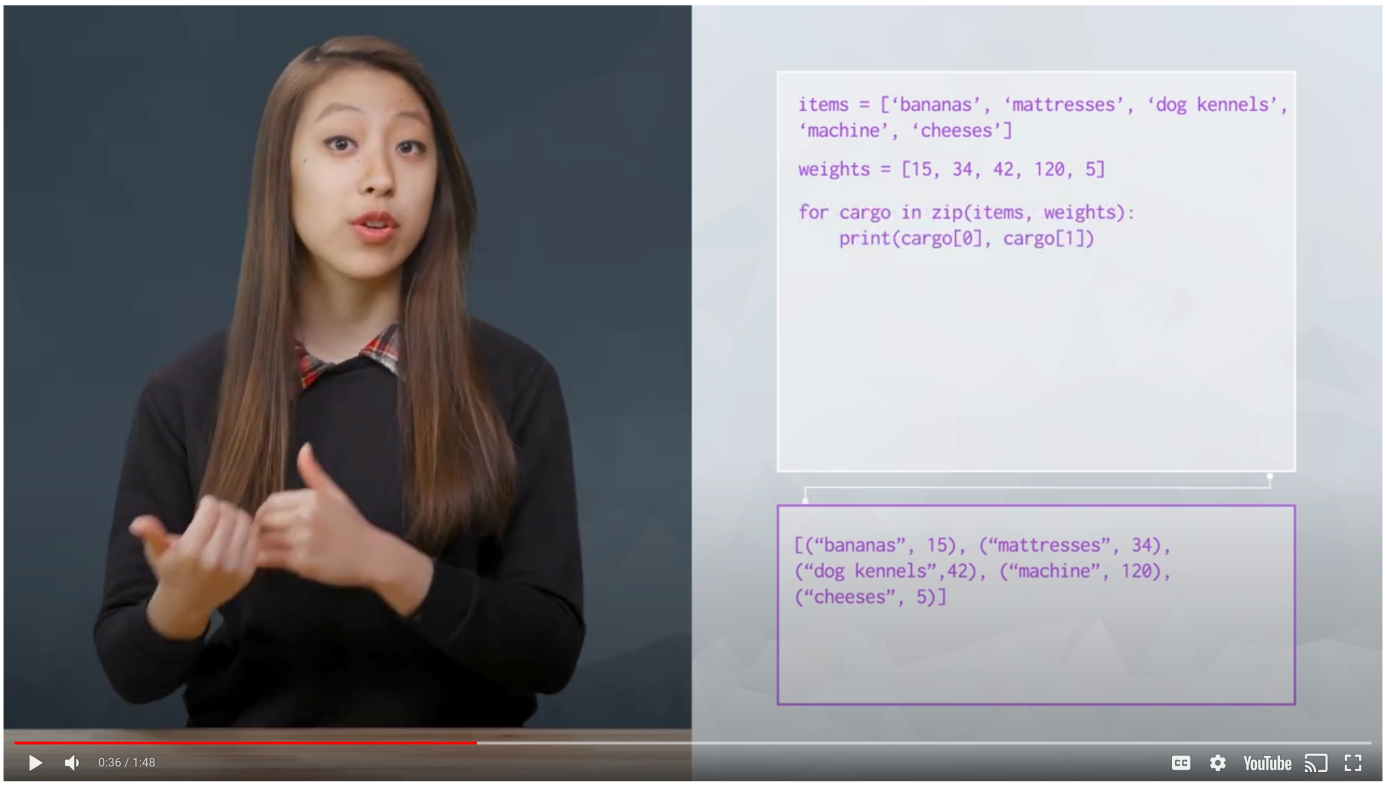
Zip and Enumerate



In the video above, at the 0:55 mark, the instructor says "... you can separate it into an items and weights list, like this," but she should instead say, "... you can separate it into an items *tuple* and a weights *tuple*, like this."

Zip and Enumerate

`zip` and `enumerate` are useful built-in functions that can come in handy when dealing with loops.

Zip

`zip` returns an iterator that combines multiple iterables into one sequence of tuples. Each tuple contains the elements in that position from all the iterables. For example, printing

`list(zip(['a', 'b', 'c'], [1, 2, 3]))` would output `[('a', 1), ('b', 2), ('c', 3)]`.

Like we did for `range()` we need to convert it to a list or iterate through it with a loop to see the elements.

You could unpack each tuple in a `for` loop like this.

```
letters = ['a', 'b', 'c']
nums = [1, 2, 3]

for letter, num in zip(letters, nums):
    print("{}: {}".format(letter, num))
```

In addition to zipping two lists together, you can also unzip a list into tuples using an asterisk.

```
some_list = [('a', 1), ('b', 2), ('c', 3)]
letters, nums = zip(*some_list)
```

This would create the same `letters` and `nums` tuples we saw earlier.

Enumerate

`enumerate` is a built in function that returns an iterator of tuples containing indices and values of a list. You'll often use this when you want the index along with each element of an iterable in a loop

an iterable in a loop.

```
letters = ['a', 'b', 'c', 'd', 'e']
for i, letter in enumerate(letters):
    print(i, letter)
```

This code would output:

```
0 a
1 b
2 c
3 d
4 e
```

← Previous Next → Give Page Feedback