Perl - Data Types

Perl is a loosely typed language and there is no need to specify a type for your data while using in your program. The Perl interpreter will choose the type based on the context of the data itself.

Perl has three basic data types: scalars, arrays of scalars, and hashes of scalars, also known as associative arrays. Here is a little detail about these data types.

Sr.No.	Types & Description
1	**Scalar** Scalars are simple variables. They are preceded by a dollar sign ($). A scalar is either a number, a string, or a reference. A reference is actually an address of a variable, which we will see in the upcoming chapters.
2	**Arrays** Arrays are ordered lists of scalars that you access with a numeric index, which starts with 0. They are preceded by an "at" sign (@).
3	**Hashes** Hashes are unordered sets of key/value pairs that you access using the keys as subscripts. They are preceded by a percent sign (%).

Numeric Literals

Perl stores all the numbers internally as either signed integers or double-precision floating-point values. Numeric literals are specified in any of the following floating-point or integer formats −

Type	Value
Integer	1234
Negative integer	-100
Floating point	2000
Scientific notation	16.12E14
Hexadecimal	0xffff
Octal	0577

String Literals

Strings are sequences of characters. They are usually alphanumeric values delimited by either single (') or double (") quotes. They work much like UNIX shell quotes where you can use single quoted strings and double quoted strings.

Double-quoted string literals allow variable interpolation, and single-quoted strings are not. There are certain characters when they are proceeded by a back slash, have special meaning and they are used to represent like newline (\n) or tab (\t).

You can embed newlines or any of the following Escape sequences directly in your double quoted strings −

Escape sequence	Meaning
\\	Backslash
\'	Single quote
\"	Double quote
\a	Alert or bell
\b	Backspace
\f	Form feed
\n	Newline
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\0nn	Creates Octal formatted numbers
\xnn	Creates Hexideciamal formatted numbers
\cX	Controls characters, x may be any character
\u	Forces next character to uppercase
\l	Forces next character to lowercase
\U	Forces all following characters to uppercase
\L	Forces all following characters to lowercase
\Q	Backslash all following non-alphanumeric characters
\E	End \U, \L, or \Q

Example

Let's see again how strings behave with single quotation and double quotation. Here we will use string escapes mentioned in the above table and will make use of the scalar variable to assign string values.

Live Demo

```
#!/usr/bin/perl

# This is case of interpolation.
$str = "Welcome to \ntutorialspoint.com!";
print "$str\n";

# This is case of non-interpolation.
$str = 'Welcome to \ntutorialspoint.com!';
print "$str\n";

# Only W will become upper case.
$str = "\uwelcome to tutorialspoint.com!";
print "$str\n";

# Whole line will become capital.
$str = "\UWelcome to tutorialspoint.com!";
print "$str\n";

# A portion of line will become capital.
$str = "Welcome to \Ututorialspoint\E.com!";
print "$str\n";

# Backsalash non alpha-numeric including spaces.
$str = "\QWelcome to tutorialspoint's family";
print "$str\n";
```

This will produce the following result −

```
Welcome to
tutorialspoint.com!
Welcome to \ntutorialspoint.com!
Welcome to tutorialspoint.com!
WELCOME TO TUTORIALSPOINT.COM!
Welcome to TUTORIALSPOINT.com!
Welcome\ to\ tutorialspoint\'s\ family
```