|  |  |
| --- | --- |
|  | Chapter 3 |
| Writing Programs |

Topics Covered In This Chapter:

* Flow of execution
* Strings
* String concatenation
* Data types (such as strings or integers)
* Using the file editor to write progams
* Saving and running programs in IDLE
* The print() function
* The input() function
* Comments
* Case-sensitivity

That’s enough math for now. Now let’s see what Python can do with text. In this chapter, you’ll learn how to store text in variables, combine text, and display text on the screen.

Almost all programs display text to the user, and the user enters text into your programs through the keyboard. You’ll also make your first program in this chapter. This program displays the greeting, “Hello World!” and asks for the user’s name.

## Strings

In Python, text values are called strings. String values can be used just like integer or float values. You can store strings in variables. In code, string values start and end with a single quote ('). Try entering this code into the interactive shell:

>>> spam = 'hello'

The single quotes tell Python where the string begins and ends. They are not part of the string value’s text. Now if you type spam into the interactive shell, you will see the contents of the spam variable. Remember, Python evaluates variables to the value stored inside the variable. In this case, this is the string 'hello':

>>> spam = 'hello'

>>> spam

'hello'

Strings can have any keyboard character in them and can be as long as you want. These are all examples of strings:

'hello'

'Hi there!'

'KITTENS'

'7 apples, 14 oranges, 3 lemons'

'Anything not pertaining to elephants is irrelephant.'

'A long time ago, in a galaxy far, far away...'

'O\*&#wY%\*&OCfsdYO\*&gfC%YO\*&%3yc8r2'

## String Concatenation

String values can combine with operators to make expressions, just like integer and float values do. You can combine two strings with the + operator. This is string concatenation. Try entering 'Hello' + 'World!' into the interactive shell:

>>> 'Hello' + 'World!'

'HelloWorld!'

The expression evaluates to a single string value, 'HelloWorld!'. There is no space between the words because there was no space in either of the two concatenated strings, unlike this example:

>>> 'Hello ' + 'World!'

'Hello World!'

The + operator works differently on string and integer values because they are different data types. All values have a data type. The data type of the value 'Hello' is a string. The data type of the value 5 is an integer. The data type tells Python what operators should do when evaluating expressions. The + operator will concatenate string values but add integer and float values.

## Writing Programs in IDLE’s File Editor

Until now, you’ve been typing instructions into IDLE’s interactive shell one at a time. When you write programs though, you type in several instructions and have them run all at once. Let’s write your first program!

IDLE has another part called the file editor. Click on the **File** menu at the top of the interactive shell window. Then select **New Window**. A blank window will appear for you to type your program’s code into, like in Figure 3-1.

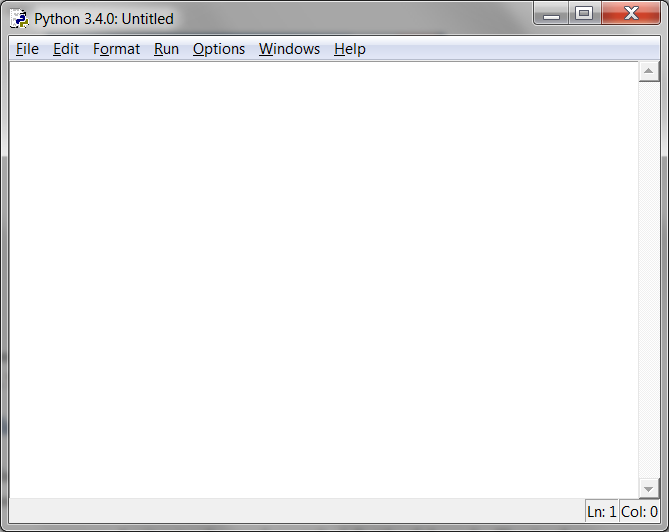
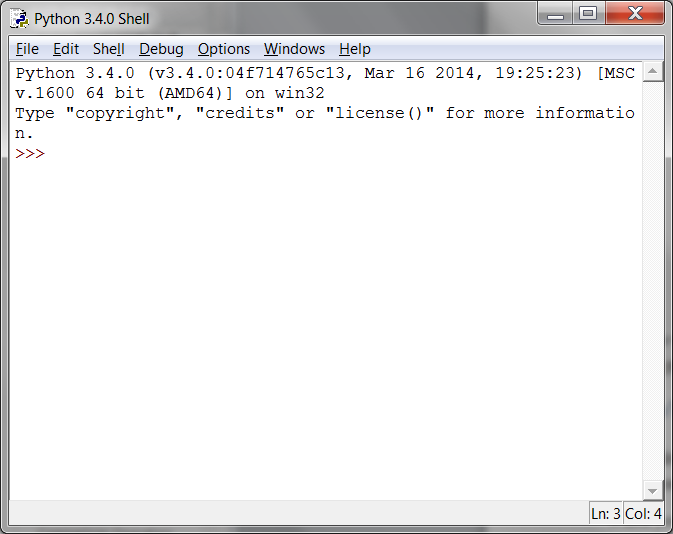
 

Figure 3-1: The file editor window (left) and the interactive shell window (right).

The two windows look similar, but just remember this: The **interactive shell window** will have the >>> prompt. The **file editor window** will not.

## Hello World!

It’s traditional for programmers to make their first program display “Hello world!” on the screen. You’ll create your own Hello World program now.

When you enter your program, don’t enter the numbers at the left side of the code. They’re there so this book can refer to code by line number. The bottom-right corner of the file editor window will tell you where the blinking cursor is. Figure 3-2 shows that the cursor is on line 1 and column 0.

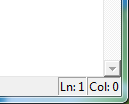


Figure 3-2: The bottom right of the file editor window tells you what line the cursor is on.

hello.py

Enter the following text into the new file editor window. This is the program’s source code. It contains the instructions Python will follow when the program is run.

**IMPORTANT NOTE!** The programs in this book will only run on Python 3, not Python 2. When the IDLE window starts, it will say something like “Python 3.4.2” at the top. If you have Python 2 installed, you can have Python 3 installed at the same time. To download Python 3, go to https://python.org/download/.

hello.py

1. # This program says hello and asks for my name.

2. print('Hello world!')

3. print('What is your name?')

4. myName = input()

5. print('It is good to meet you, ' + myName)

The IDLE program will write different types of instructions with different colors. After you’re done typing the code, the window should look like this:

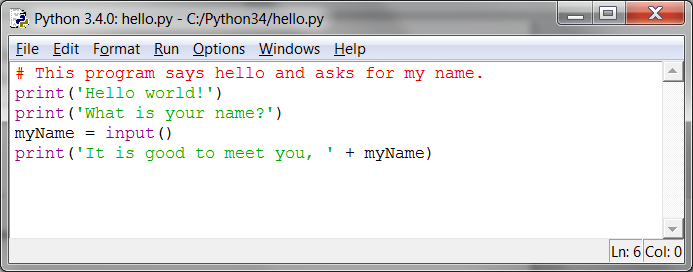


Figure 3-3: The file editor window will look like this after you type in the code.

## Saving Your Program

Once you’ve entered your source code, save it by clicking on **File ►** **Save As**. Or press Ctrl-S to save with a keyboard shortcut. Figure 3-4 shows the Save As window that will open. Enter hello.py in the **File name** text field then click **Save**.

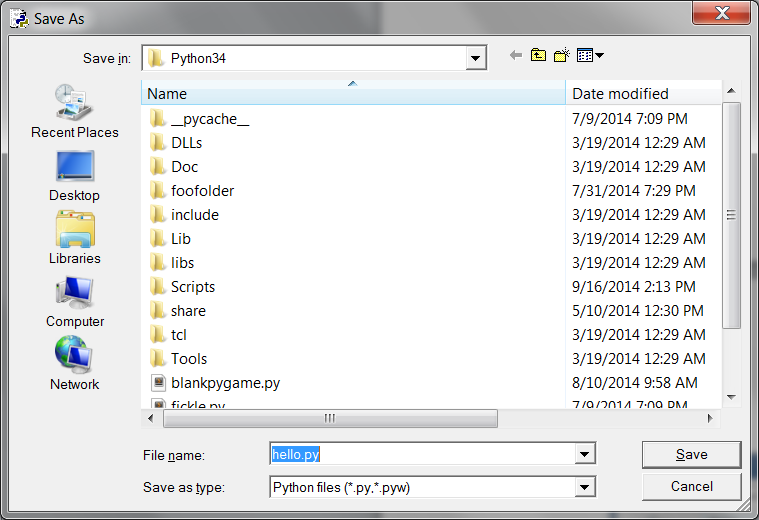


Figure 3-4: Saving the program.

You should save your programs often while you type them. That way, if the computer crashes or you accidentally exit from IDLE you won’t lose much work.

## Opening The Programs You’ve Saved

To load your previously saved program, click **File ► Open**. Choose the file in the window that appears and click the **Open** button. Your saved hello.py program will open in the File Editor window.

Now it’s time to run the program. Click **File ► Run ► Run Module** or just press F5 from the file editor window. Your program will run in the interactive shell window.

Enter your name when the program asks for it. This will look like Figure 3-5.

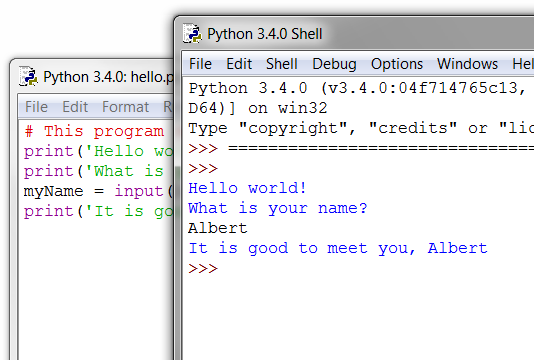


Figure 3-5: The interactive shell after running hello.py.

When you type your name and push enter, the program will greet you by name. Congratulations! You’ve written your first program and are now a computer programmer. Press F5 again to run the program a second time and enter another name.

If you got an error, compare your code to this book’s code with the online diff tool at http://invpy.com/diff. Copy and paste your code from the file editor into the web page and click the **Compare** button. This tool will highlight any differences between your code and the code in this book, like in Figure 3-6.

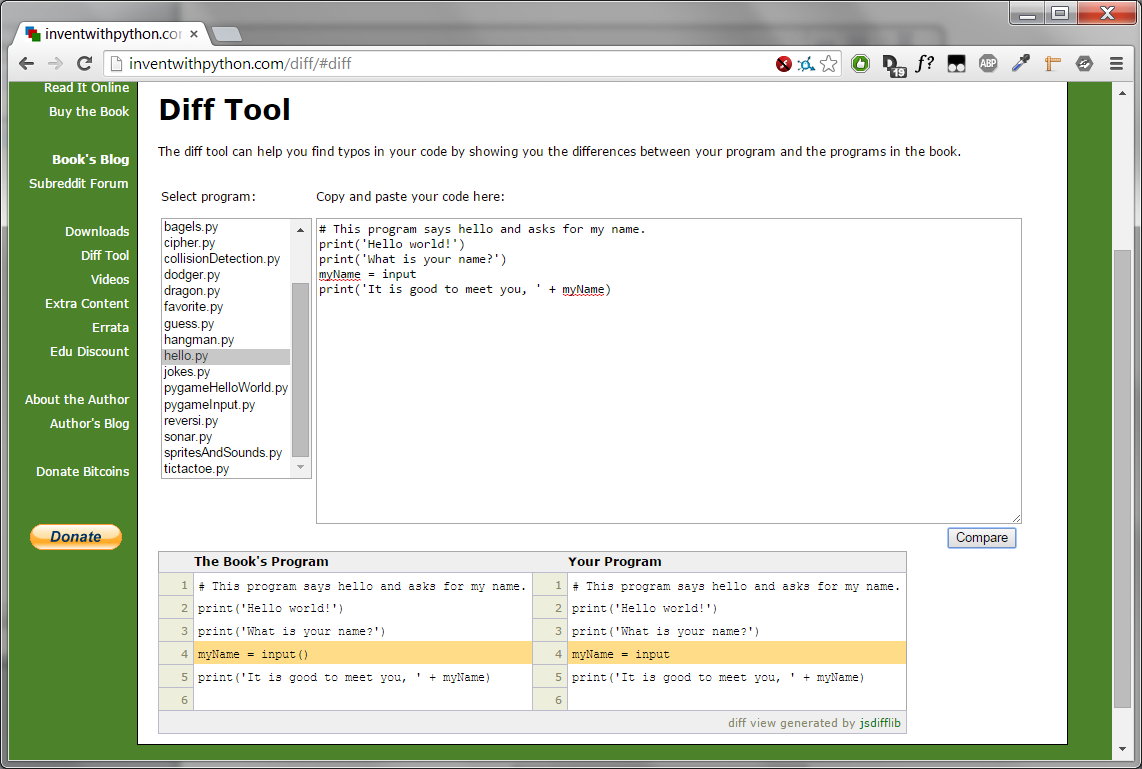


Figure 3-6: The diff tool at http://invpy.com/diff

While coding, if you get a NameError that looks like this:

Hello world!

What is your name?

Albert

Traceback (most recent call last):

File "C:/Python26/test1.py", line 4, in <module>

myName = input()

File "<string>", line 1, in <module>

NameError: name 'Albert' is not defined

...that means you are using Python 2, instead of Python 3. Install a version of Python 3 from https://python.org/download. Re-run the program with Python 3.

## How the “Hello World” Program Works

Each line of code is an instruction interpreted by Python. These instructions make up the program. A computer program’s instructions are like the steps in a cookbook recipe. Each instruction executes in order, beginning from the top of the program and going down the list of instructions.

The step Python is at in the program is called the execution. When the program starts, the execution is at the first instruction. After executing the instruction, the execution moves down to the next instruction.

Let’s look at each line of code to see what it’s doing. We’ll begin with line number 1.

Comments

1. # This program says hello and asks for my name.

This instruction is a comment. Any text following a # sign (called the pound sign) is a comment. Comments are not for Python, but for you, the programmer. Python ignores comments. Comments are the programmer’s notes about what the code does. You can write anything in a comment. To make it easier to read the source code, this book prints comments in a light gray-colored text.

Programmers usually put a comment at the top of their code to give their program a title.

Functions

A function is kind of like a mini-program inside your program. Functions contain several instructions to execute when the function is called. Python provides some built-in functions already. Two functions, print() and input(), are described next. The great thing about functions is that you only need to know what the function does, but not how it does it.

A function call is an instruction that tells Python to run the code inside a function. For example, your program calls the print() function to display a string on the screen. The print() function takes the string you type between the parentheses as input and displays the text on the screen.

To display Hello world! on the screen, type the print function name, followed by an opening parenthesis, followed by the 'Hello world!' string and a closing parenthesis.

The print() function

2. print('Hello world!')

3. print('What is your name?')

Lines 2 and 3 are calls to the print() function. A value between the parentheses in a function call is an argument. The argument on line 2’s print() function call is 'Hello world!'. The argument on line 3’s print() function call is 'What is your name?'. This is called passing the argument to the print() function.

In this book, function names have parentheses at the end. This makes it clear that print() means this book is talking about a function named print(), and not a variable named print. This is like the quotes around the number '42' telling Python that you are talking about a string '42' and not an integer 42.

The input() function

4. myName = input()

Line 4 is an assignment statement with a variable (myName) and a function call (input()). When input() is called, the program waits for the user to enter text. The text string that the user enters becomes the value that the function call evaluates to. Function calls can be used in expressions anywhere a value can be used.

The value that the function call evaluates to is called the return value. (In fact, “the value a function call returns” means the same thing as “the value a function call evaluates to”.) In this case, the return value of the input() function is the string that the user typed in-their name. If the user typed in “Albert”, the input() function call evaluates to the string 'Albert'. The evaluation looks like this:

myName = input()

▼

myName = 'Albert'

This is how the string value 'Albert' gets stored in the myName variable.

Using Expressions in Function Calls

5. print('It is good to meet you, ' + myName)

The last line is another print() function call. The expression 'It is good to meet you, ' + myName in between the parentheses of print(). However, arguments are always single values. Python will first evaluate this expression and then pass that value as the argument. If 'Albert' is stored in myName, the evaluation looks like this:

print('It is good to meet you, ' + myName)

▼

print('It is good to meet you, ' + 'Albert')

▼

print('It is good to meet you, Albert')

This is how the program greets the user by name.

Ending the Program

Once the program executes the last line, it terminates or exits. This means the program stops running. Python forgets all of the values stored in variables, including the string stored in myName. If you run the program again and enter a different name, the program will think that is your name.

Hello world!

What is your name?

Carolyn

It is good to meet you, Carolyn

Remember, the computer does exactly what you program it to do. Computers are dumb and just follow the instructions you give it exactly. The computer doesn’t care if you type in your name, someone else’s name, or just something silly. Type in anything you want. The computer will treat it the same way:

Hello world!

What is your name?

poop

It is good to meet you, poop

## Variable Names

Giving variables descriptive names makes it easier to understand what a program does. Imagine if you were moving to a new house and you labeled every moving box “Stuff”. That wouldn’t be helpful at all!

Instead of myName, you could have called this variable abrahamLincoln or nAmE. Python doesn’t care. It will run the program just the same.

Variable names are case-sensitive. Case-sensitive means the same variable name in a different case is considered a different variable. So spam, SPAM, Spam, and sPAM are four different variables in Python. They each contain their own separate values. It’s a bad idea to have differently cased variables in your program. Use descriptive names for your variables instead.

Variable names are usually lowercase. If there’s more than one word in the variable name, capitalize each word after the first. This makes your code more readable. For example, the variable name whatIHadForBreakfastThisMorning is much easier to read than whatihadforbreakfastthismorning. This is a convention: an optional but standard way of doing things in Python programming.

Short variable names are better than long names: breakfast or foodThisMorning is more readable than whatIHadForBreakfastThisMorning.

This book’s interactive shell examples use variable names like spam, eggs, ham, and bacon. This is because the variable names in these examples don’t matter. However, this book’s programs all use descriptive names. Your programs should use descriptive variable names too.

Summary

Once you learn about strings and functions, you can start making programs that interact with users. This is important because text is the main way the user and the computer will communicate with each other. The user enters text through the keyboard with the input() function. The computer will display text on the screen with the print() function.

Strings are just values of a new data type. All values have a data type, and there are many data types in Python. The + operator can concatenate strings.

Functions are used to carry out some complicated instruction as part of your program. Python has many built-in functions that you’ll learn about in this book. Function calls can be used in expressions anywhere a value is used.

The instruction in your program that Python is currently at is called the execution. In the next chapter, you’ll learn more about making the execution move in ways other than just straight down the program. Once you learn this, you’ll be ready to create games.