Exercise 38: Doing Things To Lists

You have learned about lists. When you learned about `while-loops` you "appended" numbers to the end of a list and printed them out. There was also Study Drills where you were supposed to find all the other things you can do to lists in the Python documentation. That was a while back, so go find in the book where you did that and review if you do not know what I'm talking about.

Found it? Remember it? Good. When you did this you had a list, and you "called" the function `append` on it. However, you may not really understand what's going on so let's see what we can do to lists.

When you type Python code that reads `mystuff.append('hello')` you are actually setting off a chain of events inside Python to cause something to happen to the `mystuff` list. Here's how it works:

1. Python sees you mentioned `mystuff` and looks up that variable. It might have to look backward to see if you created with `=`, look and see if it is a function argument, or maybe it's a global variable. Either way it has to find the `mystuff` first.
2. Once it finds `mystuff` it then hits the `.` (period) operator and starts to look at *variables* that are a part of `mystuff`. Since `mystuff` is a list, it knows that `mystuff` has a bunch of functions.
3. It then hits `append` and compares the name "append" to all the ones that `mystuff` says it owns. If append is in there (it is) then it grabs *that* to use.
4. Next Python sees the `(` (parenthesis) and realizes, "Oh hey, this should

   be a function." At this point it *calls* (aka runs, executes) the function just like normally, but instead it calls the function with an *extra* argument.

5. That *extra* argument is ... `mystuff`! I know, weird, right? But that's how Python works so it's best to just remember it and assume that's alright. What happens then, at the end of all this is a function call that looks like: `append(mystuff, 'hello')` instead of what you read which is `mystuff.append('hello')`.

For the most part you do not have to know that this is going on, but it helps when you get error messages from Python like this:

```
$ python
Python 2.6.5 (r265:79063, Apr 16 2010, 13:57:41)
[GCC 4.4.3] on linux2
Type "help", "copyright", "credits" or "license" for
more information.
>>> class Thing(object):
...     def test(hi):
...             print "hi"
...
>>> a = Thing()
>>> a.test("hello")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: test() takes exactly 1 argument (2 given)
>>>
```

What was all that? Well, this is me typing into the Python shell and showing you some magic. You haven't seen `class` yet but we'll get into that later. For now you see how Python said `test() takes exactly 1 argument (2 given)`. If you see this it means that Python changed `a.test("hello")` to `test(a, "hello")` and that somewhere someone messed up and didn't add the argument for `a`.

That might be a lot to take in, but we're going to spend a few exercises getting this concept firm in your brain. To kick things off, here's an exercise that mixes strings and lists for all kinds of fun.

```python
ten_things = "Apples Oranges Crows Telephone Light Sugar"

print "Wait there's not 10 things in that list, let's fix
```

```
that."

stuff = ten_things.split(' ')
more_stuff = ["Day", "Night", "Song", "Frisbee", "Corn",
"Banana", "Girl", "Boy"]

while len(stuff) != 10:
    next_one = more_stuff.pop()
    print "Adding: ", next_one
    stuff.append(next_one)
    print "There's %d items now." % len(stuff)

print "There we go: ", stuff

print "Let's do some things with stuff."

print stuff[1]
print stuff[-1] # whoa! fancy
print stuff.pop()
print ' '.join(stuff) # what? cool!
print '#'.join(stuff[3:5]) # super stellar!
```

What You Should See

```
$ python ex38.py
Wait there's not 10 things in that list, let's fix
that.
Adding:  Boy
There's 7 items now.
Adding:  Girl
There's 8 items now.
Adding:  Banana
There's 9 items now.
Adding:  Corn
There's 10 items now.
There we go:  ['Apples', 'Oranges', 'Crows',
'Telephone', 'Light', 'Sugar', 'Boy', 'Girl', 'Banana',
'Corn']
Let's do some things with stuff.
Oranges
Corn
Corn
Apples Oranges Crows Telephone Light Sugar Boy Girl
Banana
Telephone#Light
```

Study Drills

1. Take each function that is called, and go through the steps outlined above to translate them to what Python does. For example, `' '.join(things)` is `join(' ', things)`.
2. Translate these two ways to view the function calls in English. For example, `' '.join(things)` reads as, "Join things with ' ' between them." Meanwhile, `join(' ', things)` means, "Call join with ' ' and things." Understand how they are really the same thing.
3. Go read about "object-oriented programming" online. Confused? I was too. Do not worry. You will learn enough to be dangerous, and you can slowly learn more later.
4. Read up on what a "class" is in Python. *Do not read about how other languages use the word "class."* That will only mess you up.
5. What's the relationship between `dir(something)` and the "class" of `something`?
6. If you do not have any idea what I'm talking about do not worry. Programmers like to feel smart so they invented object-oriented programming, named it OOP, and then used it way too much. If you think that's hard, you should try to use "functional programming."

Common Student Questions

Didn't you say to not use `while-loops`?

Yes, so just remember sometimes you can break the rules if you have a good reason. Only idiots are slaves to rules all the time.

What does `stuff[3:5]` do?

That's getting a "slice" from the `stuff` list that is from element 3 to element 4, meaning it does *not* include element 5. It's similar to how `range(3,5)` would work.

Why does `join(' ', stuff)` not work?

The way the documentation for `join` is written doesn't make sense. It does not work like that and is instead a method you call on the *inserted* string to put between the list to be joined. Rewrite it like `' '.join(stuff)`.

Purchase The Videos For $29.59

For just $29.59 you can get access to all the videos for Learn Python The Hard Way, **plus** a PDF of the book and no more popups all in this one location. For $29.59 you get:

- All 52 videos, 1 per exercise, almost 2G of video.
- A PDF of the book.
- Email help from the author.
- See a list of everything you get before you buy.

When you buy the videos they will immediately show up **right here** without any hassles.

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

[✓ VISA MasterCard American Express Pay With Credit Card (by Stripe™)]

PayPal Use your PayPal™ account.

Buy Learn Python The Hard Way, 3rd Edition