The parser repeats the offending line and displays a little ‘arrow’ pointing at the earliest point in the line where the error was detected. The error is caused by (or at least detected at) the token preceding the arrow: in the example, the error is detected at the function [print()](https://docs.python.org/3/library/functions.html#print), since a colon (':') is missing before it. File name and line number are printed so you know where to look in case the input came from a script.

Even if a statement or expression is syntactically correct, it may cause an error when an attempt is made to execute it. Errors detected during execution are called exceptions and are not unconditionally fatal: you will soon learn how to handle them in Python programs.

Some objects define standard clean-up actions to be undertaken when the object is no longer needed, regardless of whether or not the operation using the object succeeded or failed. Look at the following example, which tries to open a file and print its contents to the screen.

The problem with this code is that it leaves the file open for an indeterminate amount of time after this part of the code has finished executing. This is not an issue in simple scripts, but can be a problem for larger applications. The [with](https://docs.python.org/3/reference/compound_stmts.html#with) statement allows objects like files to be used in a way that ensures they are always cleaned up promptly and correctly.

After the statement is executed, the file f is always closed, even if a problem was encountered while processing the lines. Objects which, like files, provide predefined clean-up actions will indicate this in their documentation.