# Lesson 3.16 — Boolean Logic

## Overview

**Objectives — *Students will be able to...***

- **Write** a game, similar to rock-paper-scissors or a pre-existing dice/card game.
- **Implement** the Applied Design stages in order to create a program with an end-user in mind.

**Assessments — *Students will...***

- **Submit** a program at the end of 2 or 3 class periods.
- **Submit** any Applied Design documents and resources.

**Homework — *Students will...***

- **Outline** Chapter 5 (up to and including BJP 5.3)

## Materials & Prep

- **Projector and computer**
- **Whiteboard and markers**
- **Classroom copies** of WS 3.16 (RPS, Pig), DeMorgan's Law, Poster 3.16.1, Poster 3.16.2
- **Link** to rock-paper-scissor game (http://tinyurl.com/bubyvtu)
- **Poster** 3.16.1
- **Poster** 3.16.2

Truth tables are an important tool, especially for some AP test questions. If you are not familiar with truth tables, watch this 5 minute tutorial online (http://tinyurl.com/mw8ohof). We recommend instructors draw out the &&, ||, and ! truth tables, and maybe do an intermediate example of a two-operator expression, before getting into the De Morgan's law example later in class.

## Pacing Guide: Day 1

| Section | Total Time |
|---|---|
| Attendance & student play | 5min |
| Introduction to Boolean Logic*Add 5–10 minutes if using truth tables* | 10min |
| Student programming activity | 40min |

## Pacing Guide: Day 2

| Section | Total Time |
|---|---|
| Attendance & outline collection | 5min |
| Whole-group troubleshooting and discussion | 10min |
| Student programming activity | 40min |

## Procedure

In place of bell-work, invite students to warm up for class by visiting the online Rock Paper Scissor game at the link above. After a quick review of Boolean logic and variables, students will be asked to build their own Rock Paper Scissor game. This programming project should take between 2 and 3 55-minute class periods to complete

**Attendance & Student Play [5 minutes]**

**Introduction to Boolean Logic [10 minutes]**

1. Students should have already reviewed this material as part of last nights' homework assignment. Before moving on to purely mathematical examples, start with a real-life example of how we apply logic. Be sure to change P and Q to statements that are relevant to your students.

   - P: It is a holiday. Q: My family is having dinner together.

     !(p || q)  It is not the case that (it is a holiday OR my family is having dinner together)

     !p && !q  It is not a holiday AND my family is not having dinner together.

   - Review `&&`, `||`, and `!`, including non-examples:

```
 Operator         Expression        Result

   AND       (4 == 4) && (2 > 1)    Evaluates to: True
   OR        (1 < 2)   (2 < 1)     Evaluates to: True
   NOT       !(2 < 1)               Evaluates to: True
```

```
if (q == 1 || 2 || 4) {     // ERROR: You must use full Boolean expressions.
   statement;
   statement;
}
if (q == 1 || q == 2 || q == 4) {      // Correct
   statement;
   statement;
}
```

2. As a special note on negating Boolean expressions, review De Morgan's law (poster 3.16.1). Have students write De Morgan's law on their Tricky Code Cheat Sheet.

   - If you feel confident working with truth tables, work through the following illustration of De Morgan's laws. On the board or projector, only write table headers as you go (putting them all up at once may lead to panic/distraction for some students).

```
 p          q          p   q    !(p   q)      !p          !q      !p && !q

 F          F
 F          T
 T          F
 T          T
```

     Have students help you fill out every possible combination of Boolean values for p and q.

   - Ask students to evaluate the logical expression for each value of p and q.

```
 p          q          p   q    !(p   q)      !p          !q      !p && !q

 F          F          F
 F          T          T
 T          F          T
 T          T          T
```

   - Now have students negate all of the values from the previous column.

```
 p          q          p   q    !(p   q)      !p          !q      !p && !q
```

| | |
|---|---|
| F | T |
| T | F |
| T | F |
| T | F |

- Ask students to complete the values for !p and !q, referring to the values from the first column.

| p | q | p | q | !(p && q) | !p | !q | !p && !q |
|---|---|---|---|---|---|---|---|
| F | F | | | | T | T | |
| F | T | | | | T | F | |
| T | F | | | | F | T | |
| T | T | | | | F | F | |

- Now have students apply the && operator to !p and !q.

| p | q | p | q | !(p && q) | !p | !q | !p && !q |
|---|---|---|---|---|---|---|---|
| | | | | | T | T | T |
| | | | | | T | F | F |
| | | | | | F | T | F |
| | | | | | F | F | F |

- Point out to your students that these two columns are the same. Whenever two columns of a truth table are the same, we say that the expressions (column headings) are equivalent, or interchangeable.

| p | q | p | q | !(p && q) | !p | !q | !p && !q |
|---|---|---|---|---|---|---|---|
| | | | | T | | | T |
| | | | | F | | | F |
| | | | | F | | | F |
| | | | | F | | | F |

In the illustration above, we showed that `!(p && q)` is equivalent to `!p && !q`. Invite students to show the equivalence of `!(p && q)` and `!p || !q`.

3. Review operator precedence on your classroom poster 3.16.2 (or projected overhead, if you're having a student make the poster for you during class).
4. Check for student understanding by having students complete self-check questions Self-Check 5.27: assertions1 and Self-Check 5.29: assertions3.

**Student Programming Activity [40 minutes]**

On the projector, board or as a handout (WS 3.16), give students the following programming prompt. A link to the NY Times article about rock paper scissors is included in the Materials section of this lesson plan.

---

### PROGRAMMING ACTIVITY
### Applied Design and Creating a Program for an End-User
Your task is to create a computer game for an end-user (a friend, classmate, teacher, relative, etc). You can create a program similar to Rock-Paper-Scissors or similar to some type of dice or simple card game that you maybe have encountered before (you can use similar rules, similar game play, etc). "War" is an interesting card game that you could alter for this purpose and "Pig" is an interesting dice game. You can read the rules for each of these online.
You will first learn about and understand the interests/likes/passions/hobbies of your end-user by interviewing them and asking them questions. You will then, in collaboration with the end-user, decide what type of game you are going to create and what type of changes you might make to an original Rock-Paper-Scissors, card or dice game.

Some examples that you and your end-user might want to change include: - Different options for play. Rather than playing rock, paper or scissors, the end-user might have suggestions for different plays or attacks, or they might want to add new attacks. - Different options for scoring. The end-user might want to add scoring features such as players being able to risk double the points on certain plays which would mean they would lose double the points, or win double the point. - Change the format of gameplay by having players win "two-in-a-row" or by having players win "best of 7" in order to be declared the victor. - Add the ability to play the computer or another player. - Add more dice to an existing game and alter the scoring accordingly. - Create new scoring for a dice game or a randomized number game.
You and your end-user may also choose to create an entirely new game. Just make sure that if you do this, you ensure that the logic of the game is sound. This may require careful design and testing.
**To get full credit on this assignment, you must include a structure diagram and/or pseudocode explaining your strategy.**
*Before you begin, take a moment to decide how your computer will pick rock, paper, or scissor. Should the computer pick randomly? Should it pick the same item always? Should it repeat the same item for a time, then switch strategies? Read through the New York Times article on Rock Paper Scissor, and any other online sources you choose to help you draft a plan for your program.*

- Allow students to work in pairs, and encourage pairs to test out each others' programs, look at each others code (to check for errors), etc. If students appear to be working too closely, remind them that each team is responsible for writing their own code.
- Start grading student note-books in small batches (so students are not without their notebooks for too long!)

---

**Final Project** The small game you design above will implement interesting game play ideas. Boolean logic is used to determine winners and losers and to alter the score. Keep these small games in mind as you continue to think about the final project in the course. You will need to have an in-depth understanding of boolean logic as you create your final game.

---

## Accommodation and Differentiation

Invite your artistic students to create posters 3.16.1 and 3.16.2 for your classroom. If needed, work through the 2 practice questions as a whole class.

For your more advanced students, you might encourage them to create more complex algorithms, or more advanced interaction with the user. If they are interested in AI and machine learning, invite them to research the topic and experiment with different techniques on their Rock Paper Scissor program.

## Misconceptions

- Students often have the misconception that logical OR (`||`) is exclusive OR. In a student's daily life, they can have either brownie or the cupcake. This implies they can have one or the other, not both. However, in boolean logic, "a or b" is also true if *both* are true.
- Students ask why the symbol `&&` for AND an `||` for OR? And why double `&&` vs single `&`.

  Java was derived from the C programming language. The designers Kernighan and Richie made design decisions based on prior programming languages of their time. The history of C can be found here: https://www.bell-labs.com/usr/dmr/www/chist.html. Java's use of `&` and `|` is historically based on C.
- A common syntax error by beginner and experienced programmers alike is typing a single ampersand `&` or vertical bar `|`. Single `&` and `|` are *bitwise* operators, and are not the same as double `&&` and `||` which are logical AND and OR, respectively.
- Logical operators AND and OR do not follow English language syntax. For example, testing whether a dice roll is either 7 or 11 could be translated incorrectly to:

```
if (roll == 7 || 11)           // INCORRECT: Logical operator misconception
if (roll == 7 || roll == 11)   // CORRECT
```

## Video

- BJP 5-4, *The Boolean Type* http://media.pearsoncmg.com/aw/aw_reges_bjp_2/videoPlayer.php?id=c5-4

## Grading Scheme/Rubric

| | |
|---|---|
| Program demonstrates complex use of boolean logic | 2 points |
| Game play demonstrates effective design and planning skills | 2 points |
| User-interface is clear, consise and easy to use. | 2 points |
| Rules, game play and scoring demonstrate logic | 3 points |
| **Total** | **9 points** |
| **Applied Design Steps and Log** | |
| Understanding Context components are complete and thorough | 1 point |
| Defining components are complete and thorough | 1 point |
| Ideating/Prototyping are components complete and thorough | 1 point |
| Testing components are complete and thorough | 1 point |
| Sharing components are complete and thorough | 1 point |
| **Total** | **5 points** |
| **Total** | **14 points** |

## Forum discussion

Lesson 3.16 Boolean Logic (TEALS Discourse account required)