Week7 lecture5.2

So up to now we've been talking about indefinite loops, and that's the use of the while keyword that just runs until some logical condition is false or you hit a break.

Definite loops are finite.

They are going over, say, all the lines in the file or all the items in a list

or all the characters in a string or something.

And for this, we use the "for" construct.

And it's more predictable; and it's easier even to validate them

because we have a set of things that we're going to go through,

and it might be a lot, but we're going to go through all the things in the set.

So here's a little loop, the for loop.

So the for key – the for is the keyword.

And so, while was the keyword for indefinite loops,

and for is the key word for definite loops.

So the first thing we see in a for loop is we see

the iteration variable is explicitly just part of the syntax.

i is, you can pick any variable you like.

I happen to pick i, and everyone picks i for integer iteration variables.

in is another Python reserved word.

And then we have various forms of

collections and we'll find that files are collections, lists are collections.

This happens to be a list of five integers: 5, 4, 3, 2, 1.

And what we're doing in this for statement is we're saying – and again,

it's like, it's got an indented block.

And you got a de-indent,

so the block has this stuff.

I don't know, one of these days I'll stop writing all those little pictures.

I can't help it.

But at some level, what we're saying is execute this block five times.

And I want i to take on the successive values: first time through 5, second time through 4,

next 3, then 2, then 1.

So it executes this one, two, three, four, five times.

And the first time, i is 5, print.

The second time, i is 4, print.

The third time, i is 3, print.

2, 1, and then we're all done.

Blastoff. And so we just, we construct it in a way that we just have directed Python to do all the work. We have an iteration variable.

We have a set of things we want to loop through.

And it just guarantees that it's going to run this once,

code once, for each of these five variables.

Or that's another way to draw that would be, you know, run it once for 5, run it once for 4,

run it once for 3, run it once for 2, and run it once for 1.

5, 4, 3, 2, 1, Blastoff. Right?

So, that's very definite.

Python takes care of everything for us.

We don't have to have a logical condition to stop or anything like that.

It's really quite nice.

This doesn't have to be a list of numbers.

This is a list of strings.

And so, I put this in a variable.

And this is where you've got to be careful because I named it friend and friends,

which makes perfect sense, but don't believe for a moment that Python understands what plurals are. So, friends is a list of three strings,

and friend is an iteration variable that's going to go through it.

So, I stick this list into friends.

So square brackets, list of strings: Joseph, Glenn, and Sally.

And then I have this iteration variable friend that's

going to go through each of the strings in friends,

so that means that friend is going to go once Joseph,

the next time it's going to run the loop again with Glenn,

and the third time it's going to run the loop with Sally.

And so it says "Happy New Year: Joseph,"

"Happy New Year: Glenn," "Happy New Year, Sally."

So you know that this code is going to run three times,

and friend is going to take on Joseph, Glenn,

and Sally in successive iterations through the loop.

So you can think of the for statement as sort of a complex thing.

It is dealing with the fact of when it's time to

continue and when you're done, it knows about that.

And it also sets i to the new value.

So it's like, are we done yet?

No. Move i ahead to the new value.

Print it.

Are we done yet? No. Move i to the new value then print.

On and on and on.

Oh! Now we're done.

All this logic here, this yellow bit, is constructed by you using the for statement.

You can tell it what to do, and it just writes the code.

You can do this with an i increment or a subtracting,

and a while loop, and while i greater than zero, you could do this.

But a for loop just does it for you.

So for does a few things for you.

And I like, I like in,

I think it's basically saying for each of the values in

5, 4, 3, 2, 1,

have i take on the successive values and run that loop one time.

And that's the "in".

I think the in is a well-thought-through Python keyword because that's how I think of it.

If you're a math person, think of it, you know,

the member of function, I think that's the member of.

Boy, it's been a while since I did set theory.

I think that's member of.

So, here's an interesting thing.

You can kind of,

you certainly can think of this as an iteration,

but at some level,

it's really just a contract that says run this

code five times with i taking on its, on the values.

So you hardly know the difference between it saying i equals 5,

print i; i equals 4, print i; i equals 3,

print i; i equals 2, print i; i equals 1, print i.

So, in a sense, this is what happens,

meaning that the for loop is telling us how long we're going to run

and it's managing the successive values of i

to make sure that our happy little line of code gets

executed five times with the right value of i.

That is the job of the for statement.

The for statement takes care of all this for us.

So that's one way to think about the for statement,

even though this is a more common way to think about the for statement.

This is sort of equivalent and think of those two things as equivalent.

So, like I said, definite loops are for lists or lines in a file or characters in a string,

and, you know, they iterate through members of a set.

Up next, we're going talk about sort of loop idioms,

and that's how we make use of loops and how we use loops

to do things like find the largest or find the smallest or

check to see if a value is present or something like that.

So that's what's coming up next.