7 Built-in functions

Earlier, we saw that Perl has various "operators" represented by mathematical-type symbols. Sometimes these are the same symbols used in familiar school maths, such as `+` for addition and `-` for subtraction; sometimes they are slightly different symbols adapted to the constraints of computer keyboards, such as `*` for multiplication and `**` for raising to a power; and sometimes the symbols represent operations that we do not usually come across in maths lessons, e.g. "`.`" for concatenation.

Perl has many more built-in functions that could conveniently be represented by special symbols, though.[10] Most are represented by alphabetic codes. For instance, taking the square root of a number is a standard arithmetic operation, but the usual mathematical symbol, √, is nothing like any character in the ASCII character-set, so instead Perl represents it as `sqrt`.

When a function is represented by letters rather than special symbols, the alphabetic code is followed by a pair of round brackets, containing the expression to which the function is applied. Thus:

```
$c = 2;
print sqrt($c);

1.4142135623731
```

– `$c` here is said to be the "argument" of the function `sqrt()`.

Maastricht University
Leading in Learning!
Join the best at the Maastricht University School of Business and Economics!
Top master's programmes
• 33rd place Financial Times worldwide ranking: MSc International Business
• 1st place: MSc International Business
• 1st place: MSc Financial Economics
• 2nd place: MSc Management of Learning
• 2nd place: MSc Economics
• 2nd place: MSc Econometrics and Operations Research
• 2nd place: MSc Global Supply Chain Management and Change
Sources: Keuzegids Master ranking 2013; Elsevier 'Beste Studies' ranking 2012; Financial Times Global Masters in Management ranking 2012
Maastricht University is the best specialist university in the Netherlands (Elsevier)
Visit us and find out why we are the best!
Master's Open Day: 22 February 2014
www.mastersopenday.nl

Perl is fond of offering short-cuts, and commonly it is allowable to omit the brackets round function arguments; `sqrt $c` works as well as `sqrt($c)`. But although Perl does not require the brackets, it is probably a good idea to include them in your programs as a visual reminder of what is going on, at least until you grow in confidence sufficiently not to need that support – and in this book I show a pair of brackets with names of functions, to make it obvious that this is what they are.

(The term `print` is itself really a kind of built-in function, so that to be fully consistent with my own principle I ought to have been writing e.g. `print($a)` rather than `print $a`. Perl happily accepts either; I have made an exception and omitted brackets round arguments to `print`, so as to avoid a confusing piling-up of brackets in a case like `print(sqrt($c))`.)

Other built-in functions stand for operations that have no well-known mathematical symbol. For instance, `int()` gives the whole-number part of a decimal number:

```
print int(3.756);
```

```
3
```

(Notice that `int()` does not round to the *nearest* whole number – it merely throws away the fraction part. We shall see later how to proceed if we want a function that rounds up as well as down.)

I have talked about "operators" and "functions", but looked at logically these are the same kind of thing. Both of them are devices which accept one or more values as arguments, and use them to deliver a new value. The symbol `*` accepts two arguments and delivers their product. The symbol `sqrt` accepts one argument and delivers its square root. Perl programmers talk about "operators" when the symbol is non-alphabetic and is placed between its arguments, if it has more than one; they talk about "functions" when the symbol is alphabetic and precedes its argument(s). Most "functions" have only one argument, but some have more; most "operators" have multiple arguments, but a few have just one. The distinction is one of terminology only, not a real contrast, and even as a distinction of terminology it is blurry.

We shall not give a comprehensive list of the built-in functions here; this is a topic to explore gradually with the help of a fullscale Perl manual, as your programming needs develop.

Without trying to survey the complete list, it is worth noticing from the start that by no means all functions take a numerical argument and deliver a numerical result, as `sqrt()` and `int()` do.

So, for instance, `length()` gives the number of characters in a string, that is, it takes a string argument and delivers a number:

```
$cabbage = "The quality of mercy is not strained";
print length($cabbage);
```

```
36
```

On the other hand, `chr()` is a function which takes a number as argument and delivers the character whose ASCII code that number is:

```
print chr(65);
```

```
A
```

Some functions have strings for both argument and result, e.g. `lc()` makes a string all lower-case and `uc()` makes it all upper-case:

```
print lc("God Save the Queen!");
```

```
god save the queen!
```

The function `substr()` takes multiple arguments, normally one string and two numbers, in order to extract a substring from a longer string:

```
$letters = "abcdefghi";
print substr($letters, 4, 2);
```

```
ef
```

– the second argument, in this case 4, locates the starting-point within the first argument (counting the initial character of the string as character 0), and the third argument, in this case 2, gives the desired length of substring. Furthermore, while `substr()` most commonly takes three arguments, as here, it can take more – if another string is included as fourth argument, it will replace the substring within the first argument:

```
$letters = "abcdefghi";
print substr($letters, 4, 2, "XYZ");
```

```
ef
```

```
print $letters;
```

```
abcdXYZghi
```

or it can take fewer – if only the first two arguments are specified, `substr()` will deliver everything from the starting-point onwards:

```
$letters = "abcdefghi";
print substr($letters, 4);
```

```
efghi
```

Related to `substr()` is `index()`, which shows where in a string a substring occurs; provided the substring does occur, `index()` returns the location in the longer string of its first character (starting from zero), if it does not occur `index()` gives –1:

```
$a = "curiosity shop";
print index($a, "y s");
print "\t";
print index($a, "ys");
print "\n";
```

```
8   -1
```

A few Perl functions take no arguments at all. The function `rand()` is commonly used with no arguments, to produce a random number between 0 and 1:

```
print rand();
```

```
0.672787631469877
```

– though one can alternatively have the number drawn from the interval between zero and a different upper bound *x* by supplying *x* as an argument to the function:

```
print rand(4.5);
```

```
3.874540028261
```

In all Perl has about eighty built-in functions; a reader who looks through the list in a comprehensive Perl manual will probably find some which are specially useful for his particular application area.