Goals of this tutorial

You should be able to...

• understand and perform Boolean algebra.
• understand and use conditional expressions.
• write sufficient test cases for functions containing conditional expressions.
Review: Boolean-valued functions

Boolean-valued functions produce Boolean values: true and false. These functions are also called predicates.

Standard Racket uses #t and #f, or #true and #false; these will sometimes show up in basic tests and correctness tests.

Racket provides many built-in Boolean functions (for example, to do numerical comparisons: (≥ x y), (≡ x y))
Review: Boolean-valued functions

Note that comparison functions are often specific to certain data types (for example, \((= x y)\) vs. \((\text{symbol}=? x y)\))

The naming convention for most predicates is to append a question mark to the name (for example, `even?`, `symbol?`)
Review: Boolean Operators

and, and, or are special forms in Racket.

and, and, or may have any number of arguments.

Their arguments are evaluated from left to right.

and:

- If an argument evaluates to false, the entire expression evaluates to false.

- Otherwise, the next argument is evaluated.

- If there are no arguments remaining, the expression evaluates to true.
Review: Boolean Operators

or:

- If an argument evaluates to true, the entire expression evaluates to true.
- Otherwise, the next argument is evaluated.
- If there are no arguments remaining, the expression evaluates to false.

not:

- not must have exactly one argument.
- If the argument evaluates to true, the entire expression evaluates to false.
- If the argument evaluates to false, the entire expression evaluates to true.
Clicker Question - Boolean Expression

Which of the following expressions evaluates to true?

A  \((\equiv \text{ 'blue 'blue})\)

B  \((\neg (\neg \text{ false}))\)

C  \((\text{check-expect} (\,3\ 7) 10)\)

D  \((\text{or} (\equiv 25 24) (\lt 27 28))\)

E  \((\text{or false (not true))}\)
Review: Conditional Expressions

The general form of a conditional expression is

\[
\text{(cond }
\begin{array}{l}
\text{[question1 answer1]} \\
\text{[question2 answer2]} \\
\ldots \\
\text{[questionk answerk]})
\end{array}
\]

where \text{questionk} could be \text{else}.
• Each of the questions must evaluate to a **boolean** value.

• The questions are evaluated from **top to bottom**.

• If a question evaluates to **true**, no more questions are evaluated and the cond expression is reduced to just the answer for that question.

• If none of the questions evaluate to **true**, then the result is the answer in the **else** clause.

• If there are no questions that evaluate to **true** and there is no **else** clause, then Racket will report an error.
Clicker Question - Cond Expression

What does the following cond expression evaluate to?

```
(cond
  [(< 16 7) 'blue]
  [(and true (not false)) 'red]
  [(= (/ 3 0) 2) 'yellow]
  [else 'green])
```

A  'blue  
B  'red  
C  'yellow  
D  'green  
E  Nothing. There is an error.
Group Problem - Converting cond to booleans

Rewrite the function \( f \) without cond. You may use and, or and not.

\[
\text{(define } (f \ x) \\
\quad (\text{cond} \\
\quad \quad [(\text{not } (\text{p1? } x)) (\text{p2? } x)] \\
\quad \quad [\text{else } (\text{p1? } x)])))
\]
Rewrite the function $g$ using only a single cond in its body. You may use and, or and not.

(define (g x)
  (cond
    [(p1? x) (cond
      [(p2? x) (f1 x)]
      [(p3? x) (f2 x)]
      [else (f3 x)])]
    [else (f4 x)]))
Note: Tests for conditional expressions

- Test for each clause in the cond expression
- If the function specifications allow for this:
  - Test all boundary points
  - Write at least one test for each interval (not including the boundary)
- DrRacket highlights unused code
  - Having no code highlighted does not mean that your code is fully tested
  - However, highlighted code means your testing is incomplete
(define (foo x)
  (cond
    [(< x 0) (exp x)]
    [(<= x 100) (sub1 x)]
    [(< x 1000) x]
    [else (add1 x)]))

Minimally, how many tests would be required for this function?

A  4  
B  5  
C  6  
D  7  
E  8
Group Problem - pokemon-battle

You’ve finally made it to the CS135 Gym and it is time to battle. Here are the rules:

- only 3 types are allowed: Fire, Water and Rock
- Water beats both Fire and Rock
- Rock beats Fire
- Any type can beat itself

Write a function `pokemon-battle` that consumes 2 Pokemon types and produces the winning type according to the rules above. Use the symbols ‘fire’, ‘water’ and ‘rock’ to represent the types. Include a design recipe.
Group Problem - prime? with boolean operators

Write a function `prime?` that consumes an integer between 2 and 120 (inclusive) and produces true if that number is prime. For this question, you may not use `cond` and you may not explicitly check for each prime number in that range. Include a design recipe.

Hint: note that $120 < 11 \cdot 11$
(Optional) Group Problem - prime? with cond

Rewrite an implementation of prime? that only uses cond statements (no boolean operators allowed). You don’t need to include a design recipe.