Week5 Lecture 3.2

So welcome back, now we're going to build up and talk about some slightly more

complex conditional execution patterns that you can build up.

So as you do the if-then-else, again, one of the things to do is to visualize the blocks, start thinking of the blocks.

And in this case, you'd think of the block as sort of starting at the if and then ending after the last indented line.

But the if and the else kind of are a piece of one thing and you can sort of think of this as the block, right?

And one of the things about these blocks is they have one entrance and one exit,

not that that's a big deal.

But it's a good way to think about how you mentally start drawing the blocks.

You see one entrance and one exit, and there's some complexity, there's some logic that you're sort of building. And that's how these blocks work.

从 ::56 开始播放视频并学习脚本0:56

So the next step up in complexity is a multi-way if.

And that uses a keyword called elif, which is really a combination of else and if.

从 :1:9 开始播放视频并学习脚本1:09

So the way it works is, probably easier to do this on a GPS version of it.

Is it comes down here, forget what's in here, it asks yes or no.

If it's yes, it runs this one, and then it's all done.

If it's no and then yes, it runs this one, and it's all done.

And if it's no, no, no, it runs this one and it's done.

A key thing is it's only going to do one of these three.

It's not going to do two, it's only going to do one.

And it checks these questions in order.

从 :1:38 开始播放视频并学习脚本1:38

And then this else is like a catch-all.

从 :1:42 开始播放视频并学习脚本1:42

So let's take a look at how this particular one works.

It depends on the value of x. So let's, what if x were 0?

Well, if x were 0, it would come in, ask the question, is x less than 2?

Yes, it is. So it'd run this code and it would finish down here.

So it would come over here, true, run, and now skip all the way down.

从 :2:4 开始播放视频并学习脚本2:04

If you draw blocks, the box would be this box right here.

It skips out of the box, or if you draw the box over here, the box would be right here.

Once you're done with this line of code, with this block of text right there, or block of code, you exit the block.

You don't like come back and look at this question.

You have run, one thing has turned true, and you're done.

从 :2:25 开始播放视频并学习脚本2:25

If, on the other hand, x was 5, you would see a situation where it would come in.

This would be false, and so it would skip.

Then it comes to this next elif, this becomes true.

So then it jumps in and does this code, runs out, and is done.

Does not run that, does not run that, does run that. So, comes in.

No, it's not less than 2. It is less than 10, so we'll run that little block of code, and

now we are all done completely.

So no, no, but yes, okay? And so that's how it works.

These are done in sequence.

They're not looked at sort of in parallel or all at the same time.

If, on the other hand, x was 20, it would say false, skip, question mark.

False, skip. Oh, else. Then the else always gets triggered if it gets that far, and

then it runs that one.

So here we say if it's less than 2, no it's not, is it less than 10, no it's not.

And if it's there, then we just hit the else part, and we finish and

we continue on. So this part doesn't run, this part doesn't run, and that part runs.

The rule is one of the three will run, and the other two will not.

It only triggers once. Once it's triggered, then it's done with the whole if statement.

And again, I think of this as the block, and you'll see in a second.

Once one of these things hits true, it runs this and then exits the block completely.

从 :3:48 开始播放视频并学习脚本3:48

So there's other variations on this. You can if you want have no else.

So there's no need to have an else.

What we've done is we've simply deindented this next line. There's no else here, but that's okay. It does mean that for some value of, they might not either execute,

right, because there's no else. If there's else, then at least one will execute.

But if there's no else, then it could be possible that zero executed.

In this case, x is 5, it's not going to do this one but it is going to do that one because x is less than 10.

But if x was, for example, 50, then that would be false, that would be false, and we just go.

And then neither of these two things would execute if x were 50.

Okay? So it just means you don't have to have an else if you don't want to.

从 :4:37 开始播放视频并学习脚本4:37

Further, you can have lots of elifs. If x is less than 2, do this, elif, elif, elif, elif, elif.

And remember, it checks them in order, first, second, third, fourth, fifth.

从 :4:51 开始播放视频并学习脚本4:51

So if x was 15, this would be false, this would be false, this would be true, so it would run this code.

And then it would come down whatever is next down here.

So again, only one of these is going to trigger.

No, no, yes, no, no, no.

I mean, 15 is indeed less than 40. If it came here, it would be true, but it doesn't matter, because one is triggered, and so then it takes the entire block, and there's our entire block.

So as soon as it executes one, the next thing to do is exit right out of the block.

Okay, got it? Okay, so here are a couple of puzzles.

I'll give you a second to pause this.

And the question is, to look at some of these, depending on for

a particular value of x, you will have none of them execute.

So which will never happen for some particular value of x?

Meaning you can pick any value for x you want, but

there's some that you can't cause to execute. Okay?

So I'll pause for a second and let you pause the video if you want and

then I'll come back and explain it to you.

从 :6:6 开始播放视频并学习脚本6:06

Okay, you had some time to pause.

Hopefully, you did pause or didn't pause, but it doesn't matter.

You can still pause while I'm talking until I start drawing and

telling you the answer. Okay, so in this one, if x is less than 2,

do this, else if x greater than or equal to 2.

The one that's never going to execute is this one right here.

And that's because, no matter what value of x, it is either less than

equal to 2 or greater than or equal to 2.

So for any value of x, no matter what you pick, it's either going to run this one or

this one. But it's never going to run that one for any value of x, okay?

So, that's a little tricky.

I just happened to have constructed my logical questions in such a way that

they covered all values of x, and so the else was kind of irrelevant.

Now, I wouldn't even draw this this way.

If I was going to draw this, or write this code,

I would probably just make this be an else : and this not be there.

But this was more of a puzzle than anything else.

So in this next one, we have to remember that these things happen in order.

So if x is less than 2, we're going to run this.

If it's less than 20, we're going to run this.

If it's less than 10 we're going to run this.

But the problem is, all values of x that are less than 10, for

which this would become true, this is always true.

So a value like 6, this becomes true.

So that means that if it's something like 6, it's going to run this and

come out and never ask this question, right?

So that's the key, even though this is true and this is true for the 6,

it never even gets here because this one triggered first.

从 :7:42 开始播放视频并学习脚本7:42

Okay, and so that's why this is the line of code that no matter what the value for

x, will never run.

从 :7:53 开始播放视频并学习脚本7:53

Okay, so the last conditional code is what's called the try and

except structure.

And if you learn other programming languages,

this sort of catching errors is a more advanced concept.

But in Python, we tend to have to use it earlier because there are things where if

you don't use it, the code blows up.

And the whole idea of a try except is that you have a bit of

code that you know might fail, and so you kind of want a take

out an insurance policy on it and say, hey, give this a try.

If it works, great, if it doesn't, do this other thing.

Don't blow up, don't get a traceback.

And so this is a way to eliminate or catch a traceback.

Something that would otherwise be a traceback.

从 :8:44 开始播放视频并学习脚本8:44

That's what this is for.

So if you got some line of code, and you know that this might blow up and

have a traceback, then you use try/except around it, okay?

So let's take a look at something.

从 :8:55 开始播放视频并学习脚本8:55

This is a sample we had from the previous code.

And it comes down, it sticks Hello Bob in, then it converts this to an integer.

And we know that if these things are aren't digits, this code blows up, and

so it runs and we get the traceback.

And the traceback happens because of this line right here, but the key thing about

the traceback is that the traceback, as I told you before, it stops.

And it stopped at line 2, which means this is the last line it executed, but

it doesn't continue, which means this code is gone.

I mean, it never gets there because it's like, I'm confused, I'm quitting, I have quit at line 2, okay? I quit at line 2.

从 :9:40 开始播放视频并学习脚本9:40

So that code, it's as if it's not there.

Now sometimes that's fine with you, you just want to blow up and

you want to see the message, you want to go look at line 2 and fix line 2 or

maybe you typed the wrong stuff in.

从 :9:53 开始播放视频并学习脚本9:53

But sometimes you want to control for this.

You want to say, you know what, I know what I want to do here,

and I don't want to die, I don't want to blow up.

I want to continue, I want to put out an error message instead.

And so, the key is, is when this code blows up, it's something that you kind of

take personally because you are that set of instructions, and

when a traceback happens inside the memory or CPU, that's you that's being vaporized.

You've been traced back, and so we take it kind of personally.

I mean, if you were to use software that I built, like the autograder for

this class and you started getting tracebacks,

I'm like, hey, that's kind of a personal thing.

I didn't do my job well, I didn't catch all the errors.

I didn't think of everything.

You could type something that will cause my code to blow up.

And so I take that kind of personally.

And so we have to be able to compensate for

situations that we know might cause errors.

Especially those where the user can type something

that can cause my program to blow up.

That's really, like, I'm going to let you blow my program up?

I am going to compensate.

I'll tell you, sorry, that's bad data.

But I don't want you to ever see a traceback

because it's kind of shameful to see a traceback for a professional programmer.

Okay, so here's how it works, it's a bit of stuff with some indentation and colons.

It looks like a lot, but don't worry, you'll figure it out.

So the idea is, is you have a line of code that you know is dangerous.

So this conversion of an integer,

let's just say this came from an input statement.

In this case, we'll just make it be Hello Bob.

We know this is going to fail.

And so, this is the line in which we kind of want to take out insurance on it.

So instead of just putting this line in here,

like we did in the previous example, we just had it right there.

Instead of taking that line there, we'd say, you know what, we're going to take and

stick this in a try and except block.

So we say the word try. Try ends in a colon, which means it's an indented block of code.

And then we put the dangerous line in there, and then we put except.

And then the except is kind of like an else, an if-then-else, but

what it really is is code that Python will execute if something goes wrong.

So this is either going to run and work and skip this.

Or if it goes bad, it's going to run, blow up, and then run this stuff and

then continue on.

But in no case you will get a traceback, meaning if this line is going to generate

a traceback, it actually just runs the except clause.

So it's kind of like if things work out, do this, if things don't work out, do this other thing.

So in this case, when this runs, this is going to fail, because that's Hello Bob.

And then it's going to come out here and set this to -1.

So that's going to blow up.

Set this to -1, that's going to print it out.

So it says, First equals -1.

And so we didn't traceback.

In the previous time we ran this, it traced back because we caught it.

Now, the way the try/except works is, if everything is fine, it has no effect.

So if we, the next thing we're going to convert is 123, the digits, 1, 2, 3,

in a string.

We do a try, and we try to convert it and it works, and so we just keep on going.

We don't run the except. So this code does not run because this code succeeded, there was no traceback that was going to be generated.

There's a traceback generated up here.

There was no traceback generated here, so it comes through, and

the result is istr ends up with a integer, 123.

And so it's an insurance policy or it says, I know this might blow up and

if it does, I'm giving you alternate text or alternate code to run.

从 :13:25 开始播放视频并学习脚本13:25

So the thing about the try/except block, and you might be tempted to do this, and

that is, if you're getting tired of tracebacks and

blowing up, you might want to put your entire program in a try and except block.

And you might say try blah, blah, blah, blah, and

then except something bad happens.

The problem is, is if your program's blowing up,

you actually want to know about it.

And the way the try and except block works is, if it's in the middle of a try and

except block and something goes wrong, like in this particular line,

it doesn't come back and finish the try and except block.

It actually exits to the except and then comes out.

And to draw this in a diagram, so here we go,

we start this thing, we're in the try block.

We're doing print, print's safe, doesn't hurt anything.

We do this, this blows up with a traceback.

Traceback and then that says go to the except block,

run whatever is in the except block and then continue on.

What's not going to happen is it's not going to go back up and do this, or

back up and try this one again.

No, no, no, no, no.

Once it gets the except block, there's only one way out to the bottom, so

this line of code never executes.

And so that's one of the things we try to do where we just,

you don't put too much stuff in.

You would put this print statement out here and this print statement out here,

and you'd only put one line in the try/except block, if possible.

Sometimes you put a few more lines in there, but you try to minimize.

从 :14:48 开始播放视频并学习脚本14:48

You know what line is dangerous. Print is not dangerous, these two prints are not really dangerous, don't put them in the block.

Because any line in the block, as soon as it hits a bad line with a traceback,

it's out of the block, runs the except, and then continues on.

从 :15:3 开始播放视频并学习脚本15:03

So here's a more practical example,

从 :15:9 开始播放视频并学习脚本15:09

Where we're going to read a number from the user and

print out either Nice work or Not a number.

And so, we take an input statement, which stops and waits for

us to type, and then we type 42.

And 42 goes into rawstr and then we know that this int is dangerous.

Right? And this rawstr came from the user, whatever the user typed.

And so we put it in the try block, and if it's 42, it converts and

it says ival is greater than 0, we print it out, so it says Nice work.

Now, we'll run this code again.

Okay, we run it a second time and now we enter something, it says forty-two

but it's like f-o-r-t-y.

And so forty-two is what goes in here.

We as the programmer had no control over what our crazy user was typing, right?

You're starting to be a programmer and crazy users do crazy things to your poor

programs, even if they're only like seven lines long.

So we got forty-two coming in here,

we know this is going to blow up, this int is going to blow up with a traceback.

从 :16:15 开始播放视频并学习脚本16:15

But that's okay, we've compensated for that and we told Python, hey,

we know that might happen and if you detect a traceback,

jump straight into the except block, run this, set it to -1, and then continue on.

So this is the de-indent of the try/except block, and

if it's greater than 0 we say Nice work.

But in this case, it's not, and we say, Not a number, and so it comes out with Not a number. What's not here is a traceback.

There is no traceback in this, that's what we achieved.

And it doesn't hurt. When it works, the try/except kind of does nothing because the except code, when it works, is ignored.

So it's like code you add in case something happens in an other line of code.

从 :16:56 开始播放视频并学习脚本16:56

Pretty cool, actually.

从 :16:59 开始播放视频并学习脚本16:59

So you have a couple exercises and I've got some videos of those exercises.

So in summary, what we talked about in this chapter is comparison operators, logical questions. A key is that these comparison operators don't change their arguments.

You can say if x is less than 5, doesn't change the value for x.

We have indentation and how important indentation is.

从 :17:23 开始播放视频并学习脚本17:23

One-way decisions with if, two-way decisions with if-then-else.

Nested decisions where you have an if inside of an if that moves on in.

Else-if, and then try and except to catch errors that you want to catch, okay?

So, thanks a lot.