LIST OPERATIONS

An Introduction to Computer Science

Let's learn about List Operations.

Doing Stuff with Lists

- Indexing
- Subscripting
- Membership Test
- Appending
- Popping

Once we have data inside of a list, we can do stuff with it.
There are actually many, many operations you can do on a list, so we'll learn about a few of them here.

Indexing a List

```
>>> names = ["Alice", "Bob", "Carol"]
>>> names[1]
Bob
```

Second Element

```
>>> [10, 20, 30, 40][0]
10
```

First Element

```
>>> [10, 20, 30, 40][-1]
40
```

Last Element

Much like strings, you can use square brackets to access a specific element of the list.
Notice that the syntax is the same, including the fact that we start counting from zero.
Do not be confused by the square brackets here - when they are first, they are list creation.
When they are second, they are list indexing.

Subscripting a List

```
>>> names = ["Alice", "Bob", "Carol"]
>>> names[1:]
["Bob", "Carol"]
```

Second until End

```
>>> [10, 20, 30, 40][1:3]
[20, 30]
```

Second until Fourth

```
>>> [10, 20, 30, 40][-1:]
[40]
```

Second to last until End

Much like strings, you can use a colon and square brackets to access a range of elements from a list.
The rules are the same as for strings: the first number is the starting index, and the second element is the ending index.

List Membership

```
>>> names = ["Alice", "Bob", "Carol"]
>>> "Carol" in names
True

>>> "David" not in names
True

>>> "Ellie" in names
False
```

Much like strings, you can ask if a list has a specific value in it.
The "in" operator has a list on the right, and then any kind of value or variable on the left.
There is also a "not in" operator.

Appending to a List

```
>>> names = ["Alice", "Bob", "Carol"]
>>> names.append("David")
>>> names
['Alice', 'Bob', 'Carol', 'David']

>>> names + "Ellie"
```

Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: can only concatenate list (not "str") to list

Can't use +

You cannot add elements to a list using the + operator.
Instead, you must call a method of the list, called "append".
The word "Append" means "add to the end".
Append lets you add one thing to the end of the list.

Pop from a List

```
>>> names = ["Alice", "Bob", "Carol"]
>>> names.pop()
'Carol'
>>> names
['Alice', 'Bob']
```

You can also remove one thing from the end of the list using the "pop" method.
You will not need to do this too often in this course, because removing things from list can actually be a little tricky.