While loops

Previously, we introduced iterative statements in Python and focused on for loops.
An iterative statement is code that repeatedly executes a set of instructions.
In this video, we'll explore another type of iterative statement: the while loop.
When we used for loops, the code repeatedly executed based on a specified sequence.
While loops still repeatedly execute, but this repetition is based on a condition.
As long as the condition is true, the loop continues to execute.
But when it becomes false, the while loop stops.
This while loop, for example, sets a condition where the variable time must be less than or equal to 10.
This means it will keep running until the variable time is greater than 10.
Similar to the for loop, a while loop has a header.
It consists of the keyword while, the condition, and a colon.
The while loop starts with the keyword while.
The keyword while signals the beginning of a while loop and is followed by the condition that evaluates to a Boolean value of either True or False.
The condition contains the loop variable.
This variable is used to control the number of loop iterations.
However, there is an important distinction in the variables used in for and while loops.
With while loops, the variable isn't created within the loop statement itself.
Before writing the while loop, you need to assign the variable.
Then you'll be able to reference it in the loop.
When the condition containing the loop variable evaluates to True, the loop iterates.
If it does not, then the loop stops.
This condition will evaluate to True while the variable time is less than or equal to 10.
Finally, the loop header ends with a colon.
Just like a for loop, a while loop has an indented body that consists of the actions to take while the loop iterates.
The intention of this code is to print the value of a variable that represents the time and increase its value by two, until it becomes greater than 10.
This means the first action in this while loop is to simply print the current value of the time variable.
Since while loops do not include a sequence to iterate through, we have to explicitly define how the loop variable changes in the body of the while loop.
For example, in this while loop, we increase the loop variable time by two every iteration.
This is because we only want to print the time every two minutes, so this while loop prints out all even numbers less than or equal to 10.
Now that we know the basics of while loops, let's explore a practical example.
Imagine we have a limitation on how many devices a user can connect to.
We can use a while loop to print a message when the user has reached their maximum number of connected devices.
Let's create a while loop for this.

Before we start our while loop, we need to assign values to two variables.
First, we'll set the maximum value of connected devices to five.
Then, we'll set our loop variable.
We'll use i for this and set it to a value of one.
Unlike with for loops, with while loops, we set this variable outside of the loop.
Next, we'll create the header of our while loop.
In this case, the condition is that the first variable is less than the second variable.
Those variables are the loop variable "i" and max_devices.
Since we know the value of max_devices is five, we can understand that this loop will run as long as the current value of "i" is less than five.
Then we indicate what we want our while loop to do.
Because this loop runs as long as the user can still connect to devices, we'll first have it print a "user can still connect to additional devices" message.
After this, with each iteration, we'll increment i by one.
When the loop repeats, it will use the new value of the i variable.
Python will exit the loop when i is no longer less than five.
Let's also print a message when this happens.
We stop indenting because this next action occurs outside of the loop.
Then we'll print "user has reached maximum number of connected devices." We're ready to run this.
Because of the loop, the first message prints a total of four times.
The loop stops when the value of i increments to five.
At this point, it exits the loop and prints the second message.
When you combine this new understanding of for and while loops with what you already know about conditional statements and variables, you have a lot of options in Python.
Great work!

Revision #1
Created 2023-12-11 07:24:08 UTC by naruzkurai
Updated 2023-12-19 03:38:35 UTC by naruzkurai