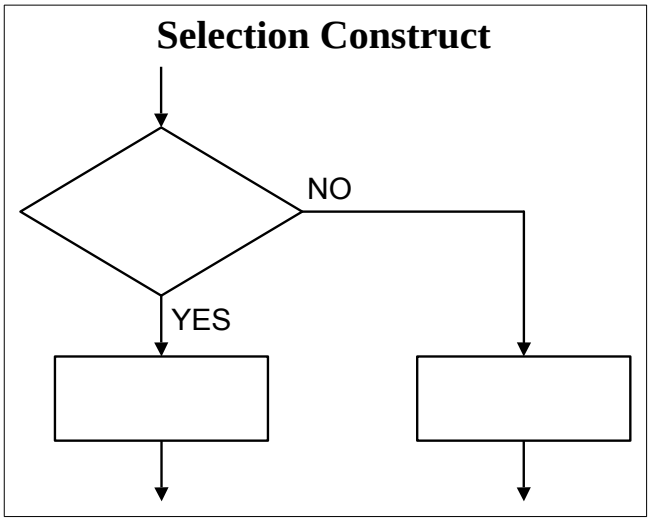
Topic 3.1: Selection (`if`)

We make decisions based on conditions everyday. For example, when we leave home in the morning, we may decide: if it is raining, I will carry an umbrella with me. We have selected a course of action based on the condition of the weather being rainy or not.

For a program to be useful, it also needs to react to different situations. The code will need to choose what to do based on some piece of information. In an algorithm, this ability is accomplished with the *selection* construct. The flowchart representation of the selection construct, shown to the right, uses a decision block where neither of the output paths flow back to re-enter above the decision block. It can be a useful exercise to be able to convert between a flow chart and program code.



Conditions

In the example of taking an umbrella if it is rainy, the action is based on the condition: *it is rainy.* It may either be true that it is rainy, or it may be false that it is rainy. This leads us to a definition for *condition* that is concise, and fits well for use in computer science:

condition: an expression that evaluates to a Boolean value – either `True` or `False`.

Recall that expressions with comparison operators, listed below, will evaluate to a boolean value.

Comparison Operators

symbol	*operation*
`==`	equal to
`!=`	not equal to
`>`	greater than
`>=`	greater than or equal to
`<`	less than
`<=`	less than or equal to

The Syntax of `if` in Java

Look at the following code snippet to refresh your memory about the syntax of an if statement in Java.

```
1 int x = -3;
2 if(x < 0) {
3     System.out.println("The value " + x + " is negative.");
4 }
5 System.out.println("The program is complete.");
```

```
1 The value -3 is negative.
2 The program is complete.
```

The condition must be enclosed in parentheses, and the code block that is to be conditionally executed is enclosed in curly braces. (If there is only a single statement in the code block, the curly braces are optional.)

Although not demonstrated with this one example, the first line of the output will not be printed if the value of `x` were set to something greater than or equal to zero, but the string "`The program is complete.`" will be printed whether or not the condition is true.

The Syntax of `if` in Python

Now examine the equivalent code in Python.

```
1 x = -3
2 if x < 0:
3     print("The value", x, "is negative.")
4 print("The program is complete.")
```

```
1 The value -3 is negative.
2 The program is complete.
```

Notice that there are no curly braces denoting the start and end of the conditional code block in Python. The colon signals to the Python interpreter that the condition is complete, and it is the indentation of the code that signals when a code block is complete. Whereas in many languages code indentation is best practice, in Python it is a strict requirement. Indentation can be any number of spaces (four spaces is the most common), but each line of the code block must be indented exactly the same amount. The table below summarizes the rules for writing a simple if statement.

- The line starts with the keyword `if`.
- After the condition, there is a colon (`:`).
- The block of code that belongs to the `if` must be indented (usually four spaces). Python uses indentation to know where the block starts and ends.
- When the condition is `False`, the indented block is skipped, and the program continues with the first unindented line after the `if`. This line must have the same indentation as the line that starts the `if` statement.

If your if statement has errors, first check to ensure you've remembered the colon after the condition and that the indentation is correct, as these are very common for beginning Python programmers.

The Syntax of `if..else` in Java

Here is another Java code segment.

```
1 int x = +3;
2 if(x < 0) {
3     System.out.println("The value " + x + " is negative.");
4 } else {
5     System.out.println("The value " + x + " is positive.");
6 }
7 System.out.println("The program is complete.");
```

```
1 The value +3 is positive.
2 The program is complete.
```

The code would have produced the same results as the previous code, had the initialization of variable `x` still set the value to `-3`.

The Syntax of `if..else` in Python

And the equivalent code in Python.

```
1 x = +3
2 if x < 0:
3     print("The value", x, "is negative.")
4 else:
5     print("The value", x, "is positive.")
6 print("The program is complete.")
```

```
1 The value 3 is positive.
2 The program is complete.
```

And the rules for `if..else` statements in Python:

- The `if` part is written exactly as before: keyword `if`, a condition, a colon (`:`), then an indented block of code.
- After the `if` block, on a new line with the same indentation as the if, write the keyword `else` followed by a colon (`else:`).
- There is no condition after else.
- The block of code that belongs to the `else` must be indented relative to the `else` line (again, usually four spaces). Python uses indentation to know where the `else` block ends.
- When the condition is `True`, the `if` block runs and the `else` block is skipped. When the condition is `False`, the `else` block runs and the `if` block is skipped. Exactly one of the two blocks will execute.
- After both blocks, the program continues with the next line that has the same indentation as the `if` and `else` lines.

An `if..else` statement will either execute the `if` block or the `else` block, never both. The `else` block catches everything that was not handled by the if block, so one of the two blocks will always execute.

Chained `if` in Java

Chained `if` in Python