

It's now time to introduce methods, and just like with attributes in a class, when we define a method, we must define it within the class definition first.

And when we define a method's definition, we are defining the methods interface.

So, if you think back to function modules and forms, they have an interface where we fill in parameters. Is it, let's say, called a function module and then it will pass by parameters.

Same thing with methods. We first define the method parameters, then we must define the method return type often referred to as the method signature. And then we define any method exceptions, and we'll cover method exceptions later.

Now, methods may be designated as public or private, just like we did with attributes, so depending on where you create the method definition, whether it's in the public section or private section, will determine the visibility of the method.

Public methods can be called externally by any calling program that instantiates an object, but a private method can only be called from within another method of that class itself. It keeps things self-contained and doesn't allow anybody else access to it.

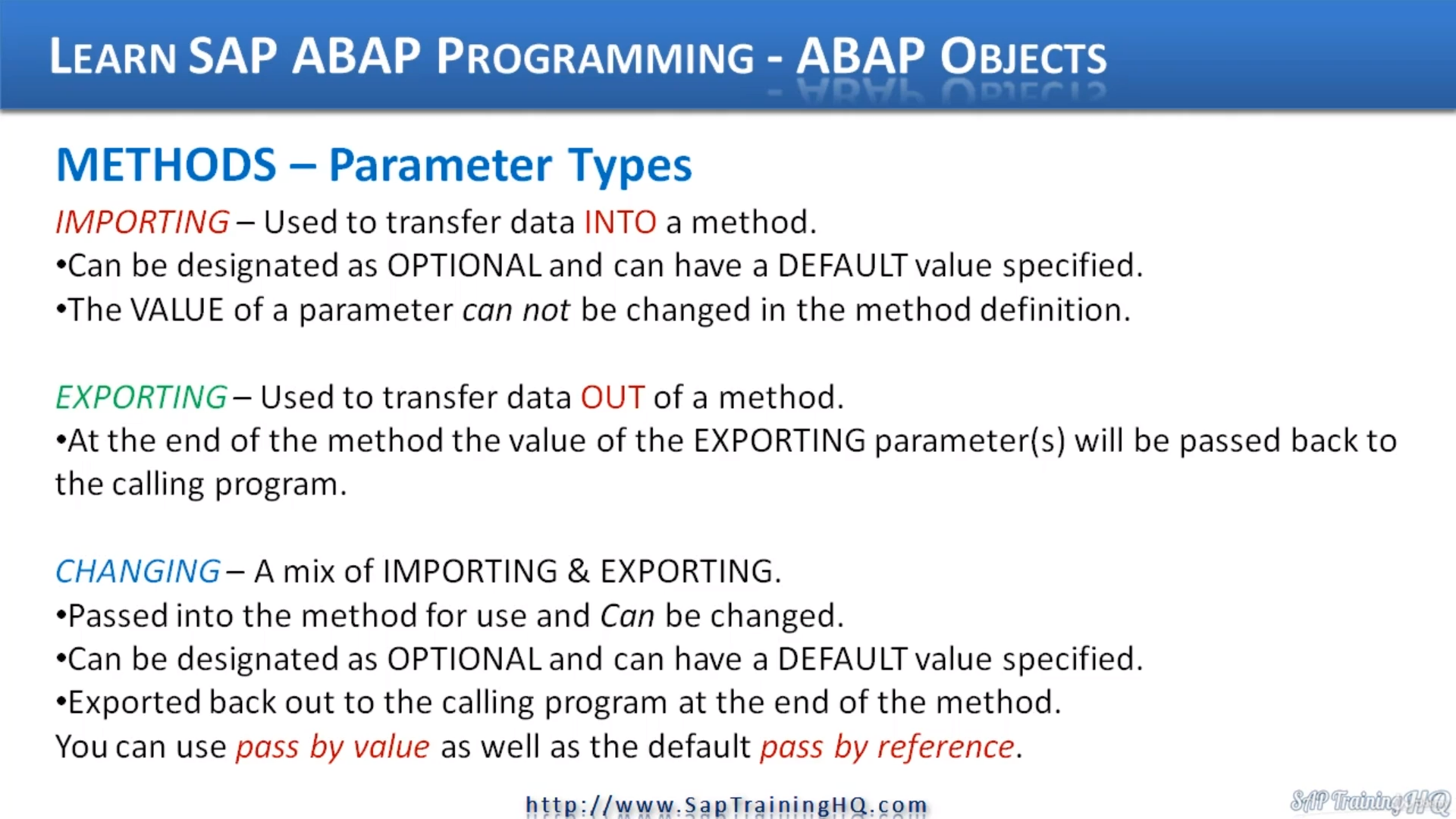
Now, just like we have with attributes where we have class data static attributes, we can also have class methods. But there's some important things to keep in mind here, class methods can only access class data.

OK, so if we make a class method, if you think about it, that means we don't have to instantiate an object first before we call a class method.

So, if we have instantiated any objects, then there's no standard, you know, normal data existing in any objects, the only data that would exist are class data, static attributes. So, it sort of makes sense that a class method can only access class data.

And because class methods don't have to be instantiated, we have two ways of calling them.

We can reference the actual class name itself or an object based on the class.



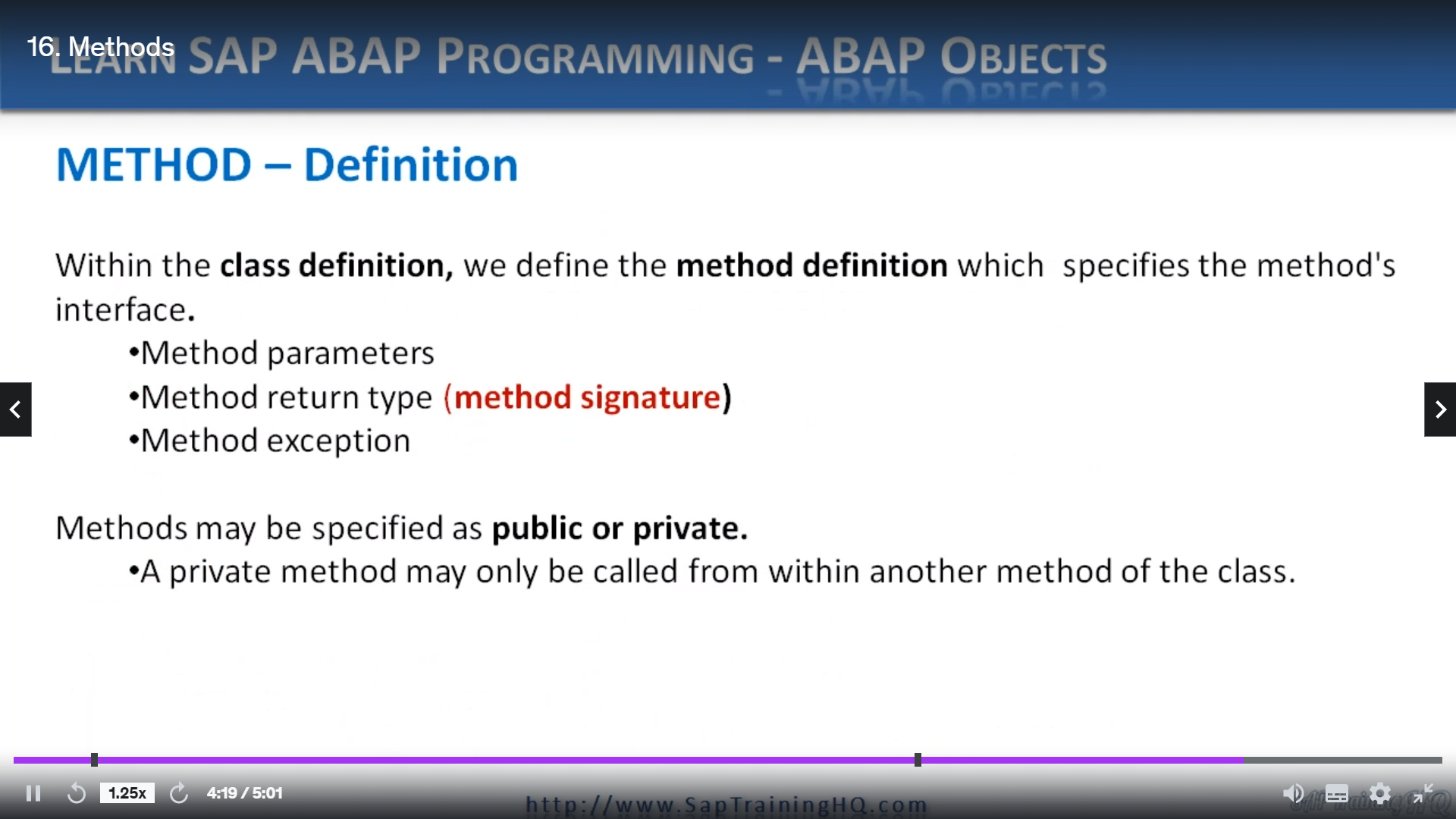
And finally, we have changing this is really a mix of importing and exporting, you know, if you're familiar with Function Module's, this will make sense.

Changing parameters are passed into the method for the method to use and can be changed, they can be designated as optional.

So, there's no need for the calling program to always define the change in parameter and they can have a default values specified as well, just like the important parameters.

Now, when we pass data back out to the method, a changing parameter gets exported back to our calling program once the method has been completed.

And if you think back to when we discussed forms earlier in this course, we also have the option here of using the pass by value as well as the pass by reference options.



Now, we've just had a description of, you know, various bits and pieces within methods and the different types of methods. I want to go over it again because it is very important. I won't spend too long in, but I just want to hammer home, you know, some of the key features of methods.

So, within a class definition, we define the method definition, which specifies the methods interface.

In the interface, we define the methods, parameters, the method return type and the method exceptions. And then for any method just with attributes, we may specify the method as public or private and a private method may only be called from within another method of that class.