So now what we're going to do is we're going to keep working on this.

We're going to do something in the loop, we're going to do something before and

something after.

And we're going to accomplish different things.

So the first thing that we're going to accomplish is counting something, right?

I'm counting the number of things that we are going to be looping through.

Now, in this case, it's kind of silly.

But, just, in general, these are called counters.

So, let's just say we had dot dot dot dot, a whole bunch of these things.

And we didn't exactly, we can look at this and say there are six of them, that's not the point.

The point is, if we have a loop and in loop we want to keep track of how many we've seen. Well, you do this counting thing, okay?

So the way it works is you set a variable to 0.

This is the top part, normally this would be called count, but I'm going to use

a bad non-mnemonic value called zork, just so you don't get too used to it.

So we so far how many have we seen? Well we've seen 0.

And then we're going to go it thing is going to be 9, 41, 12.

So thing is going to be the iteration variable.

Each time through the loop, we're going to add 1 to zork,

increment is another word for this, zork = zork + 1.

We print out the thing we're looking at and

we print out the current value for zork.

So, the first time through 9, and then, that's the first thing we saw, 41,

the second. So, and each time, this goes up by 1.

It doesn't really matter what these things are, but then, we run this many times.

And when we're done then we say, okay, loop is all done, right?

There's the end of the block. And when it's all said and done, there were 6 things.

Now again, 6 is kind of obvious.

We could just look at it and know that there are 6.

But this is a way to keep track of the number of times that a loop is executed.

You set it to 0 at the place, you add 1 to it, and then you print out the end.

At the beginning, before it starts set it to 0, add 1 each time through, and

then print out the count at the end. So that's counting.

The next thing we're going to do is total up a series of values.

So in this case we're going to total them up.

It makes a lot more sense now and we're going to have another variable, zork.

I would probably call this total.

During the loop, it'll be the running total and at the end of loop it will be the total.

And so total = 0, before we, that's the running total of what we have seen.

So now 9 comes out and 0 + 9 is 9, so the running total is 9 and 9 was what we looked at. Then 41 comes in, then we take 9 plus 41.

Running total's 50, we read 41, go back up, 12 comes in.

50 plus 12 is 62.

So, that's our running total.

Next time we get 3. 62 plus 3 is 65.

So, this line prints out. Then we get 74.

So, 65 plus 74 is 139. So, that's the running total.

We go and do the last one, 15. 139 plus 15 is 154.

We print that out.

And now it goes back up but the for loop is done, so out we come.

So at the end, when it's all said and done, this this running total become the total, 154.

So, we know certainly when you print all that stuff out, right,

you wouldn't print that out.

But at the end if you do this.

And so the difference between the count and the total is instead of adding 1 here,

you add the thing you're running that you're totaling up.

从 :3:28 开始播放视频并学习脚本3:28

So that's how we compute totals.

And of course, we can do this with an average, right?`

So now we're going to have a counter which we set to 0,

a running total which we set to 0 in the variable sum.

We're going to print out 0, 0.

So, the count and the total is 0.

We run through. Each time through, we add 1.

So, it's going to be 1, 2, 3, 4, 5, 6 each time through.

And we have the sum = sum + value, so, 9, 50, 62, 65, 139, 154.

So this is the running total.

This is the count. Actually that's the sum in this case, the sum variable.

And then, this is the thing that we're reading in.

And then we finally get done.

because there was only six things.

The for loop dumps this out.

And then we got, okay, how many did we get?

We got 6. What was the total? It was 154.

And then divide sum by count and that should be 25.0.

从 :4:22 开始播放视频并学习脚本4:22

Okay? And so we divide sum by count and now we have calculated the average,

actually this should be 154 divided by 6.

I think that should be 25.6666 or something like that. Okay?

So you get the idea, and so you sum and so you can do more than one of these things in a particular loop.

Now, filtering is the idea that we're looking for something that meets some criteria.

So we're going to go through and look at all the things in the loop.

And we're trying to figure out if something is greater than 20 and

we're going to declare that large number. So this is how you put an if in the loop.

So sometimes this loop will do nothing, sometimes this loop will print out Large number.

And so, value is going to be 9 and that's false, so it doesn't do anything.

Value is 41, that's true, so it prints out this.

Value's 12, so it does nothing.

Value's 3, it does nothing.

Value's 74 and so it prints this out.

Value's 15, so that's false. Done.

Now the for loop knows oh, we're done, go on to the next line. Okay?

从 :5:29 开始播放视频并学习脚本5:29

So this is kind of a filtering pattern where we are going to do some if statement

and conditionally run some code based on the value that we are looking at for now.

So that's like searching for large numbers in our long list of numbers.

从 :5:48 开始播放视频并学习脚本5:48

Sometimes instead of printing something out in the middle of the loop, just like in

functions, we don't often print in functions.

We tend to prefer using return values, sometimes we just want a variable that

tells us whether something was found or not.

And so we're going to use a boolean variable.

So boolean is another type of a variable.

You've got integer variables, you've got string variables, you've got floating point variables. Boolean is another kind of variable.

Boolean is a kind of variable that either has the value True or False,

that's it, it can only have two.

And so False is a constant in Python.

And so I'm going to say found.

found is a mnemonic variable. I use the word found for this all the time, but

don't get stuck on it.

It just happens to be a variable with the value False.

So we print it out before, it's False.

And now we're going to go through each of these values again.

And we're looking for 3.

And we want to know did we find 3 or not.

So we have 9.

从 :6:44 开始播放视频并学习脚本6:44

And it stays False.

We have 41, it stays False.

We have 12, stays False.

Now we have 3, this all of a sudden clicks to True and so

then we change the variable found, which is the same found here, to be True.

So this time when we come out up to 3 we run this code, then we print this, and

it's True. Now at 74, this is False, but we've changed the variable for found, so

it doesn't change back to False.

从 :7:15 开始播放视频并学习脚本7:15

So we run the rest of the loop.

And when we're done, True.

And so, at some level, if you didn't print all this stuff out.

从 :7:27 开始播放视频并学习脚本7:27

Before, after, that means we found a 3.

There could be a million times that we ran this and

all we know when we're done is there was a 3 in there.

So that's what it's saying.

Found starts out False, it loops and loops and loops and loops.

And if it ever finds it then found becomes True and

then we can detect by the loop, yes we found it.

Yes, that happened.

No matter how many numbers we looked at, one of them at one point,

at least one of them was a 3.

从 :7:57 开始播放视频并学习脚本7:57

Now, if we were a little tricky we would realize that we could put a break right here.

从 :8:2 开始播放视频并学习脚本8:02

Right there, because once it's True it's not going to go back to False, so we could stop.

从 :8:8 开始播放视频并学习脚本8:08

So we've done a bunch of different things, right? We found the largest,

we searched for a value.

We did some sums, we did some averages and now,

we're going to do the smallest value.

从 :8:17 开始播放视频并学习脚本8:17

So, just before we do the smallest value let's review what we did for

the largest value.

从 :8:23 开始播放视频并学习脚本8:23

Okay, and so, we have this variable called largest\_so\_far

that we set to -1, and then we print it out, it starts being -1.

And then we're going to go through this loop, 9.

And the largest so far is 9 and then we go 41.

We like that one better. Then we hit 12. We don't want that.

We went 3, we don't want that.

74, we keep that and then 15 doesn't make any difference.

Then it pops out and we get 74.

That is the largest, not the largest so far, but the largest you would get.

So the question is, what would you do to change this to make it search for the smallest value in the list?

Now you can look, humans are good at this.

Like hey, what's the smallest value?

It's 3, yeah of course, you know that.

No, we've got to think like a computer.

从 :9:12 开始播放视频并学习脚本9:12

How are we going to find the smallest value in the list?

What would we change here?

What are some of things you might change?

There is at least couple things you would change, right?

So the first thing you would do is, let's just change this from a greater than sign to a less than sign.

从 :9:32 开始播放视频并学习脚本9:32

And let's just change the word large to small.

从 :9:47 开始播放视频并学习脚本9:47

Is that going to fix it?

从 :9:51 开始播放视频并学习脚本9:51

Take a second, if I changed all the word largest to the word smallest, and

I changed the greater than to a less than, is that going to fix it?

If I did that, what number's going to come out here?

从 :10:4 开始播放视频并学习脚本10:04

Is it going to be 3? Is that going to come out?

从 :10:13 开始播放视频并学习脚本10:13

So, like, if I did this?

从 :10:19 开始播放视频并学习脚本10:19

Is that going to make it better?

Is it going to print 3?

Obviously if I'm staring at you, probably it's not going to, this was trick question, right?

It was a trick question.

This is what happens when you run it.

从 :10:33 开始播放视频并学习脚本10:33

So you start out, smallest\_so\_far is -1.

We come in with 9.

Is 9 less than -1?

No, it is not, so we skip.

从 :10:46 开始播放视频并学习脚本10:46

So, the smallest we've seen so far is -1, and we go back up.

Is 41 less than -1?

No, it is not, so we skip. So the smallest so far, is -1.

12? No.

3 is not smaller than -1, 74 is not smaller than -1.

从 :11:3 开始播放视频并学习脚本11:03

15 is not smaller than -1.

So out we come and the smallest we've seen so far is -1.

从 :11:13 开始播放视频并学习脚本11:13

No, it is not. So, what's the flaw in our logic here?

First off, the word smallest so far doesn't mean anything.

This could be gorp or zap or x.

So just changing the variable to say smallest so far is highly misleading, okay?

So, at this point, point right at the screen to the one place that has the fallacy, the mistake in this code.

从 :11:43 开始播放视频并学习脚本11:43

If you pointed right here, you're correct, right?

So baked into that when we were doing largest so far

was that these are positive numbers.

And smallest\_so\_far, if we're starting at -1.

If they are positive numbers, well smallest\_so\_far

is already smallest than the smallest possible numbers.

So what number might we put there to make this work better?

Well, some of you might suggest that we would just say, that would be okay.

I'll make that be 100.

It's going to work fine, right?

9 will tick, 41 will ignore, 12 will not work,

3 will tick. And it will work, it will work, that's fine.

That would work if you made it 100.

But then what if there were three-digit numbers in here, how will that work?

Well, let's make it 1,000.

从 :12:36 开始播放视频并学习脚本12:36

Yeah, that's it.

No, no, no, let's make it a million, or like more, right?

What's a big enough number?

从 :12:46 开始播放视频并学习脚本12:46

As a matter of fact, largest so, that largest so far code that we did,

从 :12:51 开始播放视频并学习脚本12:51

if there were negative numbers, and

they're all negative, this is not going to work so well.

Wait, larger so far, yeah.

This is just not going to work very well.

Because what if they're all negative numbers?

We just assumed they were positive numbers, and

it worked just fine, didn't it?

从 :13:6 开始播放视频并学习脚本13:06

Whoops, whoops, whoops, whoops, whoops, okay.

So, there's kind of a flaw.

从 :13:13 开始播放视频并学习脚本13:13

We were using this -1, there's a word for it.

It is kind of like a flag value, it's not really the smallest number we have seen.

从 :13:23 开始播放视频并学习脚本13:23

It's like an indicator that we haven't seen any numbers.

从 :13:30 开始播放视频并学习脚本13:30

And we can't really pick a number for this.

Frankly, either in the largest or the smallest, it's just not going to work.

So what are we going to do?

Okay, so, this is what we do.

从 :13:44 开始播放视频并学习脚本13:44

We have yet another type of variable, remember I had boolean, True and False,

integer, floating point.

There is a variable called None type, None.

It only has one constant in it.

So booleans have true and false, integers have a whole bunch, and

then floats have a whole bunch, and None types have one thing, None.

We think of it as the absence of a value.

The lack of a value, okay? And so,

what we're going to do instead, is in our smallest variable, that we're going to do.

We're going to say, you know what,

before this loop starts, the smallest number we've seen is nothing.

We've seen no numbers whatsoever.

And that's going to be our marker to indicate that we've seen no numbers. Okay?

This is kind of like the found code where we set found to be False and

then later we set found to be True.

从 :14:33 开始播放视频并学习脚本14:33

But we're going to use smallest equals None.

None is a variable, a value, that we can distinctly detect different than numbers.

So we can say is the contents of smallest None?

If smallest is None, is is like it's more powerful than double equal sign.

从 :14:56 开始播放视频并学习脚本14:56

Means is it exactly the same as.

And so if we are asking is smallest None,

that's only true if we've got a None in there.

If we put 17 in, smallest is not None.

So this is how it works.

Start with smallest equals None.

从 :15:12 开始播放视频并学习脚本15:12

And then in the code we have a little bit more intelligence.

从 :15:16 开始播放视频并学习脚本15:16

If smallest is None, this means that it's the first time.

从 :15:22 开始播放视频并学习脚本15:22

If the smallest is None, smallest is value.

So 9 is coming in here.

从 :15:29 开始播放视频并学习脚本15:29

And so, if the smallest that we've seen so far is empty,

we have seen nothing, then we'll grab the first one as the smallest one.

So our 9 becomes the smallest one.

And so that's how we print out the smallest so far is 9 and we just saw 9.

Then we come up.

The second time through this loop, this is going to always be false from now on,

because smallest is a number and number is not None.

Okay, so we have this flag value that's None just for

the first time through the loop.

So that's false,

so it's going to run here, elif value is less than the smallest, which is 41.

Is 41 less than 9 in this case?

And the answer is no, it is not.

So, it stays 9, then 12, 3 works.

So 3 causes this code to run, so we grab it.

So 3 is it and we go through 74 and 15.

It runs two more times.

And then, when we're all done, we get 3.

This is a flag value

从 :16:29 开始播放视频并学习脚本16:29

that we're checking and through the loop it will only be smaller, smallest will only be

None the first time through the loop but after that we'll just ignore this little part.

This is another way to think about this is this is priming us.

从 :16:43 开始播放视频并学习脚本16:43

This is getting started.

从 :16:48 开始播放视频并学习脚本16:48

Okay?

And this technique is a good way to do the largest and the smallest.

To use this None and then have a little bit of code that triggers the first time

through the loop to get your sort of loop-carried variable setup.

从 :17:1 开始播放视频并学习脚本17:01

Okay, so the is and is not operator, they're both operators.

Is and is not are like less than or less than or equal to or not equal to.

They don't hurt their operands and they return you a true or a false.

Okay? So is, None, and is not is also a logical operator.

And there are other times we use it.

You shouldn't use is when you should be using double equals,

usually you're using them for a True, False, or None.

So that we don't overuse is, because is, is a really, really strong equality.

So it's a stronger equality than double equals to,

so the double equals is mathematically equal to with potential conversion.

从 :17:44 开始播放视频并学习脚本17:44

But is is a stronger thing, and is not is the opposite of is.

从 :17:51 开始播放视频并学习脚本17:51

Okay, so in this chapter, we've talked about some definite loops,

some indefinite loops.

break to get out of loops, continue to pop back up.

continue works in for loops as well as while loops.

Iteration variables, how with while loops you construct them yourself and in for

loops, for constructs them for you.

And things like largest, smallest, counting, average,

etc., etc., etc.

So we looked at some loop idioms where you do something at the beginning,

do something in the middle, and then you get the payoff at the end.