

So when we talk about a data object and we talk about data objects quite a bit in about objects, these

are the variables and a data object really helps describe the type of data we are holding in the variable a lot better than the standard data type would.

On what I mean by this is we might have a data type of, let's say, NUMC eight, which just means an eight digit numeric character string.

But the data object that makes use of this NUMC could be something like employee number.

So when we define employee number as a variable, as a data object, it really helps describe what data is being held a lot better than the actual data type itself.

And then when we talk about data types in general, in ABAP, in Sap, we really have two sets of data types.

We have complete data types and incomplete data types.

Complete data types have a predefined fixed size with a specified format and an example of this will be an integer or maybe a float or a string and so on.

So in up an integer is a fixed size, a float is a fixed size, so is a string.

I think a string can go to something like two gigabytes in size.

And as you can imagine, with something like that, it's useful for maybe important files into memory that we can then process up incomplete data types.

On the other hand, do you have a format specified?

But the storage size does vary.

When you use an incomplete data type to create your data objects, you have to specify the set size

when you declare the variable.

So, for example, if we use a character string, they have a data type C, we have to specify how big

we want that variable to be.

Now, with the character string, there is a default of one character if we don't specify a size.

But in general, we have to specify how large we want that variable to be.

So here you can see a table of the different data types that we use in our body.

We start with the integer and you can see that's a full byte, whole number, and it goes up to two

point one billion.

Then we have a float.

We have string x string date and time.

These are complete data types with a fixed size already defined in the system.

The incomplete data types are the ones in blue and gold.

So these are the character string.

We have to specify a size the.

And which is a numeric character string, same thing again by sequence and numbers, and you'll see

down here, if you're not familiar with numbers, they are very useful in our programs with a number.

The SAP system handles any calculations that we do with it.

And what I mean by this is the AB system itself carries out the rounding rather than the underlying

operating system.

So this is useful.

I could just say four for rounding decimals and things like that for monetary fields so we don't lose

pennies here and there in our calculations.

And you'll see, as I've put here on this slide, the pattern number, some people get a bit confused

with it.

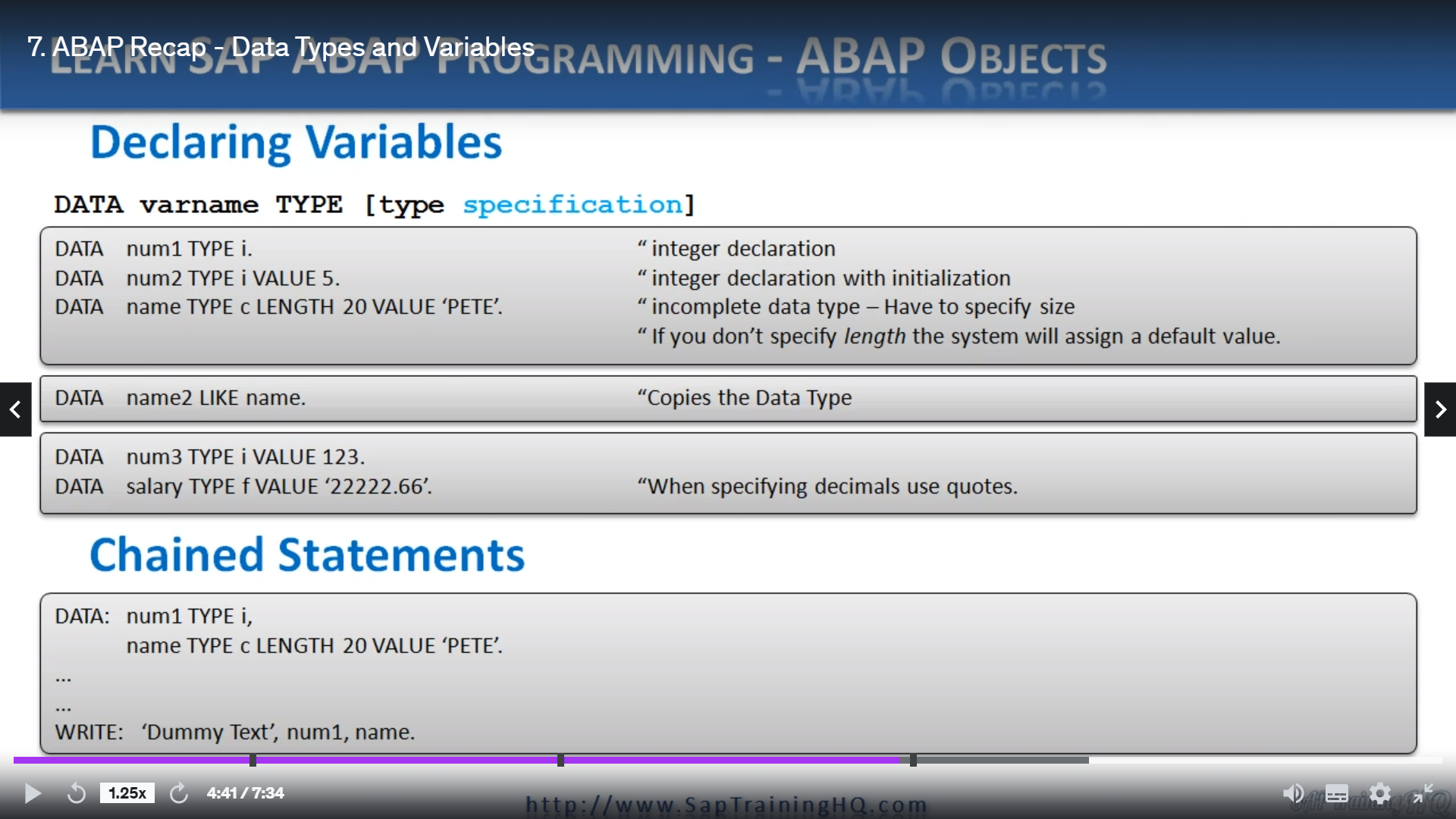
It's like a compressed number and the actual size of the field is determined by the calculation you

see here.

So if I specify a size of 13, the field can actually hold 25 digits.

So this would be 13 as the size times two is 26.

And then we take off one character for the sign, the positive or negative.



So let's have a look at how we declare variables.

Pretty straightforward, and you should be familiar with this, which starts off with the data statements,

then we specify our variable name, then we specify our type, we use type and then give the actual

type so integer and so on.

Here's some examples.

We have data.

Number one is a variable name and we're going to use a type I.

So I just declared no integer next line name to Type II with a value of five.

And this just shows that we are initializing the integer with a specific value as soon as we declare

it.

Next we have the date to name type C and here this is an incomplete data type.

So we're having to specify the size.

We're doing a size of twenty and then we're initialize in it with a value of.

And as you can see, because it's a character string here Type C, we're having to put the content within

single close.

Next down, we can use the like statement and with the like, instead of specifying the actual type,

we can use a built in type in the system.

So you can see here we define the name variable with a type C, lens 20.

So now we can create name two to be exactly the same as name.

And finally, just to go through decimal places and things like that, we have two data statements here.

We have number three, type II value, one, two, three, no decimals.

But when we do specify a decimal, we have to put our value within quotes because remember, in ABCP,

we terminate a statement with the full stop with the period.

But if we do that.

Without putting single quote in our data statement here, the app syntax checker will throw an error

and say you're terminating the statement too early and we have the sixty 66 left on the end.

So when you're defining decimal places, use single quotes.

Then finally, let's have a look.

We have changed statements and remember, with any statement in app you can use what we call chaining

statements together.

So you can see here we have data and then we're using a colon to tell the system we're changing statements.

And when we change statements, we have our declaration here, number one variable.

But then instead of a period, we use the comma and then we can specify our second statement.

Once we're finished listing all the statements down, we terminate our changed statement with the period

at the end.