Chapter 7.2

Okay, so now we know how to open a file.

Now we have to learn how to read through the file, and

it couldn't be easier, it really couldn't be easier.

It's almost at the point where you probably would have

guessed it if I didn't tell you.

Well, so xfile. remember, is not the data,

it's a way to get to the data. It's a sequence.

Well, we already have a loop, the determinant loop, the for loop, and

all you do is you put the handle here.

Now this is not the same as putting a string there, the for loop is smart.

This is some kind of a sequence of things.

So a file handle to the for loop looks like a sequence of lines.

It's a sequence of lines.

So again, the for loop is going to run this code multiple times where the iteration

variable, cheese in this case, is going to take on the successive lines.

If this file has 10 lines, this loop is going to run 10 times.

Cheese is going to be the first line, the second line, third line.

So that's it, that's it.

Isn't it pretty?

No while loop, no end of file, no nothing.

Run this as many times as necessary, one time through for

each of the lines in the file.

You take care all the stuff.

And if you've done this in other programming languages, you'll be like,

Python, you are my friend.

Because it's simple.

And you can look at it and go like, I know what it's doing.

Whereas all the other programming languages have like total gibberish ways

of doing this.

And yes, you can figure it out and after you've used C or

some other JavaScript language or whatever to read a file, Python is better.

Because Python is elegant and straightforward,

and it has to do with this for and in clause which I'm so in love with.

So that just says treat a file handle as a sequence of lines,

and allow us to iterate through it automatically using the for loop.

从 :1:58 开始播放视频并学习脚本1:58

So we can write some smart things, right?

So a counting loop, let's just do counting.

So now we actually have something to count where we don't exactly know how many

things there are, right?

If you go back a couple chapters, we were counting things,

we knew how many things there are.

But now we don't.

So, we set the counter to 0, and then we loop through each line, or

don't do anything with the line, count = count + 1, and we're done.

We know that this file has 132045 lines, right?

And away we go.

So, we have built a line counter,

a file length counter, just by looping through that. Simple stuff.

Now sometimes, not so much when we're doing files but

often when reading data from elsewhere, we want to read it all.

And so this time, we will read the whole thing in with .read.

Now the thing about this is, it doesn't split it into lines.

It actually just reads all the stuff with a newline, all the stuff with a newline,

so you've got to be careful.

But of course if you print it out,

it will give you all the newlines because the print will show the newlines.

But you have to realize you've got the whole as one big blob of

characters punctuated by newlines.

Whereas the for loop knows to go to the newline and then give you the first line,

go to the newline and give you the second line, etc., etc.

But we can read the entire file, so

in this case we're reading mbox-short.txt, read the whole thing into a string.

So that takes all of that file, whatever is on that file and

takes the characters and sticks them in.

Then we can say how many things did we get?

Well, we got 94626 characters in this case.

And we can just print like with slicing, start at the beginning,

up to but not including the 20th character.

So that's really the first 20 characters that we would see

from that file into a single string.

And sometimes we'll use that, sometimes we'll write a for loop,

it depends on what we're trying to achieve.

从 :3:51 开始播放视频并学习脚本3:51

We can search through things, and

we can show the lines that meet a search criteria.

So we open this mailbox, we're going to loop through line by line.

And then we're going to ask if the line starts with From: print.

So this is going to read the lines, and then once in a while it'll run this loop,

and then read some more lines, and

then once in a while it'll run this loop, read the lines.

So it's like a search operation through the file.

And then so this is going to, there may be many lines, but then there's a From: .

And so it's skip, skip, skip, skip, skip, and then prints that one out.

So it's a very, very nice way.

Now, if you run this code, you're going to see that it prints out

the first From line, the second From line, but the question is, something is wrong.

We're seeing lots of blank lines.

Lots of blank lines, blank line, blank line, blank line, blank line.

It's like, what's going on here?

Why are they?

Well, it turns out that the print statement,

as I mentioned before, or the print function, adds a newline.

从 :4:55 开始播放视频并学习脚本4:55

But, actually, I'm pointing at the wrong newline, the print

从 :5:4 开始播放视频并学习脚本5:04

adds a newline, this newline right here, but the file that,

when we read it from the file, the string, has the newline at the end.

So this newline is this part, this part is what we read in from the file, and

this is the newline that was added by the print statement.

So then we end up with two newlines which leads to a newline

when there's no characters, means a blank line.

That's how we end up with blank lines.

The text from the file did this and

then the print statement did one more and so then we ended up with new blank lines.

So how do we deal with this?

Well, we have a function, a function we talked about in the last lecture,

called rstrip, which strips off whitespace,

newlines are part of whitespace because you don't see them.

And so we read through.

We're going to read each of the lines in the file and

then we're going to grab and strip off the whitespace from the right side.

So we're going to wipe out the newline that was here on that string and

then we're going to print it out.

And print's going to add a newline but now we don't end up with two newlines.

So you are going to write these lines of code over and over and over again.

Open a file, read through all the lines, throw away the whitespace at the end, and

now do something fun and interesting with those lines of file.

So these three lines you're going to get to know really,

really well before the end of this class.

从 :6:27 开始播放视频并学习脚本6:27

Sometimes we'll write a if statement where we say if we're

从 :6:33 开始播放视频并学习脚本6:33

throwing away most of the lines in the file but keeping some.

So this case, these are the good lines, right?

The ones that start with From are the ones we're interested in so

we're going to print them out.

Sometimes what you want to do is actually flip the logic, right?

So in this case, it's a different way of writing this same loop but

just a little bit differently.

We do these three lines, that's the same, stripping the whitespace.

And here what we do is if the line does not start with From, so

that's the opposite, if line starts with From, this not turns it, flipping it, continue.

So this is a way of going back up.

Remember how continue works?

Goes back to the top of the loop, so

all the lines that are uninteresting to us we're going to skip.

You might want to call this skip.

That's what we're kind of doing, skip.

But it's just skip this line.

Go get the next line.

Skip, skip, skip, oh wait a sec, this one became true.

We found one and so we print that one out.

And so that's going to produce the exact same

exact same as the previous one.

从 :7:34 开始播放视频并学习脚本7:34

It's just what we're doing is we're skipping the lines

we're not interested in.

And then because often and the amount of code we're going to do is quite a bit to

blah, blah, blah, blah, add some things, parse them in different ways.

It's better to sort of skip the bad lines and then fall through for the good lines.

It's just sort of a stylistic thing between this code and this code.

It's just a stylistic difference not something really to worry about.

从 :7:58 开始播放视频并学习脚本7:58

And so this, again, is code.

We can do all kinds of things.

We can look for all the lines that have uct.

We can use in, right?

And so we can say if uct is not in the line,

if not uct in the line, we're going to skip it.

And so that's going to find the various lines that somewhere inside those lines

have uct, and print those lines out.

uct, uct, uct, and so this is dot, dot, dot, dot, dot, there,

that's the output that comes out of that particular loop.

从 :8:33 开始播放视频并学习脚本8:33

Now often you're going to get assignments from me where you have to do it for

the long mail box and then the short mail box.

And so it's good to learn how to read and write the name of the file.

Read the name of the file from input.

So we just use input.

And we say, enter the filename, so we type in the filename,

that of course is a string that goes into this thing.

I call it fname, I tend to use that variable over and over, fname.

And then I open it, get a handle, then I loop through it and

count the number of subject lines. Right?

That's what this is doing.

If line starts with subject colon, count = count + 1.

And then it just prints out how many subject lines there were.

So it's going to count them.

In the big mailbox there are 1797 subject lines in the big mailbox.

And if we run the same code again mbox-short.txt,

then there were 27 subject lines in mbox-short.txt.

So the idea is this is the different file name.

We can teach Python to open file names, instead of just using strings,

we prompt for the name of the file on input, and then we open that file instead.

从 :9:37 开始播放视频并学习脚本9:37

Now, if the user types in a bad file name it's going to blow up.

Let's say we want to deal with that.

We can, we can deal with that.

So in this case we're going to take input and that's going to work because no

matter what they type, even if it's junk, the input's going to work.

But the open is going to blow up if the file name is a bad file.

And so we indent that and put it in a try block.

So we don't know if this is going to work or not.

Of course, if it works, it's going to just keep on going.

If it fails, if something goes bad here, it jumps to the except block.

And so in the except block, we put out a message, File cannot be opened,

and then print out the file name, be nice to our user.

And then we put this quit statement in. The quit statement says, I'm quitting,

I don't want to continue.

Otherwise it would continue down here and

the quit statement is a function that it goes into but never returns from.

And so it makes sure that you don't actually fall through and

run this code because fhand is not properly defined.

So it's going to just traceback down here if it runs, so you've got to add this quit.

So when things are bad, when something is not working, this blows up.

It runs this code.

It prints that out and then it quits.

And so otherwise you get tracebacks down here if you didn't put that quit in.

You'll get used to this.

So when you,

there's no reason to continue because you didn't get the data you expected.

Now some of you are going to want to change this so that it's a while loop.

We'll talk about that later so it re-prompts for the file name.

从 :11:7 开始播放视频并学习脚本11:07

Little complex like right now.

Right now I just want to show you how the try and except can compensate when you're

prompting the user for a bad file name and you're trying to read that bad file.

But you don't want to have a traceback, you just want to print out a nice message.

So that kind of gets us through opening files and looping through files and

doing something intelligent in the loop and we're going to do a lot of this.

We're going to be opening and stripping and looping and parsing and

doing all kinds of fun things.

So, coming up next,

we're going to learn switch gears a little bit and learn about data structures.