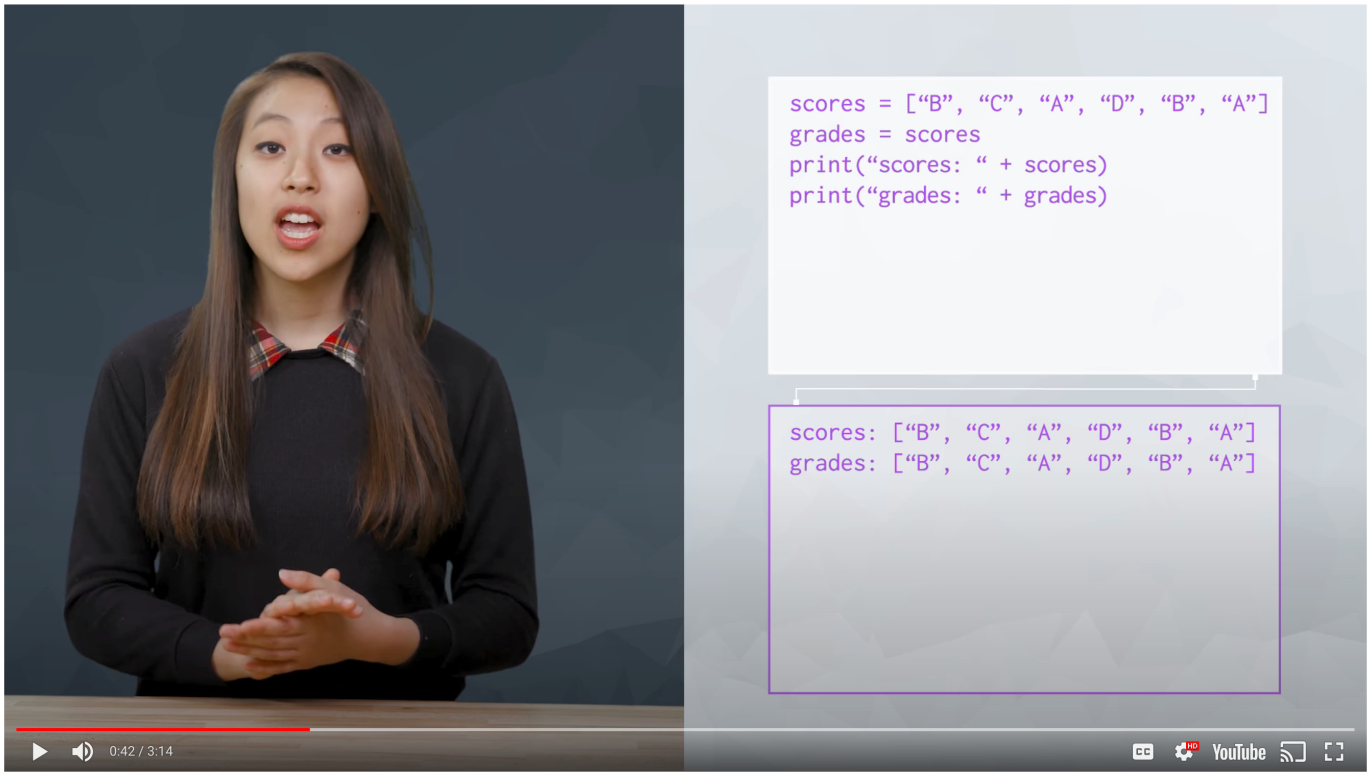
List Methods



```
scores = ["B", "C", "A", "D", "B", "A"]
grades = scores
print("scores: " + scores)
print("grades: " + grades)
```

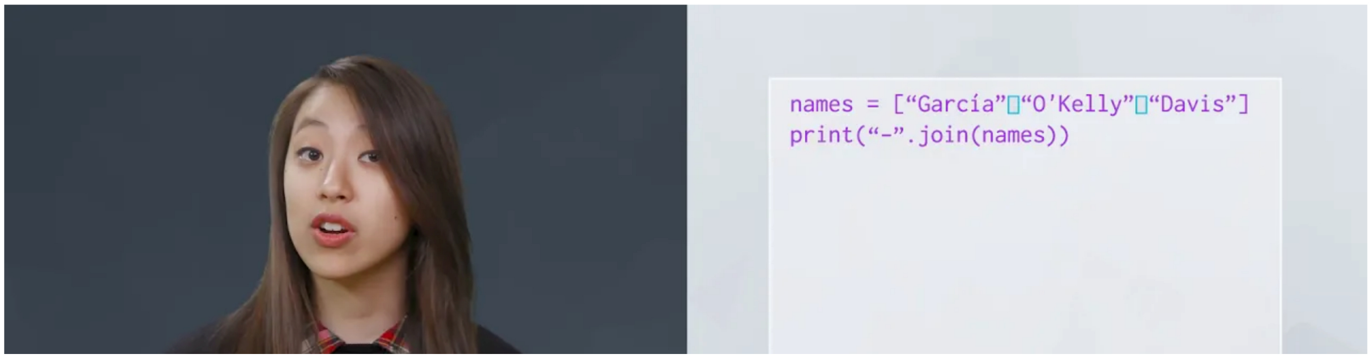
```
scores: ["B", "C", "A", "D", "B", "A"]
grades: ["B", "C", "A", "D", "B", "A"]
```

Correction: In the above video, at timestamp 0:42, the code should read

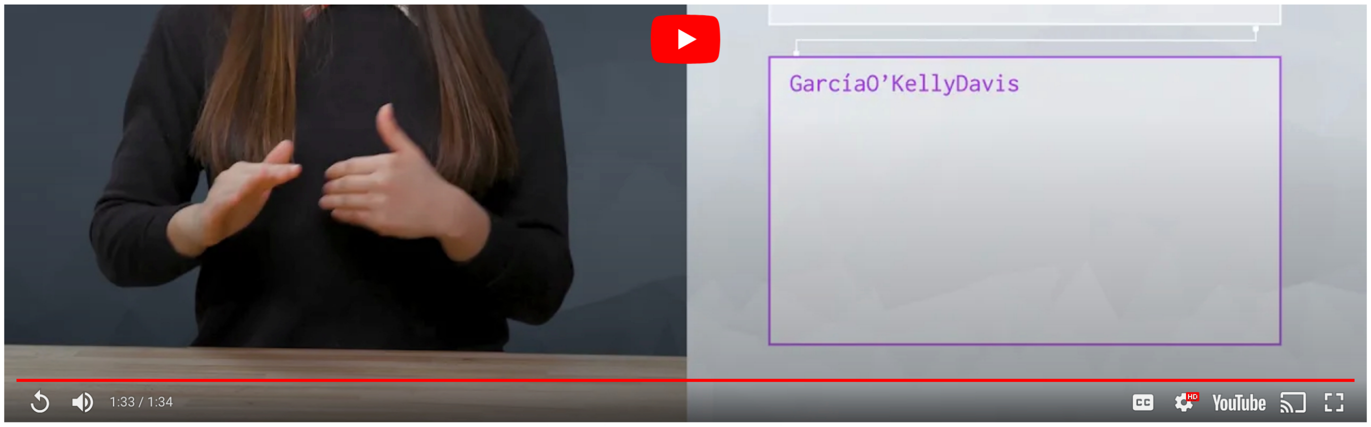
`print("scores: " + str(scores))`
`print("grades: " + str(grades))`

Useful Functions for Lists I

1. `len()` returns how many elements are in a list.
2. `max()` returns the greatest element of the list. How the greatest element is determined depends on what type of objects are in the list. The maximum element in a list of numbers is the largest number. The maximum element in a list of strings is the element that would occur last if the list were sorted alphabetically. This works because the the `max()` function is defined in terms of the greater than comparison operator. The `max()` function is undefined for lists that contain elements from different, incomparable types.
3. `min()` returns the smallest element in a list. min is the opposite of max, which returns the largest element in a list.
4. `sorted()` returns a copy of a list in order from smallest to largest, leaving the list unchanged. Note again that for string objects, sorted smallest to largest means sorting in alphabetical order.



```
names = ["García"▯"O'Kelly"▯"Davis"]
print("-".join(names))
```



Useful Functions for Lists II

`join` method

Join is a string method that takes a list of strings as an argument, and returns a string consisting of the list elements joined by a separator string.

```
new_str = "\n".join(["fore", "aft", "starboard", "port"])
print(new_str)
```

Output:

```
fore
aft
starboard
port
```

In this example we use the string `"\n"` as the separator so that there is a newline between each element. We can also use other strings as separators with .join. Here we use a hyphen.

```
name = "-".join(["García", "O'Kelly", "Davis"])
print(name)
```

Output:

```
García-O'Kelly-Davis
```

It is important to remember to separate each of the items in the list you are joining with a comma (,). Forgetting to do so will not trigger an error, but will also give you unexpected results.

`append` method

A helpful method called `append` adds an element to the end of a list.

```
letters = ['a', 'b', 'c', 'd']
letters.append('z')
print(letters)
```

Output:

```
['a', 'b', 'c', 'd', 'z']
```

Try It Out!

In the beginning of the first video, you saw how the behaviour of variables containing mutable and immutable objects is very different and might even seem surprising at times! Experiment, use the print functions and double-check your work where you can, to make sure that your programs correctly keep track of their data. While you experiment with lists, try out some of the useful functions above.