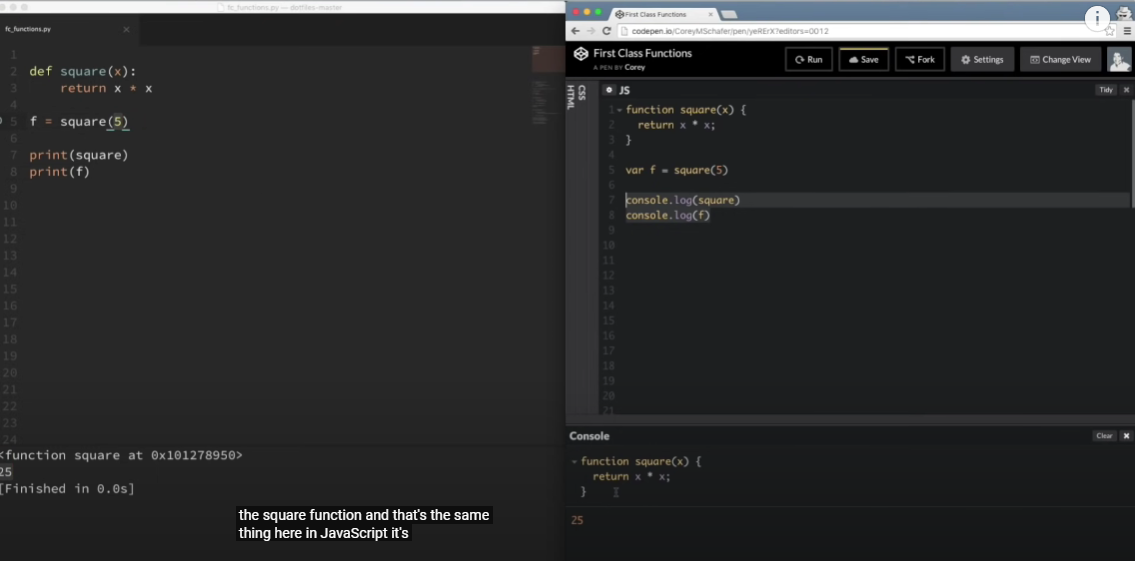
Hey everybody, how's it going? In this programming terms video, we'll be going over the term first-class functions. Now we're going to look at first-class functions in both JavaScript and in Python, so that you can see the concept in two different languages, and that may help the term sink in a little bit. But don't worry, if you don't know either of these languages, what's important to take away from the video is the concept of first-class functions and not the specific syntax of any one language. So understanding first-class functions will help you understand other terms such as higher order functions, currying, and closures. So, it's a very useful term to know. Okay, so what is a first class function? If you look up the definition on Wikipedia, one of the first lines you'll see is that it says "A programming language is said to have first class functions if it treats functions as first-class citizens."

And then if you look up first class citizens for programming, it says that "A first class citizen (sometimes called a first-class objects) in a programming language is an entity which supports all the operations generally available to other entities. These operations typically include being passed as an argument, returned from a function, and assigned to a variable.

Okay, so what exactly does that mean? So that means that we should be able to treat functions just like any other object or variable. So, I think this will be more clear once we take a look at these coding examples.

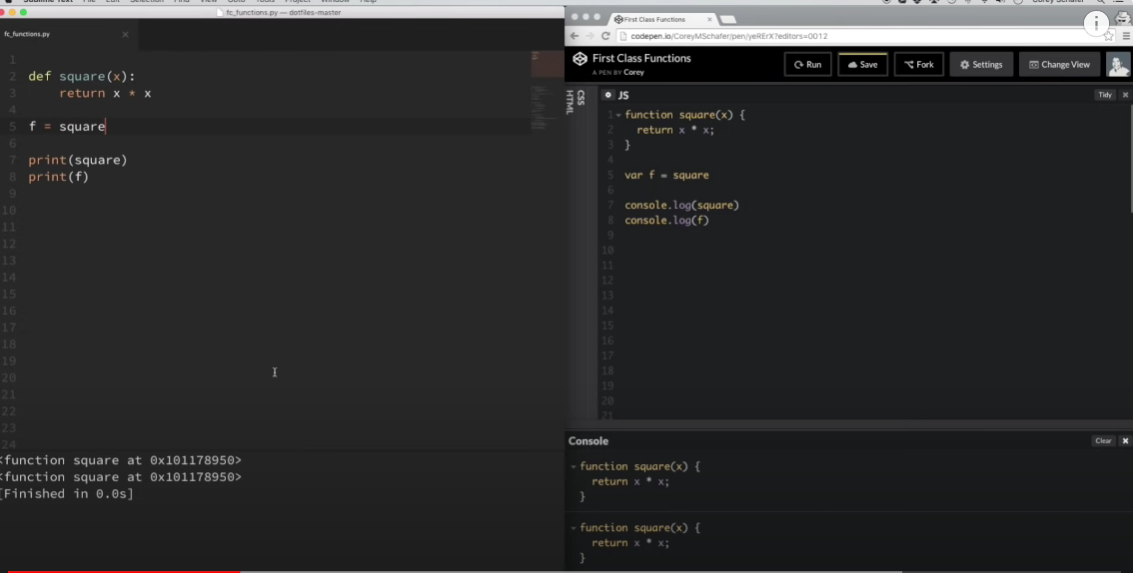
So here on the left I have Python, and here on the right I have JavaScript. So you can see this concept working in two different languages.

Okay, so first of all, let's see what it means to assign a function to a variable. Now this doesn't mean that we are assigning the result of the function to a variable. This is where some people get confused. So, in the example I have pulled up here, this is what we're used to doing with functions. So, we have a function called square, and we're setting this variable F equal to the square of five. Now down here on this line, I'm printing out the function square, and I'm printing out this variable F. So, I'm gonna go ahead and run that, and then I'm gonna do the same thing in JavaScript, and I'll go ahead and run that. So, you can see whenever I print out the function square, you can see that it prints out that it is a function called square. Then when I print out F, it's printing out 25, and that's because we passed in five to the square function. And that's the same thing here in JavaScript. It's printing out the function square that we logged out and we're also logging f which is the square of five.



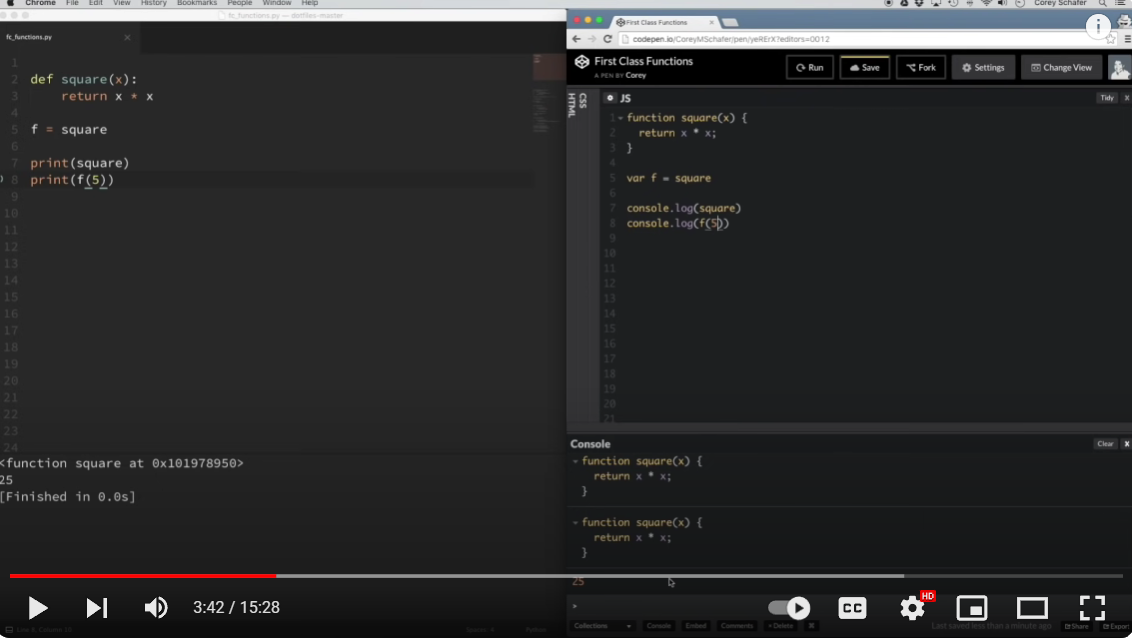
Now what a lot of people don't realize is that we can remove these parentheses here and set our variable f to the square function. So, if I take out these parentheses and save that and let me do the same thing here in JavaScript. And now we did take out those parentheses. A lot of people get tripped up at this point and they only take out the arguments and they leave in the parentheses, but you don't want to do that because the parentheses mean that we're going to execute the function.

We don't want to execute the function, we just want to set our variable f equal to the function, not executed. So now that I have my variable f equal to square, what if I run this now? So let me run this code. So now you can see that when it prints out f, f is equal to our square function. So, this is one of the aspects of what it means to be a first-class function. So now we can treat the variable f as a function.



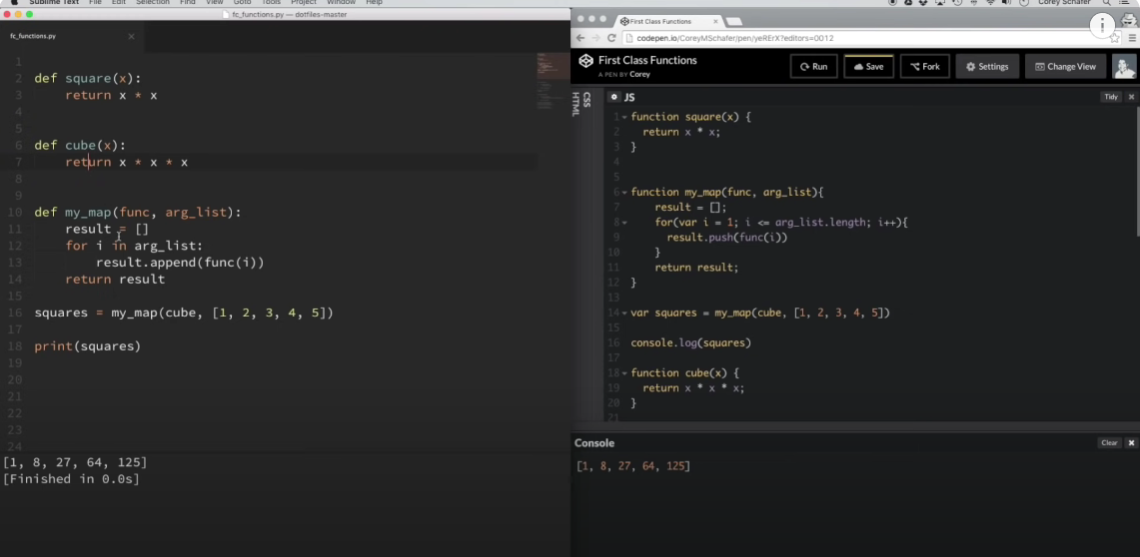
So now I can use f just like I would use square. So instead of printing out f, what if I print out f of five and I run it just like a function? So, you can see now I get 25 there.

If I do the same thing in JavaScript, then I will get 25 there as well



So, so far, we assigned a function to a variable. But we can also pass functions as arguments and return functions as the result of other functions. So let's take a look at both of those examples. And by the way, if a function accepts other functions as arguments or returns functions as their result, that's what you call a higher order function. So first, let's look at an example of passing a function as an argument to another function. So a great example of this is the map function in both Python and in JavaScript. So, the map function takes a function and an array as its arguments, and it runs each value of that array through the provided function, and then returns a new array of those results.

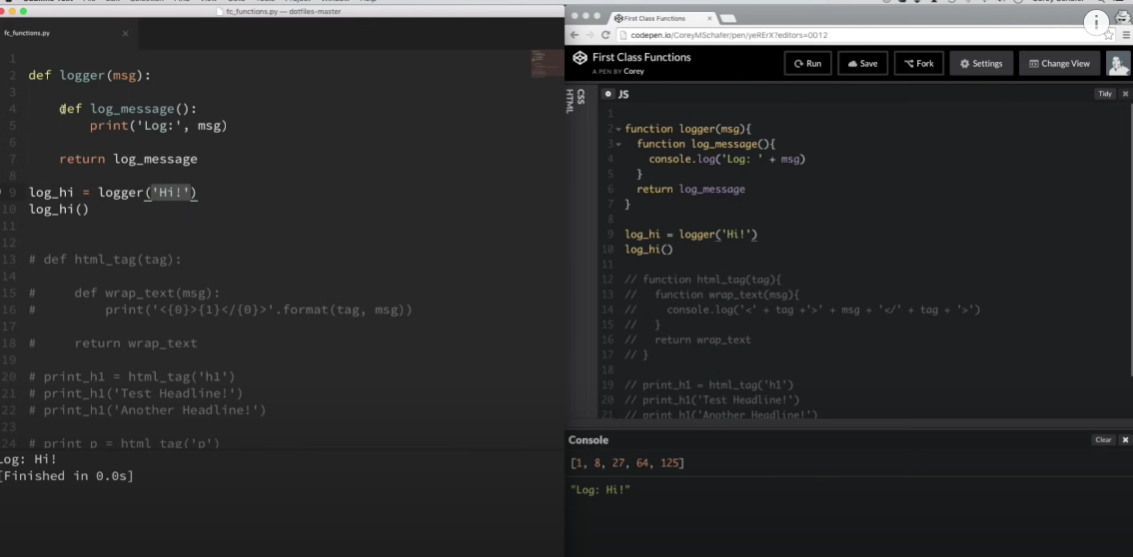
So, I think that'll be more clear if we take a look at an example here. So I have a sample map function here. This isn't the built-in map function, but what we're going to do is just build our own from scratch, so that way we can see exactly what's going on. Okay, so let me take out all these old variables here and uncomment out this, and I'll push these down for now. And let me do the same thing over here in JavaScript. So let me go ahead and move these up. Okay, and let me go ahead and uncomment out this code here. Okay, so we have a custom-built map function here, which is gonna behave kinda like the built-in map function. And what we're doing is we are taking in a function as an argument, and then we're taking in an array as an argument. And then here we're going to create an empty result array, loop through all of the items in the passed in array, and run each of those items through the function that we passed in as an argument, and append that to the result. And then after we're done, we're going to return the result. So you can see here I have an example where I'm setting this squares variable equal to the my map function, I'm passing in the square function that we created up here, and then I'm also passing in a list of values from one to five. And again, both here in Python and in JavaScript, notice that I'm not using the parentheses when I pass this square function in to this other function, because adding the parentheses would try to execute the function, and that's not what we're doing here. We aren't executing the function, until this line here within our map function. So let me go ahead and run this so that you can see the results whenever we pass in this list and the square function to our map function. So you can see here we printed out the squares. It took this list of one through five and squared all of those numbers. And if I do that over here in JavaScript, then it does the same thing. And actually to get the same result here, I'm gonna have to start at one. and go to the end of that. So let me save that and rerun it. Okay, so now you can see that we're getting the same results. Okay, so this is useful because now we can pass any function into our custom map function that we created.



So, for example, I have a cube function down here. Now let me grab this cube function and just pass it in to our map function and I'll do the same thing over here in JavaScript. So let me clear that out and rerun that. And in my Python example, I'm going to have to actually cut this out and move it above here by the square function. So let me save that and rerun it. So you can see all that we did differently was pass in this new function as an argument to our map function, and then it used that function within here to execute on each of these values instead of the square function. So it's useful to be able to pass around functions like this.

Okay, so lastly, let's see what it looks like to return a function from another function. So that is one of the aspects of what it means to be a first-class function. Now this seems to be what trips people up the most, and I have to admit I get tripped up on this all the time too. Because it can get complicated to keep track of everything. And sometimes it can be hard to visualize exactly what's going on all at once. But let's take a look at some simple examples and see if we can see what this looks like. So let me go ahead and just remove all the code that I had. and uncomment out these other examples that I have down here.

Okay, so first let's take a look at this really simple logger function that I have. So I'm gonna uncomment out these and save these.

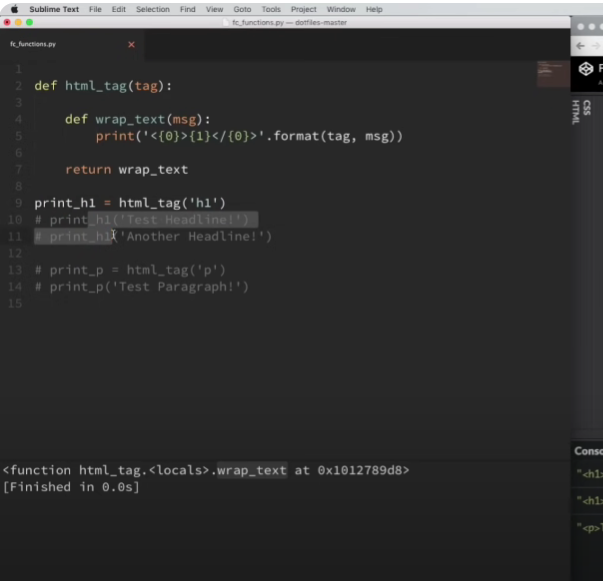


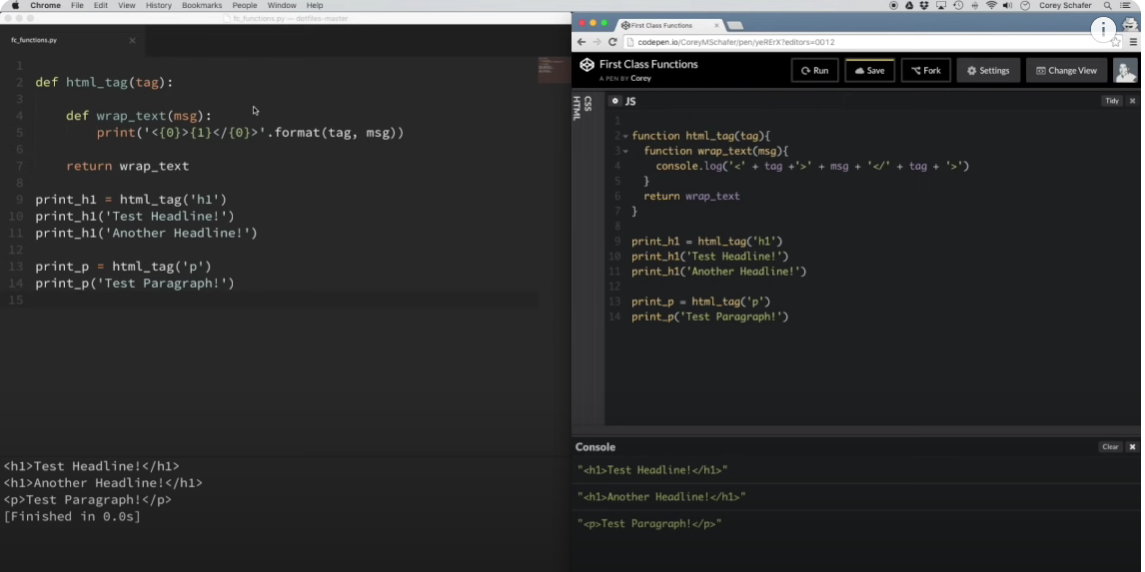
Okay, so this is one of the simpler examples that I could think of to start out with. So we have this function here called logger. And this logger function takes in an argument, a message argument, and then within this function we have another function called logMessage, and this function doesn't take in any arguments. And all it does when you execute this logMessage function is that it prints out this log statement with the message that we passed in to our logger function. And now down here at the bottom, we are returning this logMessage function that is within our logger function.

Now I know that might sound confusing, but let's go ahead and run an example and walk through it and see if we can see what's going on. Okay, so let me go ahead and execute this in both Python and in JavaScript and let's walk through exactly what just happened. So what we did is we set this log high variable equal to our logger function and we passed in a message of high. And then, so this came in and it saw that we had a function created here and then we returned our that function, and again, no parentheses, so that function did not execute at that time. So now our log high variable is equal to this function. That's what got returned from this.

So now we can run this log high variable just like it's a function, because it is a function. So now we can treat this variable just like this function here, and it doesn't take in any arguments. So if we put in these parentheses here, and what it's gonna do is it's going to execute this log message, and then it came in here and printed out the log with our message. So one thing important to point out here is that from this step to when we executed the function at this step, that it remembered our initial message that we passed in to this initial logger function. Now this is what we call a closure, but again, I'll go in more in depth into closures in a future video,

But this just kind of knocks the point home that understanding these first class functions will help you understand these more complex terms like closures and higher order functions. Okay, so why would returning a function from another function like this be useful?





Well, I tried to, some of the practical examples of this can get really complicated really fast. So I tried to make a semi-practical example that we could look at and still understand pretty well. So I'm just gonna remove this logger function that I have here and uncomment out the rest of the code that we have here. So let me do this in Python also and I'll just go ahead and run that and over here I'm gonna clear out the console and run that code.

Okay, so let's see if we can understand what's going on here. So I have a function here called HTML tag which takes in an argument called tag and within this function, I have another function called wrapText, and wrapText takes in an argument called message. And when you execute this wrapText function, it prints out the tag within opening brackets here, and then it prints out the message, and then it prints out the tag again in closing brackets, and it does the same thing here in JavaScript. So then, back to our HTML tag function outside of the wrapText, what we're doing is returning this wrap text function. Okay, so now let's look at our examples. So I have this variable called print h1, and I'm setting this equal to our HTML tag function, and I'm passing in a string of h1.

So at this point, whenever I do that, actually to kind of explain this a little bit better, I'm gonna comment it out these. So at this point, it's not actually doing anything yet. So you can see if I rerun this code, it hasn't actually done anything. And if I come down here and just print out this print h1 variable, then you can see that it's just equal to this wrap text function. So it's just waiting to be executed. So now we can use this print h1 variable just like it's a function because it is a function. And if I uncomment out these lines here, Now remember, the function that we returned, this wrap text function, it takes in an argument called message. So I can pass in, I'm gonna execute this print h1 variable that we just created, and I'm gonna pass in the message of test headline.

And then down here, I'm running this code again, and I'm passing in a message of another headline. So whenever I run this, now you can see that it ran our wrap text function. and it remembered the tag that we passed in before. So it prints out the tag that we passed in to HTML tag function here, and then it also printed out our message that we passed in to each of these variables, which is equal to our wrap text function. And everything that I'm describing over here in Python is also the exact same thing that's happening over here in JavaScript. You can see it's the exact same process. Setting this variable equal to HTML tag function passing in the tag of h1. It returns the wrap text function waiting for this message argument.

So now our print h1 variable is equal to that function that got returned waiting for that message. So now we can execute this just like any other function. We passed in our message and it printed out our tags with the message and then the closing tags. And then just to illustrate the point a little further, You can see that I did the same thing down here, starting off with a paragraph tag instead. So you can see that that worked as well. Now there's a lot more practical examples that people use this functionality for. A lot of people use this for logging, and in Python it's similar to how we use decorators. And you can see how returning functions like this can get extremely complicated and confusing because it's easy to lose track of exactly what's going on.

And like I said, I'll go further into closures in a future video, but to understand those, we really need to have a grasp on these first class functions and how we can use these functions, not just execute the functions, but we can also pass them around just like any other variable. We can assign them to variables, we can pass them in as arguments like we did in our previous example, and we can return these functions from other functions. So hopefully after this video, you get a sense of all the different things that you can do with those. So I think that's going to do it for this video. If you do have any questions, just feel free to ask in the comment section below.

Be sure to subscribe for future videos and thank you all for watching.