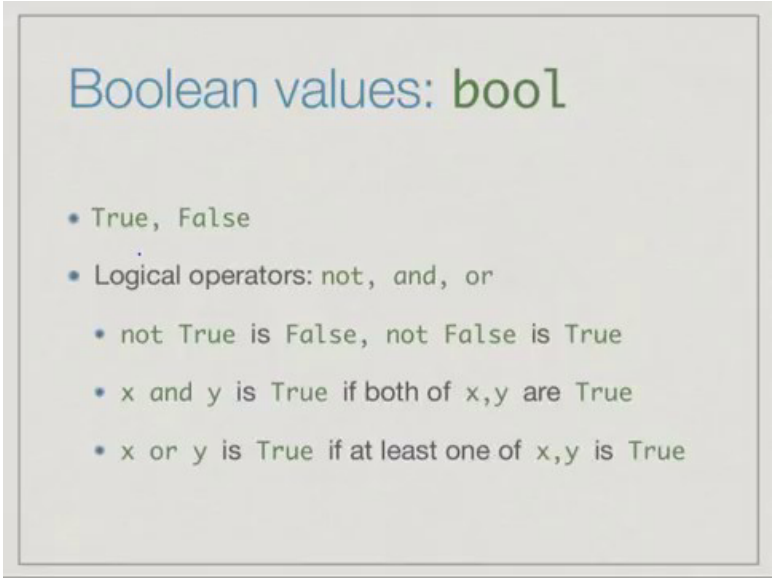
(Refer Slide Time: 15:05)



Another important class of values that we use implicitly in all our functions are Boolean values which designate truth or falseness. So, there are two constants or two basic values of this type which in python are called true with the capital "T" and false with the capital "F". So, true is the value which tells something is true. So, when we remember we wrote conditions like if something happens if x is equal to y do something, x mod 7 is equal to something, to something in our gcd function. So, the output of such an expression where we compare something to another expression compare an expression on the left to an expression on the right is to determine whether this comparisons succeeds or fails when it succeeds it is true and when it fails it is false.

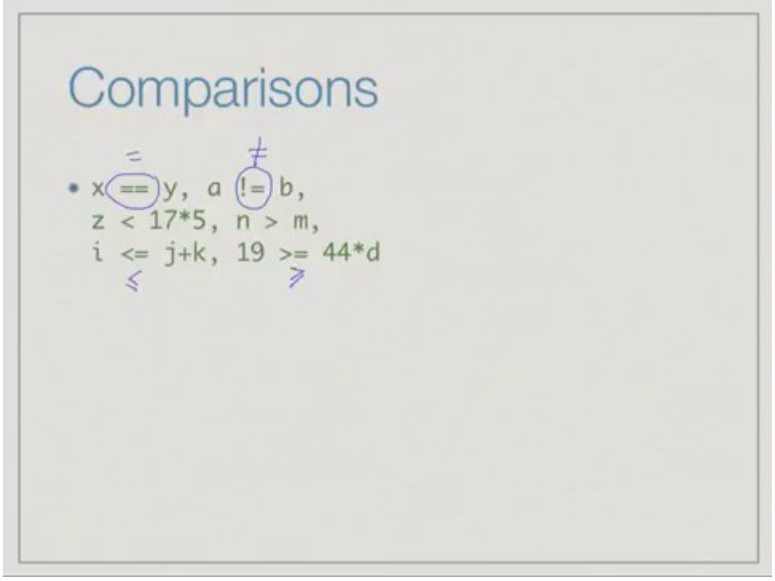
These are implicitly used to control the execution of our program. So, we need to have a way of recording these values and manipulate it. The basic values are true or false and typically there are three functions which operate on these values. So, not negates the value. So, true is the opposite of false. So, not applied to true will give us false not applied to false will give us true and follows the usually English meaning of and so, when we say that something is true and something else is true we mean that exactly both of them are true. So, x and y two values of Boolean type will be the expression x and y will be true provided at the moment x has a value true and y also has a value true. If either of them is not true then the output x and y is false.

"Or" again has an English meaning, but the meaning in computer science and logic is

slightly different from what we mean. So, normally when we say or we mean 1 or the other. So, you might say either i will wake up in time or i will miss my bus. So, what you will mean is that one of these two will happen it is unlikely that you mean that you will wake up in time and you will miss your bus.

It is when we use or in English we usually mean either the first thing will happen or the second thing will happen, but not both, but in computer science and logic or is a so, called inclusive or not exclusive, its not exclusively one will happen or the other, but inclusive both may happen. So, x or y is true if at least one is true. So, one of them must be true, but it also possible when both are them true.

(Refer Slide Time: 17:28)

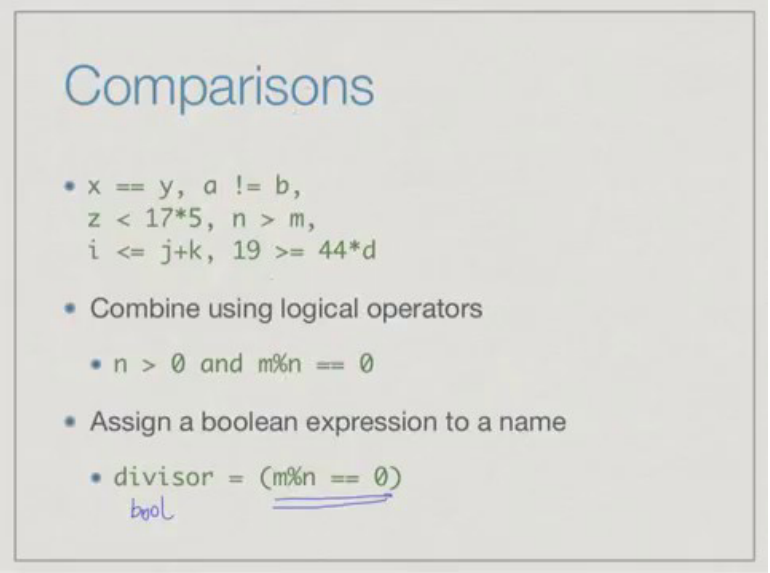


The most frequent way in which we generate Boolean values is through comparisons we have already seen the two of these. So, we have seen equal to - equal to. This is the actual equality of mathematics not the single equal to which is the assignment. So, if x equal to equal to y checks, whether the value of x is actually the same as the value y and if so, it returns the value true otherwise, it returns false.

And the corresponding inequality operator is exclamation mark followed. So, this is not equal to exclamation is equal to is a symbol for not equal to and this is the usual mathematical. And then of course you have for values which can be compared as smaller or larger you have less than, greater than this is less than equal to and this is greater than equal to. So, we have these 6 logic logical comparison operators' arithmetic comparison

operators which yield a logical value true or false.

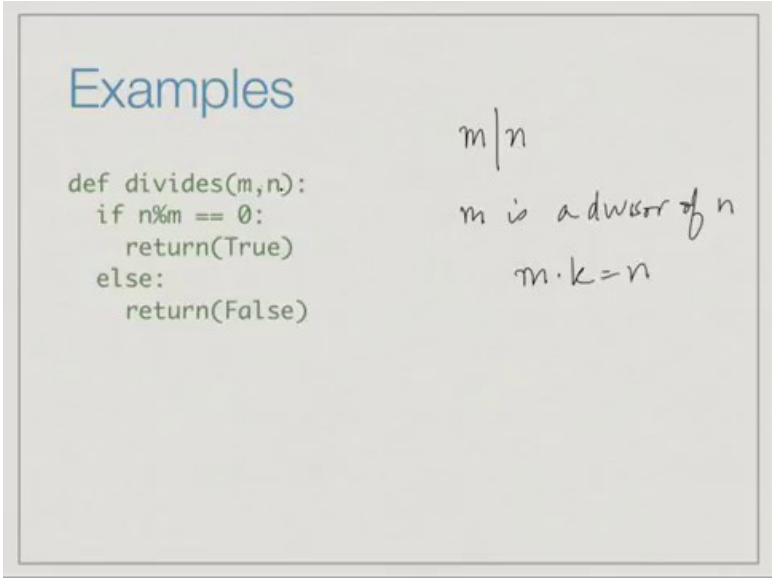
(Refer Slide Time: 18:24)



And the usual thing we will do is combine these. So, we might want to say that check if the reminder when divided by n is 0 provided n is 0 not 0. So, if we say n is greater than 0 and this it will require n to be number bigger than 0 and the reminder n divided by n to be equal to 0. So, this says n is a multiple of n and n is not 0. And we can take an expression of this file kind of comparison, which yields as we said a Boolean value, and take this Boolean value and assign it to a name.

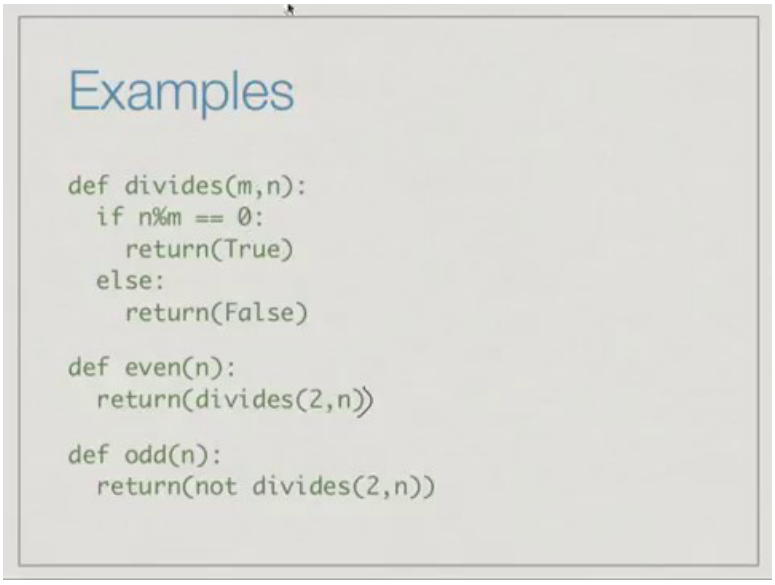
So, we can say that n is a divisor of m if the reminder of m divided by n is equal to 0. And we can say that the fact that it is a divisor its true provided this happens. So, divisor is now of type bool right and it has a value true or false depending or not whether or not n divides m evenly.

(Refer Slide Time: 19:22)



Let us look at an example of how we would use Boolean values. So, let us get back to the divides example. In mathematics we write m divides n to say that m is a divisor of n. So, this means that m times k is equal to n for some k. So, m divides n if the reminder of n divided by m is 0. If so you return true else you return false right. This is a very simple function it takes two arguments and checks if the first argument divides the second argument.

(Refer Slide Time: 20:00)

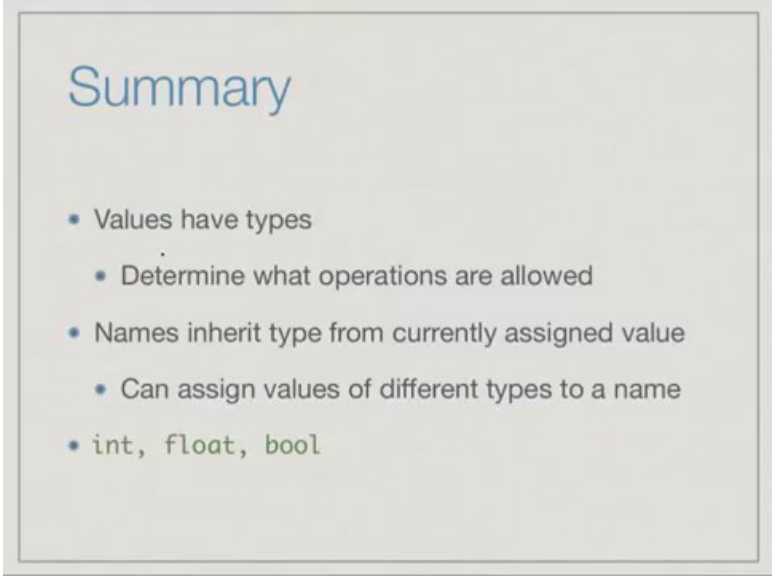


Now what we can do is define an another function called even whose value is derived

from here. So, we check whether two is a divisor of this number. So, we check whether 2 divides n; if 2 divides n, then n is even, we return true; if 2 does not divide n, n is odd, we return false. So, similarly we could say define odd n else the negation of the previous case. So, if 2 divides n then n is not odd.

You take the answer about whether 2 divides n or not, and reverse it to get the answer odd. So, if 2 divides n, you negate it and say odd is false; if 2 does not divide n, you get false back and you negate and say odd is true. So, we just wanted to emphasise that Boolean values can be computed, assigned, passed around just like numerical values are.

(Refer Slide Time: 20:50)



To summarise what we have seen is that the basic type of statement is to assign a name to a value values have type and these determine what operations are allowed. So, we can use for instance arithmetic operations on numeric types, we can use logical operations like and, or, and not on Boolean types, but the important difference between python and traditional languages where we declare names in advance is that python does not fix types for names. So, we cannot say that 'i' has the type int forever; 'i' will have a type depending on what it is assigned. A name inherits the type from its currently assigned value and its type can change as a program evolves depending on what values have been assigned.

What we have seen in this particular lecture are 3 basic type int, float and bool. As we go along this week, we will see more types with interesting structures and interesting

operations defined on them.