<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>a == b</th>
<th>a != b</th>
<th>a and b</th>
<th>a or b</th>
<th>not a</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>
Boolean Logic

Programming languages make decisions in binary terms

Two boolean values: True and False

True and False can be assigned to variables, just like strings and numbers

Logical operators for boolean values: ==, !>, and, or, and not
**Equivalence**

\[ a \text{ == } b \]

Evaluates to True when \( a \) and \( b \) are the same:

- Both \( a \) and \( b \) are True
- Both \( a \) and \( b \) are False
Negation

\[ a \neq b \]

Evaluates to True when \( a \) and \( b \) are not the same:

- \( a \) is True and \( b \) is False
- \( a \) is False and \( b \) is True
And

\( a \text{ and } b \)

Evaluates to True when both \( a \) and \( b \) are True:

- \( a \) is True and \( b \) is True
Or

\( a \text{ or } b \)

Evaluates to True when \( a \) is True or \( b \) is True:

- \( a \) is True and \( b \) is True
- \( a \) is True and \( b \) is False
- \( a \) is False and \( b \) is True
Not \( \neg a \)

Evaluates to True when \( a \) is False and False when \( a \) is True:

- \( a \) is False
- \( b \) is False
### Truth Table

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>a == b</th>
<th>a != b</th>
<th>a and b</th>
<th>a or b</th>
<th>not a</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>
Order of Execution
Highest to Lowest

\[ a = b \]
\[ a \neq b \]
\[ \text{not } a \]
\[ a \text{ and } b \]
\[ a \text{ or } b \]

Brackets ( ) override operator precedence
Conditional Statements

Allow you to change the flow of a Python program

Consist of if-statements

Introduce code blocks
If/Else-Statements

if
else

Begins with the keyword if
Followed by a Boolean expression
May be followed by an else block for alternate conditions
If/Else If-Statements

if

elif

If-statement with more than one condition

Multiple elif blocks are permitted

else block is optional
Semantic Indentation

Code blocks are meaningful in Python.

Indentation marks blocks of code and is not an optional format.

Code blocks must be indented consistently by the same amount.

As a result, Python code is clearer and more legible.
Introduction to Computer Programming

CSCI-UA 2

Control Structures

Boolean Logic and If-Statements