Hello Class,

The `and` and `or` operators are logical operators. In programming, logical operators are used in conditional expressions to evaluate two Boolean conditions (`True` or `False`), returning a single Boolean condition based on the operator rule (`and` and `or` rule).

In brief and plain terms:

The operator `and` evaluates Boolean conditions based on the following logical rules:

Table 1

Truth Table for and

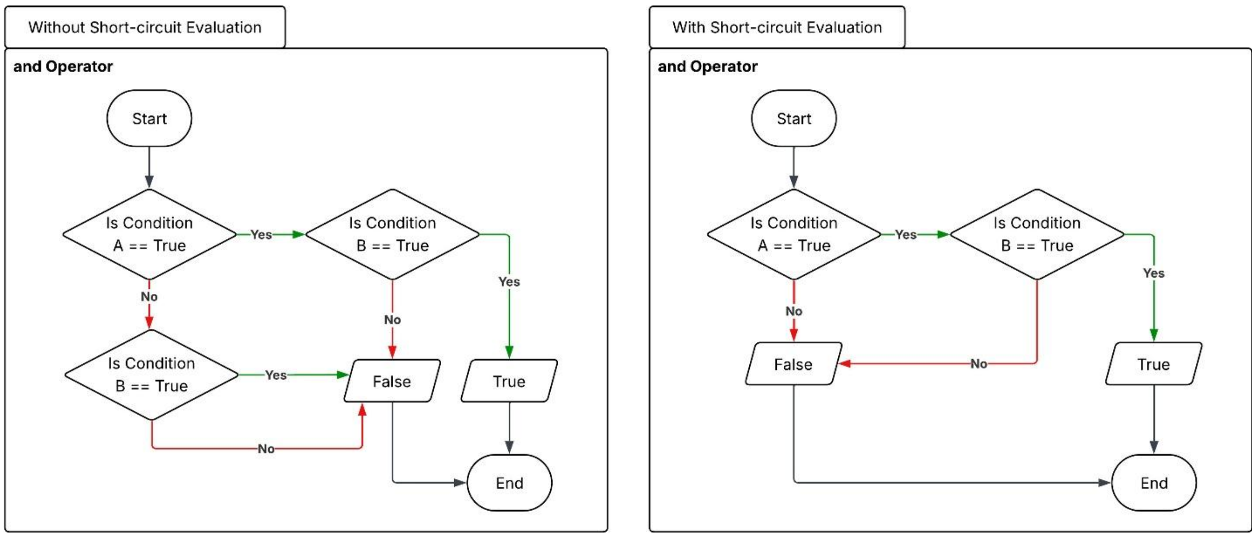
Condition A	Condition B	A and B
False	False	False
False	True	False
True	False	False
True	True	True

Note: The table provides all possible outcomes of the logical operators `and` , comparing two conditions. Modified table from "Short-Circuit Evaluation in Programming" (Ricciardi, 2025)

In programming, it evaluates from left to right, obeying the short-circuit's rules that we discussed in module 2. See Figure 1 below for the decision flow of the `and` operator:

Figure 1

Truth Table for `and`



Note: The diagram illustrates the logical operator `and` decision flows without and with short-circuit evaluation. Modified figure from "Short-Circuit Evaluation in Programming" (Ricciardi, 2025).

The `and` operator could be used to evaluate what kind of day it is weather-wise:

Pseudocode:

```
IF temperature > 20 AND is_sunny THEN
  PRINT "It's a warm and sunny day."
ELSE IF temperature > 20 AND NOT is_sunny THEN
  PRINT "It's warm but not sunny."
ELSE IF temperature <= 20 AND is_sunny THEN
  PRINT "It's a cold but sunny day."
ELSE
  PRINT "It's cold and not sunny."
ENDIF
```

In Python:

```python
if temperature > 20 and is_sunny:
    print("It's a warm and sunny day.")
elif temperature > 20 and not is_sunny:
    print("It's warm but not sunny.")
elif temperature <= 20 and is_sunny:
    print("It's a cold but sunny day.")
else:
    print("It's cold and not sunny.")
```

On the other hand, the operator `or` evaluates Boolean conditions based on the following logical rules:

Table 1

Truth Table for or

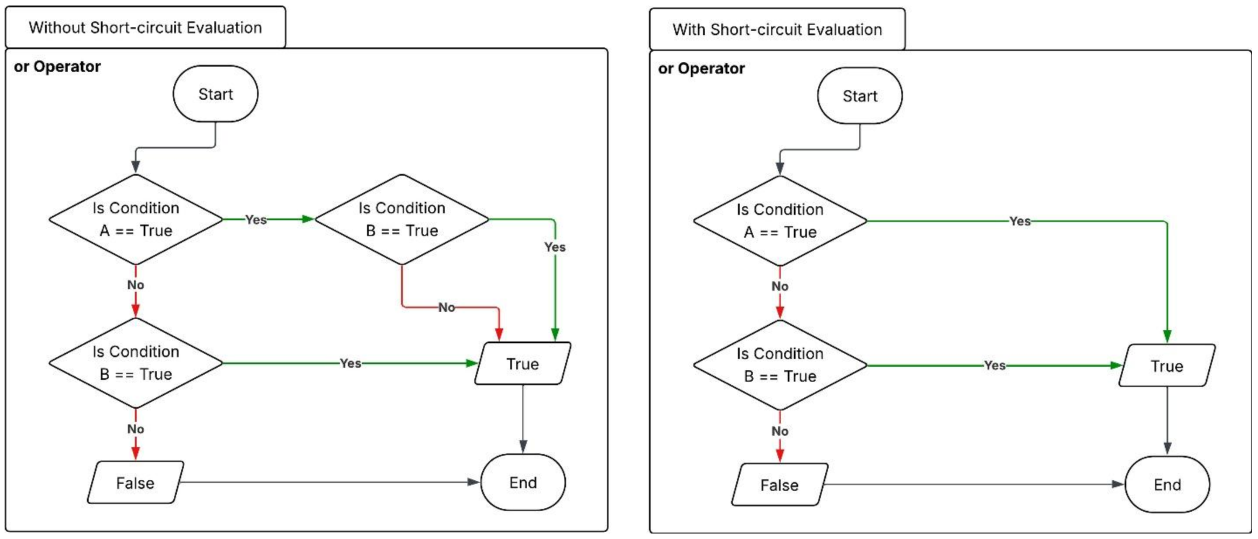
Condition A	Condition B	A `or` B
False	False	False
False	True	True
True	False	True
True	True	True

Note: The table provides all possible outcomes of the logical operators `or`, comparing two conditions. Modified table from "Short-Circuit Evaluation in Programming" (Ricciardi, 2025)

In programming, it evaluates from left to right, obeying the short-circuit's rules that we discussed in module 2. See Figure 2 below for the decision flow of the `and` operator:

Figure 2

Truth Table for `or`



Note: The diagram illustrates the logical operator `or` decision flows without and with short-circuit evaluation. Modified figure from "Short-Circuit Evaluation in Programming" (Ricciardi, 2025)

The `or` operator could be used to evaluate what drinks I have:

Pseudocode:

```
IF have_coffee_hot OR have_coffee_ice OR have_tea_hot OR have_tea_ice
  IF have_coffee_hot OR have_tea_hot THEN
    PRINT "I have a hot drink."
  ENDIF
  IF have_coffee_ice OR have_tea_ice THEN
    PRINT "I have something cold to drink."
  ENDIF
ELSE
  PRINT "I don't have drinks."
ENDIF
```

In Python:

```
if have_coffee_hot or have_coffee_ice or have_tea_hot or have_tea_ice:
    if have_coffee_hot or have_tea_hot:
        print("I have a hot drink.")
    if have_coffee_ice or have_tea_ice:
        print("I have something cold to drink.")
else:
    print("I don't have drinks.")
```

We can mix both operators in a compound conditional expression, for instance, the drink example could be written as:

Pseudocode:

```
IF (have_coffee_hot OR have_tea_hot) AND (have_coffee_ice OR have_tea_ice) THEN
  PRINT "I have both hot and cold drinks."
ELSE IF (have_coffee_hot OR have_tea_hot) AND NOT (have_coffee_ice OR
have_tea_ice) THEN
  PRINT "I have only hot drinks."
ELSE IF (have_coffee_ice OR have_tea_ice) AND NOT (have_coffee_hot OR
have_tea_hot) THEN
  PRINT "I have only cold drinks."
ELSE
  PRINT "I don't have drinks."
ENDIF
```

In Python:

```python
if (have_coffee_hot or have_tea_hot) and (have_coffee_ice or have_tea_ice):
    print("I have both hot and cold drinks.")
elif (have_coffee_hot or have_tea_hot) and not (have_coffee_ice or have_tea_ice):
    print("I have only hot drinks.")
elif (have_coffee_ice or have_tea_ice) and not (have_coffee_hot or have_tea_hot):
    print("I have only cold drinks.")
else:
    print("I don't have drinks.")
```

Note that I used "( )" to encase the `or` logical expression; not doing so could change the result/meaning of the condition, as the `and` operator has higher precedence than or. See Table 2 for the precedence of logical operators.

Table 2

Precedence of Logical Operators

Expression	Precedence Illustration
A or B and C	A or (B and (!C))
A and B or C and D	(A and B) or (C and D)
A and B and C OR D	((A and B) and C) OR D
!A and B or C	((!A) and B) and C

Note: The table provides examples that illustrate how logical operator precedence is implemented within compound conditional expressions. Modified table from "Short-Circuit Evaluation in Programming" (Ricciardi, 2025).

Finally, the `and` operator is a conjunction (restrictive operator); it is used when all conditions must be true to proceed.
On the other hand, the `or` operator is a disjunction (permissive operator); it is used when only one condition needs to be true.

-Alex

References:

Ricciardi, A. (2025, September 20). *Short-circuit evaluation in programming.* Level Up Coding – Medium. https://medium.com/gitconnected/short-circuit-evaluation-in-programming-f922c5f0eec8