Exercise 32: Loops and Lists

You should now be able to do some programs that are much more interesting. If you have been keeping up, you should realize that now you can combine all the other things you have learned with `if-statements` and boolean expressions to make your programs do smart things.

However, programs also need to do repetitive things very quickly. We are going to use a `for-loop` in this exercise to build and print various lists. When you do the exercise, you will start to figure out what they are. I won't tell you right now. You have to figure it out.

Before you can use a for-loop, you need a way to *store* the results of loops somewhere. The best way to do this is with a `list`. A list is exactly what its name says, a container of things that are organized in order. It's not complicated; you just have to learn a new syntax. First, there's how you make a list:

```
hairs = ['brown', 'blond', 'red']
eyes = ['brown', 'blue', 'green']
weights = [1, 2, 3, 4]
```

What you do is start the list with the `[` (left bracket) which "opens" the list. Then you put each item you want in the list separated by commas, just like when you did function arguments. Lastly you end the list with a `]` (right bracket) to indicate that it's over. Python then takes this list and all its contents and assigns them to the variable.

Warning

This is where things get tricky for people who can't program. Your brain has been taught that the world is flat. Remember in the last exercise

where you put `if-statements` inside `if-statements`? That probably made your brain hurt because most people do not ponder how to "nest" things inside things. In programming this is all over the place. You will find functions that call other functions that have `if-statements` that have lists with lists inside lists. If you see a structure like this that you can't figure out, take out a pencil and paper and break it down manually bit by bit until you understand it.

We now will build some lists using some loops and print them out:

```
1  the_count = [1, 2, 3, 4, 5]
2  fruits = ['apples', 'oranges', 'pears', 'apricots']
3  change = [1, 'pennies', 2, 'dimes', 3, 'quarters']
4
5  # this first kind of for-loop goes through a list
6  for number in the_count:
7      print "This is count %d" % number
8
9  # same as above
10 for fruit in fruits:
11     print "A fruit of type: %s" % fruit
12
13 # also we can go through mixed lists too
14 # notice we have to use %r since we don't know what's in
15 it
16 for i in change:
17     print "I got %r" % i
18
19 # we can also build lists, first start with an empty one
20 elements = []
21
22 # then use the range function to do 0 to 5 counts
23 for i in range(0, 6):
24     print "Adding %d to the list." % i
25     # append is a function that lists understand
26     elements.append(i)
27
28 # now we can print them out too
29 for i in elements:
       print "Element was: %d" % i
```

What You Should See

```
$ python ex32.py
This is count 1
```

```
This is count 2
This is count 3
This is count 4
This is count 5
A fruit of type: apples
A fruit of type: oranges
A fruit of type: pears
A fruit of type: apricots
I got 1
I got 'pennies'
I got 2
I got 'dimes'
I got 3
I got 'quarters'
Adding 0 to the list.
Adding 1 to the list.
Adding 2 to the list.
Adding 3 to the list.
Adding 4 to the list.
Adding 5 to the list.
Element was: 0
Element was: 1
Element was: 2
Element was: 3
Element was: 4
Element was: 5
```

Study Drills

1. Take a look at how you used `range`. Look up the `range` function to understand it.
2. Could you have avoided that `for-loop` entirely on line 22 and just assigned `range(0,6)` directly to `elements`?
3. Find the Python documentation on lists and read about them. What other operations can you do to lists besides `append`?

Common Student Questions

How do you make a 2-dimensional (2D) list?

That's a list in a list like this: `[[1,2,3],[4,5,6]]`

Aren't lists and arrays the same thing?

Depends on the language and the implementation. In classic terms a list is very different from an array because of how they're implemented. In Ruby though they call these arrays. In Python they call them lists. Just call these lists for now since that's what Python calls them.

How come a for-loop can use variables that aren't defined yet?

It defines that variable, initializing it to the current element of the loop iteration, each time through.

Why does `for i in range(1, 3):` only loop two times instead of three times?

The `range()` function only does numbers from the first to the last, *not including the last*. So it stops at two, not three in the above. This turns out to be the most common way to do this kind of loop.

What does `elements.append()` do?

It simply appends to the end of the list. Open up the Python shell and try a few examples with a list you make. Any time you run into things like this, always try to play with them interactively in the Python shell.

Already Paid? Reactivate Your Purchase Right Now!

Buying Is Easy

Buying is easy. Just fill out the form below and we'll get started.

Full Name

Email Address

Pay With Credit Card (by Stripe™)

Use your PayPal™ account.

Buy Learn Python The Hard Way, 3rd Edition