Week5 lecture3.1

Now we're going to talk about conditional execution.

Conditional execution is where we start sort of adding more intelligence.

Sequential execution, you just type them in and Python does it.

But sequential execution is where you can choose one of two paths.

Turn left, or turn right, or go straight.

You can make choices and

this is an essential part of making computers seem intelligent.

And so the key to this is the if statement.

So the if statement is the reserved word that indicates that we're going to do

something conditionally and it's not that hard to read.

So the if statement has a condition, it's really a question.

Unlike x = 5, which changes potentially the value of x,

x less than 10 is asking a question. It's asking, is x less than 10?

And that returns a true or a false.

In this point because x is 5, x is indeed less than 10 so it's true.

These if statements end in a colon.

And they start an indented block of text.

It's kind of like an outline that basically says, if this is true,

do this statement.

Otherwise, skip the statement.

So in this case, it comes in, x is less than 10, so it does the statement.

Then it comes back and says,

if x is greater than 20, this one evaluates to false, and so that skips it.

So that's why we see Smaller print out and Finis print out,

and this line never executes.

So that's why I call it conditional.

Depending on this question that's being asked,

the line inside the indented block will either execute or not execute.

And so you can kind of look at this as if you were driving a car, right?

You execute this statement then it goes to the next one, then it asks the question.

Is x less than 10? If the answer is yes, it turns this way and drives down here and

runs this code, does that thing.

And then it rejoins.

And then it says, oh, is x less than 20?

Well, since x is 5, the answer is no.

And so that skips that completely and then it runs this and then it's done.

And so this code never runs, okay?

And so when we say conditional, we say, yes that one ran but that one did not run.

And so that's why we call it conditional because it may or may not run.

The other thing that's important about this is this indenting notion.

That we start with an if, have this colon, and then we indent.

The indenting can be a variable.

We tend to indent by four spaces.

That's the recommended pattern.

And we'll see in a bit the indents can be longer than one line.

This is only a one-line indent where you have one line that's conditional but

you will see in a second how we can do this with more than one line.

从 :2:41 开始播放视频并学习脚本2:41

So, good time to talk about the comparison operators.

Again, the problem here has to do with the limitations of the keyboards in the 1950s

and the 1960s of computers when most of these languages were kind of designed.

So less than is on keyboards,

less than followed by an equal sign is less than or equal to.

Double equal, this is probably the part that's the hardest.

Double equal is a question mark.

Remember equals is an assignment statement, it has kind of direction.

x = 1, right? x = 1.

That puts 1 in x.

This, if you say x == 1, you're asking the question,

is x equal to 1 and it doesn't harm x,

So that's probably, it takes a while.

从 :3:19 开始播放视频并学习脚本3:19

If I was designing a language, I would say, assign x into 1 and

then this would be single equals.

But, sorry, I didn't design Python so that's not the way it is.

So, double equals is the question mark version of equality.

Greater than or equal to and greater than, and then not equal is exclamation point.

Another word for exclamation is bang.

We say bang equal or not equal and

so the exclamation point is like not equal, sort of like emphasis.

I don't know.

That's how I remember it.

从 :3:51 开始播放视频并学习脚本3:51

Remember that none of these harm the data that they're looking at.

They evaluate and then return us a true or a false.

So here's an example of all these things kind of in action.

For x = 5, they're all going to be true.

If x = 5, remember that's the question mark.

If x is greater than 4, and the answer's yes, print Greater than 4.

If x greater than or equal to 5, that's also true so this part runs.

You can also, if it's only one line of stuff,

you can sort of pull this line up to the end here.

If x less than 6, print Less than 6.

从 :4:25 开始播放视频并学习脚本4:25

If x less than or equal to 5, print Less than or equal 5.

If x is not equal to 6, print 6.

And so you see this pattern of indent, end of indent.

Indent, end of indent.

Indent, end of indent.

Indent, end of indent.

So this is an important part of Python.

Not a lot of languages make the indenting of lines

a syntactically meaningful thing, but that is how Python works.

And so if you don't indent it, it's not going to work the way that you expected.

Especially if you're coming from a programming language like JavaScript,

or Java, or C where the actual spacing doesn't matter.

In Python, the spacing does matter.

从 :5:3 开始播放视频并学习脚本5:03

And so this is an example of a longer block of text that's indented.

And you can now see kind of how this indent works.

x is 5, we print Before 5.

If x = 5, this is going to be true because it is.

Colon starts an indented block.

Now by maintaining the same indent,

you make this all part of the conditionally executed block.

So this is three lines of code that it runs,

it runs this line then sequentially runs this line, sequentially runs this line.

And we indicate when it is that we want to get out of this block and

then continue by de-indenting, okay?

So when it's true it runs all of them, then now it's running sequentially,

sequentially, sequentially.

And now it says, oh if, if x = 6.

Well this one's going to be false, then that skips all of these.

None of these run.

Skips all of the indented blocks.

And so this indenting is a way to, in effect,

make bigger blocks of conditional code, or multi-line blocks of conditional code.

从 :6:8 开始播放视频并学习脚本6:08

So, indentation is important in Python.

It's more important in Python than literally almost any other

programming language.

We always use indentation in other programming languages to mentally

keep track of blocks of text, but

it's not where the programming language will complain if you get it wrong.

So you have to sort of think why did you increase the indent, which is like

if, after an if or a for or a while statement, which we'll see in a bit.

You maintain the indent, which means you stay the same, or then you de-indent or

you reduce the indent back, and that is the way to end a block.

So you reduce it.

Blank lines don't matter, and

then comments by themselves on a line don't make any difference for indenting.

And so, as you're doing this indenting,

it's really natural to hit the tab key on your computer and to move in.

So you do if blah, blah, blah, and then on the next the line, you hit tab.

The problem is is that Python can get very confused if you sometimes

are using four spaces, space, space, space, space, and sometimes you use tab.

And the danger is it can look right on your screen, but

Python will still complain to you.

And so if you're using a text editor that might be putting tabs into your document,

从 :7:20 开始播放视频并学习脚本7:20

find where it turns them off.

So a programmer editor, we're not talking a word processor now,

we're talking about a programmer editor like Notepad or Text Wrangler.

These have a setting, whether or not to turn tabs into spaces or not.

So it either puts a tab in the file or puts spaces.

And so this is a good time, if you're using one of these text editors,

to make sure to turn tabs off, or cause tabs to be expanded.

So you don't end up with, in a few weeks or a few days,

with Python errors that you just don't understand, and that makes you grouchy.

Here's the good news, and that is, if you're using our recommended

Atom text editor, once you save the file with a .py extension,

it automatically just uses, you can hit the tab key and it moves in spaces.

And so that is very nice and it's just one of the many reasons that we recommend

and like Atom to edit these files.

So just understand that tab and spaces might look the same on your screen,

but to Python, they may or may not be equivalent.

And you can up with spurious errors that Python will give you

if you get your tabs wrong, okay?

So here's just some of the screens.

Depending on your tool that you're using,

your editor that you're using, you might have to find it.

But it's like auto-expand tabs, or replace tabs by spaces, etc.

The idea is don't put tabs in your document.

The best practice is to use spaces and

don't put tabs in your document when you're writing Python programs.

So, you have to think about this, about the indenting and

the de-indenting, when do you go in, and when do you come back out?

And so, it's the way that we define blocks.

So we have sequential code, which maintains the same indent.

Now we see the if and we see the colon, that means we start an indent.

And then to stay inside that block of code, you maintain the indent.

And then to end that block of code, you de-indent.

So you think of this like I am de-indenting, I'm doing this on purpose,

it's not like randomly.

If this was in, then it would function differently.

The fact that this print is out means it's not part of this if block.

These two statements are part of the conditional code, the print really

is the thing after the if, right?

And so this is going to run no matter what, these two may or may not run.

In this case, because it's true, they are going to run and come back out.

So, this is conditional code, this is sequential code.

从 :9:46 开始播放视频并学习脚本9:46

And if we continue down, we haven't talked about loops yet but the for keyword is

a loop and this is telling it to run this thing five times, and it ends in a colon.

And then you go in and then you have some sequential code, then you have an if, this is called nested.

从 :10:2 开始播放视频并学习脚本10:02

That's a block within a block.

So the if has another colon so we go in even farther.

And then we de-indent to this level of this if, and

later we de-indent to the level of the for.

So you kind of have to match these things up and

decide how you're going to end your conditional code, or end the looping code,

which we're going to learn in the next section how that works.

从 :10:24 开始播放视频并学习脚本10:24

So what you want to do after a while, and it won't take you too long, but sort of

train yourself to start thinking about the fact that these are blocks of code.

That you can sort of draw these squares around the blocks of code.

And I've drawn the square here, the if starts it.

You can also kind of use the colon as your little mechanism.

And then the de-indent stops it, and so this is the conditional code.

And then you can see where the if starts and then it's the line

that lines back up with the for, that is the scope of the entire for loop.

Same here with with the if.

The thing that lines up, that is the scope of that if block.

So mentally, you have to realize that the de-indenting and

indenting adds meaning to your program.

I mean, they absolutely define which things are going to run as part of an if

statement or regardless of the value in the if statement.

So start seeing these, and in terms of nesting,

you can see them inside one another.

It's like the Russian dolls, where you've got a block of code,

within it you've got another block of code.

And believe me, we'll go down, down, down, down, down, out, out, out, out, out.

And so, start seeing these indenting as blocks.

Just start feeling that.

从 :11:35 开始播放视频并学习脚本11:35

Like I said, nested decisions, if there was a for loop with an if,

we can also have a decision where there's an if within an if.

And so in this case, x is 42, if x is greater than 1, that's true.

So then we're going to print this, if x is less than 100, that's again true.

So we're going to go in and run this, and then we've de-indented twice.

And so this does line up with the if, so

this functions as the end of both this block and the end of this block.

Because it lines up out here.

So, if we look at it from a GPS perspective, it's true,

x is greater than 1, we print that.

We ask another question, that's a question, and it's yes and

now, we're continuing back down.

So we've kind of come out of the indent block.

And as you've drawn, we kind of see this is indented once and

this one is indented twice, just the way I drew it.

And the same is true here.

This is indented once, once and then getting too busy here.

This one's indented once, this one's indented once,

this one is indented twice, all right?

So start counting those indents and knowing how the de-intents works.

So we have one-way decisions and very commonly, we have a two-way decision.

We call this else.

We want to basically do one thing or another, it's like a fork in the road.

If something is true, in this case, is x greater than 2?

Well, it's 4, so it is true. We do one thing.

And if it's false, we do another thing.

In this particular example, we don't do this one.

We only do this one over here.

But you got the idea that with one if test, we choose alternatively between one and the other of the choices.

So this is how we write that in Python.

We say sequential code, question mark, colon.

And then we have the true part, this is the part that executes if the question evaluates to true. And then we de-indent and use the keyword else, so else is a Python keyword.

And you see another colon here so we sort of de-indent, say else and

then re-indent and then we have this part.

And so that's how we capture the other half and then of course it runs no matter what.

So, if we were to do this.

If it's true, it runs this and then skips that.

And if it were false, which it's not, it would skip this, run this, and then continue.

The point is, if you use an if-else structure, one of these two things is going to run.

And it's sort of the same as this, it comes in here, there's no scenario in which both run.

It's going to run one, and by running one, it's not going to run the other one.

That's how an if-then-else happens.

So coming up next, we're going to talk about some more conditional execution

patterns, a little more complex stuff, else if, etc.