Multiple Conditions

by Sophia

WHAT'S COVERED

In this lesson, you will learn how to use nested conditional statements and their outcomes with multiple variables. Specifically, this lesson covers:

1. Nested Conditionals with One Variable
2. Nested Conditionals with Multiple Variables

1. Nested Conditionals with One Variable

So far, we have looked at conditional statements using if/else:

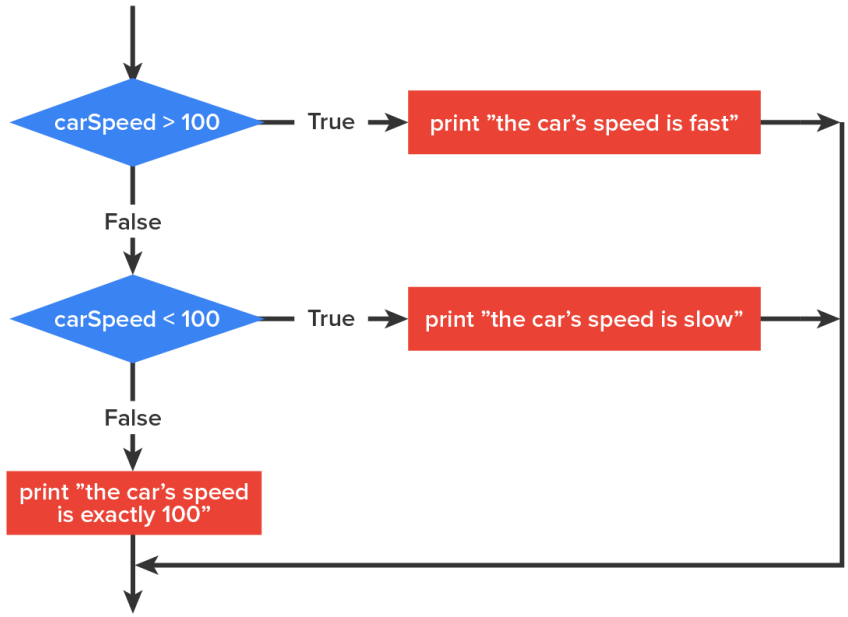
EXAMPLE

```
grade = 75
if grade > 60:
    print("You passed")
else:
    print("You did not pass")
```

We also looked at chained conditional statements with if/elif/else statements.

EXAMPLE

Here is an example in a flowchart.



EXAMPLE

Here is a code example.

```
carSpeed = int(input("Enter in the car's speed between 0 and 200: "))
if carSpeed > 100:
    print("The car's speed is fast!")
elif carSpeed < 100:
    print("The car's speed is slow.")
else:
    print("The car's speed is exactly at 100.")
```

If the car’s speed is greater than 100, we should get the results of the first condition.

```
Enter in the car's speed between 0 and 200: 120
The car's speed is fast!
```

If the car’s speed is less than 100, we should get the results from the second condition.

```
Enter in the car's speed between 0 and 200: 99
The car's speed is slow.
```

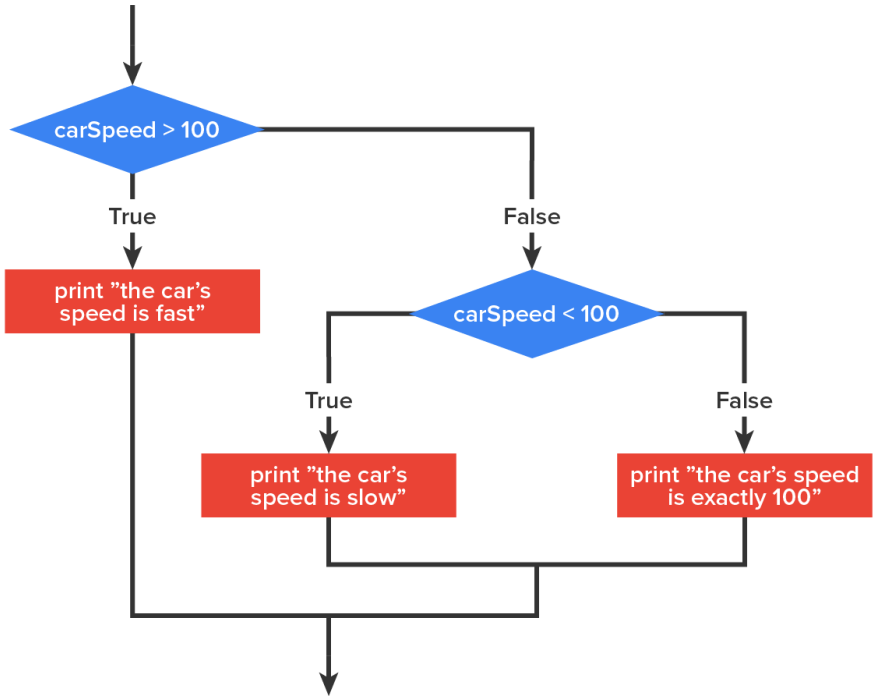
If the car’s speed is exactly at 100, the result is the third output from the else.

```
Enter in the car's speed between 0 and 200: 100
The car’s speed is exactly 100.
```

Conditional statements are able to be **nested conditionals**, meaning they can appear in one of the branches of another conditional statement using multiple conditions. You can represent chained conditional statements using nested conditionals.

Using our second example in a nested conditional statement, let's see what a flowchart could look like.

EXAMPLE



As we can see, the outer conditional contains two branches. The first branch contains a simple statement. The second branch contains another if statement, which has two branches of its own. Those two branches are both simple statements, although they could have been conditional statements as well.

Remember the importance of indenting with conditional statements? In a past lesson we explained that to finish a conditional statement, we end the statement with a colon character (:) and the line(s) after the if statement is/are indented. This applies to a nested condition as well, only now each nested condition's block needs to be indented further to match the nested condition.

Now, let's look at a nested condition with Python code. Notice the indentation of the nested condition. The entire nested if statement is indented so you can see which else statement links to which if statement.

EXAMPLE

```
carSpeed = int(input("Enter in the car's speed between 0 and 200: "))
if carSpeed > 100:
    print("The car's speed is fast!")
else:
    if carSpeed < 100:
```

```
        print("The car's speed is slow.")
    else:
        print("The car's speed is exactly at 100.")
```

We can see that the same results are displayed as before.

```
Enter in the car's speed between 0 and 200: 120
The car's speed is fast!

Enter in the car's speed between 0 and 200: 99
The car's speed is slow.

Enter in the car's speed between 0 and 200: 100
The car’s speed is exactly 100.
```

The conditions end up being very similar to one another. But the chained conditional statement may be easier to follow logically, because there is only one variable being tracked in the nested conditional statements with the `carSpeed`.

Let’s revisit an example of a chained conditional statement that we used in a previous lesson to see how the use of nested conditionals can make the logic easy to follow.

EXAMPLE

```
grade = 85
if grade > 90:
    print("You got an A")
elif grade > 80:
    print("You got a B")
elif grade > 70:
    print("You got a C")
elif grade > 60:
    print("You got a D")
elif grade <= 60:
    print("You got a F")
```

Remember this one? We used the elif statement to determine what the correct output should be.

Instead of using the elif for each condition, we could split each condition separately and nest the conditional statements instead.

EXAMPLE

```
grade = 85
if grade > 90:
    print("You got an A")
```

```
else:
    if grade > 80:
        print("You got a B")
    else:
        if grade > 70:
            print("You got a C")
        else:
            if grade > 60:
                print("You got a D")
            else:
                if grade <= 60:
                    print("You got a F")
```

Visually, this is easier to follow compared to the chained conditional statement that we originally had.

TERM TO KNOW

Nested Conditional

A conditional statement that appears in one of the branches of another conditional statement.

2. Nested Conditionals with Multiple Variables

Nested conditional statements become easier to follow when you have multiple variables that are being tracked for various criteria. Let's take a look at an example with multiple variables based on two different conditions. The first condition is whether or not we are hungry and the second condition is whether we want a healthy meal.

The conditions may look like the following.

EXAMPLE

```
hungry = "y"
healthy = "y"
if hungry == "n" and healthy == "n":
    print("Not hungry.")
elif hungry == "n" and healthy == "y":
    print("Not hungry.")
elif hungry == "y" and healthy == "n":
    print("Getting some junk food.")
elif hungry == "y" and healthy == "y":
    print("Getting a healthy meal.")
```

You may notice that the results look like a truth table. In converting this to a nested conditional, we will have the outer if statement based on whether or not we are hungry, and the inner conditions based on whether we want to have a healthy meal or not. Each inner condition is entered only if the outer condition is satisfied, so if we are not hungry, the inner condition is never tested.

EXAMPLE

```
hungry = "y"
healthy = "y"
if hungry == "n":
    if healthy == "n":
      print("Not hungry.")
    else
      print("Not hungry.")
else:
    if healthy == "n":
        print("Getting some junk food.")
    else:
        print("Getting a healthy meal.")
```

Note that if we're not hungry, the check on the healthy meal isn't necessary, since the output is the same. This is the same as a short-circuit evaluation of the criteria. We can also further combine the criteria.

EXAMPLE

```
hungry = "y"
healthy = "y"
if hungry == "n":
    print("Not hungry.")
else:
    if healthy == "n":
        print("Getting some junk food.")
    else:
        print("Getting a healthy meal.")
```

Visually, this makes it much easier to follow than the chained conditional statement. We could make a change to get the user's input as well in choosing the answers.

EXAMPLE

```
hungry = input("Are you hungry? Enter y if you are and n if you are not: ")
healthy = input("Did you want a healthy meal? Enter y if you do and n if you do not: ")
if hungry == "n":
    print("You are not hungry.")
else:
    if healthy == "n":
        print("Getting some junk food.")
    else:
        print("Getting a healthy meal.")
```

Note that this is partially flawed, because there's no point asking the user if they want a healthy meal or not if they aren't hungry. We could change this to only prompt the user about what type of meal they want if they said

that they were hungry. To do that, we would first prompt the user to see if they were hungry. Then, within the else condition if the user is hungry, we can prompt if they want a healthy meal or not. Let's see what the code would look like with that change.

EXAMPLE

```
hungry = input("Are you hungry? Enter y if you are and n if you are not: ")
if hungry == "n":
    print("You are not hungry.")
else:
    healthy = input("Did you want a healthy meal? Enter y if you do and n if you do not: ")
    if healthy == "n":
        print("Getting some junk food.")
    else:
        print("Getting a healthy meal.")
```

This approach works much better, and is an example that couldn't be performed in a chained conditional statement.

SUMMARY

In this lesson, we learned how we can use simple conditions and chained conditions in **nested conditionals**. Using **one variable** in the `carSpeed` and grade examples, we noticed how the logic stayed the same in a chained condition example versus a nested example, but it was easier to follow the logic with one method over another depending on what was being compared. Next, we saw the hungry/healthy example that used **multiple variables**. We learned how the logic and outcome can be improved by using nested conditionals. By nesting the conditionals, it simplified the program and made it easier to follow the logic.

Best of luck in your learning!

TERMS TO KNOW

Nested Conditional
A conditional statement that appears in one of the branches of another conditional statement.