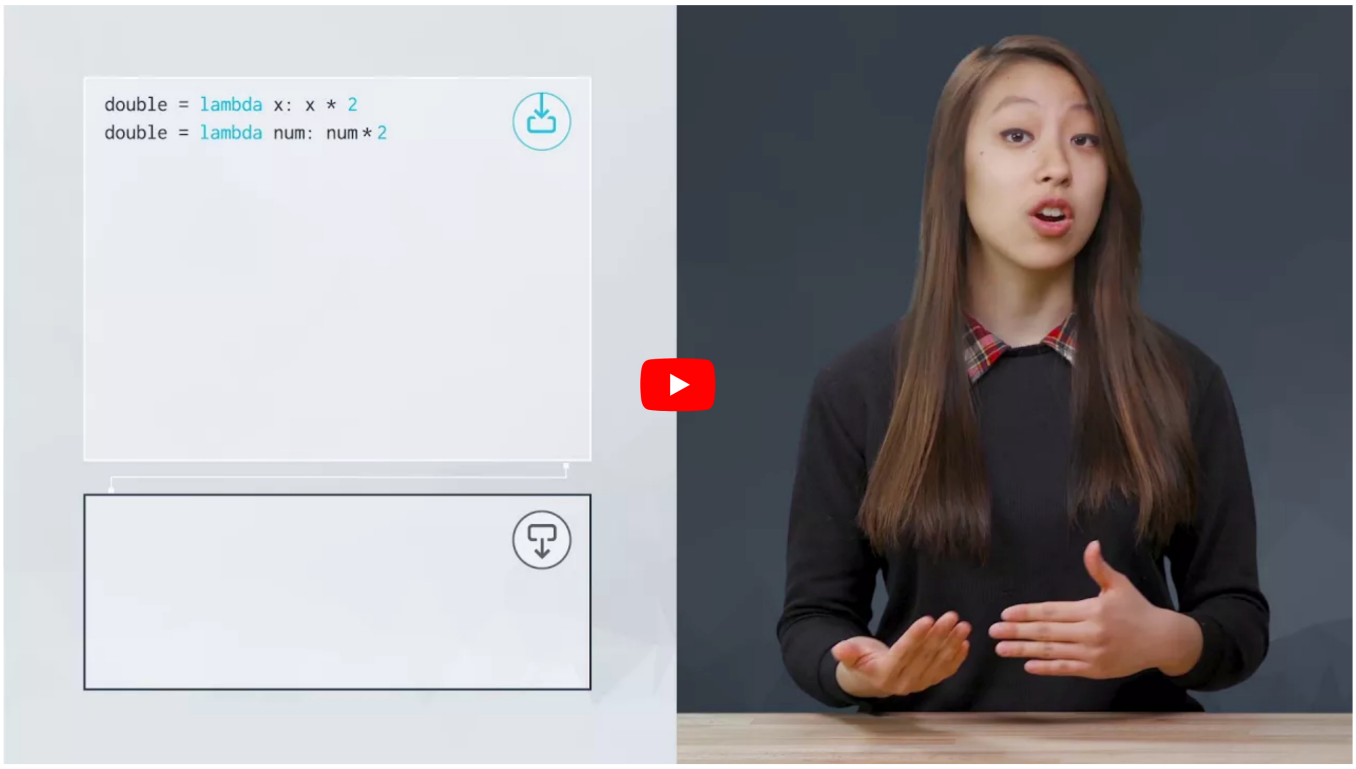
Lambda Expressions



Lambda Expressions

You can use lambda expressions to create anonymous functions. That is, functions that don't have a name. They are helpful for creating quick functions that aren't needed later in your code. This can be especially useful for higher order functions, or functions that take in other functions as arguments.

With a lambda expression, this function:

```
def multiply(x, y):
    return x * y
```

can be reduced to:

```
multiply = lambda x, y: x * y
```

Both of these functions are used in the same way. In either case, we can call `multiply` like this:

```
multiply(4, 7)
```

This returns 28.

Components of a Lambda Function

1. The `lambda` keyword is used to indicate that this is a lambda expression.
2. Following `lambda` are one or more arguments for the anonymous function separated by commas, followed by a colon `:`. Similar to functions, the way the arguments are named in a lambda expression is arbitrary.
3. Last is an expression that is evaluated and returned in this function. This is a lot like an expression you might see as a return statement in a function.

With this structure, lambda expressions aren't ideal for complex functions, but can be

With this structure, lambda expressions aren't ideal for complex functions, but can be very useful for short, simple functions.

Previous Next

Give Page Feedback