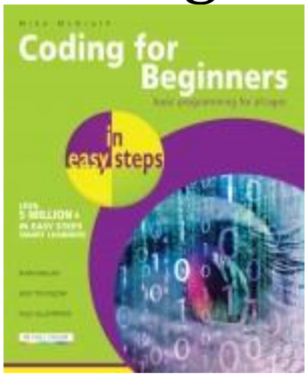
Coding for Beginners in easy steps



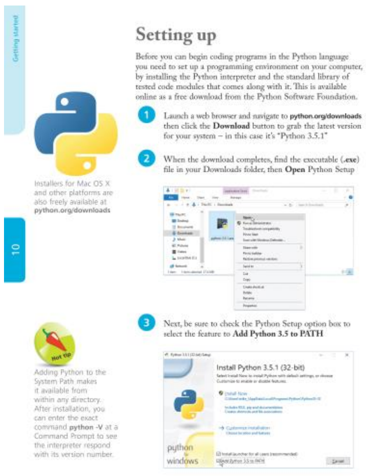
Coding for Beginners in easy steps has an easy-to-follow style that will appeal to anyone, of any age, who wants to begin coding computer programs. You need have no previous knowledge of any computer programming language so it's ideal for the newcomer, including youngsters needing to learn programming basics for the school curriculum.

Coding for Beginners in easy steps instructs you how to write code to create your own computer programs. It contains separate chapters demonstrating how to store information in *data structures*, how to control program flow using *control structures*, and how to create re-usable blocks of code in program *functions*. There are complete step-by-step example programs that demonstrate each aspect of coding, together with screenshots that illustrate the actual output when each program has been executed.

Coding for Beginners in easy steps begins by explaining how to easily create a programming environment on your own computer, so you can quickly begin to create your own working programs by copying the book's examples. After demonstrating the essential building blocks of computer programming it describes how to code powerful *algorithms* and demonstrates how to code *classes* for Object Oriented Programming (OOP). The examples throughout this book feature the popular Python programming language but additionally the final chapter demonstrates a comparison example in the C, C++, and Java programming languages to give you a rounded view of computer coding.

The code in the listed steps within the book is color-coded to precisely match the default color-coding of the Python IDLE editor, making it easier for beginners to grasp. By the end of this book you will have gained a sound understanding of coding and be able to write your own computer programs that can be run on any compatible computer.

Contents


Getting started

10

Setting up

Before you can begin coding programs in the Python language you need to set up a programming environment on your computer, by installing the Python interpreter and the standard library of tested code modules that comes along with it. This is available online as a free download from the Python Software Foundation.

1 Launch a web browser and navigate to **python.org/downloads** then click the **Download** button to grab the latest version for your system – in this case it's "Python 3.5.1"

2 When the download completes, find the executable (**.exe**) file in your Downloads folder, then **Open** Python Setup

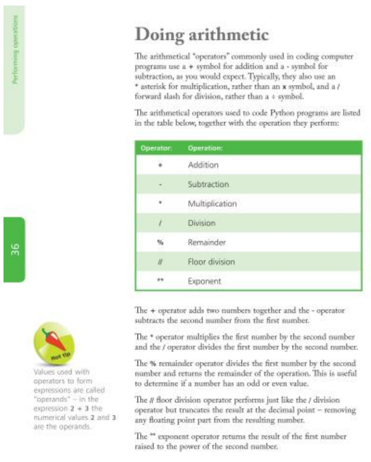
3 Next, be sure to check the Python Setup option box to select the feature to **Add Python 3.5 to PATH**

Install Python 3.5.1 (32-bit)

Installers for Mac OS X and other platforms are also freely available at **python.org/downloads**

Hot tip

Adding Python to the System Path makes it available from within any directory. After installation, you can enter the exact command **python -V** at a Command Prompt to see the interpreter respond with its version number.


Performing operations

36

Doing arithmetic

The arithmetical "operators" commonly used in coding computer programs use a + symbol for addition and a - symbol for subtraction, as you would expect. Typically, they also use an * asterisk for multiplication, rather than an x symbol, and a / forward slash for division, rather than a ÷ symbol.

The arithmetical operators used to code Python programs are listed in the table below, together with the operation they perform:

Operator:	Operation:
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Remainder
//	Floor division
**	Exponent

The + operator adds two numbers together and the - operator subtracts the second number from the first number.

The * operator multiplies the first number by the second number and the / operator divides the first number by the second number.

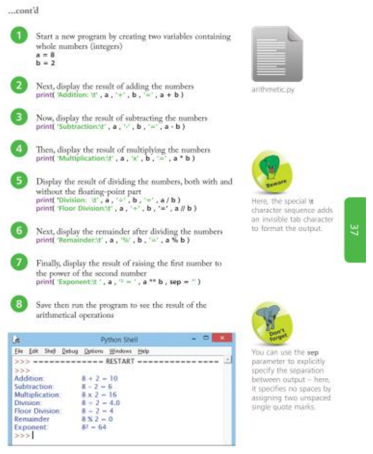
The % remainder operator divides the first number by the second number and returns the remainder of the operation. This is useful to determine if a number has an odd or even value.

The // floor division operator performs just like the / division operator but truncates the result at the decimal point – removing any floating point part from the resulting number.

The ** exponent operator returns the result of the first number raised to the power of the second number.

Hot tip

Values used with operators to form expressions are called "operands" – in the expression **2 + 3** the numerical values **2** and **3** are the operands.


...cont'd

1 Start a new program by creating two variables containing whole numbers (integers)
```
a = 8
b = 2
```

2 Next, display the result of adding the numbers
```
print( 'Addition: \t' , a , '+' , b , '=' , a + b )
```

3 Now, display the result of subtracting the numbers
```
print( 'Subtraction:\t' , a , '-' , b , '=' , a - b )
```

4 Then, display the result of multiplying the numbers
```
print( 'Multiplication:\t' , a , 'x' , b , '=' , a * b )
```

5 Display the result of dividing the numbers, both with and without the floating-point part
```
print( 'Division:  \t' , a , '÷' , b , '=' , a / b )
print( 'Floor Division:\t' , a , '÷' , b , '=' , a // b )
```

6 Next, display the remainder after dividing the numbers
```
print( 'Remainder:\t' , a , '%' , b , '=' , a % b )
```

7 Finally, display the result of raising the first number to the power of the second number
```
print( 'Exponent:\t ' , a , '² = ' , a ** b , sep = '' )
```

8 Save then run the program to see the result of the arithmetical operations

arithmetic.py

```
>>> ============= RESTART ==============
>>>
Addition:          8 + 2 = 10
Subtraction:       8 - 2 = 6
Multiplication:    8 x 2 = 16
Division:          8 ÷ 2 = 4.0
Floor Division:    8 ÷ 2 = 4
Remainder:         8 % 2 = 0
Exponent:          8² = 64
>>>
```

Beware

Here, the special \t character sequence adds an invisible tab character to format the output.

Don't forget

You can use the **sep** parameter to explicitly specify the separation between output – here, it specifies no spaces by assigning two unspaced single quote marks.

37