Chapter 9.1

Hey and welcome to Chapter 9.

This is Python dictionaries.

My favorite chapter in the whole book is files because that is where it

sort of comes together.

This may be your favorite chapter.

Python dictionaries are one of the things that people sort of fall in love with and

Python is like, I can do so many things with dictionaries.

And dictionaries are our second data structure, our second kind of collection.

And the basic definition of a collection is

more than one thing in a single variable, and a way of indexing and looking up and

manipulating things in that single variable.

And so we have the list and now we have the dictionary.

And between lists and dictionaries you pretty much get the two most important and

most basic data structures, not just in Python, but in any programming language.

So to review, the idea of a collection is we put more than one thing in,

whereas our regular old variable that is not a collection,

a non-collection variable like x, we stick a 2 in.

We say x = 4 and we sort of implicitly know that now it has a 4 in it, but

the 2 is long gone, it's been overwritten.

从 :1:10 开始播放视频并学习脚本1:10

So the two collections we've talked about, the list is a very ordered collection.

Everything is very precise, zero, one, two, three, four.

You append at the end.

And when you start deleting, it all stays compacted,

and the position is the key to it.

But a dictionary is a more of a mess.

You put things in, and you give them a label, and you get things back out.

And so they don't always stay in order, the order shifts around.

But, they're super powerful, because it's a key-value pair, and

you give each item a key.

You give it a key on the way in, and you get a key on the way back out.

从 :1:48 开始播放视频并学习脚本1:48

And so inside of a dictionary everything has to have a label.

You can't just put something in a dictionary,

you have to give it a label and a value, or a key and a value.

So the key is, when we say key,

it's like the thing to unlock or thing to find that thing.

It's a general concept in many languages called associative array and

you can kind of look up and learn a bit more about associative arrays.

And in most languages like Perl or PHP, the associative array is, again,

most people's favorite data structure because you can do so many things with it.

So dictionaries are Python's most powerful collection.

It's like a little baby database, key-value, you look something up, your

name and a phone number, or a phone number and name, but there's key and a value.

And like I said, these concepts of dictionaries has many

different names in many languages, like HashMaps in Java or Property Bags in C#,

or Associative Arrays in Perl or PHP, but they basically are the same thing.

They're memory-based key-value stores.

从 :2:53 开始播放视频并学习脚本2:53

So, like I said the whole idea

of these collections is they have more than one thing, but

the question is how you put stuff in and how you get stuff out is what changes.

And so here we're going to make a dictionary.

And this is a constructor that basically says, make me an empty dictionary object.

If we're doing this with a constant, we just use a curly braces.

Curly braces are for dictionaries.

And then so purse is a variable of type dictionary, and it now has nothing in it.

Whereas in the list we had to append stuff, we use the index operator

on the left-hand side of an assignment statement to indicate the key.

So this ends up being the value, and this is the key.

The key-value pair.

And so we're putting 12 into the purse, but

on the way in we're labeling it with money, right?

And so 12 is going in, 12 is the value and we're putting it into purse, but

we're storing it under money.

So I would say this as purse sub money equals 12.

Money is the label, the index, the tag, whatever you want to call it.

But it is the way we're going to label this number 12 as we go in.

And we're going to put 3 in and we're going to label it candy, and

we're going to put 75 in and we're going to label it tissues.

So inside of purse we have these key-value pairs.

Say hey, what's in purse?

Well, money is 3.

Under the tag tissues, we have 75.

And under the tag candy, we have 3.

It's important that the order of these dictionaries is somewhat unpredictable.

They don't always come out in the same order.

And so we'll have to learn later about how to sort them and how to do stuff to them.

But think of them for now as a unsorted bag of values.

But it does remember no matter how they get moved around

what the label-to-value mapping is.

To pull something out, we say purse['candy'].

So we use the exact label that we used to put it in.

We use that to pull it back out.

So purse['candy'] is 12.

So it says go look up and find in this dictionary

the thing that has a key of candy and then pull that number out, so

that's the number that comes out and gets printed.

从 :4:54 开始播放视频并学习脚本4:54

We can update these variables.

The dictionary contents are mutable, which means they can be changed.

So we can go grab the old value,

purse['candy'], which is a 3, add 2 to it, and then put a 5 back in.

And then we can print this out, and

then the 3 has been modified to become 5.

So we can use them on right-hand sides of expressions,

on left-hand sides of expressions, away you go.

从 :5:17 开始播放视频并学习脚本5:17

So it's really a lot like lists.

If we look at an example from a previous chapter on these lists,

now we make an empty list, we make an empty dictionary.

We add into position 0, which is because it's empty, a 21.

And we put in position 1 a 183.

Here we put a into this dictionary under the key age 21 and

into this dictionary under the key course, we put in 182.

Right?

从 :5:41 开始播放视频并学习脚本5:41

And if we print them, we see that there is an implicit position,

position 0 and position 1.

There's an implicit position so we don't have to show the position.

Python could choose to show that, but it doesn't.

And then if we print a dictionary,

it shows us with curly braces to say this is a dictionary, not a list.

And it uses this key: value, key: value.

And order again isn't the same.

We put in age then course then we got out course then age.

Just don't expect the order of dictionaries to be a predictable thing.

And then continuing on, they're mutable.

Which means we say lst sub 0 and this is an index.

So in a list the index is the position, and we put a 23.

We put a 23 into position 0 and so when we see that, we see a 23 coming out.

And here if we're going to change something, we're going to say,

go find that thing marked with the tag age and replace that with 23.

So that puts the 23 in here and replaces it with 23 and so

when we print that out, we get 23.

But basically, they're similar.

But dictionaries have these labels that we have to use everywhere and

lists have positions.

So lists are compacted, they always start at 0 even if there is only,

if there's none in there this doesn't start at 0, but there is one or

more things the first one is always 0.

And while we haven't deleted stuff from lists,

once we start deleting stuff from lists they get compacted.

But not true for dictionaries.

They just kind of remove the key because they're not particularly in any

order anyways.

从 :7:12 开始播放视频并学习脚本7:12

So, again, we can see this all happening.

I like to think of, in my mind, a picture of these things where a list

has kind of a key or the way we look something up, it's a 0, 1, and

if there were more of these, it would be 0, 1, and 2.

Whereas a dictionary, the key is some kind of a string, this is how we look it up.

And they both have values, right?

They both have values in them, and that's how it works.

And so, when we put these things in, the list gets extended,

it's always going to be extended at the end.

If you called append() again, there would be a sub 2 in here.

And we put some number in there, and we print them out.

And then, when we go and change this 23, we put it in here, it gets wiped out.

They are mutable, which means these values are mutable.

The keys are sort of not. In lists,

the keys are simply an artifact.

You can't change them, but you can delete items.

You can delete items from lists, which we haven't done yet, but

we will be able to delete items from lists.

And so the same thing is true.

We can delete items here, which we haven't done yet.

But you can also change them.

So ddd['age'] will go look up age and then stick 23 in there and that's changing.

So they're very similar structures.

The key difference, the significant difference, is that the key mechanism,

and the way we look up an entry inside the collection, is different between them.

从 :8:33 开始播放视频并学习脚本8:33

We can make constants and

the printouts of these things conveniently in Python show us the constant syntax.

So it's like, what's in this dictionary?

So we start with a curly brace for a constant, and

then we have key, value, key, value, key, value.

Now I have been using strings for keys and numbers for

values, which is actually a pretty common shape, but it doesn't have to be that.

So just because I'm using it doesn't mean that's always that way.

And so this creates a dictionary constant.

And then that gets assigned in the jjj variable.

Just like it's no different than saying x = 3,

this is an integer constant that gets assigned into x.

We print it out and it looks exactly kind of the same,

but the order's different, because dictionaries mess with order.

You can make empty dictionaries with just curly braces and print those out as well.

So up next, we're going to start talking about the kinds of applications,

the kinds of problems that we're going to solve using dictionaries.