Perl Memoire

[Installing Modules 2](#_Toc74075334)

[strings 3](#_Toc74075335)

[string functions 3](#_Toc74075336)

[arrays 4](#_Toc74075337)

[syntax 4](#_Toc74075338)

[array functions 4](#_Toc74075339)

[hashes 5](#_Toc74075340)

[syntax 5](#_Toc74075341)

[hash functions 5](#_Toc74075342)

[files 6](#_Toc74075343)

[file functions 6](#_Toc74075344)

[maths 7](#_Toc74075345)

[math functions 7](#_Toc74075346)

[Module Data 8](#_Toc74075347)

[Data::Dumper 8](#_Toc74075348)

# Installing Modules

cpan <module>

# strings

## string functions

chomp()

length()

substr()

uc()

ucfirst()

lc()

lcfirst()

chr()

chop()

index()

rindex()

sprint()

ord()

quotemeta()

split()

# arrays

## syntax

Length – array returns its length when used in a scalar context

Is Empty:

if (!@var\_array)

if (@var\_array == 0)

if (scalar @var\_array == 0)

## array functions

push(*Array, List*)

Pushes the values in *List* onto the end of *Array*. Returns the number of elements in resulting array.

pop(*Array*)

Remove the last element of *Array* and return it, or return *undef* if *Array* is empty.

shift(*Array*)

Remove the first element of *Array* and return it, or return *undef* if *Array* is empty. Remaining elements are all shifted left.

unshift(*Array, List*)

Places the values of *List* onto the beginning of *Array*, shifting existing values right. Returns number of elements in resulting array.

sort(*List*)

sort(*Subroutine, List*)

sort(*Block, List*)

Sort *List* and return result. Sorting order may be specified by *Subroutine*, or *Block.* Behaviour is undefined in scalar context. *Subroutine* must be a subroutine (or a reference to a subroutine) that returns greater-than zero, less-than zero, or zero, given two arguments, according to how those arguments are to be ordered.

Examples of *Block*:

{ $a cmp $b } # alphabetical

{ $a <=> $b } # numeric

wantarray()

Returns *true* if called within function that was called in list context. Returns *false* if called within function that was called in scalar context. Returns *undef* if called within function that was called in void context.

exists(*Expr*)

If *Expr* specifies an element of a hash, return *true* if the specified element has ever been initialized (even if corresponding value if undefined). Use of *exists()* on list elements is (strongly) discouraged due to often surprising behaviour (use *defined()* instead).

defined(*Var*)

Returns *true* if variable *Var* has value other than *undef*.

grep(*Expression, @Array*)

Extract any elements from the given *Array* which evaluate *true* given the regex *Expression*. Returns list of results in list context, or number of results in scalar context.

split(*Pattern*)

split(*Pattern, Expr*)

split(*Pattern, Expr, Limit*)

Split string *Expr* (use *$\_* if not given) into a list of strings, using *Pattern*. If *Pattern* is an empty string, *Expr* is split between characters. Returns resulting list in list context, or size of this list in scalar context. If *Limit* is given, it specifies the maximum number of fields into which *Expr* may be split (which is 1 greater than the number of splits that will be made). If *Limit* is negative, as many fields as possible will be produced. If it is unspecified, or zero, trailing empty fields are stripped. When assigning result of *split()* to list of variables, *Limit* is implicitly 1 greater than the number of variables in said list.

join(*Delim, List*)

Join the separate strings of *List* into a single string, with fields separated by *Delim*, and return it.

reverse(*List*)

Reverses values in *List*. Returns reversed list in list context, or in scalar context, concatenated string of reversed list with characters of each element also reversed.

# hashes

## syntax

Initialising:

%var\_hash = ('Welcome' => 10, 'to' => 20, 'Geeks' => 80);

Access element:

print “$var\_hash{‘Welcome’}”

Nested hash

Access element:

print “$var\_hash{‘key\_outer’}{‘key\_inner’}”

Set element:

$var\_hash{‘key\_outer’}{‘key\_inner’} = $value

## hash functions

values()

keys()

each()

delete()

# files

## file functions

open()

glob()

tell()

getc()

reverse()

rename()

print()

# maths

## math functions

exp()

hex()

srand()

sqrt()

oct()

rand()

log()

int()

sin()

cos()

atan2()

abs()

# Module Data

## Data::Dumper

# Features by version

v5.10

State (persistent) variables