In this tutorial, we will test a program using the debugger features in IDLE. This tutorial is an alternative to the previous Python debugging tutorial, and will be using a simpler example for testing.

We will be testing a sample Python script, login.py, that will be distributed with this video. Or you can simply copy the program yourself as we go along in the tutorial. First, let’s open up the script, login.py in the IDLE editor.

login.py is a simple program that accepts an inputted username and password and determines whether or not they are valid. The username and password must be equal to the definition in the code, otherwise the program will display an error message. In this case, if the user enters the credentials, ‘ceram’ and ‘pacman’ as the username and password respectively, then the program should tell them that they have successfully logged in. We can test this ourselves by going to the taskbar and clicking [Run], and then [Run Module], or by simply hitting the F5 key.

We can use the Debugger feature in IDLE to pause our program midway through execution. This can be useful whenever we are trying to figure out which lines are being run during execution. Debugging is essential when testing for certain conditions when a program is being run. In the code editor window, right click on line 7, the print statement that says “Hello Michael”, and select [Set Breakpoint]. Breakpoints are used to establish a point at which we would like our program to pause during its execution. We can set a breakpoint on any line, and can even set multiple breakpoints if we so choose. By setting out breakpoint here, we are effectively saying that we want our program to run until it reaches this print statement, at which point the program will pause and wait for further commands. Provided our program does execute this line, the execution should pause during its run.

In the IDLE main window, click on [Debug] from the taskbar, and select [Debugger]. This will bring up the Debug Control window where you can view your program running step by step. Rerun the program and keep both the Debug Control window and the main window active.

Notice how the Debug Control window now displays a line pertaining to the first line in the program. The program has been paused, and as long as we are using the Debugger, the program will always pause on the very first line of our script as shown by the Debugger. At this point, we can hit the [Go] button, which will run the rest of the program until our breakpoint is reached.

Our program has reached our first input statement asking for the username. Enter ‘ceram’ as the username and click [Enter].

Follow the same steps for the next input, setting the password as ‘pacman.’

We have successfully inputted the correct credentials, although the script has been paused at the breakpoint we set earlier. Notice in the Debugger window that line 9, the line that we set our breakpoint to, is now displayed. This is the Debugger notifying you that it has reached this line in the program and is now waiting for further instructions. From here, you have the option of clicking the [Step] button, which will simply move down to the next line in the sequence. Alternatively, you can click [Go] again to resume running until the next breakpoint. Since there is no breakpoint after line 9, the program will run until the end. The Debugger comes with another useful feature known as [Source]. What this does is it allows you to pinpoint the exact line that the Debugger has been paused on in the code. By clicking the [Source] button, the line in the code will become active, notifying you the position of our current line. In this example, clicking [Source] will bring our cursor to line 9 in our script file. It sounds simple enough, but when dealing with a file that could contain hundreds of lines of code, this feature can certainly be useful.

Notice that our program still hasn’t finished, as shown by our output in the main IDLE window. Click the [Step] button to move down to the next line, line 10. Our print statement from line 9 should now appear in the window. You can click through [Step] again or simply hit [Go] to finish execution.

Leave the breakpoint on line 9 and rerun the program with the Debugger open. Click [Go] and then, this time, enter false credentials when prompted.

Notice how the Debugger did not pause the program during its execution. This is because our breakpoint was never reached, as putting in false credentials went down the other path of our condition, the one that prints an error statement. Since line 9 is never reached at this point, the program runs like normal. You can guarantee that the Debugger will pause the program if you were to put a breakpoint on line 7, our error statement.

To close the debugger, click the close button in the top right corner of the window.

This concludes the Introductory Python Debugging Tutorial.