**Introduction**

Perl is a high level, interpreted, general-purpose programming language originally developed for text manipulation. It borrows many features from C and Shell script and is used for system administration, networking, and other applications that involve user interfaces.

Perl is the Swiss Army chainsaw of scripting languages it is both powerful and adaptable. It was developed by a systems administrator at NASA, named Larry Wall in 1980's.

Perl was developed to make report processing easier. Since then, it is used for different works like automating system administration, acting as glue between different computer systems and, of course, being one of the most popular languages for CGI programming on the Web.

Perl plays well with your personal programming style.

**Introduction to –**

1. **Functions and Statements**
2. **Numbers, Strings and Quotes**
3. **Variables**
4. **Comments**
5. **Functions and Statements**

Perl has a rich library of built-in *functions*. They’re the verbs of Perl, the commands that the interpreter runs. You can see a list of all the built-in functions in the [perlfunc](http://perldoc.perl.org/index-functions.html) man page (perldoc perlfunc, from the command line). Almost all functions can take a list of commma-separated *parameters*.

The print function is one of the most frequently used parts of Perl. You use it to display things on the screen or to send information to a file. It takes a list of things to output as its parameters.

print "This is a single statement.";

print "Look, ", "a ", "list!";

A Perl program consists of statements, each of which ends with a semicolon. Statements don’t need to be on separate lines; there may be multiple statements on one line. You can also split a single statement across multiple lines.

print "This is "; print "two statements.\n";

print "But this ", "is only one statement.\n";

### Numbers, Strings, and Quotes

There are two basic data types in Perl: numbers and strings.

Numbers are easy; we’ve all dealt with them. The only thing you need to know is that you never insert commas or spaces into numbers in Perl. Always write 10000, not 10,000 or 10 000.

Strings are a bit more complex. A string is a collection of characters in either single or double quotes:

'This is a test.'

"Hi there!\n"

The difference between single quotes and double quotes is that single quotes mean that their contents should be taken literally, while double quotes mean that their contents should be interpreted. Remember the character sequence \n? It represents a newline character when it appears in a string with double quotes, but is literally the two characters backslash and n when it appears in single quotes.

use feature ':5.10';

say "This string\nshows up on two lines.";

say 'This string \n shows up on only one.';

(Two other useful backslash sequences are \t to insert a tab character, and \\ to insert a backslash into a double-quoted string.)

### Variables

If functions are Perl’s verbs, then variables are its nouns. Perl has three types of variables: scalars, arrays, and hashes. Think of them as things, lists, and dictionaries respectively. In Perl, all variable names consist of a punctuation character, a letter or underscore, and one or more alphanumeric characters or underscores.

Scalars are single things. This might be a number or a string. The name of a scalar begins with a dollar sign, such as $i or $abacus. Assign a value to a scalar by telling Perl what it equals:

my $i = 5;

my $pie\_flavor = 'apple';

my $constitution1789 = "We the People, etc.";

### Comments

Some of the code samples from the previous section contained code comments. These are useful for explaining what a particular piece of code does, and vital for any piece of code you plan to modify, enhance, fix, or just look at again. (That is to say, comments are important.)

Anything in a line of Perl code that follows a # sign is a comment, unless that # sign appears in a string.

use feature ':5.10';

say "Hello world!"; # That's more like it.

# This entire line is a comment.