BRIAN YU: All right, welcome, everyone, to Web Programming with Python and JavaScript. My name is Brian Yu. And in this course, we'll dive into the design and implementation of web applications.

In lectures, we'll have an opportunity to discuss and explore many of the ideas and tools and languages that are central to modern web programming. And through hands-on projects, you'll have an opportunity to take those ideas and put them into practice, designing multiple web applications of your own, culminating in a final project of your own choosing. Throughout the term, we'll cover a number of topics in this world of web programming, beginning with HTML5 and CSS3, two of the languages that are central to the understanding of web pages. HTML is a language we'll use to describe the structure of a web page. And CSS is the language we'll use to describe the style of a web page, the colors and the fonts and the layouts and the spacing that make the web page look exactly the way we want it to look.

After that, we'll turn our attention to Git, a tool not specific to web programming per se, but that we can use in order to version control our programs, to keep track of the different changes we make to our web programs, and to allow us to be able to work on various different parts of the web application at the same time before merging those pieces back together. After that, we'll take a look at Python, one of the first main languages that we're going to be exploring in the course, which is the language that we are going to use in order to build our web applications. Specifically, we'll use Python using a framework called Django.

Django is a web programming framework written in the Python programming language that we're going to use to make it easy to design and develop our web applications Django in particular makes it easy to design web applications that interact with data. So after that, we'll turn our attention to SQL, a language that we can use to interact with databases, in particular looking at how Django allows us to use models and migrations to interact with data and allow users to interact with data all the more easily.

Next, we'll turn our attention to the second of the main programming languages that we'll be exploring in this class, JavaScript, and looking at how we can use JavaScript to run in users' web browsers to make web pages just a little bit more interactive. In particular, we'll use JavaScript in the context of user interfaces, looking at modern user interfaces and exploring how it is that those interfaces work and how we can develop those user interfaces with a combination of Python and JavaScript.

Next, we'll turn our attention to testing and CI/CD, or Continuous Integration and Continuous Delivery, which are tools that we can use and software best practices to make sure that we're able to design and develop code more efficiently. And testing in particular makes sure that as we make changes to our code, we're not breaking existing parts of our web application by making sure that we have a whole suite of tests that we can use to ensure that our web application is always behaving as it should.

And finally, we'll turn our attention to scalability and security on the internet, thinking about what happens as our web application grows larger. As more and more different users start to use our web application, how do we load balance between those people? And what do we need to change about our database to make sure lots of users are able to connect to our web application at the same time? Moreover, we'll look at the security implications behind designing our web applications. What might an adversary do if we're not careful? And how should we proactively be designing our web application to make sure that it's secure?

But today, we begin our conversation with HTML and CSS, two of the languages that are foundational to understanding web pages and how web browsers are able to display those web pages. And we'll start with each HTML, or HyperText Markup Language, which is a language that we can use to describe the structure of the web page, all of the buttons and the text and the forms and other parts of the web page that the user ultimately sees and interacts with.

Our very first HTML page is going to look a little something like this. It's going to be text-based code that we write that a web browser, like Safari or Chrome or Firefox, is then able to look at, parse, understand, and display to the user. So let's take a look at this page one line at a time and get an understanding for how it works.

Even if you don't quite understand all the nuances of the syntax, there are probably a couple of things that stand out to you. You might notice the word title, which probably reflects the title of the web page, for example, which in this case appears to be the word hello. And then down further below, we see that we have the body of the web page that seems to contain the words hello world.

So what is this web page actually going to look like? Well, let's take a look at it. We'll go ahead and open up a text editor. You can use any text editor you want. But for this course, I'm going to use Microsoft's Visual Studio Code. And I'm going to open up a new file that I'm just going to call hello.html.

Inside of hello.html I'm going to write the same HTML that we just saw a moment ago. And we'll explain each of these lines in due time. But recall that we had a title of the page that said something like hello and then a body of the page where we said something like hello world, for example.

So this is our very first HTML page. And if I go ahead and open that HTML page as my opening hello.html HTML, for example, inside of a web browser, what I'll see is something like this. In the body of the page, I see the words hello world. And if you notice up here at the top of my web browser, I see the title bar, where I have the title for this page, which in this case is just the word hello.

So this is our very first web program that we've been able to develop just using HTML. And now let's explore in more detail how exactly this program works. So here again was the web page that we were just looking at. And this very first line here, DOCTYPE html is what we might call a DOCTYPE declaration. It's a way of telling the web browser what version of HTML we're using in this particular web page, because depending on the version of HTML, the web browser might want to display different information or it might need to parse the page a little bit differently. Each version of HTML has had a slightly different way of indicating that version. But this line here, DOCTYPE html is our way of saying that this HTML page is written using HTML5, the latest version of HTML.

After that our HTML page is structured as a series of nested HTML elements, where an HTML element describes something on the page. And we might have elements that are inside of other elements. Each of those elements is indicated by what we're going to call an HTML tag, enclosed using those angled brackets. And right here, we'll see the beginning of the HTML tag, which means that this is the beginning of the HTML content of our page.

Down below this slash HTML means that this is the end of the HTML content of the page. And in between is the actual HTML content of the page, which might include other HTML elements. You might also notice that in this HTML tag we've specified what we're going to call an HTML attribute, some additional information that we're giving about this tag. In particular, we're giving it a lang, or language, attribute, which is equal to en, or English. This just tells the web browser or anyone looking at the HTML of this page that this page is written in a language, and the language it's written in is English. And this is helpful for search engines, for example. When they're looking through many different web pages trying to figure out what language each web page is in we can just tell the search engine or anyone else who's looking at the page that this page is written in English.

Now, inside of the HTML body of the page, we have a number of different elements that are going to describe what we want on this page, starting with the head section of the web page, which describes stuff not in the main body of the web page, the part of the web page the user sees, but other information about the web page that's going to be helpful or useful for web browsers to know about. For example, one important thing that a web browser needs to know is, what is the title of the web page?

And here, we see a title tag, again, indicated by the word title in those angled brackets, followed by the end of the title tag, indicated by a slash before the title. And in between the two title tags is the word hello, which means the title of this page should be the word hello. And that's all the information we'll have in the head of the page. We'll add more information there later, but for now all the web page needs to know is that it has a title and the title is the word hello.

Next up comes the body of the page, again, indicated by a body tag and that ends with a tag with slash body, meaning this is the end of the body of the page. And the body of the page, again, is just the visible part of the page that the user can see. And what do we want inside the body of the page? For now, we just want the text, hello world. And that's the information that's going to be displayed when someone visits this web page.

And so that's all there really is to this HTML page. We specified in the header that there's a title of the page called hello. And inside the body, we're saying the page should say the words hello world.

And if you want to visually think about the way that all these HTML elements are structured, it can sometimes be helpful to think about an HTML page in terms of a tree-like structure that we call the document object model, or DOM. And so, here, for example, is what the DOM for this web page might actually look like. Here on the left is the HTML content that we just saw a moment ago. And over here on the right is the DOM, the document object model, the tree-like structure that describes how all of these HTML elements are related to each other.

So we start up here with the HTML element. And this parent element, so to speak, has two child elements within it, a head element and a body element. As we can see here, we're inside of HTML. We have a head section and a body section.

And the indentation here that we're including in the HTML text, it's not strictly necessary. The web browser doesn't care what the indentation is. But it can be helpful for someone who's reading the page just to see the indentation to understand visually that the head is inside of the HTML element and the body is inside of the HTML element too.

So inside of the head element, we have a title element. And inside of the title element is just the text, the word hello. And likewise, inside of the body element, we also have some text, the text hello world. So thinking about HTML and HTML documents in terms of this structure can be helpful for understanding which HTML elements are inside of which other HTML elements. And that's going to make it easier for us to reason about these pages later on. And especially as we later transition into the world of JavaScript, JavaScript is going to make it all the more powerful and give us the ability to actually modify parts of this DOM as well. But we'll certainly get to that in due time.

So now, let's take a look at some of the other common HTML tags and HTML elements that we might be interacting with in our web page. And we'll start by thinking about HTML headings, so big banners at the top of the page, for example, some headline that describes what a page is about. So I'll go ahead into my text editor and create a new file that I'll call headings.html.

And the structure of this page is going to be pretty similar to the pages that we've seen before already. So I'm going to start by just using the hello.html text and paste it in here. I'll change the title of the page. Instead of hello, we'll go ahead and call it headings. But inside the body of this page now, I want something a little bit different. I'm going to, inside the body the page, use an h1 element and say this is a heading, for example.

So h1 is a tag that I can use to create a large heading at the top of my page, like for the title of the page, for example. So if I open up headings.html. I might see something that looks like this, a big heading at the top of my page that says, this is a heading.

h1, where the h stands for heading and the 1 stands for the largest possible heading. And in fact, HTML gives us a number of different tags that we can use in order to create headings of various sizes. So, for example, I could also say h2 inside of which I say, this is a smaller heading. If h1 is the largest heading, h2 is the second largest heading.

So if I load this page, for example, I now see the h1 at the very top. This is the big heading. And then beneath that, I see this is a smaller heading, the h2. And it turns out there's also h3, h4, h5, all the way down to h6, which is the smallest heading, such that if I load this page now, I have a big heading, a smaller one, and then here's the smallest.

So we can often use these h1, h2, h3 tags just for visually organizing text inside of a page. If I want the title of the page, but also I want titles for each of the various different sections and subsections that might be contained within that page as well. So those are headings.

And now, let's also take a look at some other elements that we might want to add. On web pages, we see not just titles and not just text, but we might also see lists, for example. Like if you've ever used a to do list program on a web page, for example, you might see a list of things that you need to do or other web pages might display lists of information.

And it turns out that HTML has two basic types of lists. We have ordered lists for things that are in a particular order, like item number 1, item number 2, item number 3. And we have unordered lists for lists that don't have any particular order. So just bullet point, bullet point, bullet point, for example. And both are quite easy to use.

I'll go ahead and create a new file. And we'll call this lists.html. And again, in list.html, I'll copy the same structure from hello.html. We're again going to have DOCTYPE html just to indicate the version of HTML. Most of the heading is the same. I'm just going to change the title from Hello to Lists. And then we're going to replace the body of this page to show some different information here.

So let me first show what an ordered list might look like, something that has numbers, 1, 2, 3. An ordered as an HTML tag is just ol, ol for ordered list. So I can add a tag that says ol.

And now inside of my ol element, my ordered list element, I need a new element for every list item. List item we're going to abbreviate to just li. So the li tag in HTML is what we're going to use to designate an item inside of an HTML list.

So here, for example, I could say li and then first item. Then I could do the same thing, li second item and then again li third item. So what I have here are some elements and then elements nested within other elements. I have an ordered list element inside of which are three other HTML elements, three list items that are each indicating each of the individual items that are inside of my HTML list.

I can now open this up by opening lists.html. And this is what I see. I see an ordered list, where I have item number 1, first item, second item, third item. Note that I didn't actually need to in the HTML anywhere specify the number 1, the number 2, and the number 3. When my web browser reads this should be an order list, my web browser, Chrome in this case, just adds those numbers in for me because it knows what an ordered list means and it knows how to take the HTML that I've written and display it in the way that I intend to the user.

Now, in addition to ordered list that all have numbers, 1, 2, 3, we also have unordered lists that are just bullet points, bullet points of information. So I could, up above, add some more content to this HTML page. I can say here is an unordered list.

And just as an ordered list we represented using the ol tag in HTML, ol standing for ordered list, likewise we can use the ul tag in HTML to create an unordered list, u for unordered. So here, we're going to add a ul tag. And again, my text editor here is automatically adding the closing tag here, this slash ul, meaning the end of the unordered list. And many text editors will do this now just so you, the programmer, don't forget to add that.

And now inside of this unordered list, we're again going to have some list items. Also, using the li tag, here is one item. And here is another item. And here is yet another item.

If I go ahead and refresh the page now, I'm still on list.html, I now see that on top of my ordered list, I have an unordered list, where each item instead of being numbered 1, 2, 3, is instead labeled with just bullet point, bullet point, bullet point, where each of these bullet points and each of these numbered items is a list item element, or an li. So hopefully now we can see that as we start to explore these various different HTML tags and nesting HTML tags inside of one another, we're able to create more and more interesting web pages as a result.