Stepping Through the Code

If you want to step through your code one line at a time, then you can use the **step** (or simply "s") command. Here's a session for your viewing pleasure:

```
C:\Users\mike>cd c:\py101

c:\py101>python -m pdb debug_test.py
> c:\py101\debug_test.py(4)<module>()
-> def doubler(a):
(Pdb) step
> c:\py101\debug_test.py(11)<module>()
-> def main():
(Pdb) s
> c:\py101\debug_test.py(16)<module>()
-> if __name__ == "__main__":
(Pdb) s
> c:\py101\debug_test.py(17)<module>()
-> main()
(Pdb) s
--Call--
> c:\py101\debug_test.py(11)main()
-> def main():
(Pdb) next
> c:\py101\debug_test.py(13)main()
-> for i in range(1,10):
(Pdb) s
> c:\py101\debug_test.py(14)main()
-> doubler(i)
(Pdb)
```

Here we start up the debugger and tell it to step into the code. It starts at the top and goes through the first two function definitions. Then it reaches the conditional and finds that it's supposed to execute the **main** function. We step into the main function and then use the **next** command. The **next** command will execute a called function if it encounters it without stepping into it. If you want to step into the called function, then you'll only want to just use the **step** command.

When you see a line like > **c:py101debug_test.py(13)main()**, you will want to pay attention to the number that's in the parentheses. This number is the

current line number in the code.

You can use the **args** (or **a**) to print the current argument list to the screen. Another handy command is **jump** (or **j**) followed by a space and the line number that you want to “jump” to. This gives you the ability to skip a bunch of monotonous stepping to get to the line that you want to get to. This leads us to learning about breakpoints!