

IDLE is an integrated development environment that combines several development tools into one program, including the following:

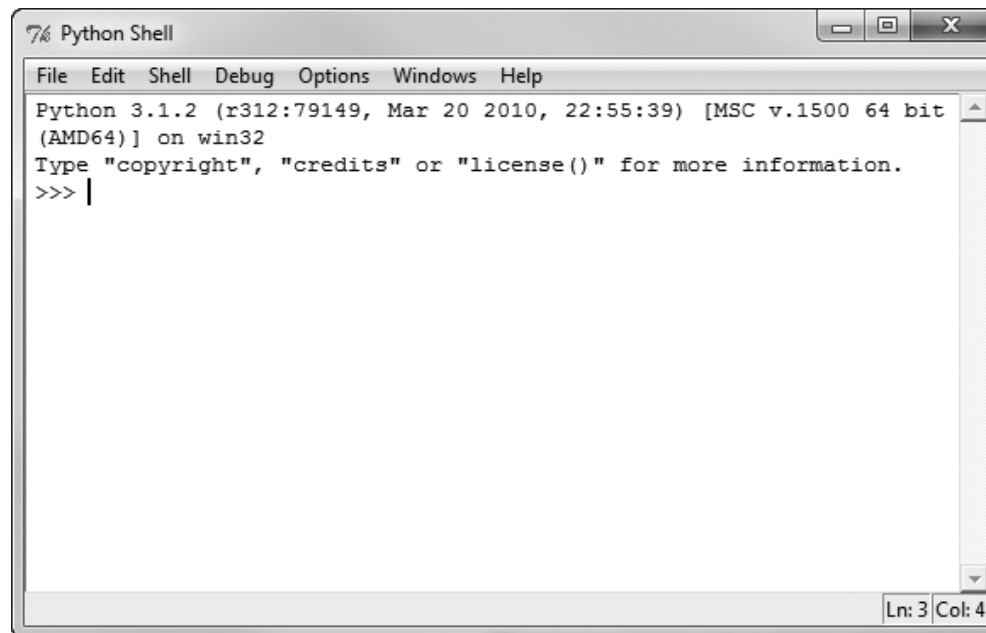


- A Python shell running in interactive mode. You can type Python statements at the shell prompt and immediately execute them. You can also run complete Python programs.
- A text editor that color codes Python keywords and other parts of programs.
- A “check module” tool that checks a Python program for syntax errors without running the program.
- Search tools that allow you to find text in one or more files.
- Text formatting tools that help you maintain consistent indentation levels in a Python program.
- A debugger that allows you to single-step through a Python program and watch the values of variables change as each statement executes.
- Several other advanced tools for developers.

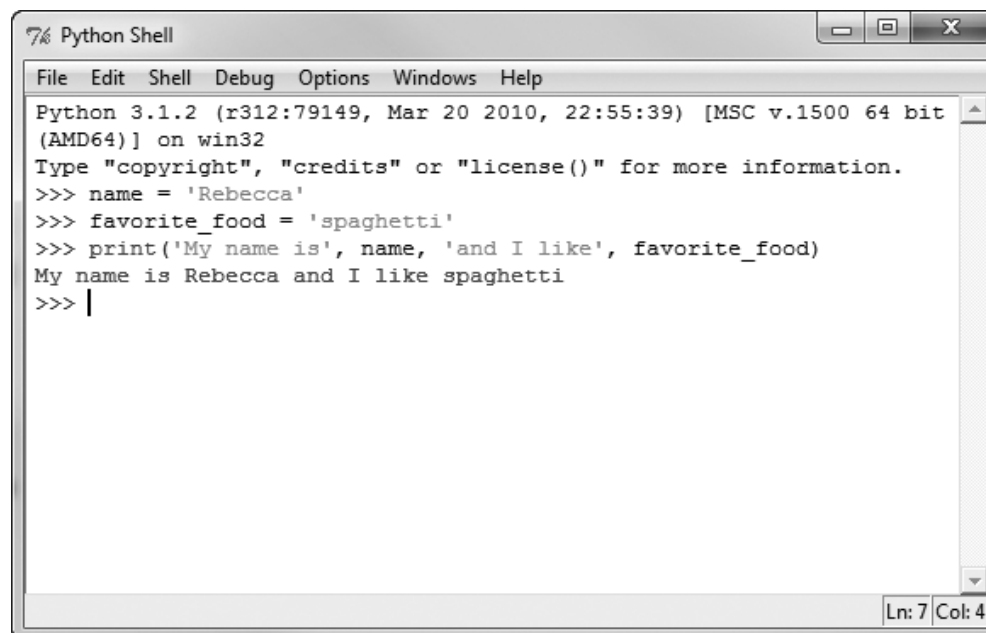
The IDLE software is bundled with Python. When you install the Python interpreter, IDLE is automatically installed as well. This appendix provides a quick introduction to IDLE, and describes the basic steps of creating, saving, and executing a Python program.

Starting IDLE and Using the Python Shell

After Python is installed on your system a Python program group will appear in your Start menu’s program list. One of the items in the program group will be titled *IDLE (Python GUI)*. Click this item to start IDLE and you will see the Python Shell window shown in Figure B-1. Inside this window the Python interpreter is running in interactive mode, and at the top of the window is a menu bar that provides access to all of IDLE’s tools.

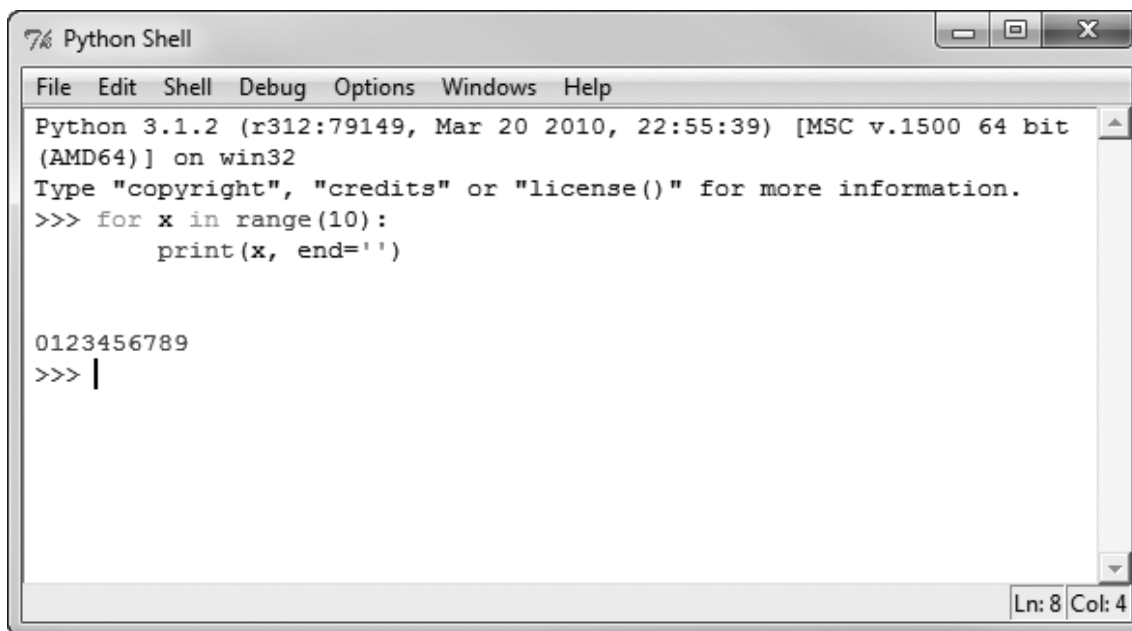
Figure B-1 IDLE shell window

The >>> prompt indicates that the interpreter is waiting for you to type a Python statement. When you type a statement at the >>> prompt and press the Enter key, the statement is immediately executed. For example, Figure B-2 shows the Python Shell window after three statements have been entered and executed.

Figure B-2 Statements executed by the Python interpreter

When you type the beginning of a multiline statement, such as an `if` statement or a loop, each subsequent line is automatically indented. Pressing the Enter key on an empty line indicates the end of the multiline statement and causes the interpreter to execute it. Figure B-3 shows the Python Shell window after a `for` loop has been entered and executed.

Figure B-3 A multiline statement executed by the Python interpreter

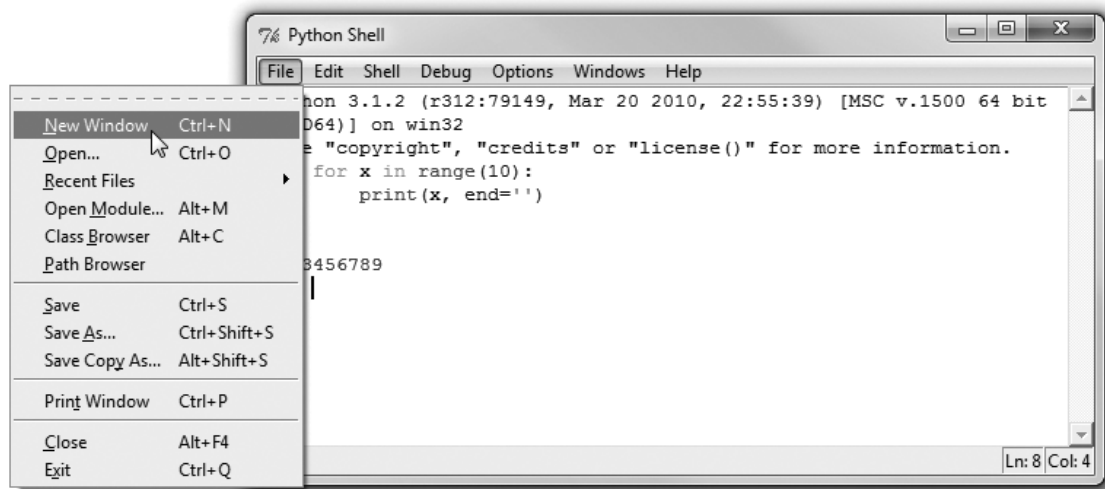
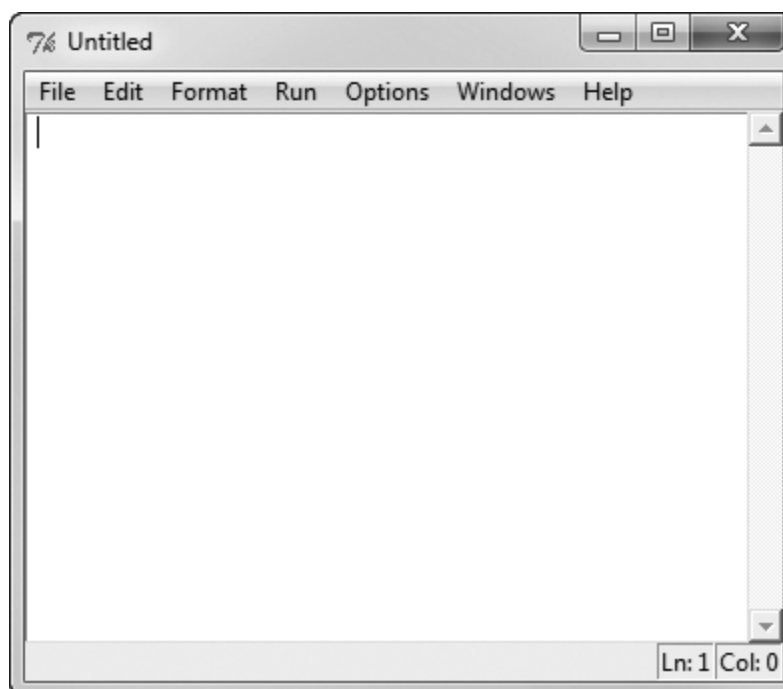
A screenshot of the Python Shell window. The window title is "Python Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The text area shows the following content:

```
Python 3.1.2 (r312:79149, Mar 20 2010, 22:55:39) [MSC v.1500 64 bit  
(AMD64)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>> for x in range(10):  
    print(x, end='')  
  
0123456789  
>>> |
```

The status bar at the bottom right indicates "Ln: 8 Col: 4".

Writing a Python Program in the IDLE Editor

To write a new Python program in IDLE you open a new editing window. As shown in Figure B-4 you click File on the menu bar, then click New Window. (Alternatively you can press Ctrl+N.) This opens a text editing window like the one shown in Figure B-5.

Figure B-4 The File menu**Figure B-5** A text editing window

To open a program that already exists, click File on the menu bar, then Open. Simply browse to the file's location and select it, and it will be opened in an editor window.

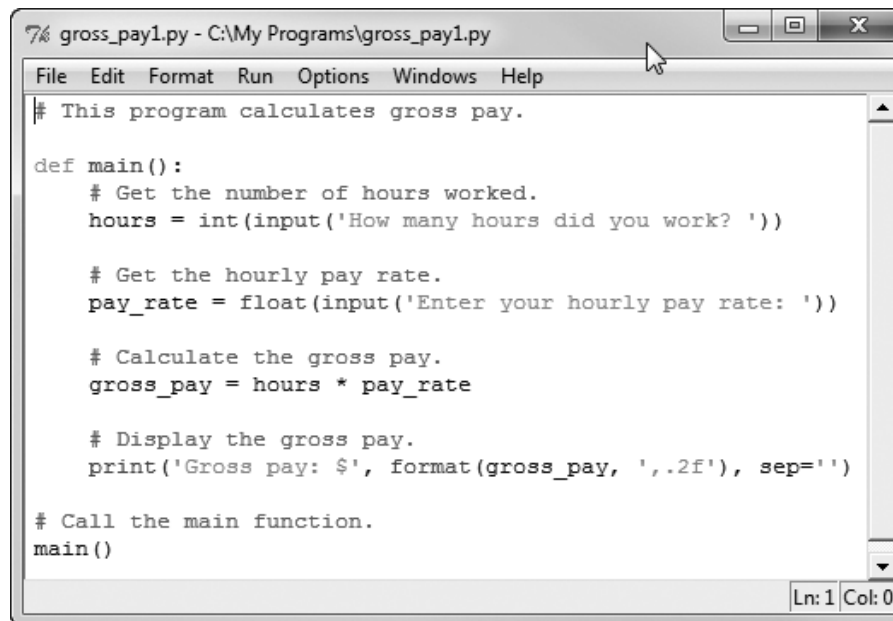
Color Coding

Code that is typed into the editor window, as well as in the Python Shell window, is colored as follows:

- Python keywords are displayed in orange.
- Comments are displayed in red.
- String literals are displayed in green.
- Defined names, such as the names of functions and classes, are displayed in blue.
- Built-in functions are displayed in purple.

Figure B-6 shows an example of the editing window containing colored Python code.

Figure B-6 Colorized code in the editing window



```
7% gross_pay1.py - C:\My Programs\gross_pay1.py
File Edit Format Run Options Windows Help
# This program calculates gross pay.

def main():
    # Get the number of hours worked.
    hours = int(input('How many hours did you work? '))

    # Get the hourly pay rate.
    pay_rate = float(input('Enter your hourly pay rate: '))

    # Calculate the gross pay.
    gross_pay = hours * pay_rate

    # Display the gross pay.
    print('Gross pay: $', format(gross_pay, ',.2f'), sep='')

# Call the main function.
main()
Ln: 1 Col: 0
```



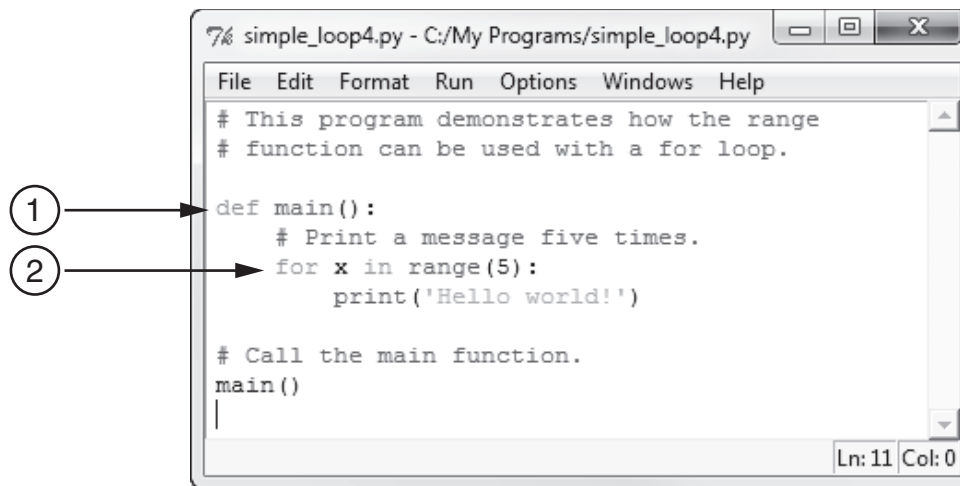
TIP: You can change IDLE's color settings by clicking Options on the menu bar, then clicking Configure IDLE. Select the Highlighting tab at the top of the dialog box, and you can specify colors for each element of a Python program.

Automatic Indentation

The IDLE editor has features that help you to maintain consistent indentation in your Python programs. Perhaps the most helpful of these features is automatic indentation. When you type a line that ends with a colon, such as an `if` clause, the first line of a loop, or a function header, and then press the Enter key, the editor automatically indents the lines

that are entered next. For example, suppose you are typing the code shown in Figure B-7. After you press the Enter key at the end of the line marked ①, the editor will automatically indent the lines that you type next. Then, after you press the Enter key at the end of the line marked ②, the editor indents again. Pressing the Backspace key at the beginning of an indented line cancels one level of indentation.

Figure B-7 Lines that cause automatic indentation



By default, IDLE indents four spaces for each level of indentation. It is possible to change the number of spaces by clicking Options on the menu bar, then clicking Configure IDLE. Make sure Fonts/Tabs is selected at the top of the dialog box, and you will see a slider bar that allows you to change the number of spaces used for indentation width. However, because four spaces is the standard width for indentation in Python, it is recommended that you keep this setting.

Saving a Program

In the editor window you can save the current program by performing any of these operations from the File menu:

- Save
- Save As
- Save Copy As

The Save and Save As operations work just as they do in any Windows application. The Save Copy As operation works like Save As, but it leaves the original program in the editor window.

Running a Program

Once you have typed a program into the editor, you can run it by pressing the F5 key, or as shown in Figure B-8, by clicking Run on the editor window's menu bar, then Run Module. If the program has not been saved since the last modification was made, you will see the dialog box shown in Figure B-9. Click OK to save the program. When the program runs you will see its output displayed in IDLE's Python Shell window, as shown in Figure B-10.

Figure B-8 The editor window's Run menu

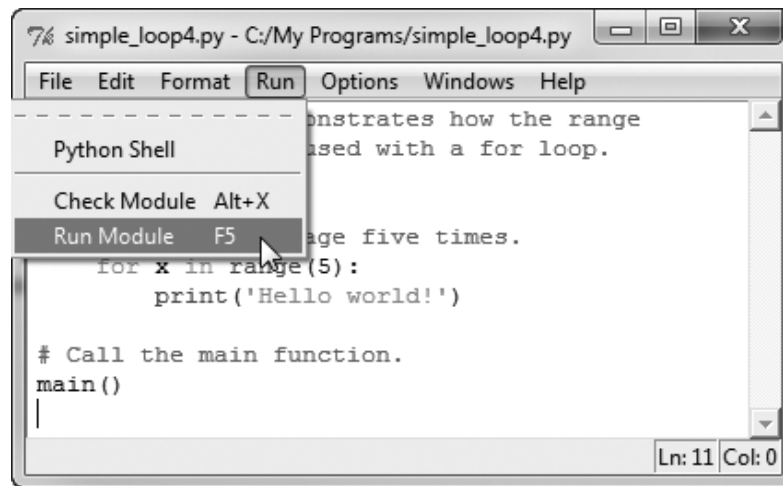


Figure B-9 Save confirmation dialog box

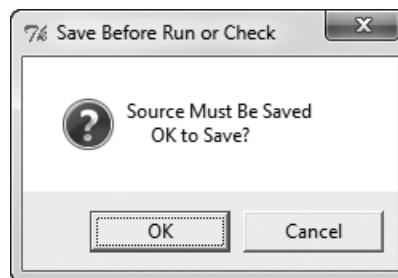
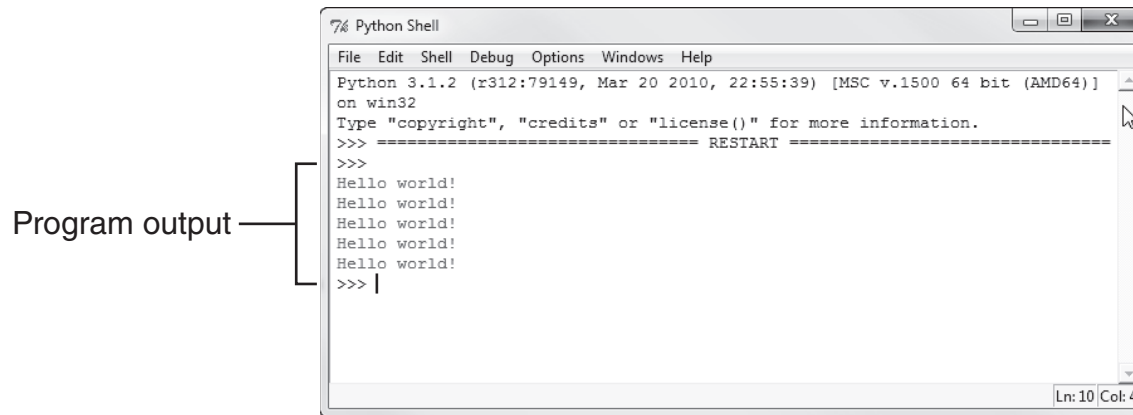
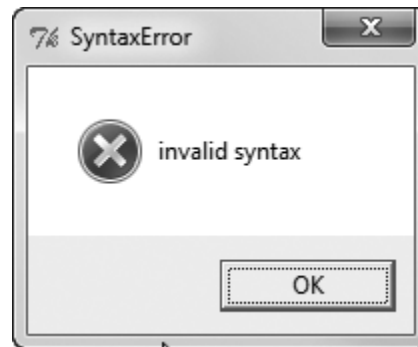


Figure B-10 Output displayed in the Python Shell window

If a program contains a syntax error, when you run the program you will see the dialog box shown in Figure B-11. After you click the OK button the editor will highlight the location of the error in the code. If you want to check the syntax of a program without trying to run it, you can click Run on the menu bar, then Check Module. Any syntax errors that are found will be reported.

Figure B-11 Dialog box reporting a syntax error

Other Resources

This appendix has provided an overview for using IDLE to create, save, and execute programs. IDLE provides many more advanced features. To read about additional capabilities, see the official IDLE documentation at www.python.org/idle.