6.S189 Lecture 2 Notes

Conditional Operators

Yesterday, we talked about Boolean types - True or False. (Notice there's no quotes - these aren't strings.) Now we'll introduce new operators that can be used to compare two values, which we'll call a and b. You can use these operators in expressions of the form *a op b*, where op is the operator. Expressions of this form evaluate to True or False.

1. > (Greater than)
2. >= (Greater than or equal to)
3. < (Less than)
4. <= (Less than or equal to)
5. == (Equal to)
6. ! = (Not equal to)

Other operators include *and*, *or*, and *not*. These operators are commonly used when a and b are the Boolean values True or False. These statements also evaluate to True or False. The not operator is unique in that it is used with only one value, e.g., not b.

- An and statement evaluates to True when both a and b are True.
- An or statement evaluates to True if either a or b is True.
- A not statement evaluates to the opposite value of the variable it modifies.

if, else, and elif Statements

Sometimes we only want to execute chunks of code if certain conditions are met. For example, in a program that models a game of hangman, we only want to display a victory message if the user has successfully guessed the secret word. If statements are useful mechanisms to control when code is executed and when it is ignored.

A boolean expression controls when the code is executed. If the boolean expression evaluates to True, then the code is run. Otherwise, the block of code is skipped.

else statements must be preceeded by an if statement, and are executed if the boolean statement evaluated to False.

elif statements are similar to else statements except that they check another boolean expression to determine whether to run the code they contain.