Using context managers

1. Using context managers

00:00 - 00:07

In this lesson, I'll introduce the concept of context managers and show you how to use these special kinds of functions.

2. What is a context manager?

00:07 - 00:21

A context manager is a type of function that sets up a context for your code to run in, runs your code, and then removes the context. That's not a very helpful definition though, so let me explain with an analogy.

3. A catered party

00:21 - 00:28

Imagine that you are throwing a fancy party, and have hired some caterers to provide refreshments for your guests.

4. A catered party

00:28 - 00:33

Before the party starts, the caterers set up tables with food and drinks.

5. A catered party

00:33 - 00:38

Then you and your friends dance, eat, and have a good time.

6. A catered party

00:38 - 00:40

When the party is done,

7. A catered party

00:40 - 00:42

the caterers clean up the food and remove the tables.

8. Catered party as context

00:42 - 01:07

In this analogy, the caterers are like a context manager. First, they set up a context for your party, which was a room full of food and drinks. Then they let you and your friends do whatever you want. This is like you being able to run your code inside the context manager's context.

Finally, when the party is over, the caterers clean up and remove the context that the party happened in.

9. A real-world example

01:07 - 01:49

You may have used code like this before. The "open()" function is a context manager. When you write "with open()", it opens a file that you can read from or write to. Then, it gives control back to your code so that you can perform operations on the file object. In this example, we read the text of the file, store the contents of the file in the variable "text", and store the length of the contents in the variable "length". When the code inside the indented block is done, the "open()" function makes sure that the file is closed before continuing on in the script. The print statement is outside of the context, so by the time it runs the file is closed.

10. Using a context manager

01:49 - 01:58

Any time you use a context manager, it will look like this. The keyword "with" lets Python know that you are trying to enter a context.

11. Using a context manager

01:58 - 02:09

Then you call a function. You can call any function that is built to work as a context manager. In the next lesson, I'll show you how to write your own functions that work this way.

12. Using a context manager

02:09 - 02:13

A context manager can take arguments like any normal function.

13. Using a context manager

02:13 - 02:19

You end the "with" statement with a colon as if you were writing a for loop or an if statement.

14. Using a context manager

02:19 - 02:39

Statements in Python that have an indented block after them, like for loops, if/else statements, function definitions, etc. are called "compound statements". The "with" statement is another type of compound statement. Any code that you want to run inside the context that the context manager created needs to be indented.

15. Using a context manager

02:39 - 02:49

When the indented block is done, the context manager gets a chance to clean up anything that it needs to, like when the "open()" context manager closed the file.

16. Using a context manager

02:49 - 03:17

Some context managers want to return a value that you can use inside the context. By adding "as" and a variable name at the end of the "with" statement, you can assign the returned value to the variable name. We used this ability when calling the "open()" context manager, which returns a file that we can read from or write to. By adding "as my_file" to the "with" statement, we assigned the file to the variable "my_file".

17. Let's practice!

03:17 - 03:27

You'll learn how to write your own context managers in the next lesson. For now, you can practice using them in order to understand how they work.