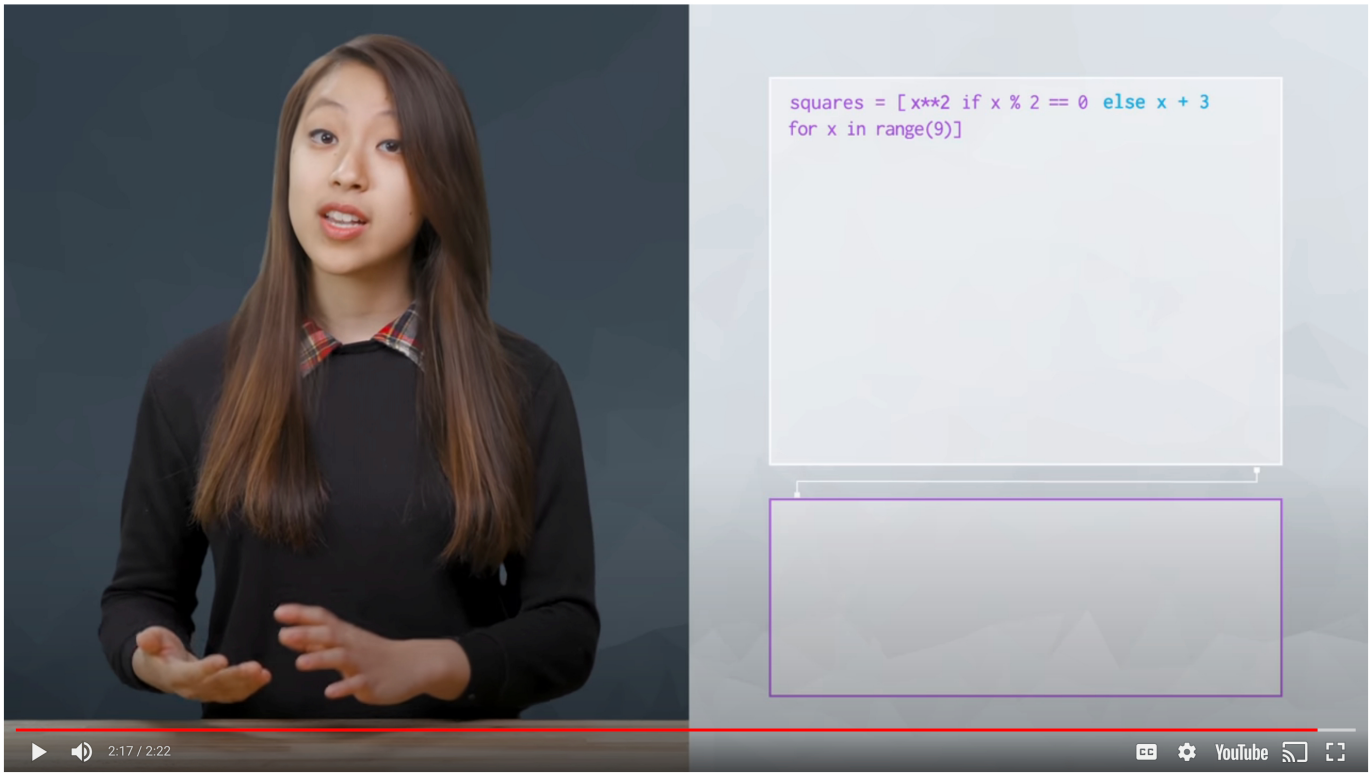
List Comprehensions



List Comprehensions

In Python, you can create lists really quickly and concisely with list comprehensions. This example from earlier:

```
capitalized_cities = []
for city in cities:
    capitalized_cities.append(city.title())
```

can be reduced to:

```
capitalized_cities = [city.title() for city in cities]
```

List comprehensions allow us to create a list using a `for` loop in one step.

You create a list comprehension with brackets `[]`, including an expression to evaluate for each element in an iterable. This list comprehension above calls `city.title()` for each element `city` in `cities`, to create each element in the new list, `capitalized_cities`.

Conditionals in List Comprehensions

You can also add conditionals to list comprehensions (listcomps). After the iterable, you can use the `if` keyword to check a condition in each iteration.

```
squares = [x**2 for x in range(9) if x % 2 == 0]
```

The code above sets `squares` equal to the list [0, 4, 16, 36, 64], as x to the power of 2 is only evaluated if x is even. If you want to add an `else`, you will get a syntax error doing this.

```
squares = [x**2 for x in range(9) if x % 2 == 0 else x + 3]
```

If you would like to add `else`, you have to move the conditionals to the beginning of the

listcomp, right after the expression, like this.

```
squares = [x**2 if x % 2 == 0 else x + 3 for x in range(9)]
```

List comprehensions are not found in other languages, but are very common in Python.

Previous Next Give Page Feedback