How to Start the Debugger

You can start the debugger three different ways. The first is to just import it and insert **pdb.set_trace()** into your code to start the debugger. You can import the debugger in IDLE and have it run your module. Or you can call the debugger on the command line. We'll focus on the last two methods in this section. We will start with using it in the interpreter (IDLE). Open up a terminal (command line window) and navigate to where you save the above code example. Then start Python. Now do the following:

```
import pdb

pdb.run('debug_test.main()')
#2
#4
#6
#8
#10
#12
#14
#16
#18
#> <string>(1)<module>()
#(Pdb) *** SyntaxError: invalid syntax (<stdin>, line 1)
#(Pdb)
```

Here we import our module and pdb. Then we execute pdb's **run** method and tell it to call our module's **main** method. This brings up the debugger's prompt. Here we typed **continue** to tell it to go ahead and run the script. You can also type the letter **c** as a shortcut for continue. When you **continue**, the debugger will continue execution until it reaches a breakpoint or the script ends.

The other way to start the debugger is to execute the following command via your terminal session:

```
python -m pdb debug_test.py
```

If you run it this way, you will see a slightly different result:

```
-> def doubler(a):
(Pdb) c
2
4
6
8
10
12
14
16
18
The program finished and will be restarted
```

You will note that in this example we used **c** instead of **continue**. You will also note that the debugger restarts at the end. This preserves the debugger's state (such as breakpoints) and can be more useful than having the debugger stop. Sometimes you'll need to go through the code several times to understand what's wrong with it.

Let's dig a little deeper and learn how to step through the code.