III B.Tech.

Computer Science & Engineering

CSE304: PYTHON PROGRAMMING WITH WEB FRAMEWORKS

UNIT – I: Testing & Debugging using Python IDLE

By
Mrs. S. KAMAKSHI, AP-III / CSE
School of Computing

Testing and Debugging

- Goal of Testing
  - To find all error before it is put into production
- Goal of Debugging
  - Fix (correct) the errors before it is put into production

Types of Errors

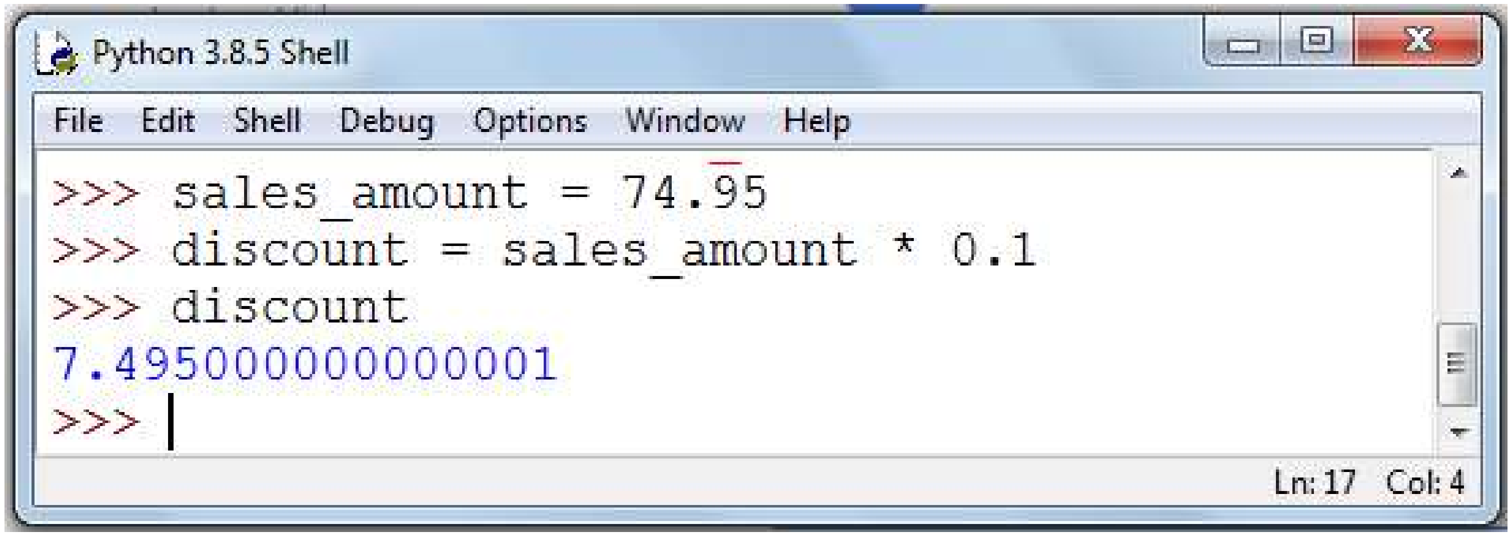
- Compile time Error (syntax error)
  - Violates the rules for Python
  - Identified during compilation
- Run time Error
  - Throws exceptions during the execution of the program
  - Identified during execution
- Logical Error
  - Produces wrong results
  - Hard to identify

Common Syntax Errors

- Misspelling keywords
- Forgetting colon(:) at the end of the opening line of function definition, if clause, else clause, while statement, for statement, try clause, except clause etc.
- Forgetting opening or closing quotation marks or parentheses
- Improper Indentation

Problems with floating point arithmetic

- Imprecise arithmetic results due to precision problem
- Eg.



```
>>> sales_amount = 74.95
>>> discount = sales_amount * 0.1
>>> discount
7.495000000000001
>>> 
```

Recommendations

- Test the program with valid input data and make sure the results are correct
- Test the program with invalid data or unexpected user actions and make sure that the program fails without giving incorrect answers
- Test the program for a wide range of input entries
- Test the program on all boundary cases

Debugging Logical Errors

- Insert print() functions at key points in the code to display the intermediate results
- Follow top-down approach during program development and test it at every stage

Debugging Features in IDLE

- Breakpoints
  - For setting: Right click on the line and select Set breakpoint from context menu
  - For removing: Right click on the line and select Clear breakpoint from the context menu
- Debugger
  - Turn on debugger: switch to shell and select Debug→ Debugger to get the debugger window
  - To start debugging, go back to editor and run the program
  - It displays the code in the Debug Control Window
  - Go button: Executes until next breakpoint
  - Step: Executes one statement at a time including statements in called functions
  - Over: Step the code one statement at a time, skipping over called functions, but still executing them
  - Out: Finish executing the current function and return to the calling function
  - Quit: End the execution of the program
- Source checkbox
  - Highlight the current line in the editor window
- Locals checkbox
  - Local variables in current scope are displayed at the bottom of the Debug Control Window
- Globals checkbox
  - To show global variables

Stack Viewer

- To open: Select Debug → Stack Viewer after an exception occurs
- To automatically open: Select Debug →Auto-open Stack Viewer
- Lists the functions in reverse order in which they are called
- Expand the folders to examine the values of global and local variables
- To jump to the line of code in editor, double-click the line in Stack Viewer