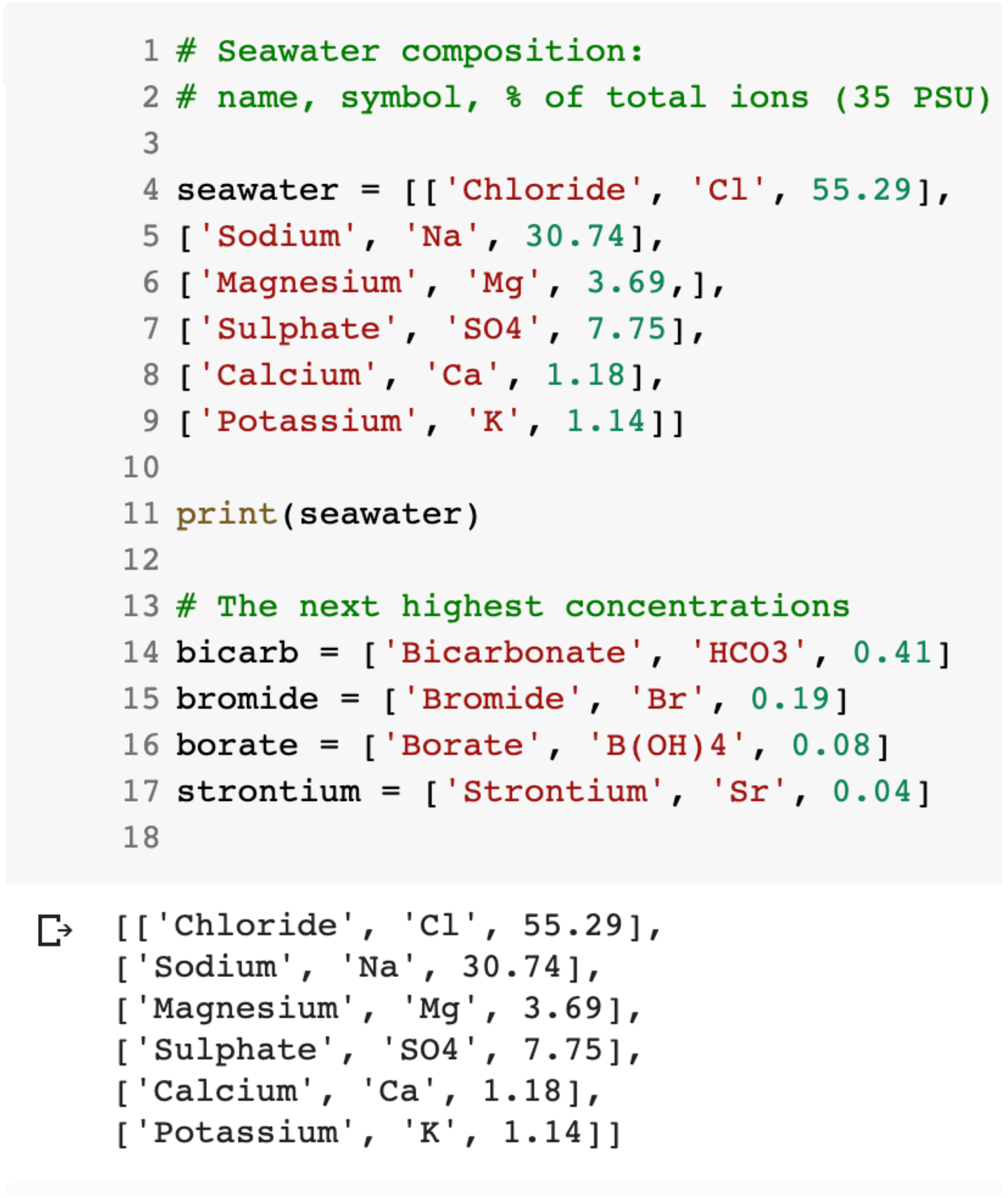
Reverses the order that a list is in



```python
seawater.reverse()
print(seawater)
```

```
[['Potassium', 'K', 1.14],
['Calcium', 'Ca', 1.18],
['Sulphate', 'SO4', 7.75],
['Magnesium', 'Mg', 3.69],
['Sodium', 'Na', 30.74],
['Chloride', 'Cl', 55.29]]
```

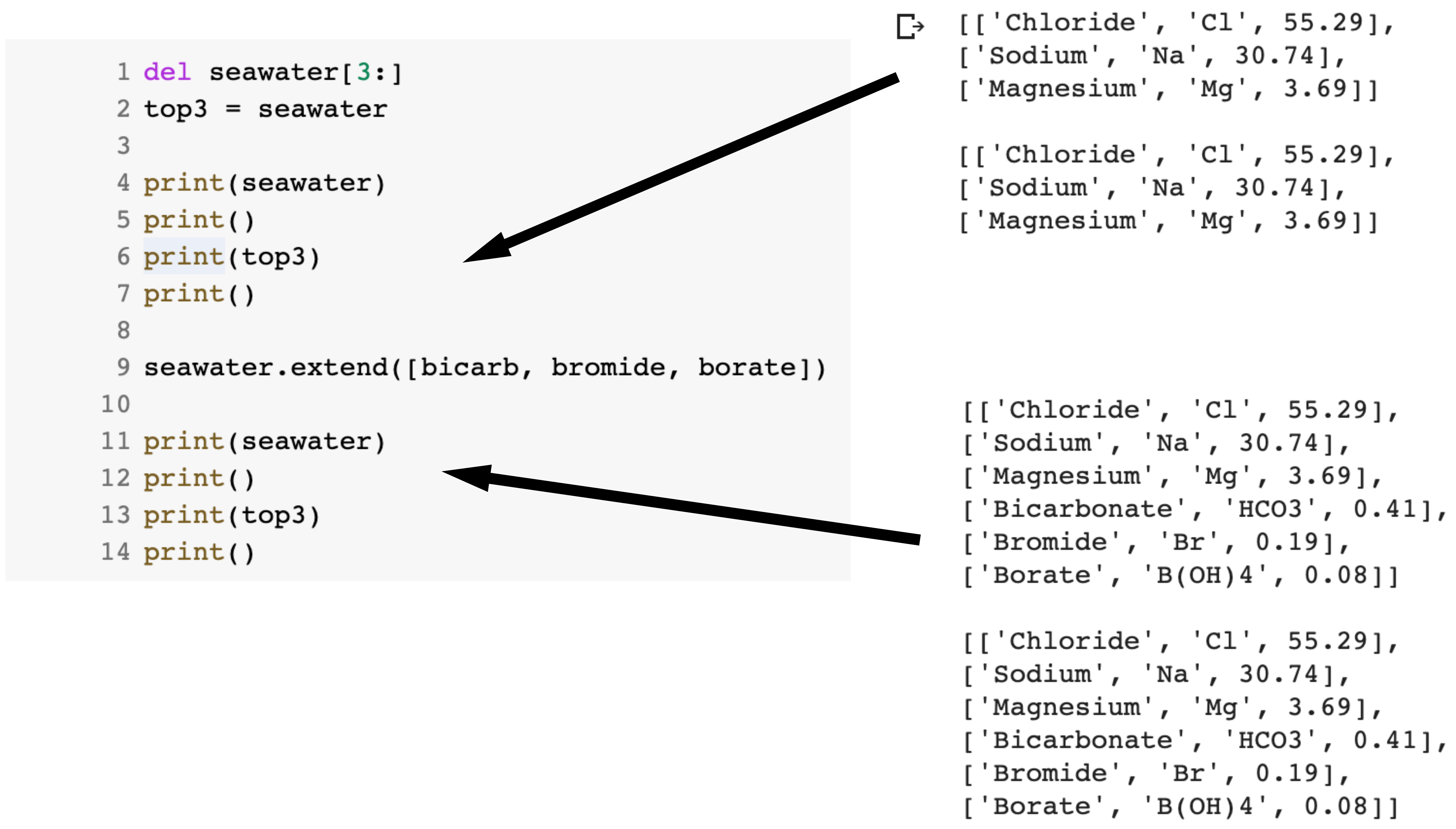
List functions



```
# Seawater composition:
# name, symbol, % of total ions (35 PSU)

seawater = [['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69,],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]

print(seawater)

# The next highest concentrations
bicarb = ['Bicarbonate', 'HCO3', 0.41]
bromide = ['Bromide', 'Br', 0.19]
borate = ['Borate', 'B(OH)4', 0.08]
strontium = ['Strontium', 'Sr', 0.04]

```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]
```

Setting a variable equal to a list and then changing the list, changes the variable too.

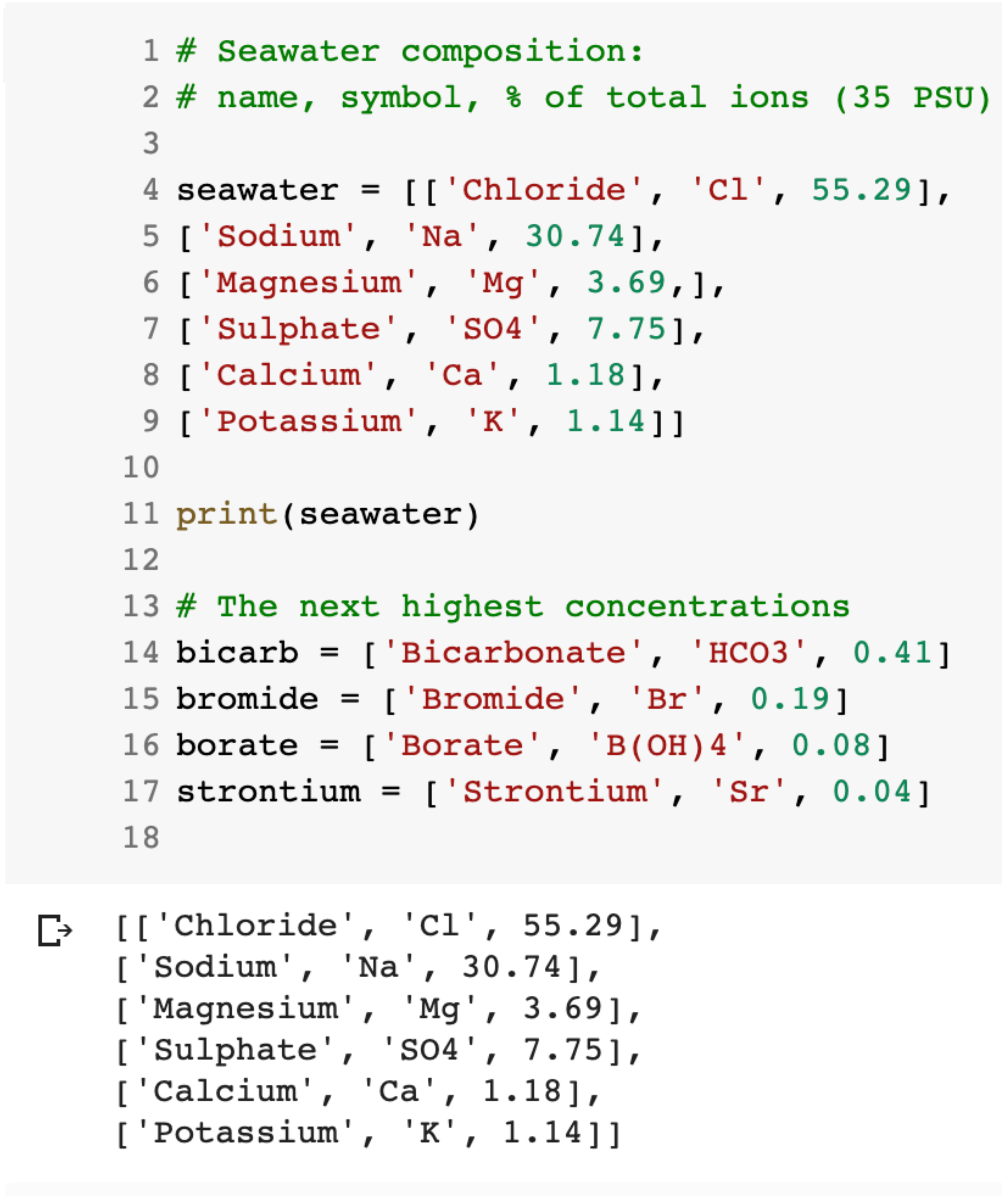


```
del seawater[3:]
top3 = seawater

print(seawater)
print()
print(top3)
print()

seawater.extend([bicarb, bromide, borate])

print(seawater)
print()
print(top3)
print()
```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69]]

[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69]]
```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Bicarbonate', 'HCO3', 0.41],
['Bromide', 'Br', 0.19],
['Borate', 'B(OH)4', 0.08]]

[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Bicarbonate', 'HCO3', 0.41],
['Bromide', 'Br', 0.19],
['Borate', 'B(OH)4', 0.08]]
```

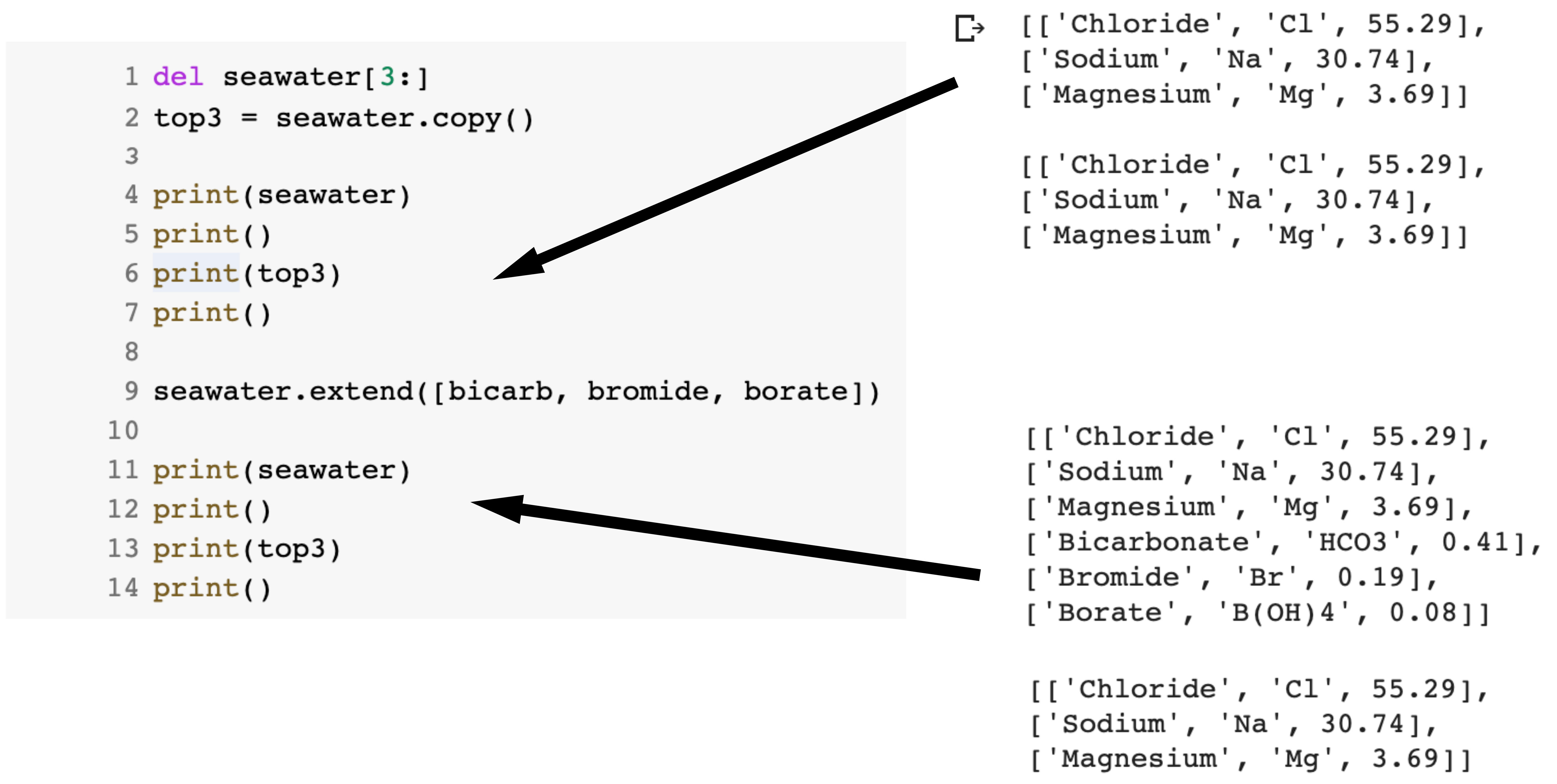
List functions



```python
# Seawater composition:
# name, symbol, % of total ions (35 PSU)

seawater = [['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69,],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]

print(seawater)

# The next highest concentrations
bicarb = ['Bicarbonate', 'HCO3', 0.41]
bromide = ['Bromide', 'Br', 0.19]
borate = ['Borate', 'B(OH)4', 0.08]
strontium = ['Strontium', 'Sr', 0.04]

```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Sulphate', 'SO4', 7.75],
['Calcium', 'Ca', 1.18],
['Potassium', 'K', 1.14]]
```

Setting a variable equal to a list and then changing the list, changes the variable too. Avoid this by using **copy( )**



```python
del seawater[3:]
top3 = seawater.copy()

print(seawater)
print()
print(top3)
print()

seawater.extend([bicarb, bromide, borate])

print(seawater)
print()
print(top3)
print()
```

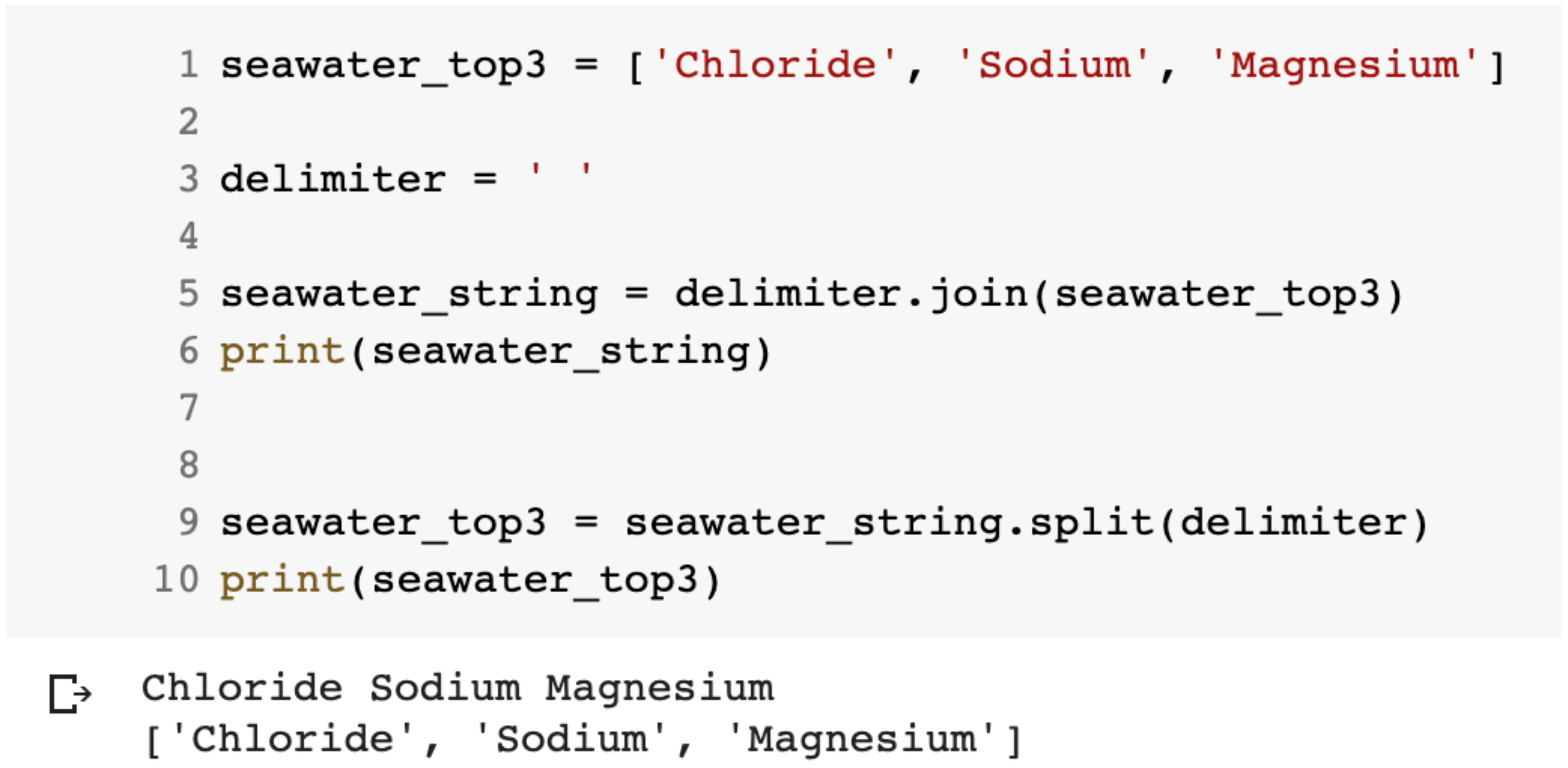
```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69]]

[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69]]
```

```
[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69],
['Bicarbonate', 'HCO3', 0.41],
['Bromide', 'Br', 0.19],
['Borate', 'B(OH)4', 0.08]]

[['Chloride', 'Cl', 55.29],
['Sodium', 'Na', 30.74],
['Magnesium', 'Mg', 3.69]]
```

List functions

When a list has only strings in it, you can combine the different string items into a single string object.

join() **split()**



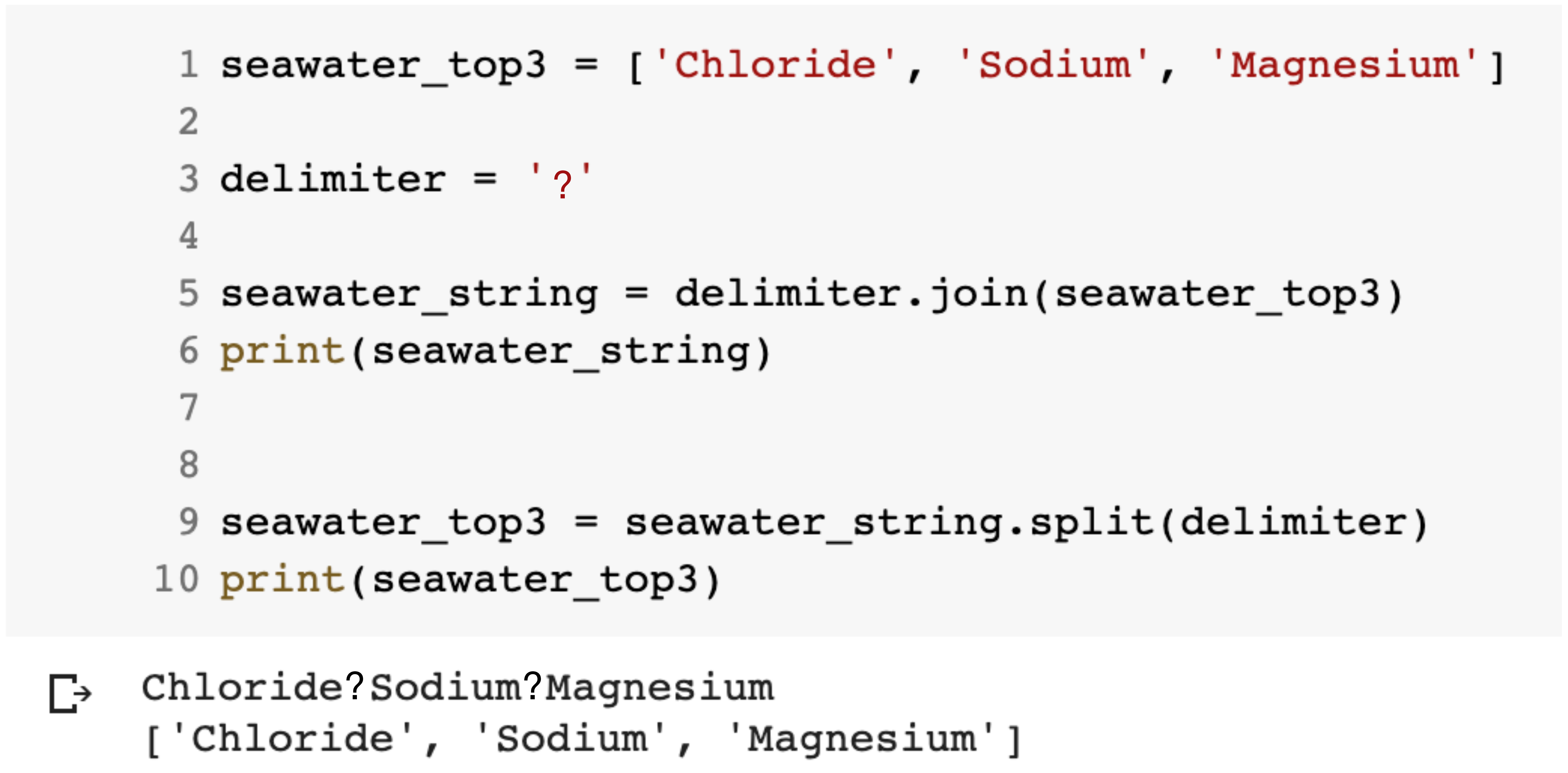
```python
seawater_top3 = ['Chloride', 'Sodium', 'Magnesium']

delimiter = ' '

seawater_string = delimiter.join(seawater_top3)
print(seawater_string)


seawater_top3 = seawater_string.split(delimiter)
print(seawater_top3)
```

```
Chloride Sodium Magnesium
['Chloride', 'Sodium', 'Magnesium']
```

List functions

When a list has only strings in it, you can combine the different string items into a single string object.

join() **split()**



```python
seawater_top3 = ['Chloride', 'Sodium', 'Magnesium']

delimiter = '?'

seawater_string = delimiter.join(seawater_top3)
print(seawater_string)


seawater_top3 = seawater_string.split(delimiter)
print(seawater_top3)
```

```
Chloride?Sodium?Magnesium
['Chloride', 'Sodium', 'Magnesium']
```

List functions

append	Put single item on the end	-	`list.append( object )`
extend	Put multiple items on the end	Items must be put into a list	`list.extend( [ object ] )`
+	Concatenate 2 lists	Be cautious of concatenating lists within lists	`new_list = list1 + list2`
insert	Put an item in at a specific index	Specify the index value first	`list.insert( index, object )`
remove	Delete the first occurrence of an item	Object must be in the list	`list.remove( object )`
del	Delete a slice (using indexing)	Can remove whole objects too!	`del list`
pop	Remove and output and item at a specific index (default: -1)	Keep track of how your index values change	`list.pop(index)`
reverse	Reverse the order of the list	-	`list.reverse( )`
copy	Create a separate copy of a list	This helps to keep a version of the list that is “original”	`new_list = list.copy( )`
join	Join the strings in a list into a single string	Can only be used if the list has just strings in it	`‘ ‘.join(list)`