Reading: Reference guide: Conditional statements

Conditional statements are an essential part of programming. They allow you to control the flow of information based on certain conditions. In Python, **if, elif,** and **else** statements are used to implement conditional statements. Using conditional statements to branch program execution is a core part of coding for most data professionals, so it's important to understand how they work. This reading is a reference guide to conditional statements.

Save this course item

You may want to save a copy of this guide for future reference. You can use it as a resource for additional practice or in your future professional projects. To access a downloadable version of this course item, click the following link and select "Use Template."

Reference guide: Conditional statements

OR

If you don't have a Google account, you can download the item directly from the following attachment.

Reference guide: Conditional statements

Conditionals syntax

In earlier videos, you learned some built-in Python operators that allow you to compare values, and some logical operators that you can use to combine values. You also learned how to use operators in **if-elif-else** blocks.

Note: The following code block is not interactive.

The basic syntax of **if-elif-else** statements in Python is as follows:

```
if condition1:
  # block of code to execute if the condition evaluates to True

elif condition2:
  # block of code to execute if condition1 evaluates to False
  # and condition2 evaluates to True
```

```
else:
  # block of code to execute if BOTH condition1 and condition2
  # evaluate to False
```

Here, **condition1** and **condition2** are expressions that evaluate to either True or False. If the condition in the if statement is true, then the block of code that follows is executed. Otherwise, it is skipped.

The **elif** statement stands for "else if," and it is used to specify an alternative condition to check if the first condition is false. You can have any number of **elif** statements in your code. If the preceding condition is false and the **elif** condition is true, then the block of code that follows the **elif** statement is executed.

The **else** statement is used to specify what code to execute if both the if statement and any subsequent **elif** statements are false.

Here is an example that uses all three kinds of statements:

```
x = 8
if x > 5:
  print('x is greater than five')
elif x < 5:
  print('x is less than five')
else:
  print('x is equal to five'
```

Output:

```
x is greater than five
```

Omitting else

Often, you'll find that there is no need to use an **else** statement, because it is superfluous in the logical context of your code. Consider this example:

```
def greater_than_ten(x):
    if x > 10:
        return True
    else:
        return False

print(greater_than_ten(15))
print(greater_than_ten(2)
```

Output:

```
True
False
```

This function will return **True** if **x** is greater than 10, otherwise it will return **False**. Notice that **else** is on its own line, followed by an indented **return** statement in the last line. This is perfectly fine code, and you'll encounter code like this routinely.

But there's another form that you'll routinely encounter that skips this last step. Consider this code, which is equivalent to the function above:

```
def greater_than_ten(x):
    if x > 10:
        return True
    return False

print(greater_than_ten(15))
print(greater_than_ten(2)
```

Output:

```
True
False
```

In this case, there is no **else** statement, but it doesn't make a difference. The logic is the same. The function will begin execution from the top and move down. When it gets to line 2, it will evaluate whether or not **x** is greater than ten, and if so, it will move to the indented line 3, which is a **return** statement. Once this **return** statement executes, the function exits. It does not continue any further execution.

However, if the condition in line 2 does not evaluate as **True**, the function will not execute line 3, because it is an indented sub-condition of line 2. Instead it will proceed directly to line 4, which says that the function must return False and exit. Note also that the final statement, **return False**, is at the same indentation level as the **if** statement.

As you continue to gain skills and experience in programming, you might find that you prefer one way over the other. Use whichever way is best for you, and make sure you're familiar with both ways.

Key takeaways

Some important things to note about conditional statements in Python:

- The **elif** and **else** statements are optional. You can have an **if** statement by itself.
- You can have multiple **elif** statements.
- You can only have one **else** statement, and only at the end of your logic block.
- The conditions must be an expression that evaluates to a Boolean value (True or False).
- Indentation matters! The code associated with each conditional statement must be indented below it. The typical convention for data professionals is to indent four spaces. Indentation mistakes are one of the most common causes of unexpected code behavior.