1. **Familiar functions**

Out of the box, Python offers a bunch of built-in functions to make your life as a data scientist easier. You already know two such functions: [**print()**](https://docs.python.org/3/library/functions.html#print) and [**type()**](https://docs.python.org/3/library/functions.html#type). You've also used the functions **[str()](https://docs.python.org/3/library/functions.html" \l "func-str" \t "_blank)**, **[int()](https://docs.python.org/3/library/functions.html" \l "int" \t "_blank)**, [**bool()**](https://docs.python.org/3/library/functions.html#bool) and [**float()**](https://docs.python.org/3/library/functions.html#float) to switch between data types. These are built-in functions as well.

Calling a function is easy. To get the type of 3.0 and store the output as a new variable, result, you can use the following:

result = type(3.0)

The general recipe for calling functions and saving the result to a variable is thus:

output = function\_name(input)

# Help!

Maybe you already know the name of a Python function, but you still have to figure out how to use it. Ironically, you have to ask for information about a function with another function: [**help()**](https://docs.python.org/3/library/functions.html#help). In IPython specifically, you can also use ? before the function name.

To get help on the [**max()**](https://docs.python.org/3/library/functions.html#max) function, for example, you can use one of these calls:

help(max)

?max

Use the Shell on the right to open up the documentation on [**complex()**](https://docs.python.org/3/library/functions.html#complex). Which of the following statements is true?

# Multiple arguments

In the previous exercise, the square brackets around imag in the documentation showed us that the imag argument is optional. But Python also uses a different way to tell users about arguments being optional.

Have a look at the documentation of [**sorted()**](https://docs.python.org/3/library/functions.html#sorted) by typing help(sorted) in the IPython Shell.

You'll see that [**sorted()**](https://docs.python.org/3/library/functions.html#sorted) takes three arguments: iterable, key and reverse.

key=None means that if you don't specify the key argument, it will be None. reverse=False means that if you don't specify the reverseargument, it will be False.

In this exercise, you'll only have to specify iterable and reverse, not key. The first input you pass to [**sorted()**](https://docs.python.org/3/library/functions.html#sorted) will be matched to the iterable argument, but what about the second input? To tell Python you want to specify reverse without changing anything about key, you can use =:

sorted(\_\_\_, reverse = \_\_\_)

Two lists have been created for you on the right. Can you paste them together and sort them in descending order?

Note: For now, we can understand an **[iterable](https://docs.python.org/2/glossary.html" \l "term-iterable" \t "_blank)** as being any collection of objects, e.g. a List.

# String Methods

Strings come with a bunch of methods. Follow the instructions closely to discover some of them. If you want to discover them in more detail, you can always type help(str) in the IPython Shell.

A string place has already been created for you to experiment with.

* Use the [**upper()**](https://docs.python.org/3/library/stdtypes.html#str.upper) method on place and store the result in place\_up. Use the syntax for calling methods that you learned in the previous video.
* Print out place and place\_up. Did both change?
* Print out the number of o's on the variable place by calling [**count()**](https://docs.python.org/3/library/stdtypes.html#str.count) on place and passing the letter 'o' as an input to the method. We're talking about the variable place, not the word "place"!

# 5. List Methods

Strings are not the only Python types that have methods associated with them. Lists, floats, integers and booleans are also types that come packaged with a bunch of useful methods. In this exercise, you'll be experimenting with:

* [**index()**](https://docs.python.org/3/library/stdtypes.html#str.index), to get the index of the first element of a list that matches its input and
* [**count()**](https://docs.python.org/3/library/stdtypes.html#str.count), to get the number of times an element appears in a list.

You'll be working on the list with the area of different parts of a house: areas.

* Use the [**index()**](https://docs.python.org/3/library/stdtypes.html#str.index) method to get the index of the element in areas that is equal to 20.0. Print out this index.
* Call [**count()**](https://docs.python.org/3/library/stdtypes.html#str.count) on areas to find out how many times 9.50 appears in the list. Again, simply print out this number.
  1. **List Methods (2)**

Most list methods will change the list they're called on. Examples are:

* [**append()**](https://docs.python.org/3/library/stdtypes.html#typesseq-mutable), that adds an element to the list it is called on,
* [**remove()**](https://docs.python.org/3/library/stdtypes.html#typesseq-mutable), that removes the first element of a list that matches the input, and
* [**reverse()**](https://docs.python.org/3/library/stdtypes.html#typesseq-mutable), that reverses the order of the elements in the list it is called on.

You'll be working on the list with the area of different parts of the house: areas.

* Use [**append()**](https://docs.python.org/3/library/stdtypes.html#typesseq-mutable) twice to add the size of the poolhouse and the garage again: 24.5 and 15.45, respectively. Make sure to add them in this order.
* Print out areas
* Use the [**reverse()**](https://docs.python.org/3/library/stdtypes.html#typesseq-mutable) method to reverse the order of the elements in areas.
* Print out areas once more.