Apply debugging strategies

Let's say our co-workers need some help getting their code to work, and we've offered to debug their code to make sure it runs smoothly.
First, we need to know about the purpose of the code.
In this case, the purpose of the code is to parse a single line from a log file and return it.
The log file we're using tracks potential issues with software applications.
Each line in the log contains the HTTP response status codes, the date, the time, and the application name.
When writing this code, our co-workers consider whether all these status codes needed to be parsed.
Since 200 signals a successful event, they concluded that lines with this status code shouldn't be parsed.
Instead, Python should return a message indicating that parsing wasn't needed.
To start the debugging process, let's first run the code to identify what errors appear.
Our first error is a syntax error.
The error message also tells us the syntax error occurs in a line that defines a function.
So let's just scroll to that part of the code.
As you might recall, these function headers should end with a colon.
Let's go ahead and add that to the code.
Now, the syntax error should go away.
Let's run the code again.
Now our syntax error is gone, which is good news, but we have another error, a "name error."
"Name error" is actually a type of exception, meaning we've written valid syntax, but Python can't process the statement.
According to the error, the interpreter doesn't understand the variable application_name at the point where it's been added to the parsed_line list.
Let's examine that section of code.
This error means we haven't assigned a variable name properly.
So now let's go back to where it was first assigned and determine what happened.
We find that this variable is misspelled.
There should be two p's in application_name, not one.
Let's correct the spelling.
Now that we've fixed it, it should work.
So let's run the code.
Great!
We fixed an error and an exception.
And we no longer have any error messages.
But this doesn't mean our debugging work is done.
Let's make sure the logic of the program works as intended by examining the output.
Our output is a parsed line.
In most cases, this would be what we wanted.
But as you might recall, if the status code is 200, our code shouldn't parse the line.

Instead, it should print a message that no parsing is needed.
And when we called it with a status code of 200, there was a logic error because this message wasn't displayed.
So let's go back to the conditional we used to handle the status code of 200 and investigate.
To find the source of the issue, let's add print statements.
In our print statements, we'll include the line number and the description of the location.
We'll add one print statement before the line of code containing "return parsed_list".
We'll add another above the if statement that checks for the 200 status code to determine if it reaches the if statement.
We'll add one more print statement inside the if statement to determine whether the program even enters it.
Now, let's run the code and review what gets printed.
Only the first print statement printed something.
The other two print statements after these didn't print.
This means the program didn't even enter the if statement.
The problem occurred somewhere before the line that returns the parsed_line variable.
Let's investigate.
When Python encounters the first return statement which sends back the parsed list, it exits the function.
In other words, it returns the list before it even checks for a status code value of 200.
To fix this, we must move the if statement, checking for the status code somewhere before "return parsed line".
Let's first delete our print statements.
This makes the program more efficient because it doesn't run any unneeded lines of code.
Now, let's move the if statement.
We'll place it right after the line of code that involves parsing the status code from the line.
Let's run our code and confirm that this fixed our issue.
Yes!
It printed "Successful event - no parsing needed." Great work!
We've fixed this logic error.
I enjoyed debugging this code with you.
I hope this video has strengthened your understanding of some helpful debugging strategies and gave you an idea of some errors you might encounter.

Revision #1
Created 2 January 2024 22:21:28 by naruzkurai
Updated 2 January 2024 22:21:42 by naruzkurai