Chapter 9.3

So now we're going to combine all this together.

We're going to read some data,

we're going to read a file,

we're going to make some dictionaries,

we're going to make some lists. And we're going to use

all these things to build a real program.

So what the real program is going to solve...

and so let's see if you can solve this, right?

Read this text really quick,

like in a hundredth of a second or so and figure out what

the most common word is and how many of these things there are.

It's kind of like the problem we showed you before,

except I was showing you names.

So figure it out. What is the most common word and how many are there?

Or maybe, let's just skip to the next slide,

because humans are so bad at this.

I mean, humans are looking at this and saying like, um,

no, I refuse to do this,

as compared to the other ones we actually tried. Now we're not even going to try.

It's like dude, how about we just write some Python code to solve this problem?

I know how this turns out.

We've seen this story before.

Okay. So here's the basic pattern.

We're going to use a dictionary.

We're going to use split. Split gives us back a list. And then we're going to loop

through the list and then we're going to do

the histogram pattern so we can go through the list.

So this program, which we'll soon switch to reading a whole file,

but for now we're just going to read a single line.

Then we just learn how to write another loop outside this loop.

So here's what we do. We start out, we make a dictionary.

We read a line of text type, type, type, type, type.

Line of text goes into this variable line, then we split it.

Right? And then we get back a list of words.

We're going to print those words out.

And then what we're doing is whatever that line was,

we're going to loop through the words. You know word,

word, word, word - whatever it is,

word - whatever's on that line.

And then we're going to update the count.

And so this is that little idiom.

We're going to grab the old value of our count for the particular word we're looking at,

or 0, add 1 to it, and stick it back in.

So this is both going to do the new and the existing.

So that's the basic program.

We're just, we're doing exactly what we did before,

except now we are taking a line from input and we're

making the list of words by using the split function.

You get the idea now.

So here we're going to run this with a bit of code.

Here's our long line of text.

I have to break it so it fits on my screen.

The clown ran after the car and the car ran into the tent and the, the, the, the

So that gets put into the variable.

Then it splits it and this is the words.

So this is just taking this line of text and breaking it based on spaces.

And then we get a list of,

you know, 15, 20 or so words.

Then we write the loop that loops through each one of these things and

makes a little histogram for every word. You know, the,

clown, you know, ran,

then it's after, and then the gets another one

because we get the the and the car gets

another one and we see these things and these grow up, you know?

And that's what we're doing down here when there's no output.

We could put output there, but it wouldn't fit on my screen.

And then at the very end, we're going to print all these counts out.

Okay? And then we can look and see oh,

the, the thing with

the highest histogram is "the" is the one with the highest histogram.

So that's how that code ultimately runs, right?

So if we look through it again,

we make a dictionary,

we take this line of text and we put it into this variable,

then we split it into words.

And then we have this word that goes iteration across this, the words in a line.

And then we do the counting trick.

And then at the end we print all these things out and we can see the largest one.

Okay? So that's, that's a common pattern in text processing, looking for words.

Now, let's talk a little bit more about some of the capabilities of dictionaries.

We've used for loops to go through strings.

We've used for loops to go through files.

We've used for loops to go through lists and

now we're going to use a for loop to go through a dictionary.

Again, for some iteration variable in some collection.

And so what, what you do here is, not because I named it key,

key's a great name for it,

but key is going to take on the successive value of the keys, not the values.

Okay? It's not going to go through the values. In the list,

it goes through the values, but in a dictionary, it goes through the keys.

So it's going to print out Jan, Chuck, and Fred.

Remember order, order doesn't matter.

I mean, order's unpredictable inside of dictionaries.

Now if I actually want to get these values,

then I just say counts sub key, right?

And so key is whatever - Jan, Chuck, and Fred

And so this is counts sub Jan,

counts, counts sub Chuck, and counts sub Fred.

And so that's how we go through the key-value pairs in a very,

very simple for loop.

Just remember, if you just put the name of a dictionary here,

the loop is going to go through the keys in the dictionary,

not the values in the dictionary,

but you can get every value.

Just say counts sub key, or whatever the dictionary name is sub key.

And so what you can do here is, you can kind of see what's going on, is we can

actually tell Python what we're doing right here is we're

telling Python to convert this dictionary variable to a list.

So lists have less information and that ends up giving the keys, right?

And so this is a list.

It doesn't have keys and values,

it just has the keys.

And so that's what happens when you, when you do that.

And you can either say make a list based on the contents of this dictionary,

or you can say take this dictionary and give me the keys.

This is a keys method within dictionaries.

And it gives us the same thing,

which is a list of the keys. List of keys.

Right? Now, if we want the values,

we can ask of that too.

We have a method inside that's called values, jjj.values.

And that says give me a list of the values.

And that gives me the values. It pulls out the values 1, 42, and 100.

Of course, the order is not the order we put them in,

but it turns out that the values do correspond in order to the keys.

So whatever the value of the key-value pairs are in the dictionary,

if you ask for the keys and you ask for the values

a moment later, they come out in corresponding order.

So Jan maps to 100,

Chuck maps to 1, and Fred maps to 42.

Even though the order is not the same as the order that you originally put them in.

One thing that we can, we'll talk about in the next is what we call tuples,

but there is another method called items that gives us back a list of key-value pairs,

which is different than the dictionary itself.

And it is a list.

So if you look at this as a list,

these little guys with parentheses are called tuples,

which is the next chapter.

But you see we see the Jan maps to 100,

Chuck maps to 1, Fred maps to 42.

This outer thing is a three-item list,

but each item itself is a data structure called a tuple.

But we'll, we'll talk about that in a short period of time.

This whole items thing can be used with a a for loop to loop through,

simultaneously, the key-value pairs.

And so items gives us back keys and values.

And so that when you write this for loop, and this is like a really cool Python thing,

you can put two iteration variables.

No other language that I know of has the ability to do more than one iteration variable.

So what happens here is we've given it two iteration variables.

And aaa goes through the keys and bbb goes through the, simultaneously, the values.

So as one is bouncing,

the other one's bouncing too.

So it is basically saying - I usually call this k and this v to mean key and value,

but I'm using non-numonic variables

so I don't confuse you.

And basically, that just means that we're

going to go through all the key-value pairs, right?

This was the key, this is the value, and we're going to run this loop

three times because there's three things in

the dictionary and we're going to hit each of the key-value pairs.

And so it prints out Jan 100, Chuck 1, 42.

So this is a very succinct and convenient way to go through

a dictionary and see all the key-value pairs and

not have to manage any of the stuff manually.

I usually name this k and v or

key and value because that's that keeps it straight in my mind.

The key is the first iteration variable and the value is the second iteration variable.

Okay. With that, we can actually now circle all the way back to Chapter 1.

Remember I told you that you would understand

every line of this at some point and this is the moment.

This is the moment that you are going to understand every single line.

And if you've been doing your homework and the work up to

now and you sort of understand the lecture we just got done doing,

you should understand every single line.

And it's not all that hard.

It's like what, 10 weeks later or whatever it is?

Or 10 days later?

Okay. So let's start at the beginning.

Aw, heck, it's easy. You're an expert now.

You know a lot of Python,

you know 9.5 chapters of Python. So what do we do first?

We put out a prompt. So it puts out a prompt.

We get a file name. words.txt ends up in name.

We open it, we get a file handle, right?

We're going to make a count, a dictionary,

a histogram pattern, so we're going to create an empty dictionary.

We're going to have this line

variable that's going to iterate through all the lines in the file,

this is going to run once for every line in the file.

We actually don't have to do the strip because

the split kind of does a strip for us automatically

it ignores spaces at the end,

and so we don't really have to do a strip.

You could do a strip here, but it wouldn't hurt.

You think of this file as like word, word, word, word, word

a second line word, word, word

word, word, word, word, word. But this outer for loop is going through line for line.

And then we do the split,

which gives us each of these words.

And now we have an inner for loop and that's going to go through

each of the words in the file, right?

So this is a word in words.

This is still one line.

So we're going to go through the words

and we're going to say counts sub word equals counts.get word;

this is that idiom, go back to that part. I won't explain that.

That just makes the histogram,

makes the histogram, right?

And we're going to end up accumulating these counts over the entire file.

So this is two loops.

We're going to each line, then we're going through the words in the line,

then we go to the next line, then we go through the words in the next line.

And then we're making a histogram as we go, you know,

a little histogram is building up for the different words and the histogram is extending.

At this point in the code,

inside counts we have a complete histogram of every word on every line of that file.

And now all we have to do is figure out what the largest one was.

And this takes us back to a couple of chapters ago where we were looking for the largest.

Okay? You can't use the max function here,

because it's all sort of hidden inside this dictionary.

So what we're going to do is we got this histogram in counts, you know.

It's the words, word, word, word,

and then, you know,

the little things counted up, right?

So we're going to look through here and we're going to look for the tallest

one and then we're going to remember both what the count was and what the word is.

And so we're going to make a variable called the biggest count that we've seen

so far and the word that is associated with that biggest count.

So I'll call my variables bigcount and bigword.

And so now I'm going to have iteration go through the items,

which is the key-value pairs.

So word is going to, the word variable is going to go through

all the keys and the count is going to go through all the values, the numbers.

And we're doing a maximum loop.

And if the big count is None,

which means we're on the first word,

or the count we're looking at for the particular item we're looking at,

you know, 4, is bigger than the

previous one, 1, if it's the first

one or if count is greater than the biggest count, then remember it.

This is kind of the remember.

We're going to remember what the word was that we

had that, where we saw the biggest number,

we're remembering the word and we're going to remember the count.

And then we'll do this a bunch of times and if we find

later, much later, some thing that's got a little higher number,

like, you know, 8 or something,

then we will fix that as well, okay?

And so this runs all the way through

all the things. And when it comes out bigword and bigcount, the residual at

the end of the loop, the residual stuff at the end of

the loop, will be the largest word and how many times.

And so all that magic is happening sort of from here to here, that magic happens.

And in words.txt, we see that the word "to" is the most common and it's seen 16 times.

And in the clown text that we've been playing with,

the word "the" happens and it's 7 times.

And that's the magic.

So this is a slide that if you don't understand every single character,

every single, you know, why is the word None?

What did the word None mean?

It's time to go back and check, right?

This is a time to review.

This is the time to understand what we're talking about,

because we're just going to get crazier from here and this is

really important basics because

this is a complete program that does something non-trivial.

OK? So remember this slide.

Okay? So we have talked about dictionaries.

We've compared them to lists.

We've shown how to use the get operation.

We've looped through files.

We took a look at tuples, tuples is what we are going to talk about next.

And we're going to sneak peek - we'll talk about sorting dictionaries coming up next.