**Introduction to python**

**(From Data Camp)**

**Create a list**

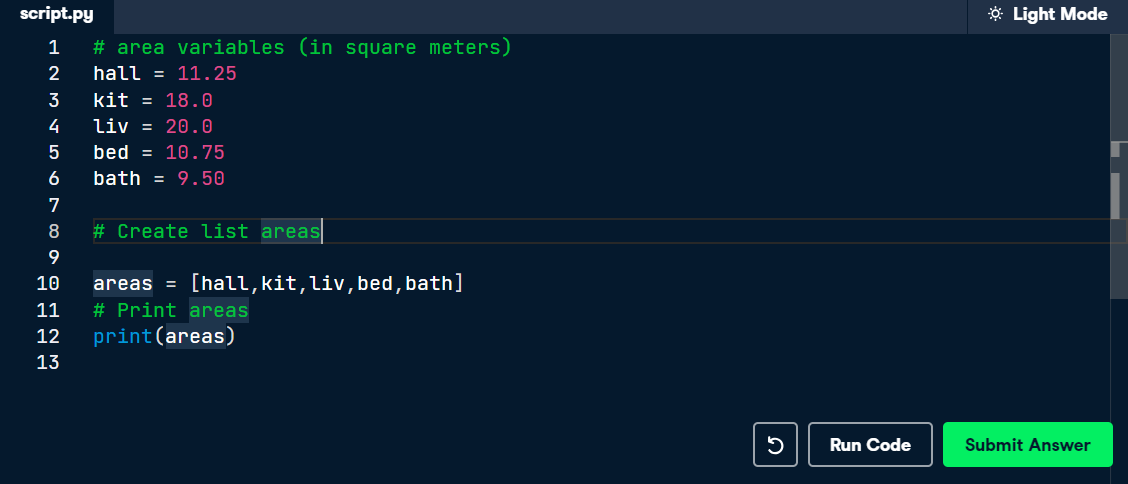
As opposed to int, bool etc., a list is a **compound data type**; you can group values together:

a = "is"

b = "nice"

my\_list = ["my", "list", a, b]

After measuring the height of your family, you decide to collect some information on the house you're living in. The areas of the different parts of your house are stored in separate variables for now, as shown in the scrip

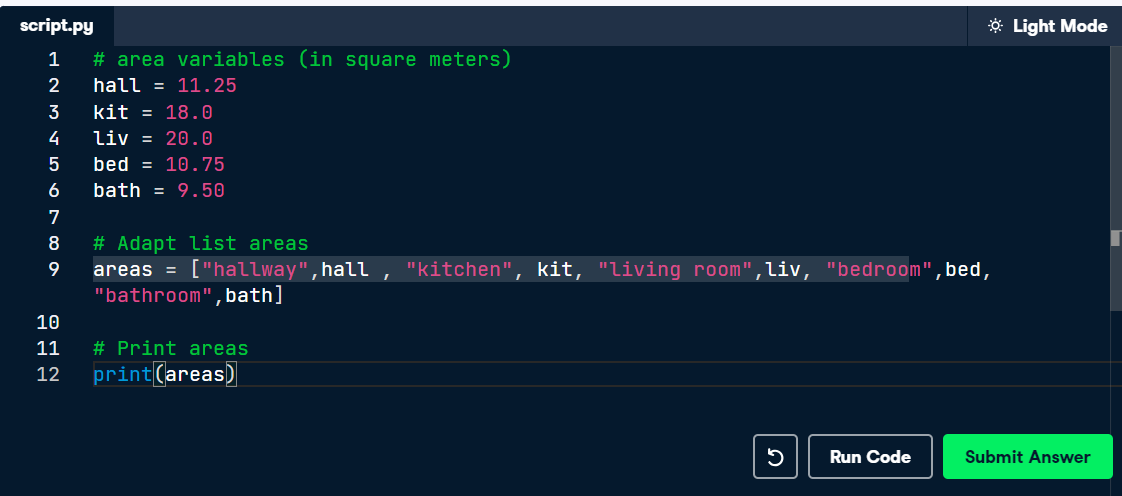


# Create list with different types

A list can contain any Python type. Although it's not really common, a list can also contain a mix of Python types including strings, floats, booleans, etc.

The printout of the previous exercise wasn't really satisfying. It's just a list of numbers representing the areas, but you can't tell which area corresponds to which part of your house.

The code in the editor is the start of a solution. For some of the areas, the name of the corresponding room is already placed in front. Pay attention here! "bathroom" is a string, while bath is a variable that represents the float 9.50 you specified earlie

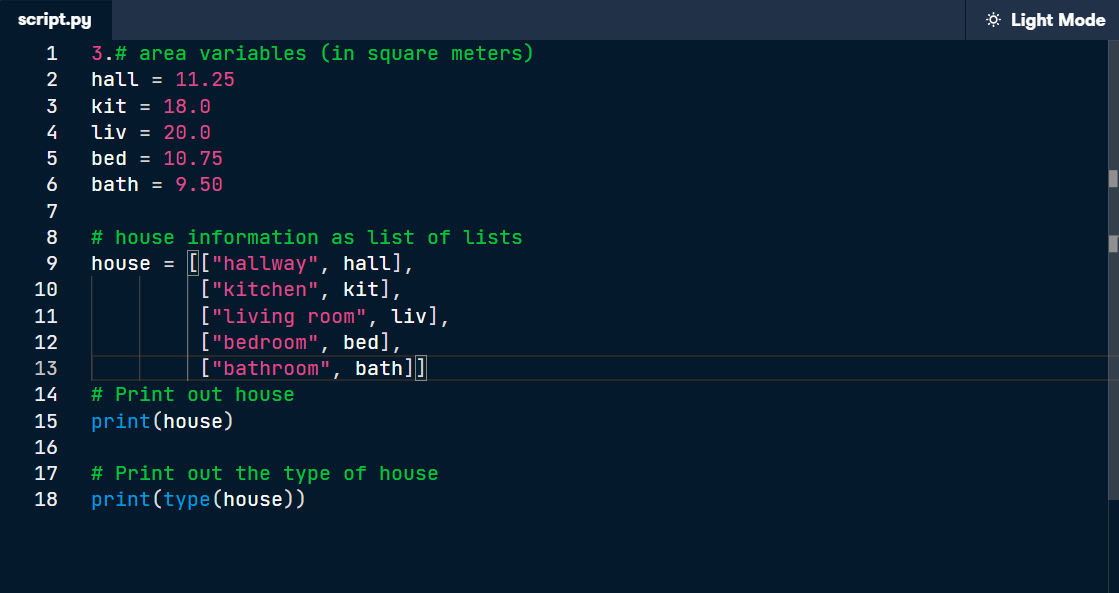


# List of lists

As a data scientist, you'll often be dealing with a lot of data, and it will make sense to group some of this data.

Instead of creating a flat list containing strings and floats, representing the names and areas of the rooms in your house, you can create a list of lists. The script in the editor can already give you an idea.

Don't get confused here: "hallway" is a string, while hall is a variable that represents the float 11.25 you specified earlier.



# Subset and conquer

Subsetting Python lists is a piece of cake. Take the code sample below, which creates a list x and then selects "b" from it. Remember that this is the second element, so it has index 1. You can also use negative indexing.

x = ["a", "b", "c", "d"]

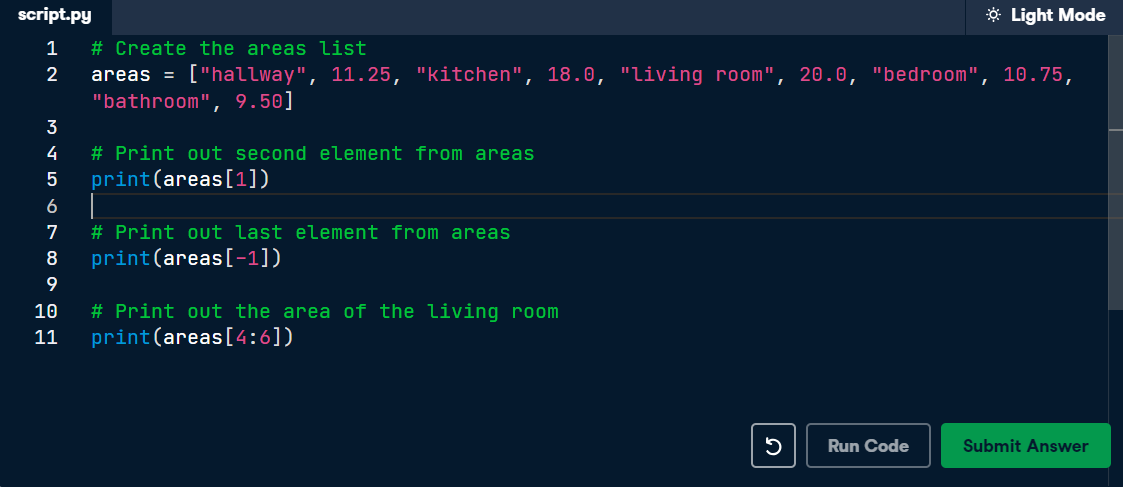
x[1]

x[-3] # same result!

Remember the areas list from before, containing both strings and floats? Its definition is already in the script. Can you add the correct code to do some Python subsetting?

**Instructions :**

* Print out the second element from the areas list (it has the value 11.25).
* Subset and print out the last element of areas, being 9.50. Using a negative index makes sense here!
* Select the number representing the area of the living room (20.0) and print it out.



# Subset and calculate

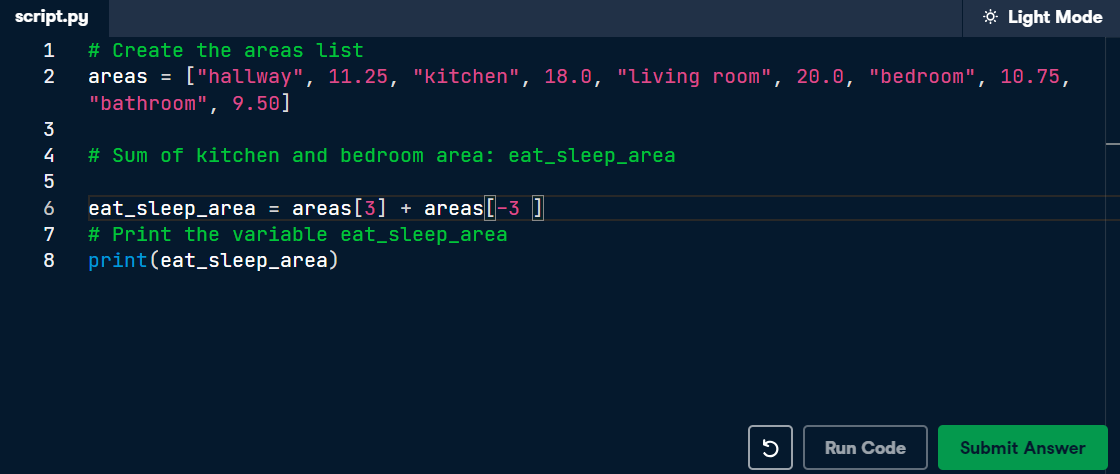
After you've extracted values from a list, you can use them to perform additional calculations. Take this example, where the second and fourth element of a list x are extracted. The strings that result are pasted together using the + operator:

x = ["a", "b", "c", "d"]

print(x[1] + x[3])

## Instructions

* Using a combination of list subsetting and variable assignment, create a new variable, eat\_sleep\_area, that contains the sum of the area of the kitchen and the area of the bedroom.
* Print the new variable eat\_sleep\_area



# Slicing and dicing

Selecting single values from a list is just one part of the story. It's also possible to slice your list, which means selecting multiple elements from your list. Use the following syntax:

my\_list[start:end]

The start index will be included, while the end index is not.

The code sample below shows an example. A list with "b" and "c", corresponding to indexes 1 and 2, are selected from a list x:

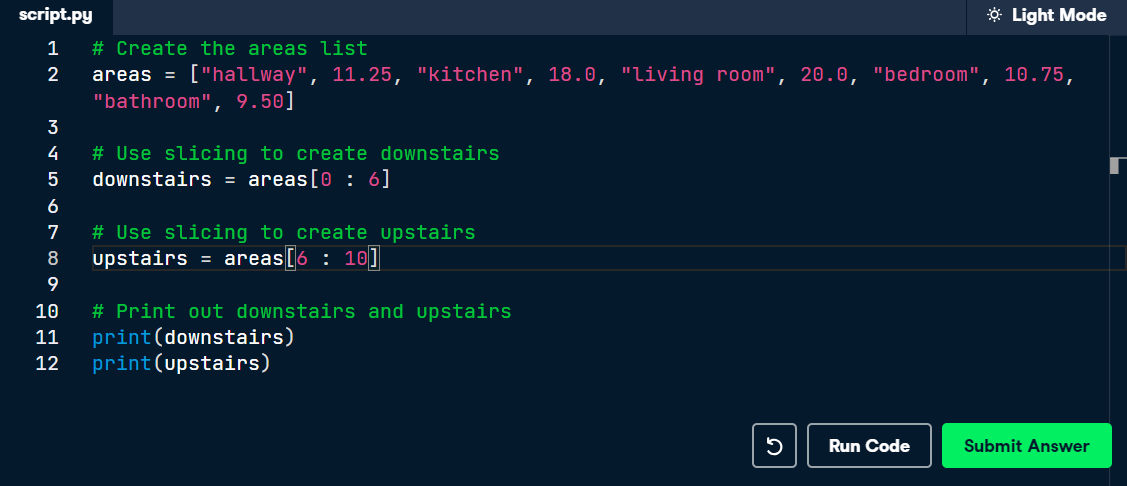
x = ["a", "b", "c", "d"]

x[1:3]

The elements with index 1 and 2 are included, while the element with index 3 is not.

## Instructions

* Use slicing to create a list, downstairs, that contains the first 6 elements of areas.
* Do a similar thing to create a new variable, upstairs, that contains the last 4 elements of areas.
* Print both downstairs and upstairs using [**print()**](https://docs.python.org/3/library/functions.html#print)



# Slicing and dicing (2)

In the video, Hugo first discussed the syntax where you specify both where to begin and end the slice of your list:

my\_list[begin:end]

However, it's also possible not to specify these indexes. If you don't specify the begin index, Python figures out that you want to start your slice at the beginning of your list. If you don't specify the end index, the slice will go all the way to the last element of your list. To experiment with this, try the following commands in the IPython Shell:

x = ["a", "b", "c", "d"]

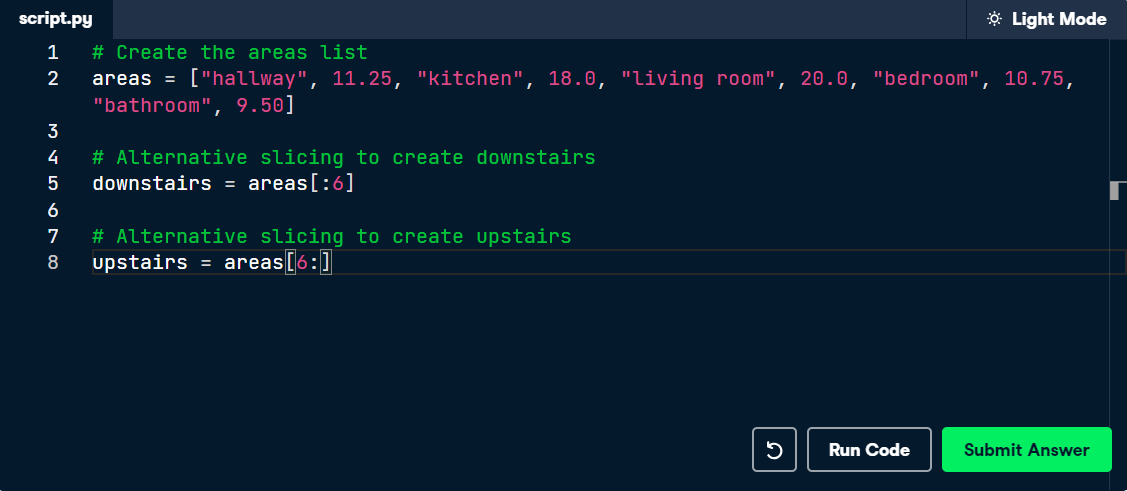
x[:2]

x[2:]

x[:]

## Instructions

* Create downstairs again, as the first 6 elements of areas. This time, simplify the slicing by omitting the begin index.
* Create upstairs again, as the last 4 elements of areas. This time, simplify the slicing by omitting the end index.



# Replace list elements

Replacing list elements is pretty easy. Simply subset the list and assign new values to the subset. You can select single elements or you can change entire list slices at once.

Use the IPython Shell to experiment with the commands below. Can you tell what's happening and why?

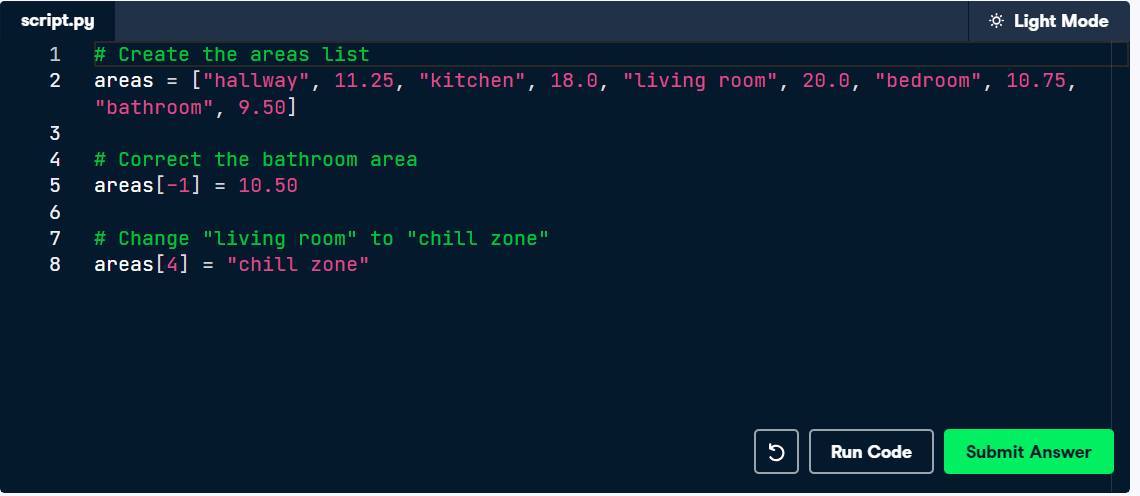
x = ["a", "b", "c", "d"]

x[1] = "r"

x[2:] = ["s", "t"]

For this and the following exercises, you'll continue working on the areas list that contains the names and areas of different rooms in a house.

## Instructions

* Update the area of the bathroom area to be 10.50 square meters instead of 9.50.
* Make the areas list more trendy! Change "living room" to "chill zone"

# Extend a list

If you can change elements in a list, you sure want to be able to add elements to it, right? You can use the + operator:

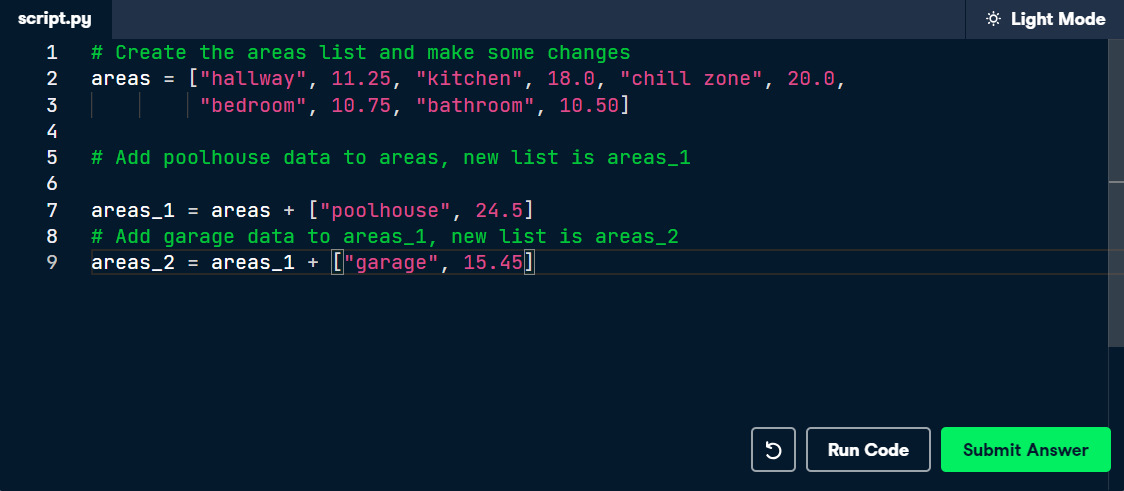
x = ["a", "b", "c", "d"]

y = x + ["e", "f"]

You just won the lottery, awesome! You decide to build a poolhouse and a garage. Can you add the information to the areas list?

## Instructions

* Use the + operator to paste the list ["poolhouse", 24.5] to the end of the areas list. Store the resulting list as areas\_1.
* Further extend areas\_1 by adding data on your garage. Add the string "garage" and float 15.45. Name the resulting list areas\_2.



# Inner workings of lists

At the end of the video, Hugo explained how Python lists work behind the scenes. In this exercise you'll get some hands-on experience with this.

The Python code in the script already creates a list with the name areas and a copy named areas\_copy. Next, the first element in the areas\_copy list is changed and the areas list is printed out. If you hit Run Code you'll see that, although you've changed areas\_copy, the change also takes effect in the areas list. That's because areas and areas\_copy point to the same list.

If you want to prevent changes in areas\_copy from also taking effect in areas, you'll have to do a more explicit copy of the areas list. You can do this with [**list()**](https://docs.python.org/3/library/functions.html#func-list) or by using [:].

## Instructions

Change the second command, that creates the variable areas\_copy, such that areas\_copy is an explicit copy of areas. After your edit, changes made to areas\_copy shouldn't affect areas. Submit the answer to check this

