3.2.9

Membership and Logical Operators

As Tom has demonstrated, decision statements play an important role in data retrieval. However, they can become more powerful and complex when you need to create compound comparison expressions or conditional statements. Situations may arise in which you may have to test two or more conditions in a data structure. This is where membership and logical operators come in handy. Next, Tom will show you how to test conditions using logical and membership operators.

Membership Operators

Membership operators are used to test if an object, like a string, integer, or other data type is present in an expression, list, tuple, or dictionary. The following table describes these operators and provides an example of each one.

Operator	Meaning	Example

in	Returns True if a sequence with the specified value is present in the object.	`counties = ["Arapahoe","Denver","Jefferson"]`
		`if "Arapahoe" in counties:`
		`print("True") else:`
		`print("False")`
		This prints `"True"` because *Arapahoe* is in the counties list.

not in	Returns True if a sequence with the specified value is **not** present in the object.	`counties = ["Arapahoe","Denver","Jefferson"]`
		`if "El Paso" not in counties:`
		`print("True") else:`
		`print("False")`
		This prints `"True"` because *El Paso* is not in the counties list.

Let's practice creating a membership operation by determining if "El Paso" is in the counties list.

In the `Python_practice.py` file, add the following code to your file and run the file.

```
counties = ["Arapahoe","Denver","Jefferson"]
if "El Paso" in counties:
    print("El Paso is in the list of counties.")
else:
    print("El Paso is not the list of counties.")
```

The output of this code will be:

```
El Paso is not the list of counties.
```

In the code, the decision statement checks if "El Paso" is in the counties list. If it is true, then the first print statement is printed to the screen. But the `if` statement is false, so the `else` statement is checked; since it is true, the second print statement is printed to the screen.

Logical Operators

Logical operators allow us to connect multiple comparison expressions to create a compound expression. There are three logical operators: **and, or,** and **not.** The following table describes these operators and provides an example of each one.

Operator	Meaning	Example

and	Evaluates two Boolean expressions into one compound expression. The compound expression is true if **both** Boolean expressions are true.
	
	If one of the expressions is false, then the compound expression is false.
		`x = 5`
		`y = 5`
		`if x == 5 and y == 5:`
		`print("True")` `else:`
		`print("False")`
		
		This prints `"True"` because *x* = 5 is true and *y* = 5 is true.

or	Evaluates two Boolean expressions into one compound expression.	`x = 5`
		`y = 5`
		`if x == 3 or y == 5:`

	The compound expression is true if **either** Boolean expression is true.
	
	If one of the expressions is false, then the compound expression is true. If both expressions are false, then the compound expression is false.
		`print("True")`
		`else:`
		`print("False")`
		
		This prints `"True"` because *x* = 3 is false and *y* = 5 is true.

not	Evaluates a Boolean expression. The expression is true if the conditional is **false**.	`x = 5`
		`y = 5`
		`if not(x > y):`
		`print("True")` `else:`
		`print("False")`
		
		This prints `"True"` because *x* is not greater than *y*. If *x* = 6, then it would print `"False"` because *x* is greater than *y*.

We can combine membership and logical operations to test certain conditions. Let's practice creating a compound membership and logical operation using the list of counties.

We will use the "and" operator to determine if two counties, Arapahoe and El Paso, are in the list of counties.

```
if "Arapahoe" in counties and "El Paso" in counties:
    print("Arapahoe and El Paso are in the list of counties.")
else:
    print("Arapahoe or El Paso is not in the list of counties.")
```

The output of this code will be:

```
Arapahoe or El Paso is not in the list of counties.
```

In the code, the decision statement checks if both Arapahoe and El Paso are in the counties list. Arapahoe is in the counties list, which is "true," but El Paso is not in the counties list, which is "false." Therefore, the compound expression is false.

Next, let's check if *either* "Arapahoe" *or* "El Paso" is in the counties list. In the previous code, replace the logical operator "and" with "or" and run the file.

```
if "Arapahoe" in counties or "El Paso" in counties:
    print("Arapahoe or El Paso is in the list of counties.")
else:
    print("Arapahoe and El Paso are not in the list of counties.")
```

The output of this code will be:

```
Arapahoe or El Paso is in the list of counties.
```

The decision statement checks if either Arapahoe or El Paso is "true." We know that Arapahoe is in the counties list, which is "true," but El Paso is not in the counties list, which is "false." Therefore, the compound expression is "true," and the first print statement is executed.

If Arapahoe and El Paso are not in the list, then both expressions are "false." Therefore, the compound expression is "false," and the following statement is printed to the terminal.

```
Arapahoe and El Paso are not in the list of counties.
```