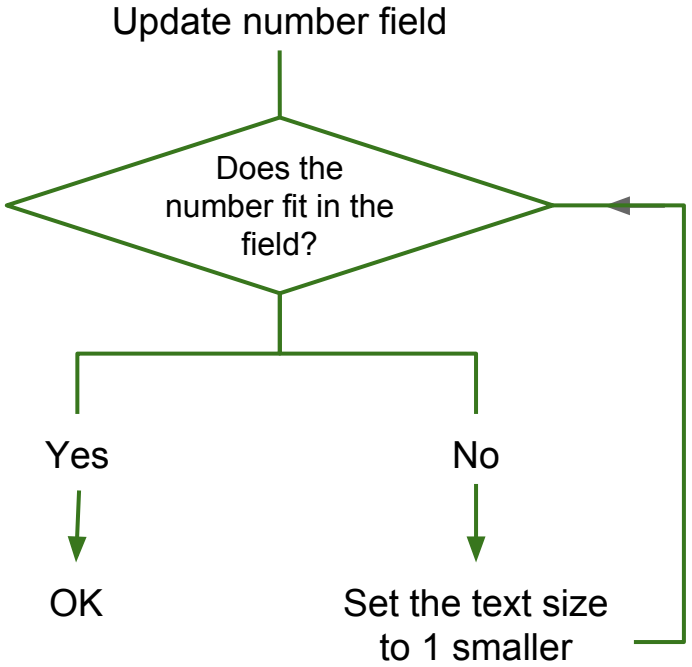
Resizing the Number to Fit

We have already limited the number of digits allowed in the field to 9, but sometimes a 9 digit number won't fit within the display field.

To deal with this we can resize the text as we put it into the field to ensure the number fits in the field.

The steps to do this are:

- update the field
- set the text size to the largest allowed size
- check if the text fits
- if not make it smaller until it does fit, checking we don't go below the minimum allowed size



LIVE CODE
Calculator

The displayChanged Command

We add a command to the card script that resizes the font size so that as the number of digits increases, they will all still fit in the display area.

1. First of all, we declare any local, temporary variables that we will be using in the handler

2. We lock the screen so that any UI transitions that occur are hidden from the user

3. Set the size of the text in the field **display** to the maximum text size

```
command displayChanged
1  local tNewSize
2  lock screen
3  set the textSize of field "display" to 72
… code continued on next slide
```

LIVE CODE
Calculator

The displayChanged Command

1. We use a repeat loop here - LiveCode will keep executing the code within the loop until the number is no longer too wide for the field. We use the **formattedWidth** property to see if the text is too large. If the number is too wide then we reduce the text size until it fits

2. In the repeat loop, we will keep shrinking the font size until it either fits in the the field or it reaches the minimum font size. The effective keyword is used to get the font size of the field, regardless of whether this has been set directly or is inherited

1
```
repeat while the formattedWidth of field "display" > \
      the width of field "display"
```

2
```
put (the effective textSize of field "display" - 1) into tNewSize
```

… code continued on next slide

LIVE CODE
Calculator

The displayChanged Command

1. Check if the minimum text size of 24 has been reached. If it has, we exit the repeat loop as we do not want the font size to be any smaller than 24

2. Set the font size of the field **display** to the newly calculated size

3. Close the repeat loop

4. Unlock the screen to show all updates to the display

```
1  if tNewSize < 24 then exit repeat
2  set the textSize of field "display" to tNewSize
3  end repeat
4  unlock screen

end displayChanged
```

LIVE CODE
Calculator

Key LiveCode Concept

formattedWidth - this property reports the width needed by an object to display its full contents without scrolling.

get the formattedWidth of object
get the formattedWidth of [chunk of] field

Use the **formattedWidth** property to adjust an object's size according to the space needed to display its contents.

LIVECODE
Calculator

Key LiveCode Concept

effective - this keyword is used with inherited properties to specify the object's own setting or the setting inherited from the object's owner, whichever is actually displayed.

put the effective textColor of me into myColor

Use the effective keyword to get the displayed color or font of an object, regardless of whether the object itself has that property set. The effective keyword can also be used to get the filename of a substack or the current rectangle of an object.

If one of an object's text, color, or pattern properties is set to empty, the owner's setting is used instead. (If the owner's setting is empty, the setting of that object's owner is used.)

For example, if a card button's backgroundColor property is empty, the backgroundColor of the card is used as the button's backgroundColor. You use the effective keyword to get the color that is actually used. If the button's backgroundColor is not set to empty, the effective keyword gets the setting for the button, since that setting is what the button displays as its background color.

LIVE CODE
Calculator

Key LiveCode Concept

repeat - This control structure executes a set of statements repeatedly.

repeat loopForm statementList end repeat

loopForm - The loopForm is one of the following forms:

- forever
- until condition
- while condition
- [for] number times
- with counter = startValue [to | down to] endValue [step increment]
- for each chunkType labelVariable in container
- for each element labelVariable in array
- for each key labelVariable in array

statementList - One or more LiveCode statements

The until condition and while condition forms repeat the statementList as long as the condition is false or as long as it is true, respectively. LiveCode re-evaluates the condition before each iteration.

Use the until condition or while condition form if you want to test a condition at the top of the loop, before the statements are executed.

LIVE CODE
Calculator

Key LiveCode Concept

lock and unlock screen - Use the **lock screen** command to rearrange items on the screen or to change locations without the user seeing the transition. Use the **unlock screen** command to allow changes to the screen appearance to be seen.

lock *screen [for* ***visual*** *effect] [in rect rect]*
unlock *screen [with* ***visual*** *[effect] effectName [speed]*

A handler may need to open a stack and then close it before a handler is completed, or to move or change the appearance of a number of objects on the screen. If the screen is locked before these changes occur, the user does not see the changes happen on screen. Locking the screen can prevent user confusion or unsightly screen flashing. It also increases the speed of the handler, since LiveCode does not have to redraw all the intermediate states of the screen.

```
lock screen
    show control 1
    show image 1
unlock screen
```

LIVE CODE
Calculator

Update Commands to Call displayChanged

Whenever the contents of the field **display** is changed, we need to call the **displayChanged** command. This is so that we can ensure that the text displayed in the field is properly sized. The following commands update the contents of the field **display**:

- equalsPressed
- numberPressed
- percentPressed
- togglePressed

Open the card script, and modify each of the commands above to call the **displayChanged** command after the contents of the field have been changed.

Remember, to call a command all you need to do is add a line of code with command name followed by any parameters. The **displayChanged** command doesn't take any parameters, so all you need to do is write the command name.