In the last video, we learned about Booleans and Boolean expressions. In this video, we're going to learn about the real power of Boolean expressions, mainly the ability to use a Boolean expression to make our programs do different things depending upon the state of our variables. In other words, we're going to learn how to write code that executes conditionally using conditional statements.

Let's take a look at an example program.

```
1   def main():
2       # get two integers from the user
3       pizzas = int(input("How many pizzas did you order? "))
4       people = int(input("How many people are there? "))
5       # multiply by 8 slices per pie and divide
6       slices = pizzas * 8 / people
7       print(pizzas, "pizzas split between", people, "can have", slices,"slices")
8
9   main()
10
```

If you spend any time at all looking at this program, you will notice that it asks the user to enter how many pizzas they have ordered as well as how many people are there. It then calculates the number of slices that each person can eat if everybody is eating the same amount, not bad.

But what happens if somebody using the program happens to enter zero for the number of people? You may wonder why anybody would do this, but reality says that you just never know what a user will do. If the user does enter a zero for the number of people, then the program will divide by zero.

As we know from mathematics, division by zero is undefined. So if I run this program, And I put in the number of pizzas as four and the number of people as zero, as expected, I get an error when the second number is zero. What we would really like to do is to stop the program from dividing if the user enters zero as the second member.

This is where the conditional statement comes into play. Let's look at what this would look like in the flow chart. So we always start our flow chart with the start block. That's the one in the oval. Next, we have a parallelogram which represents the input of the number of pizzas and the number of people.

For a condition statement we use a diamond to represent the decision block. The diamond shape has a different function than all of the other shapes. This is because we will choose between two different possible paths. You might recall from the video where we introduced the flowcharts in the introduction module, that there's only one exit from all the other shapes.

Now when we draw the diamond, the condition is placed inside the diamond in the form of a

question. In this case, we want to know that the number of people is not equal to zero. Exiting the diamond in one direction represents true, exiting the diamond in the other direction represents false.

On the true side of this decision, we want to calculate the result of dividing the number of slices, the number of pizzas times eight and divided by the number of people, put it in the variable slices, and then printing it. On the false side, we want to do nothing.

So just make sure that you correctly label which direction is which so that it's clearly understood. There, that's all there is to it. Now let's translate this to code. To write the conditional in Python, we use the keyword if right after this input. Followed by a Boolean expression that test whether or not we want to execute the code inside the conditional, the true side of the decision.

In this case, we want to know whether or not the number of people is not equal to zero, `people != 0` . That's because if it's not equal to zero then we want to do something. The Boolean expression is followed by a colon, which indicates that we're going to write a new block of code.

Recall that Python uses indentation to indicate blocks. So we must indent the statements that are on the true side of the decision under the if. If people are not zero then the condition is true and we will calculate the result by dividing the number of slices, pizzas times eight, by the number of people and print the result.

```
1   def main():
2       # get two integers from the user
3       pizzas = int(input("How many pizzas did you order? "))
4       people = int(input("How many people are there? "))
5       if people != 0:
6           # multiply by 8 slices per pie and divide
7           slices = pizzas * 8 / people
8           print(pizzas, "pizzas split between", people, "can have", slices,
9                 "slices")
10
11  main()
12
```

If people is not zero is false, nothing happens and both of these go to the end. This form of the conditional is often called a one sided conditional because something is happening on only one path after the decision. The general form is this, where `if` is a reserved word and has to be there. It lets Python know that a conditional statement is beginning.

The condition is a Boolean expression that determines whether or not the body of the condition will be executed. The colon follows the condition and indicates the start of the block. The `if` block contains one or more statements to be executed if the condition is true.

Furthermore, the statements have to be indented the same number of spaces. Sometimes this

block is called the **body** of the if. As we know, indentation is an important part of Python. Python requires the same number of spaces be used for a block of code. Some programmers will use two spaces, others will use four. Python style guides recommends four to be used to set off a block. IDLE lets us use the Tab key to insert spaces automatically.

It's important to keep the number of spaces and indentation consistent in order not to get an error message stating unexpected indentation. But also Python gets really confused if you end up with a file that mixes tabs and spaces for indentation. This can happen when you're using different editors to edit the same Python file. This is one of the reasons we're recommending that you stick with IDLE for this class.

One of the bad things about the program we have written is if zero people is entered, nothing happens. To the user, it will look like the program froze. It may be better to provide feedback to the user to indicate that the number of people can't be zero. To do this, we want to add something to the false side of the conditional.

So let's go back to our flow chart. In our flow chart, on the false side, we want to do something.

In other words, if the number of people is not zero then go ahead and calculate the result and print it. Otherwise, we want to print a message. That says that the number of people can't be zero. Print "People are not be zero", or something to that effect. This is often referred to as a **two-sided conditional**, because there is something on both sides of the condition.

The general form looks like this. As before we have the `if`. To write a two way conditional `else` is added at the same level as the `if` to which it belongs. `else` is another reserved word. And it's reserved to introduce the second part of the conditional. The `else` block is one or more statements to be executed if the condition of the `if` statement is false.

Like the `if` block, it must be indented one level deeper than the line with the `else`. Now let's add it to the if statement that we wrote in the Python code. To do this we need an `else` block. The `else` block contains the alternate block of code that is executed whenever the condition is false.

So let's go back to our program. So to do that, to add `else` to this program, we started the last line of the program where we had the `print`. We're going to add a line, change the indent so that it's back to the same level as the `if` statement.

This is going to end the `if` block. By adding the keyword `else`, we start the other portion of the conditional adding the single statement to this block, which will print that people cannot be zero. By using the `if-else` form of the conditional we know that something will definitely be printed on the screen depending upon the number of people that the user enters, even if they give us bad input.

```
def main():
    # get two integers from the user
    pizzas = int(input("How many pizzas did you order? "))
    people = int(input("How many people are there? "))
    if people != 0:
        # multiply by 8 slices per pie and divide
        slices = pizzas * 8 / people
        print(pizzas, "pizzas split between", people, "can have", slices,
              "slices")
    else:
        print("people cannot be 0")
main()

```

Awesome, we have now seen the conditional statement in Python has two forms: the one-sided form where there is only something happening on the true side of the condition, and the two-sided form where there's something happening on both the true and false sides of the condition. All of this from the power of a Boolean expression.

In the exercises that follow this video you'll have a chance to reflect on and practice this. Okay, that's all for now. Thanks for watching Align online.