More on strings

Let's take a deeper dive into strings. Yesterday, I said a string was a list-like sequence of 0 or more characters. But what does that mean?
We haven't covered lists yet, so its hard to describe the differences, but let's just say there are things you can do with lists in python that you cant do with strings. Most of these relate to the idea that strings are "immutable" in python- once a string variable is created, it cant be changed. It can be destroyed, and a new variable created using the same name, but the original variable hasn't changed, it been replaced. Lists, however, are "mutable", you change them all you like. So there are things you can do to lists (list methods) that you can't do to strings if it involves change. Even though we haven't covered them, list methods like remove(), insert() or append() (which all do what you'd expect them to do) are not allowed as string methods.
So, that's why strings are "list-like" but not lists.

"But wait Shock! You just gave us an exercise where we made all sorts of changes to a string, and now you say you can't change them?". Well, no. The string methods we used did not change the string itself, but when we used them in a print statement like:

```
print(name.upper())
```

the string method makes a new variable which is your name in uppercase and gives that to the print function to be printed. Your original name variable remains unchanged which you can confirm by

```
print(name)
```

which is not all uppercase.