

# Python Lists

INTRODUCTION TO PYTHON



Hugo Bowne-Anderson

Data Scientist at DataCamp

## Python Data Types

- float - real numbers

- int - integer numbers
- str - string, text

```
height = 1.73  
tall = True
```

bool - True, False

- Each variable represents single value

## Problem

- Data Science: many data points

- 

```
height1 = 1.73  
height2 = 1.68  
height3 = 1.71  
height4 = 1.89
```

Height of entire family

- Inconvenient

## Python List

- 

```
[1.73, 1.68, 1.71, 1.89]
```

```
[a, b, c]
```

```
[1.73, 1.68, 1.71, 1.89]
```

```
fam = [1.73, 1.68, 1.71, 1.89]
```

```
fam
```

```
[1.73, 1.68, 1.71, 1.89]
```

- Name a collection of values

- Contain any type

- Contain different types

## Python List

- 

```
fam = ["liz", 1.73, "emma", 1.68, "mom", 1.71, "dad", 1.89]
```

```
fam
```

```
[a, b, c]
```

```
['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
```

```
fam2 = [ ["liz", 1.73],  
         ["emma", 1.68],  
         ["mom", 1.71],  
         ["dad", 1.89] ]
```

```
fam2
```

```
[['liz', 1.73], ['emma', 1.68], ['mom', 1.71], ['dad', 1.89]]
```

## List type

```
type(fam)
```

```
list
```

```
type(fam2)
```

```
list
```

- Specific functionality

- Specific behavior

# Let's practice!

INTRODUCTION TO PYTHON

## Exercise

### Create a list

A list is a **compound data type**; you can group values together, like this:

```
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 in the exercise.

### Instructions

100 XP

- Create a list, `areas`, that contains the area of the hallway (`hall`), kitchen (`kit`), living room (`liv`), bedroom (`bed`) and bathroom (`bath`), in this order. Use the predefined variables.
- Print `areas` with the `print()` function.

 Take Hint (-30 XP)

## script.py

```
1 hall = 11.25  
2 kit = 18.0  
3 liv = 20.0  
4 bed = 10.75  
5 bath = 9.50  
6  
7 # Create list areas  
8 areas = [hall, kit, liv, bed, bath]  
9  
10 # Print areas  
11 print(areas)
```



Run Code

Submit Answer

## IPython Shell

## Slides

In [1]:

## Exercise

### Create lists with different types

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

You're now going to add the room names to your list, so you can easily see both the room name and size together.

Some of the code has been provided for you to get you started. Pay attention here! "bathroom" is a string, while bath is a variable that represents the float 9.50 you specified earlier.

#### Instructions

100 XP

- Finish the code that creates the areas list. Build the list so that the list first contains the name of each room as a string and then its area. In other words, add the strings "hallway", "kitchen" and "bedroom" at the appropriate locations.
- Print areas again; is the printout more informative this time?

 Take Hint (-30 XP)

## script.py

```
1 hall = 11.25
2 kit = 18.0
3 liv = 20.0
4 bed = 10.75
5 bath = 9.50
6
7 # Adapt List areas
8 areas = ["hallway",hall,"kitchen",kit,"living room",liv,"bedroom",
9 bed,"bathroom",bath]
10 # Print areas to see the output
11 print(areas)
```



Run Code

Submit Answer

IPython Shell

Slides

In [1]:

## Exercise

Light Mode

### 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 list containing strings and floats, representing the names and areas of the rooms in your house, you can create a list of lists.

Remember: "hallway" is a string, while hall is a variable that represents the float 11.25 you specified earlier.

### Instructions

100 XP

- Finish the list of lists so that it also contains the bedroom and bathroom data. Make sure you enter these in order!
- Print out house ; does this way of structuring your data make more sense?

 Take Hint (-30 XP)

### script.py

```
1  hall = 11.25
2  kit = 18.0
3  liv = 20.0
4  bed = 10.75
5  bath = 9.50
6  # House information as list of lists
7  house = [[ "hallway", hall ],
8  [ "kitchen", kit ],
9  [ "living room", liv ],
10 [ "bedroom", bed ],
11 [ "bathroom", bath ] ]
12 # Print out house
13 print(house)
```



Run Code

Submit Answer

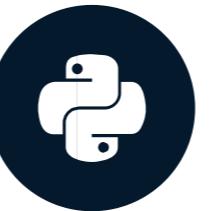
IPython Shell

Slides

In [1]:

# Subsetting Lists

INTRODUCTION TO PYTHON



**Hugo Bowne-Anderson**  
Data Scientist at DataCamp

```
fam = ["liz", 1.73, "emma", 1.68, "mom", 1.71, "dad", 1.89]  
fam
```

```
['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
```

```
fam[3]
```

```
1.68
```

# Subsetting lists

```
[ 'liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
```

```
fam[ 6 ]
```

```
'dad'
```

```
fam[ -1 ]
```

```
1.89
```

```
fam[ 7 ]
```

```
1.89
```

```
[ 'liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
```

```
fam[6]
```

```
'dad'
```

```
fam[-1] # <-
```

```
1.89
```

```
fam[7] # <-
```

```
1.89
```

## List slicing

```
fam
```

# Subsetting lists

```
[ 'liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
```

```
fam[3:5]
```

```
[1.68, 'mom']
```

```
fam[1:4]
```

```
[1.73, 'emma', 1.68]
```

[ start : end ]

inclusive

exclusive

# List slicing

```
fam
```

```
['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
```

```
fam[:4]
```

```
['liz', 1.73, 'emma', 1.68]
```

```
fam[5:]
```

```
[1.71, 'dad', 1.89]
```

# Let's practice!

INTRODUCTION TO PYTHON

## Exercise

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

100 XP

- 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.

## script.py

```
1 # Create the areas list
2 areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0,
3         "bedroom", 10.75, "bathroom", 9.50]
4
5 # Print out second element from areas
6 print(areas[1])
7
8 # Print out last element from areas
9 print(areas[-1])
10
11 # Print out the area of the living room
12 print(areas[5])
```



Run Code

Submit Answer

IPython Shell

Slides

In [1]:

## Exercise

### 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*. However, it's also possible not to specify these indexes. If you don't specify the `start` index, Python figures out that you want to start your slice at the beginning of your list.

#### Instructions

100 XP

- Use slicing to create a list, `downstairs`, that contains the first 6 elements of `areas`.
- Create `upstairs`, as the last 4 elements of `areas`. This time, simplify the slicing by omitting the `end` index.
- Print both `downstairs` and `upstairs` using `print()`.

### script.py

```
1 # Create the areas list
2 areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0,
3           "bedroom", 10.75, "bathroom", 9.50]
4
5 # Use slicing to create downstairs
6 downstairs = areas[0:6]
7
8 # Use slicing to create upstairs
9 upstairs = areas[6:]
10
11 # Print out downstairs and upstairs
12 print(downstairs)
13 print(upstairs)
```



Run Code

Submit Answer

IPython Shell

Slides

In [1]:

## Exercise

Light Mode

### Subsetting lists of lists

A Python list can also contain other lists.

To subset lists of lists, you can use the same technique as before: square brackets. This would look something like this for a list, `house`:

```
house[2][0]
```

#### Instructions

100 XP

- Subset the `house` list to get the float `9.5`.

 Take Hint (-30 XP)

script.py

```
1 house = [["hallway", 11.25],  
2     |     | ["kitchen", 18.0],  
3     |     | ["living room", 20.0],  
4     |     | ["bedroom", 10.75],  
5     |     | ["bathroom", 9.50]]  
6  
7 # Subset the house list  
8 house[4][1]
```



Run Code

Submit Answer

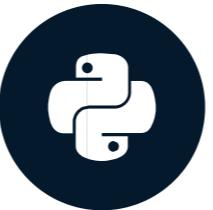
IPython Shell

Slides

In [1]:

# Manipulating Lists

INTRODUCTION TO PYTHON



Hugo Bowne-Anderson

Data Scientist at DataCamp

# List Manipulation

- Change list elements
- Add list elements
- Remove list elements

# Changing list elements

```
fam = ["liz", 1.73, "emma", 1.68, "mom", 1.71, "dad", 1.89]  
fam
```

```
fam[7] = 1.86  
fam
```

```
['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.86]
```

```
fam[0:2] = ["lisa", 1.74]  
fam
```

```
['lisa', 1.74, 'emma', 1.68, 'mom', 1.71, 'dad', 1.86]
```

```
['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
```

# Adding and removing elements

```
fam + [ "me" , 1.79 ]
```

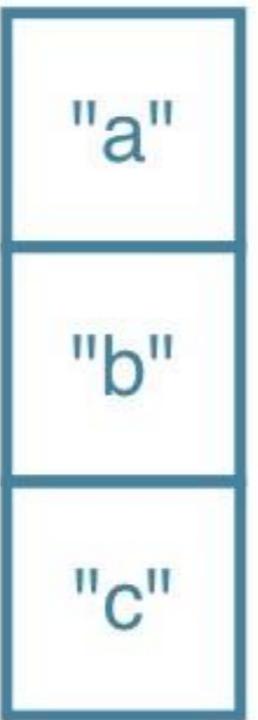
```
['lisa' , 1.74 , 'emma' , 1.68 , 'mom' , 1.71 , 'dad' , 1.86 , 'me' , 1.79]
```

```
['lisa' , 1.74 , 1.68 , 'mom' , 1.71 , 'dad' , 1.86]
```

```
fam_ext = fam + [ "me", 1.79]  
del fam[2]  
fam
```

# Behind the scenes (1)

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



# Behind the scenes (1)

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

```
y = x
```

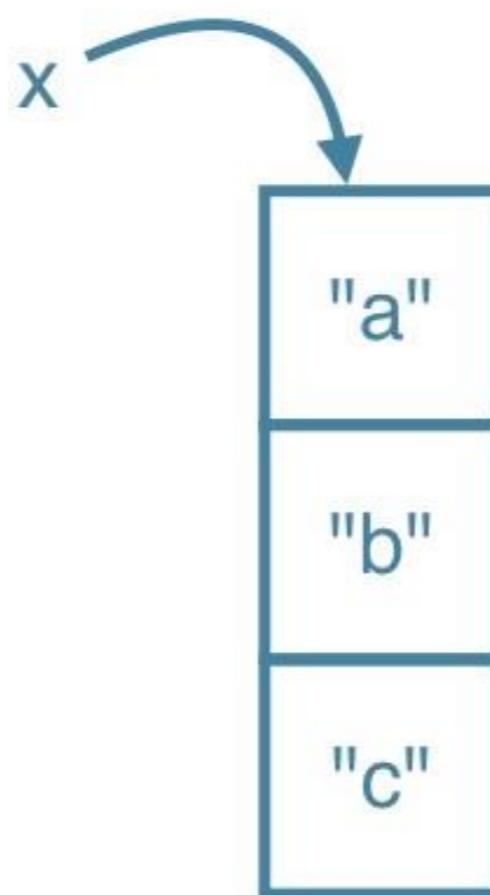
```
y[1] = "z"
```

```
y
```

```
['a', 'z', 'c']
```

```
x
```

```
['a', 'z', 'c']
```



# Behind the scenes (1)

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

```
y = x
```

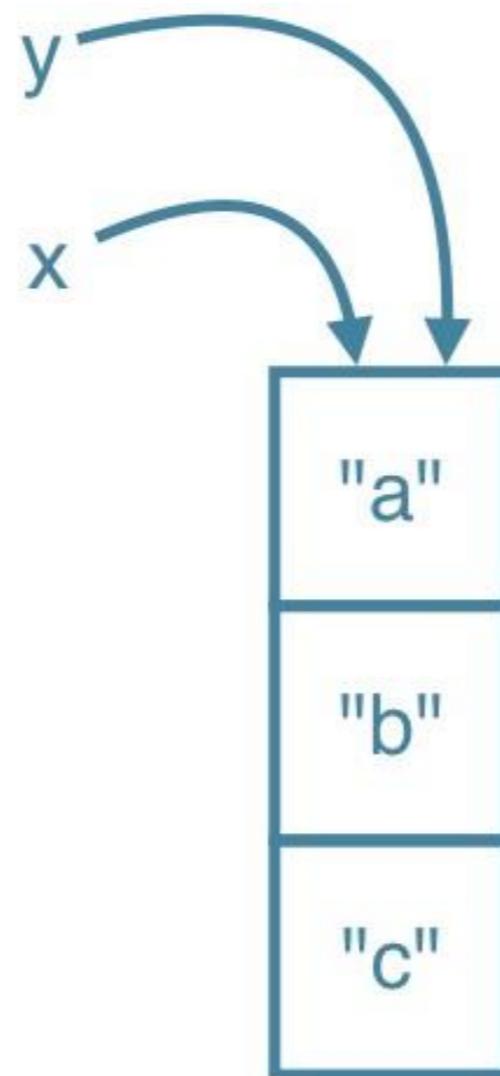
```
y[1] = "z"
```

```
y
```

```
['a', 'z', 'c']
```

```
x
```

```
['a', 'z', 'c']
```



# Behind the scenes (1)

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

```
y = x
```

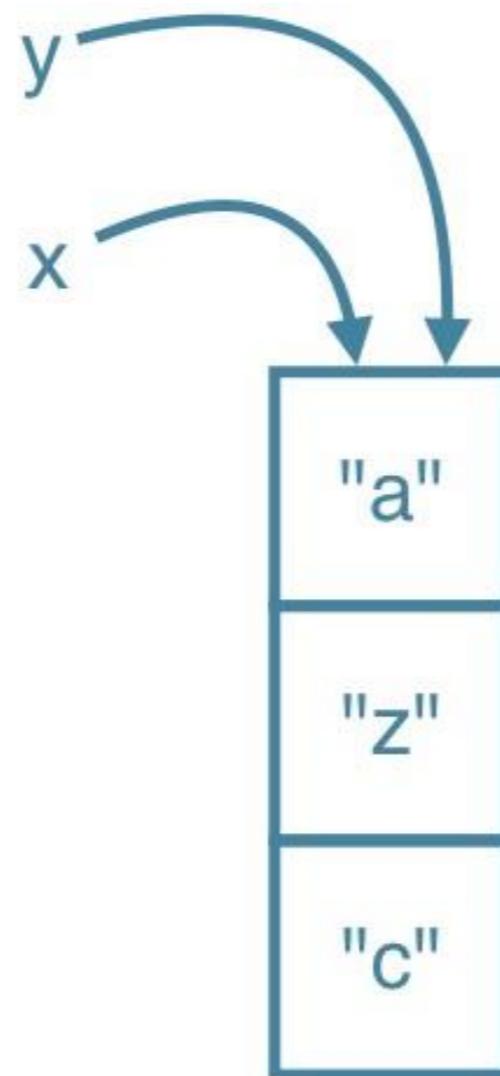
```
y[1] = "z"
```

```
y
```

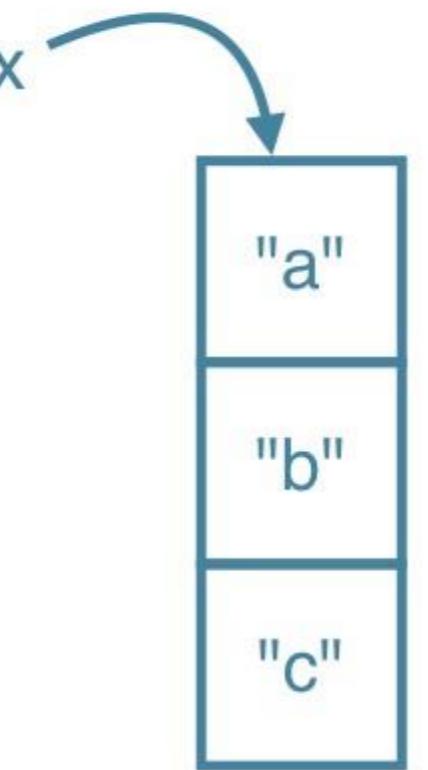
```
['a', 'z', 'c']
```

```
x
```

```
['a', 'z', 'c']
```



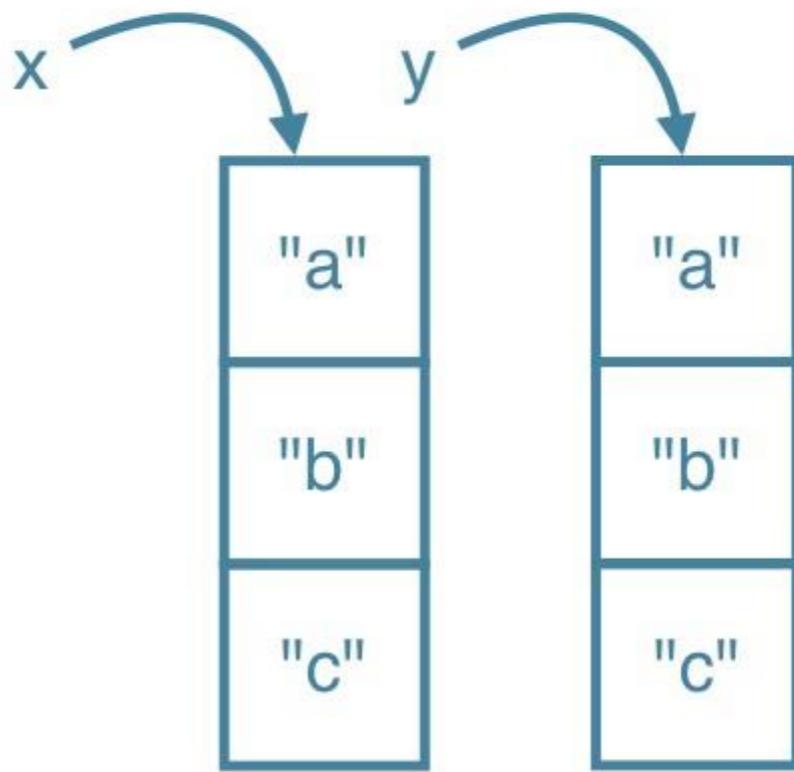
## Behind the scenes (2)



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

# Behind the scenes (2)

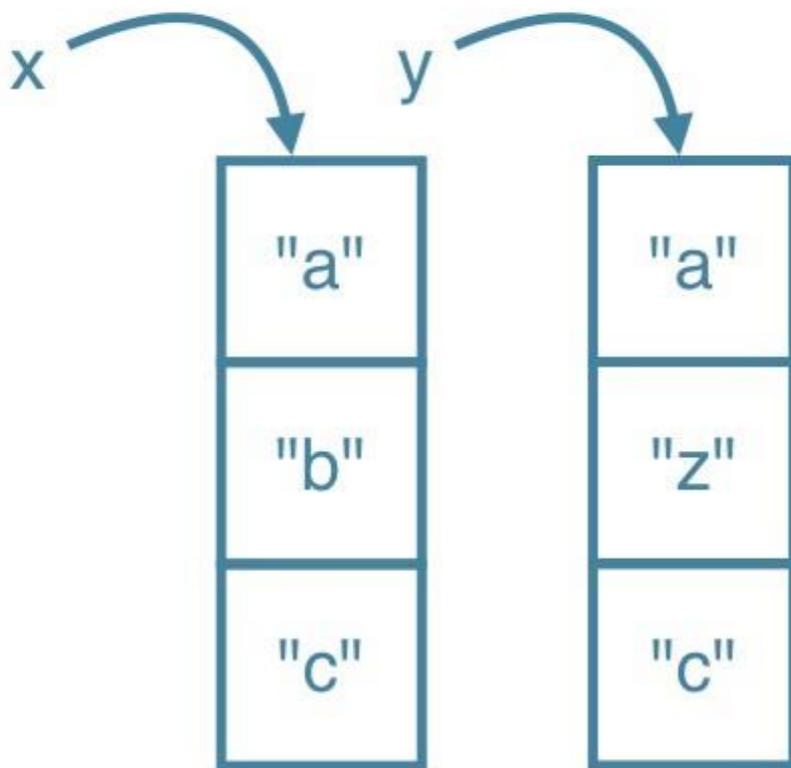
```
x = ["a", "b", "c"]  
y = list(x)  
y = x[:]
```



# Behind the scenes (2)

```
x = ["a", "b", "c"]  
y = list(x)  
y = y[:]  
y[1] = "z"  
x
```

```
['a', 'b', 'c']
```



# Let's practice!

## INTRODUCTION TO PYTHON

Exercise

### Replace list elements

To replace list elements, you subset the list and assign new values to the subset. You can select single elements or you can change entire list slices at once.

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 100 XP

- Update the area of the bathroom to be `10.50` square meters instead of `9.50` using negative indexing.
- Make the `areas` list more trendy! Change `"living room"` to `"chill zone"`. Don't use negative indexing this time.

Take Hint (-30 XP)

script.py

```
1 # Create the areas list
2 areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0,
           "bedroom", 10.75, "bathroom", 9.50]
3
4 # Correct the bathroom area
5 areas[-1] = 10.50
6
7 # Change "living room" to "chill zone"
8 areas[4] = "chill zone"
```

Run Code Submit Answer

In [1]:

## Exercise

Light Mode

### 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

100 XP

- 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`.

 Take Hint (-30 XP)

script.py

```
1 # Create the areas list and make some changes
2 areas = ["hallway", 11.25, "kitchen", 18.0, "chill zone", 20.0,
3           "bedroom", 10.75, "bathroom", 10.50]
4
5 # Add poolhouse data to areas, new list is areas_1
6 areas_1 = areas + ["poolhouse", 24.5]
7
8 # Add garage data to areas_1, new list is areas_2
9 areas_2 = areas_1 + ["garage", 15.45]
```



Run Code

Submit Answer

IPython Shell

Slides

In [1]:

## Exercise

Light Mode

### Delete list elements

Finally, you can also remove elements from your list. You can do this with the `del` statement:

```
x = ["a", "b", "c", "d"]
del x[1]
```

Pay attention here: as soon as you remove an element from a list, the indexes of the elements that come after the deleted element all change!

Unfortunately, the amount you won with the lottery is not that big after all and it looks like the poolhouse isn't going to happen. You'll need to remove it from the list. You decide to remove the corresponding string and float from the `areas` list.

#### Instructions

100 XP

- Delete the string and float for the "poolhouse" from your `areas` list.
- Print the updated `areas` list.

script.py

```
1 areas = ["hallway", 11.25, "kitchen", 18.0,
2           "chill zone", 20.0, "bedroom", 10.75,
3           "bathroom", 10.50, "poolhouse", 24.5,
4           "garage", 15.45]
5
6 # Delete the poolhouse items from the list
7 del areas[10]
8 del areas[10]
9
10 # Print the updated list
11 print(areas)
```



Run Code

Submit Answer

IPython Shell

Slides

In [1]:

## Exercise

Light Mode

### Inner workings of lists

Some code has been provided for you in this exercise: a list with the name `areas` and a copy named `areas_copy`.

Currently, the first element in the `areas_copy` list is changed and the `areas` list is printed out. If you hit the run code button 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 with `list()` or by using `[:]`.

#### Instructions

100 XP

- 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.

script.py

```
1 # Create list areas
2 areas = [11.25, 18.0, 20.0, 10.75, 9.50]
3
4 # Change this command
5 areas_copy = list(areas)
6 areas_copy = areas[:]
7
8 # Change areas_copy
9 areas_copy[0] = 5.0
10
11 # Print areas
12 print(areas)
```



Run Code

Submit Answer

IPython Shell

Slides

In [1]: